Java

Programming

Lesson

មាតិកា

មេរៀនទី ៤

ការប្រើប្រាស់ *ARRAY*

មេរៀនទី ៥

ការប្រើប្រាស់ Object **នៅក្នុង** Java

មេរៀនទី ៦

អំពី Inheritance

មេរៀនទី ៧

អំពី *Package* និង *Interface*

មេរៀនទី ៨

ការប្រើ *Exception*

មេរៀនទី ៩

String និង *Collections*

មេរៀនទី ១០

អំពីការប្រើប្រាស់ *Events*

មេរៀនទី ១១

AWT Controls, Layout Managers និង Menus

មេរៀនទី ១២

អំពី *Applet*

មេរៀនទី ១៣

អំពីការប្រើប្រាស់ *Graphics*

មេរៀនទី ១

ការណែនាំឲ្យស្គាល់ Java
Introduction to Java

១.ប្រវត្តិនៃ Java

Java គឺជាភាសាមួយដែលត្រូវបានបង្កើតឡើងដោយលោក James Gosling ។ គាត់ជាបុគ្គលិកធ្វើការនៅក្រុមហ៊ុន Sun Microsystems នៅក្នុងគំរោងស្រាវជ្រាវមួយឈ្មោះថា Green Project នៅឆ្នាំ ១៩៩១។ គម្រោងនេះបានប្រើភាសាមួយដែលមានមូលដ្ឋានឈរលើភាសា C និង C++ ។ ដំបូងឡើយភាសានេះឈ្មោះថា Oak បន្ទាប់ពីលោកបានឃើញដើម Oak តាមរយៈបង្អួចការិយាល័យរបស់គាត់ក្នុងក្រុមហ៊ុន Sun។ តែឈ្មោះនេះត្រូវបានប្តូរតាមសំណើរបស់មិត្តរួមការងាររបស់គាត់ទៅជា Java បន្ទាប់ពីគាត់ត្រឡប់ពីហាងកាហ្វេ ហើយក៏មានឈ្មោះថា Java រហូតមកដល់សព្វថ្ងៃនេះ។

ការងាររបស់ Green Project មានការលំបាកយ៉ាងខ្លាំងជាហេតុធ្វើអោយកិច្ចព្រមព្រៀមសំខាន់មួយរបស់ក្រុមហ៊ុន Sun ត្រូវបានប្រគល់អោយក្រុមហ៊ុនដទៃ។ ការងាររបស់ Green Project ស្ទើតែដួលរលំទៅហើយ តែសំណាងល្អនៅឆ្នាំ ១៩៩៣ World Wide Web បានលេចធ្លោឡើងបានធ្វើអោយអ្នកធ្វើការនៅក្រុមហ៊ុន Sun មើលឃើញយ៉ាងច្បាស់ពីអនុភាពនៃការប្រើ Java ដើម្បីបង្កើត Web Page វាបានធ្វើអោយគម្រោងនេះដើរសារជាថ្មីវិញ។

២. Version របស់ Java

-នៅឆ្នាំ ១៩៩៥ Java 1.0 ជា Version មួយមានលក្ខណៈប្រសើរសម្រាប់ប្រើលើ World Wide Web វាមាន Packages ចំនួន៨ ដោយមាន Classes ចំនួន ២១២។

-នៅឆ្នាំ ១៩៩៧ java 1.1 ជា Version មួយដែលបានបង្កើនដល់ភាសានូវមធ្យោបាយសម្រាប់បង្កើត និងប្រើ User Interface ជាពិសេសវាបានកែលំអលើផ្នែកការប្រើ events, inner class។ វាមាន Packages ចំនួន ២៣ ដោយមាន Classes ចំនួន ៥០៤។ លើសពីនេះ swing package បានកែលំអលើផ្នែកក្រាហ្វិចយ៉ាងលើសលប់ ប៉ុន្តែនៅមិនទាន់បានបញ្ចូលចំណុចសំខាន់ៗនៃភាសានៅឡើយ។

-នៅឆ្នាំ ១៩៩៩ Java 1.2 (ហៅថា Java 2) ជា Version មួយដែលបង្កើនដល់ភាសានូវមធ្យោបាយដ៏សំបូរបែបច្រើនជាងភាសាសំនេរកម្មវិធីផ្សេង២ទៀត។ វាមាន Packages ចំនួន ៥៩ ដោយមាន Classes ចំនួន ១៥២០ ។ វាបានបញ្ចូលនូវ Code និង Tools ដោយ Software Development Kit (SDK) ។ Java Foundation Classes (JFC) ដែលមានមូលដ្ឋានលើ Swing សម្រាប់កែលំអក្រាហ្វិច និង User Interfaces ក៏បានបញ្ចូលនូវចំនុចសំខាន់ៗនៃភាសាផងដែរ។ លើសពីនេះ Collection API ក៏បានបញ្ចូលដើម្បីឲ្យគេអាចប្រើ lists, sets និង hash maps។

-ឆ្នាំ ២០០០ Java 1.3 បានបង្កើតឡើង និងមាន Packages ចំនួន៧៦ ដោយមាន Classes ចំនួន ១៨៤២ ។ វាបានបង្កើនមុខងារឲ្យ Hotspot virtual machine ។

-ឆ្នាំ ២០០២ Java 1.4 បានបង្កើតឡើង និងមាន Packages ចំនួន ១៣៥ ដោយមាន Classes ចំនួន ២៩៩១ ។ វាបានកែលំអលើផ្នែក I/O និងអាចប្រើ XML...។

-ឆ្នាំ ២០០៤ Java 1.5 បានបង្កើតឡើង និងមាន Packages ចំនួន ១៦៥ ដោយមាន Classes ចំនួនច្រើនជាង ៣០០ ។ វាបានកែលំអលើផ្នែក Multithreaded និងទម្រង់ Output, metadata...។

-ឆ្នាំបន្តបន្ទាប់មកទៀតមាន Java 1.6 ត្រូវបានបង្កើតឡើងដែរ និងមាន Packages ចំនួន ២០០ ។

៣. <u>លក្ខណៈរបស់ Java</u>

គេថា Java មានលក្ខណៈ: Simple, Object Oriented, Statically Typed, Compiled and Interpreted, Architecture Neutral and Portable, Multithreaded, Garbage Collected, robust, Secure, Built-in Networking and Extensible ។

-Simple : អ្នកបង្កើត Java បានលុបបំបាត់ចោលនូវលក្ខណៈមិនចាំបាច់មួយចំនួនរបស់ភាសាសំនេរកម្មវិធី ដូចជា Java មិនប្រើ Pointers, Structures, Unions, Templates, Header file ឬ Multiple Inheritance ជាដើម។

-Object Oriented : ដូចនឹងភាសា C ឬ C++ ដែរ គឺប្រើ Classes ដើម្បីរៀបចំ Code ឲ្យទៅជាសំនុំមួយត្រឹមត្រូវ ហើយវាបង្កើត Objects តាមរយៈ Classes ។ Class របស់ Java អាចទទួលលក្ខណៈពី Class មួយផ្សេងទៀត។ ប៉ុន្តែ Class មួយមិនអាចទទួលលក្ខណៈពី Classes ច្រើនបានឡើយ។

-Statically Typed : គ្រប់ Object ទាំងអស់មុនពេលប្រើប្រាស់នៅក្នុងកម្មវិធីមួយ ដាច់ខាតត្រូវតែប្រកាសវាជាមុនសិន។ លក្ខណៈនេះអាចធ្វើឲ្យ Compiler របស់ Java រកឃើញនូវទីតាំង និងប្រាប់ឲ្យដឹងនូវប្រភេទទិន្នន័យដែលមិនត្រូវគ្នា។

-Compiled and Interpreted : មុននឹងយើងអាចដំណើរការកម្មវិធីមួយដែលសរសេរឡើងជាភាសា Java បាននោះលុះត្រាតែយើង Compile វាតាមរយៈ Compiler របស់វាជាមុនសិន។ នៅពេល Compile រួចដោយជោគជ័យ វានឹងបង្កើតបាន File មួយផ្សេងទៀតប្រភេទជា Byte-Code ដែលស្រដៀងគ្នាទៅនឹង Machine-code ហើយវាអាចដំណើរការក្នុងប្រព័ន្ធ Computer ដោយមាន Interpreter របស់ Java។ Interpreter ជាអ្នកបកប្រែពី Byte-code ទៅជាពាក្យបញ្ជាប្រភេទ machine-code ។ ហេតុនេះហើយបានជាគេថា Java មានលក្ខណៈ: Compiled and Interpreted ។

-Architecture Neutral and Portable: ដោយសារកម្មវិធីដែលសរសេរឡើងដោយ Java ត្រូវបាន Compiled ជាទម្រង់ Byte-code ដែលមានលក្ខណៈមិនអាស្រ័យនឹងទម្រង់ខាងក្នុងរបស់ Computer ជាហេតុធ្វើឲ្យកម្មវិធីដែលសរសេរឡើងដោយ Java អាចដំណើរការលើប្រព័ន្ធណាមួយក៏បាន(មិនប្រកាន់ plate form) ។

-Multithreaded: Java មាន threads សម្រាប់អនុវត្តធ្វើការប្រតិបត្តិការងារច្រើនក្នុងពេលតែមួយបាន។

-Garbage Collected : Java បានធ្វើការប្រមូលនូវអ្វីៗដែលមិនចាំបាច់ លុបចោលពី Memory ដោយខ្លួនឯង មិនចាំបាច់សរសេរ code ដើម្បីលុប Object ដែលមិនប្រើនោះទេ(បានន័យថា Variables ឬ Objects ណាដែលឈប់ប្រើហើយវានឹងលុបចោលដោយខ្លួនឯង ជួយសម្រួលដល់អ្នកសរសេរកម្មវិធីមិនព្រួយបារម្ភណ៍ពីការខ្វះខាត Memory ដោយសារ Objects មិនបានការណ៍នោះឡើយ)។

-Robust : ដោយសារ Interpreter របស់ Java ពិនិត្យគ្រប់ដំណើរការចូលទៅក្នុងប្រព័ន្ធទាំងអស់របស់កម្មវិធី ជាហេតុធ្វើឲ្យកម្មវិធីដែលសរសេរដោយ Java មិនប៉ះពាល់ដល់ប្រព័ន្ធ Computer ឡើយ។ កាលណាវាមាន Error វានឹងបង្កើតជា Exception។

-Secure : វាមិនគ្រាន់តែត្រួតពិនិត្យរាល់ដំណើរការចូលក្នុង Memory ប៉ុណ្ណោះទេ វាថែមទាំងធានាមិនឆ្លង Virus នៅពេលកំពុងដំណើរការកម្មវិធីទៀតផង ព្រោះវាមិនប្រើ Pointer ធ្វើឲ្យ Virus មិនអាចចូលទៅកាន់ Memory នៃប្រព័ន្ធ Computer បានឡើយ។

- Built-in Networking : Java បានបង្កើតឲ្យមានការប្រើលក្ខណៈជាបណ្តាញ ដោយនាំមកនូវ Classes ជាច្រើនសម្រាប់បង្កើតទំនាក់ទំនងជាមួយ Internet ។

-Extensible : Java អាចឲ្យប្រើនូវ native methods ដែលជា Functions ដែលសរសេរឡើងក្នុងភាសាផ្សេងៗ ដូចជា C ឬ C++ ជាដើម។ លក្ខណៈនេះវាធ្វើឲ្យអ្នកសរសេរកម្មវិធីសរសេរ function ដែលអាចប្រតិបត្តិការបានលឿនជាងការសរសេរ functions នៅក្នុង Java។ Native Methods ដំណើរការភ្ជាប់ទៅនឹងកម្មវិធី Java មានន័យថាវាបញ្ចូលជាមួយកម្មវិធីនៅក្នុងពេលដំណើរការកម្មវិធី។ នៅពេលដែល Java ត្រូវបានជម្រុញលើបញ្ហាល្បឿននោះ Native Methods ប្រហែលជាឥតត្រូវការទៀតឡើយ។ លើសពីនេះ Java អាចឲ្យគេប្រើជាមួយនឹងកម្មវិធីផ្សេងទៀតបាន ដូចជា Microsoft Access និង HTML ជាដើម។

៤. Java applications និង Java applets

គេអាចបង្កើតកម្មវិធីពី Java តាមពីរបៀបគឺតាមរយៈ Java applications ឬក៏តាម Java applets។

-Java Applications : គឺជាកម្មវិធីមួយដែលអាចដំណើរការនៅលើម៉ាស៊ីន Computer មួយ ក្រោមដំណើរការរបស់ប្រព័ន្ធ Computer នោះ។

-Java applets : គឺជាកម្មវិធីដែលអាចដំណើរការនៅលើ Internet វាអាចដំណើរការនៅលើ web browsers ណាក៏បាន វាអាចប្រើរូបភាព សំឡេង ឬក៏ជាវីដេអូ (video clip)។ Applets មិនគ្រាន់តែជាកម្មវិធីសម្រាប់ធ្វើឲ្យរូបភាពមានចលនា និង សម្លេងនោះទេ(Multimedia)ថែមទាំងជាកម្មវិធីមានលក្ខណៈឆ្លាតដែលអាចឲ្យអ្នកប្រើប្រាស់មានសកម្មភាពទៅវិញទៅមកក្នុងការបញ្ចូលទិន្នន័យនិងប្តូរទិន្នន័យទៀតផង។

៥. ការចាប់ផ្តើមដំណើរការកម្មវិធី

```
/* This is the first example java program.
   Save file as "Example.java" */
class Example{
```

```
//your program begins with a call to main()
public static void main(String[] arg){
        System.out.println("This a simple Java Program");
}
}
```

- ដើម្បីដំណើរការកម្មវិធីនេះជំហានទី១ យើងត្រូវ Compile វាសិនដោយអនុវត្តដូចខាងក្រោម៖

```
C:\JDK1.6\BIN\javac  Example.java
```

ចំពោះ (JDK1.6) អាស្រ័យទៅតាម Version របស់ Java។ ក្រោមពី Compile ហើយវាបានបង្កើត File មួយ ដែលមានប្រភេទជា byte-code ដែលមានឈ្មោះដូចនឹងឈ្មោះ class ហើយមាន extension (.class) គឺ Example.class ដែលជា File សម្រាប់ដំណើរការ។

- ជំហានទី ២ យើងត្រូវអនុវត្តដូចតទៅ៖

```
C:\JDK1.6\BIN\java  Example
```

ដោយមិនមាន extension (.java ឬ .class) ឡើយ។

End

មេរៀនទី ២

ប្រភេទទិន្នន័យ អញ្ញាត និងសញ្ញាណនព្វន្ត

Data Type, Variable and Arithmetic Operators

Java គឺជាភាសាមួយដែលមានលក្ខណៈ Statically typed បានន័យថារាល់អញ្ញាត កន្សោម ត្រូវបានកំណត់ប្រភេទទិន្នន័យច្បាស់លាស់មុនពេលប្រើប្រាស់ កត្តានេះជាហេតុធ្វើឲ្យ java មានភាពរឹងមាំ និងមានសុវត្ថិភាព។ នៅពេលដែលយើងហៅ អញ្ញាត ឬកន្សោមមកប្រើ compiler java បានត្រួតពិនិត្យរួចជាស្រេចចំពោះភាពត្រូវគ្នានៃទិន្នន័យ កាលណាវាមានប្រភេទទិន្នន័យមិនត្រូវគ្នា វានឹងមាន Error កើតឡើង ហើយវានឹងបញ្ឈប់ការបំលែងពី Source file ទៅជា file ដែលមាន extension (.class) ដែលជា file សម្រាប់ដំណើរការ។

១. ប្រភេទទិន្នន័យ Simple

ភាសា Java មានប្រភេទទិន្នន័យ Simple ចំនួន ៨ គឺ៖ byte, short , int, long, char, float, double និង boolean ។ដែលប្រភេទទាំងនេះត្រូវបានគេបែងចែកជា ៤ ក្រុមធំៗទៀតគឺ៖

-Integers : ក្នុងក្រុមប្រភេទ Integer នេះរួមមាន byte, short, int និង long ដែលជាប្រភេទទិន្នន័យចំនួនគត់។

-Floating point : ក្នុងក្រុមប្រភេទ Floating-point នេះរួមមាន float និង double ដែលជាប្រភេទទិន្នន័យចំនួនទសភាគ។

-Characters : ក្នុងក្រុមប្រភេទ Character នេះមានតែ char តែមួយគត់ ដែលជាតួអក្សរ សញ្ញាពិសេសផ្សេងៗ លេខមួយតួៗ ។

-Boolean : ក្នុងក្រុមប្រភេទ Boolean នេះក៏មានតែ boolean តែមួយគត់ដែរ ដែលតាងឲ្យ ពិត ឬមិនពិត(true ឬ false)។

២. អញ្ញាត (Variable)

នៅក្នុង Java programming មុននឹងយើងប្រើប្រាស់អញ្ញាតមួយបានលុះត្រាតែយើងប្រកាស់វាជាមុនសិន។ ការប្រកាសអញ្ញាតមួយមានទំរង់ដូចខាងក្រោម៖

```
type    identifier  [= value];
type    identifier [=value], identifier [=value],……;
```

- type គឺជាប្រភេទទិន្នន័យមួយរបស់ Java ឬជាឈ្មោះរបស់ class ឬអាចជា Interface។
- identifier គឺជាឈ្មោះរបស់ Variable ។ នៅក្នុង Java យើងអាចកំណត់តម្លៃតំបូងទៅឲ្យអញ្ញាតមួយ ដោយដាក់សញ្ញាស្មើ (=) និងតម្លៃដែលយើងចង់កំណត់។

<u>កំណត់ចំណាំ</u>៖ ការកំណត់ឈ្មោះរបស់អញ្ញាតមួយ មានលក្ខណៈដូចនឹងភាសា C ឬ C++ ដែរ។

ឧទាហរណ៍៖

int x, y, z; //ប្រកាសអញ្ញាតប្រភេទ Integer ចំនួនបីគឺ x, y និង z ដែលគ្មានការកំណត់តម្លៃដំបូង។

int x=9, y=1;// ប្រកាសអញ្ញាតប្រភេទ Integer ចំនួនពីរដោយ x មានតម្លៃ =៩ និង y មានតម្លៃ=១

២.១ ការកំណត់តម្លៃដែលមានលក្ខណៈប្រែប្រួល(Dynamic initialization)

Java Programming អាចអនុញ្ញាតឲ្យយើងធ្វើការកំណត់តម្លៃសដំបូងដែលមានលក្ខណៈប្រែប្រួលបាន ដោយប្រើកន្សោមលេខត្រឹមត្រូវណាមួយកំណត់ទៅឲ្យអញ្ញាតនៅពេលយើងប្រកាសវា។

ឧទាហរណ៍៖

```
class Demo{
public static void main(String[] agr){
        float a=2.0, b=3.0;
                double  c=Math.sqrt(a*a+b*b);
                System.out.println("The value of C = " + c);
        }
}
```

២.២ Scope និងLifetime របស់អញ្ញាត

ការប្រកាសអញ្ញាតនៅក្នុងកម្មវិធី Java គឺយើងអាចប្រកាសនៅក្នុង Block "{...}" ណាមួយក៏បាន នៅក្នុងកម្មវិធីជាច្រើនគេបានកំណត់ Scope របស់អញ្ញាតជាពីរគឺ៖ Global និង Local តែនៅក្នុងកម្មវិធី Java វិញកំណត់ទំហំ Scope ជាពីរដែរគឺ Class និងMethod។ Scope របស់អញ្ញាតមួយ អាចនៅក្នុង Scope មួយទៀត។

ឧទាហរណ៍៖

```
class DemoScope{
public static void main(String[] arg){
        int x;
        x=9;
        if (x==9){
                int y=10;
                System.out.println("x and y " + x + " "+ y);
                x= y * 3;
        }
        //y=30 ;                    //Error Because y here unknown out of y scope
        System.out.println("Value of x= " + x);
}
}
```

២.៣ ការបំលែងប្រភេទទិន្នន័យ(Type conversion and Casting)

នៅក្នុង Java វាមានលក្ខណៈពិសេសមួយចំនួនដូចជា៖

-កាលណាអញ្ញាតមួយ ត្រូវផ្ទេរតំលៃទៅឲ្យអញ្ញាតមួយទៀតមានប្រភេទទិន្នន័យដូចគ្នា វាធ្វើដោយស្វ័យប្រវត្តិ។

-ប្រភេទទិន្នន័យនៃអញ្ញាតអង្គខាងឆ្វេងធំជាងអង្គខាងស្តាំវាធ្វើដោយស្វ័យប្រវត្តិ។

-ប៉ុន្តែដើម្បីបំលែងប្រភេទទិន្នន័យដែលមិនត្រូវគ្នាយើងត្រូវប្រើតាមទំរង់ដូចខាងក្រោម៖

(target) value;

ឧទាហរណ៍៖ ដើម្បីបោះលែងពីប្រភេទ integer ទៅ boolean យើងត្រូវសរសេរដូចខាងក្រោម៖

```
int x;
byte b;
b=(byte) x;
```

```
/* Demonstration  conversion between integer to byte ,double to byte and double to integer*/
class DemoConversion{
    public static void main (String[] agr){
    byte b;
    int i=234;
    double d=123.456;
    b=(byte) i;
    System.out.println(" Conversion of integer to byte. ");
    System.out.println(" i  and b " + i  + " " + b);
    b=(byte) d;
    System.out.println(" Conversion of Double to byte. ");
    System.out.println(" d  and b " + d  + " " + b);
    i=(int) d;
    System.out.println(" Conversion of double to integer. ");
    System.out.println(" d  and i " + d  + " " +i);
    }
}
```

៣. <u>សញ្ញាណនព្វន្ត(Arithmetic Operators)</u>

សញ្ញាណនព្វន្តដែលប្រើនៅក្នុង Java programming ក៏ដូចជាកនៅក្នុងកម្មវិធី C ឬ C++ ដែរ ខាងក្រោមជាសញ្ញា និងអត្ថន័យរបស់វា៖

សញ្ញា	អត្ថន័យ
+	បូក(Addition) ដើរតួម៉្យាងទៀតជាសញ្ញាភ្ជាប់ឃ្លា
-	ដក (Subtraction or unary minus)
*	គុណ (Multiplication)
/	ចែក (Division)
%	ចែករកសំណល់ (Modulus)
++	កំណើនមួយឯកតា (increment)
+=	កំណត់តំលៃនៃផលបូក (Addition assignment)
--	តំហាយមួយឯកតា (Decrement)

-=	កំណត់តំលៃនៃផលដក (Subtraction assignment)
*=	កំណត់តំលៃនៃផលគុណ (Multiplication assignment)
/=	កំណត់តំលៃនៃផលចែក (Division assignment)
%=	កំណត់តំលៃនៃផលចែកសំណល់ (Modulus assignment)

៣.១ <u>សញ្ញាណនព្វន្តសំខាន់ៗ (The Basic Arithmetic Operators)</u>

នៅក្នុងនេះមានសញ្ញាណនព្វន្តសំខាន់ៗចំនួន ៤ គឺសញ្ញា បូក ដក គុណ និង ចែក។

ឧទាហរណ៍៖

```
class  DemoBasArth{
        pubic static void main (String[] arg){
                int a = 1 ;
                int b = a + 3 ;
                int c = b * 5 ;
                int d = c / 4 ;
                int e = d – a ;
                System.out.println(" a = " + a);
                System.out.println(" b = " + b);
                System.out.println(" c = " + c);
                System.out.println(" d = " + d);
                System.out.println(" e = " + e);
                double a1 = 1 ;
                double b1 = a1 + 2 ;
                double c1 = b1 * 3 ;
                double d1 = c1 / 4 ;
                double e1 = d1 – a1 ;
                System.out.println(" a1 = " + a1);
                System.out.println(" b1 = " + b1);
                System.out.println(" c1 = " + c1);
                System.out.println(" d1 = " + d1);
                System.out.println(" e1 = " + e1);
        }
}
```

៣.២ <u>សញ្ញាណនព្វន្តចែករកសំណល់កំណត់តំលៃ កំណើន និងតំហាយ</u>

- ការចែករកសំណល់ គឺជាការចែករកសំណល់នៃផលចែកណាមួយ មិនមែនចែករកផលចែកនោះទេ។

```
class DemoModulus{
        public static void main (String[] arg){
                int a = 34 ;
                double b = 89.5 ;
                System.out.println ( " a mod 10 = " + a % 10);
                System.out.println ( " b mod 10 = " + b % 10);
        }
}
```

- ការប្រើសញ្ញាកំណត់តំលៃនៅក្នុង Java ក៏ដូចជានៅក្នុង C ឬ C++ ដែរ។ ទំរង់នៃការសរសេរមានដូចតទៅ ៖

  Variable = Variable Operator Expression

  ឬអាចសរសេរ Variable Operator = Expression

  ឧទាហរណ៍៖

  a = a + 2 ; សមមូល a += 2 ;
  a = a - 4 ; សមមូល a -= 4 ;
  a = a *3 ; សមមូល a *= 3 ;
  a = a / 5 ; សមមូល a /= 5 ;
  a = a % 2 ; សមមូល a %= 2 ;

  ……

- ការប្រើសញ្ញា កំណើន ឬតំហយ នៅក្នុង Java ក៏ដូចជានៅក្នុង C ឬ C++ ដែរ។ ទំរង់នៃការសរសេរមានដូចតទៅ ៖

  ឧទាហរណ៍៖

  x = x + 1 ; សមមូលនឹងការសរសេរ x++ ;
  x = x - 1 ; សមមូលនឹងការសរសេរ x-- ;

<u>ចំណាំ</u>៖ កំណើន ឬ តំហយ ប្រើបានតែកើន ឬក៏ ថយមួយឯកតាតែប៉ុណ្ណោះ។ បានន័យថាកើន ម្តង ១ ឬក៏ថយម្តង ១ ។

៣.៣ <u>សញ្ញាណនព្វន្តធៀប(Relational Operators)</u>

ដូចគ្នាទៅនឹងភាសា C ឬ C++ ដែរ ។ អត្តន័យ និងខ្លឹមសាររបស់និមិត្តសញ្ញាមានដូចខាងក្រោម៖

សញ្ញាណនព្វន្ត	អត្តន័យ
==	ស្មើគ្នានឹង
!=	មិនស្មើគ្នានឹង (ខុសពី)
>	ធំជាង
<	តូចជា
>=	ធំជាង ឬស្មើ
<=	តូចជាង ឬស្មើ

End

មេរៀនទី ៣

ផ្នែកឆ្លាត់លក្ខខណ្ឌ
Control Structure

Control Structure នៅក្នុង Java Programming ត្រូវបានគេចែកជាបីប្រភេទគឺ៖ Selection, Iteration និង Jump។

១.ប្រភេទ Selection

គឺជាឃ្លាមួយប្រភេទដែលធ្វើការជ្រើសរើសអនុវត្តការងារណាមួយក្នុងចំណោមការងារជាច្រើន។ នៅក្នុង Java Programming Control Structure ប្រភេទ Selection ចែកចេញជាពីរគឺ If statement និង Switch Case statement ។

១.១ If statement

នៅក្នុង Java Programming, If statement មានទម្រង់បីទម្រង់ផ្សេងគ្នា៖

a- if *(condition)*
    statement(s);

b- if *(condition)*
    statement*(s)*;
  else
    statement*(s)*;

c- if *(condition 1)*
    statement*(s)*;
  else if *(condition 2)*
    statement*(s);*
  else if *(condition 2)*
    statement*(s)*;
    ……
  else if *(condition n)*
    statement*(s)*;
  else
    statement*(s)*;

១.១.១ if *(condition)*

នៅក្នុង if នេះវាធ្វើការពិនិត្យលក្ខខណ្ឌ បើ **ពិត** នោះវាអនុវត្ត statement ដែលនៅបន្ទាប់ពីវា។ statement នោះអាចមានមួយ statement ឬក៏ច្រើន statements។ បើសិនជាមានច្រើន statements គឺយើងត្រូវសរសេរនៅក្នុងសញ្ញា {…} តែបើមានតែមួយ statement ទេ យើងមិនបាច់ដាក់នៅក្នុងសញ្ញា

{...} ទេ, តែបើដាក់ក៏មិនខុសដែរ។ ប៉ុន្តែប្រសិនបើ **មិនពិត** វិញ នោះវារំលង statement មួយដែលនៅបន្ទាប់ ឬ statementទាំងឡាយណាដែលនៅក្នុងសញ្ញា {...} ហើយទៅអនុវត្ត statement*(s)* ដែលនៅបន្តបន្ទាប់ ។

ឧទាហរណ៍ទី ១៖

```
class DemoIF{
        public static void  main(String[] agr){
                boolean b;
                if (b==true)
                        System.out.println("Repaired or Change New  ");
                System.out.println("This is the Test of If condition in Form 1 ");
        }
}
```

ឧទាហរណ៍ទី ២៖

```
class DemoIF{
        public static void  main(String[] agr){
                boolean b;
                if (b==true) {
                        System.out.println("Repaired ");
                        System.out.println("Or Change New  ");
                        }
                System.out.println("This is the Test of If condition in Form 1 ");
        }
}
```

១.១.២ if ...else statement

នៅក្នុង if នេះវាធ្វើការពិនិត្យលក្ខខណ្ឌដូចគ្នានឹង if ខាងលើដែរគឺ បើ **ពិត** នោះវាអនុវត្ត statement ដែលនៅបន្ទាប់ពីវា។ statement នោះអាចមានមួយ statement ឬក៏ច្រើន statements។ បើសិនជាមានច្រើន statements គឺយើងត្រូវសរសេរនៅក្នុងសញ្ញា {...} តែបើមានតែមួយ statement ទេ យើងមិនបាច់ដាក់នៅក្នុងសញ្ញា {...} ទេ, តែបើដាក់ក៏មិនខុសដែរ។ ប៉ុន្តែប្រសិនបើ **មិនពិត** វិញ វានឹងអនុវត្ត statement(s) ទាំងឡាយណាដែលស្ថិតនៅក្រោម else block ។

ឧទាហរណ៍ទី ១៖

```
class DemoIfElse{
        public static void  main(String[] agr){
                boolean b;
                if (b==true)
                        System.out.println("You can go now!  ");
                else
                        System.out.println("Wait! Wait! You must repair first! ");
        }
}
```

ឧទាហរណ៍ទី ២៖

```
class DemoIfElse{
        public static void  main(String[] agr){
                boolean b;
            if (b==true) {
                System.out.println("You can go now. ");
                System.out.println("Have a safe trip! ");
                }
            else {
                System.out.println("Wait! Wait! You must repair first! ");
                System.out.print("Otherwise");
                System.out.println("You will have an accident cause from it  ");
                }
        }
}
```

១.១.៣ if ...else if statement

ចំពោះទម្រង់ទី ៣ នេះវាមានជម្រើសច្រើនក្នុងការអនុវត្ត មានន័យថាទី១ វាត្រួតពិនិត្យលក្ខខណ្ឌ if សិន បើ**ពិត** វាអនុវត្ត statement *(s)* ទាំងឡាយណាដែលស្ថិតនៅក្រោម block *condition ទី ១។* តែបើ *condition* ទី១ **មិនពិតទេ** វានឹងត្រួតពិនិត្យ *condition ទី២* ទៀត ហើយបើ *លក្ខខណ្ឌទី២* **ពិត** វានឹងអនុវត្តនូវរាល់ statement*(s)* ទាំងឡាយណាដែលស្ថិតនៅក្រោម *conditionទី២* នោះ ផ្ទុយទៅវិញបើ *condition ទី២* **មិនពិត** វានឹងធ្វើការត្រួតពិនិត្យ លក្ខខណ្ឌបន្តបន្ទាប់ ដោយមានលំនាំដូចខាងលើរហូតដល់ ***condition ទី n*** ។ ជាចុងក្រោមបើ condition ពីទី១ រហូតដល់ទី n នៅតែ **<u>មិនពិត</u>** នោះវានឹងអនុវត្ត statement ចុងក្រោយគឺ else statement block ។

<u>ចំណាំ៖</u> ចំពោះ ទម្រង់ទី ៣ នេះ អាចមាន else ក៏បាន ឬមិនមានក៏បាន។ ហើយបើវាជួបលក្ខខណ្ឌណាមួយ **ពិត** ហើយ នោះវានឹងមិនត្រួតពិនិត្យលក្ខខណ្ឌបន្តបន្ទាប់ទៀតឡើយ ។

```
// Display the number of days in each month depend on year
class DemoIfElseIf{
        public static void main (String[] agr){
                boolean b; // to check the year mod 4 equal 0
                int year; // store year input
                int month; //store month input
                year=Integer.parseInt(agr[0]);
                month=Integer.parseInt(agr[1]);
                if ((year%4)==0)
                        b=true;
                else
                        b=false;
```

```
                if (month==1)
                        System.out.println(" This " + year + " January has 31 days");
                else if(month==2)
                        if (b==true)
                                System.out.println(" This " + year + " February has 29 days");
                        else
                                System.out.println(" This " + year + " February has 28 days");
                else if (month==3)
                        System.out.println(" This " + year + " March has 31 days");
                else if (month==4)
                        System.out.println(" This " + year + " April has 30 days");
                else if (month==5)
                        System.out.println(" This " + year + " May has 31 days");
                else if (month==6)
                        System.out.println(" This " + year + " June has 30 days");
                else if (month==7)
                        System.out.println(" This " + year + " July has 31 days");
                else if (month==8)
                        System.out.println(" This " + year + " August has 31 days");
                else if (month==9)
                        System.out.println(" This " + year + " September has 30 days");
                else if (month==10)
                        System.out.println(" This " + year + " October has 31 days");
                else if (month==11)
                        System.out.println(" This " + year + " November has 30 days");
                else if (month==12)
                        System.out.println(" This " + year + " December has 31 days");
                else
                        System.out.println(" Input invalid month (1-12) ");
        }
}
```

១.២. Switch case statement

Switch Case Statement ដូចគ្នាទៅនឹង if statement ទម្រង់ទី ៣ ដែរ។

ទម្រង់ទូទៅរបស់ Switch case ៖

```
switch (expression) {
        case value1 :
                statement(s);
                break;
```

```
        case value2 :
                statement(s);
                break;
        case value3 :
                statement(s);
                break;
        ……
        ……
        case value n :
                statement(s);
                break;
        default :
                statement(s);
}
```

ចំពោះ expression ត្រូវមានប្រភេទទិន្នន័យជា byte , short, int ឬក៏ char។ ចំណែក value ទាំងអស់នៅក្នុង case នីមួយៗត្រូវមានប្រភេទ ដូចនឹងប្រភេទទិន្នន័យរបស់ expression ដែរ ។ Values ទាំងអស់ត្រូវតែជាចំនួនថេរ មិនមែនជាអញ្ញាតឡើយ ហើយតម្លៃ value មិនត្រូវដូចគ្នាឡើយនៅក្នុង case នីមួយៗ។

ចំពោះ break វិញ គឺគេប្រើដើម្បីឲ្យវាបញ្ចប់ case នីមួយៗ ។ ប្រសិនបើគ្មាន break នៅក្នុង case នីមួយៗទេ នោះ Switch case statement គ្មានប្រយោជន៍អ្វីឡើយ ព្រោះវាអនុវត្តរាល់ statements ទាំងអស់ដែលនៅក្នុង Switch ដោយវាមិនបានអនុវត្តទៅតាមលក្ខខណ្ឌណាមួយឡើយ។

ឧទាហរណ៍៖

```
class DemoIfSwitchCase{
        public static void main (String[] agr){
                boolean b; // to check the year mod 4 equal 0
                int year; // store year input
                int month; //store month input
                year=Integer.parseInt(agr[0]);
                month=Integer.parseInt(agr[1]);
                if ((year%4)==0)
                        b=true;
                        else
                                b=false;
                switch (month){
                        case 1:
                                System.out.println(" This " + year + " January has 31 days");
                                break;
                case 2:
                        if (b==true)
```

```
                        System.out.println(" This " + year + " February has 29 days");
                    else
                        System.out.println(" This " + year + " February has 28 days");
                    break;
                case 3:
                    System.out.println(" This " + year + " March has 31 days");
                    break;
                case 4:
                    System.out.println(" This " + year + " April has 30 days");
                    break;
                case 5:
                    System.out.println(" This " + year + " May has 31 days");
                    break;
                case 6:
                    System.out.println(" This " + year + " June has 30 days");
                    break;
                case  7:
                    System.out.println(" This " + year + " July has 31 days");
                    break;
                case  8:
                    System.out.println(" This " + year + " August has 31 days");
                    break;
                case 9:
                    System.out.println(" This " + year + " September has 30 days");
                    break;
                case 10:
                    System.out.println(" This " + year + " October has 31 days");
                    break;
                case 11:
                    System.out.println(" This " + year + " November has 30 days");
                    break;
                case 12:
                    System.out.println(" This " + year + " December has 31 days");
                    break;
                default:
                    System.out.println(" Input invalid month (1-12) ");
            }
        }
```

២. ប្រភេទ Iteration

នៅក្នុង Java Programming ប្រភេទ Iteration ជាប្រភេទ control structure មួយប្រភេទដែលគេប្រើសម្រាប់ត្រួតពិនិត្យទៅលើដំណើរការដដែលៗ ។ ដែលក្នុងនោះរួមមាន for loop, while loop និង do-while loop ។

២.១ ការប្រើប្រាស់ for loop

យើងប្រើ for loop ដើម្បីអនុវត្តការងារដដែលៗក្នុងចន្លោះណាមួយ ដោយមានការបត់បែនទៅតាមអ្វីដែលយើងចង់បាន ។ for loop វាដំណើរការ statements ដែលនៅក្រោម block របស់វានៅពេល

ដែលលក្ខខណ្ឌវា **ពិត**។

ទម្រង់ទូទៅរបស់ for loop ៖

```
for (initialization ; condition ; iteration) {
        statement(s);
}
```

- Initialization : គឺជាកន្លែងដែលយើងផ្តល់តម្លៃចាប់ផ្តើម
- Condition : គឺជាលក្ខខណ្ឌដើម្បីត្រួតពិនិត្យថា **ពិត** ឬ **មិនពិត** ដើម្បីអនុវត្ត statement ដែលនៅក្រោម block របស់ for ។
- Iteration : គឺជាកំណើន ឬ តំហាយដែលធ្វើឲ្យ for loop បញ្ចប់ គឺវាធ្វើឲ្យលក្ខខណ្ឌ**មិនពិត**។

ឧទាហរណ៍៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីចំនួន **គូ** នៅចន្លោះណាមួយ តាមរយៈការបញ្ចូលពី keyboard នៅពេលដែលយើង **run** កម្មវិធី។

```
class DemoForLoop{
        public static void main(String[] agr){
                int n, m, i ;
                n=Integer.parseInt(agr[0]);
                m=Integer.parseInt(agr[1]);
                System.out.println ( "The Even number Between " + n + " and " + m   );
                for ( i=n ; i<m ; i++){
                        if (( i % 2 ) ==0)
                                System.out.print ( i + "    " );
                }
        }
}
```

២.១.១ <u>ការប្រកាសអញ្ញាតនៅក្នុងfor</u>

ការប្រកាសអញ្ញាតនៅក្នុង for គឺនៅពេលដែលយើងត្រូវការប្រើប្រាស់តែនៅក្នុង for block ហើយអញ្ញាតដែលប្រកាសក្នុង for នេះនៅពេលដែលចេញក្រៅ for មិនអាចប្រើប្រាស់បានទេ ។ ការប្រកាសនេះ យើងអាចប្រកាសបានតែនៅក្នុងផ្នែក Initialization តែមួយគត់ ។

ឧទាហរណ៍៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីចំនួន **សេស** នៅចន្លោះណាមួយ តាមរយៈការបញ្ចូលពី keyboard នៅពេលដែលយើង **run** កម្មវិធី។

```
class DemoForLoop{
        public static void main(String[] agr){
                int n, m ;
                n=Integer.parseInt(agr[0]);
                m=Integer.parseInt(agr[1]);
                System.out.println ( "The Even number Between " + n + " and " + m   );
```

```
                for ( int i=n ; i<m ; i++){
                    if (( i % 2 ) !=0)
                        System.out.print ( i + " " );
                }
            }
        }
```

២.១.២ ការប្រើសញ្ញា (,) នៅក្នុង for loop

ការប្រើសញ្ញា(,) នៅក្នុង for គឺនៅពេលដែលយើងចង់ប្រើអញ្ញាតពីរ ឬ ច្រើននៅក្នុងករណីដដែលៗ ដែលជំហាននៃ loop ដូចៗគ្នា ។ ការប្រើសញ្ញា (,) បានតែនៅក្នុងផ្នែក Initialization និង Iteration តែប៉ុណ្ណោះ ។

ឧទាហរណ៍៖

```
        for (int i=0, n=1 ; i<10 ; i++, n+=2){
            System.out.println(" i = " + i);
            System.out.println(" n = " +n);
        }
```

២.១.៣ ការប្រើទម្រង់ផ្សេងៗរបស់for

ឧទាហរណ៍ខាងក្រោមនេះ យើងអាចសរសេរតាមរបៀបណាមួយក៏បាន។

1.
```
int i, n;
 for ( i=0 ;  i<n ; i++ ) {
      System.out.println(" i = " + i );
 }
```
2.
```
int i, n;
 for ( i=0 ;  i<n ;  ) {
      System.out.println(" i = " + i );
     i+=1;
 }
```
3.
```
int i, n;
 i=0;
 for (  ;  i<n ;  ) {
     System.out.println(" i = " + i );
     i+=1;
     }
```
4.
```
int i, n;
 i=0;
 for ( ;  ;  ) {
```

```
        if (i<n)
                System.out.println(" i = " + i );
        else
                break;
        i +=1;
}
```

២.២ ការប្រើប្រាស់ while loop

ការប្រើ while ក៏មិនមានអ្វីខុសពី for ប៉ុន្មានដែរ ។ while វាធ្វើការត្រួតពិនិត្យលក្ខខណ្ឌជាមុនសិនមុននឹងការអនុវត្ត statements ដែលនៅក្រោម block របស់វា ។ while ដំណើរការតែនៅក្នុងករណីលក្ខខណ្ឌ **ពិត** តែប៉ុណ្ណោះ។

ទម្រង់ទូទៅរបស់ while

```
while (condition) {
        statement (s); // have iteration break,++,--
}
```

ឧទាហរណ៍៖ យើងចង់បង្ហាញលេខពី ១00 មក 0 ដោយថយម្តង ៣ ឯកតា។

```
class DemoWhile{
        public static void main(String[] agr){
                int n=100;
                while (n>0){
                        System.out.print( n + "    ");
                        n -= 3 ;
                }
        }
}
```

ឧទាហរណ៍៖ កម្មវិធីខាងក្រោមបង្ហាញពីការរាប់ចំនួន ដងនៃការបញ្ចូលពី keyboard រហូតដល់ពេលដែលយើងបញ្ចូលអក្សរ (n ឬ N)។

```
import java.util.Scanner;
class DemoWhile{
        static Scanner sc=new Scanner(System.in);
        public static void main(String[] agr){
            int num=2;
            String input;
            while (true){
                System.out.println(num + "  ");
```

```
                System.out.print(" count Countinue?(y or n)");
                input = sc.next();
                if (input.equalsIgnoreCase("N")) break;
                num+=1;
                }
            System.out.println(" Closed ! ");
        }
    }
```

២.៣ ការប្រើប្រាស់ do-while loop

ដំណើរការរបស់ do-while ក៏មិនខុសគ្នាពី while ដែរ គឺវាខុសគ្នាត្រង់ do-while យ៉ាងហោចណាស់ក៏អនុវត្តបានម្តងយ៉ាងតិច ទើបត្រួតពិនិត្យលក្ខខណ្ឌជាក្រោយ។ វាផ្ទុយពី while ត្រួតពិនិត្យលក្ខខណ្ឌមុនទើបអនុវត្តជាក្រោយ ។

```
class DemoDoWhile{
    public static void main(String[] agr){
        int n=100;
        do {
            System.out.print( n + "    ");
            n -= 3 ;
        }while (n<0);
    }
}
```

៣. ប្រភេទ Jump

នៅក្នុង Java Programming មាន Jump statement ចំនួន ៣ គឺ break, continue និង return។

៣.១ ការប្រើប្រាស់ break statement

break statement នៅក្នុង java programming គេអាចប្រើប្រាស់បាន **៣** ទម្រង់ផ្សេងៗគ្នា គឺ ប្រើសម្រាប់ចាកចេញពី switch case ,សម្រាប់ចាកចេញពី loop និងប្រើប្រាស់ដូច goto ។

៣.១.១ ការប្រើប្រាស់ break សម្រាប់ចាកចេញពី switch

យើងប្រើ break ដើម្បីចាកចេញពី switch case statement ដូចមាននៅក្នុងចំនុច **១.២** ក្នុងមេរៀននេះ ដែលនិយាយពីការប្រើប្រាស់ switch case statement ។

៣.១.២ ការប្រើប្រាស់ break សម្រាប់ចាកចេញពី loop

ការប្រើប្រាស់ break វាអាចបង្ខំឲ្យ loop មួយចាកចេញភ្លាមៗពី loop នោះដោយរំលងចោលនូវរាល់ statement*(s)* ដែលនៅសល់ក្នុង block របស់វា ។

ខាងក្រោមនេះជា ឧទាហរណ៍ដែលបញ្ជាក់ពីការប្រើប្រាស់ break សម្រាប់ចាកចេញពី loop។

```
// using break to exit from for loop
class DemoBreakForLoop{
        public static void main(String[] agr){
                int n = 100,i=0 ;
                for (int i = 0 ; i < n ; i++){
                        if ( i == 9 ) break ; // exit from the FOR loop, if i equal 9
                        System.out.print ( i + " " );   i++;
                }
                System.out.println();
                System.out.println("This statement is out of FOR LOOP! ");
        }
}

// using break to exit from while loop
class DemoBreakWhileLoop{
        public static void main(String[] agr){
                int n = 100 ;
                while ( i <n ){
                        if ( i == 9 ) break ; // exit from the WHILE loop, if i equal 9
                                System.out.print ( i + " " );
                }
                System.out.println();
                System.out.println("This statement is out of WHILE LOOP! ");
        }
}
```

៣.១.៣ ការប្រើប្រាស់ break ដូចនឹង goto

នៅក្នុង java program យើងអាចប្រើ break ជាមួយ Label ដើម្បីឲ្យវាមានលក្ខណៈដូចនឹង goto ។ ទម្រង់ទូទៅមានដូចខាងក្រោម៖

```
break Label ;
```

ឧទាហរណ៍៖

```
class DemoBreakGoto{
        public static void main(String[] agr){
                First: {
                        System.out.println( " Start First! ");
                        Second: {
                                System.out.println( " Start Second! ");
```

```
                    Third : {
                        System.out.println( " Start Third! ");
                        break Second;
                        System.out.println( " End Third! "); //not execute
                    }
                    System.out.println( " End Second! "); //not execute
                }
                System.out.println( " End First! "); //Executed
            }
        }
    }
```

៣.២ <u>ការប្រើប្រាស់ continue statement</u>

ការប្រើប្រាស់ continue មានលក្ខណៈដូចទៅនឹង goto Label ដែរ។ ប៉ុន្តែវាមានលក្ខណៈខុសគ្នាបន្តិចត្រង់ continue វារំលង statement*(s)* ដែលនៅសេសសល់ក្នុង loop ណាមួយទៅចុងបញ្ចប់នៃ loop ហើយវាបង្កើនមួយឯកតានៃ iteration របស់វា។

ឧទាហរណ៍ទី១៖

```
class DemoContinue1{
    public static void main(String[] agr){
        for ( int i = 0 ; i <10 ; i++){
            System.out.print( i + "  ");
            if ((i %2)==0) continue;
            System.out.println( " ");
        }
    }
}
```

ឧទាហរណ៍ទី២៖

```
class DemoContinue2{
    public static void main(String[] agr){
        outer: for ( int i = 0 ; i <10 ; i++){
            for (int j =0 ; j < 10 ; j++){
                if (j > i) {
                    System.out.println( );
                    continue outer;
```

```
                    }
                    System.out.print( " " + (i));
                }
            }
            System.out.println( );
        }
    }
```

៣.៣ ការប្រើប្រាស់ return statement

ឃ្លា return វាត្រូវបានប្រើសម្រាប់ឲ្យតម្លៃទៅ method មួយ។ នៅពេលណាក៏ដោយនៅក្នុង method មួយ ឃ្លា return អាចប្រើសម្រាប់ធ្វើឲ្យការប្រតិបត្តិត្រឡប់ទៅរកកន្លែងដែលហៅវាមកប្រើវិញ។

ឧទាហរណ៍៖

```
class  DemoReturn{
    public static void main (String[] agr){
        boolean b = true ;
        System.out.println(" Here before call return ");
        if (t) return;
        System.out.println(" After call RETURN "); // not Execute
    }
}
```

End

មេរៀនទី ៤

ការប្រើប្រាស់ *ARRAY*

Array

១. <u>និយមន័យ</u>

Array គឺជាបណ្តុំនៃ អញ្ញាតដែលមានប្រភេទទិន្នន័យដូចគ្នា ស្ថិតនៅក្រោមឈ្មោះតែមួយ តែវាញែកធាតុនីមួយៗដាច់ពីគ្នាដោយសារ Index ។

២. <u>Array មួយវិមាត្រ</u>

គឺជា array ដែលមានតែមួយជួរឈរ ពោលគឺមាន column តែមួយ មាន rows ច្រើន។

ទម្រង់ទូទៅនៃការប្រកាស array មួយវិមាត្រ ៖

```
Data-type  Array-name[ ] ;
```

ឬ

```
data-type [ ]  array-name ;
```

- data-type បញ្ជាក់ពីប្រភេទទិន្នន័យរបស់ array

ឧទាហរណ៍៖ យោងប្រកាស array មួយឈ្មោះថា day_in_month ដែលប្រើសម្រាប់រក្សាចំនួនថ្ងៃក្នុងខែនីមួយៗ យើងសរសេរដូចខាងក្រោម៖

```
int[ ] day_in_month ;
```

ការប្រកាសបែបនេះ វាមិនមានន័យថា យើងបានបង្កើត arrayមួយរួចហើយនោះទេ។ ដូច្នេះដើម្បីឲ្យវាបង្កើតជា array មួយយ៉ាងពិតប្រាកដ យើងត្រូវសរសេរដូចខាងក្រោម៖

```
int[ ] day_in_month ;
day_in_month = new int[12] ;
```

ឬ

```
int[ ] day_in_month = new int[12] ;
```

ឧទាហរណ៍៖

```
class DemoArray{
    public static void main(String[] agr){
        int day_in_month[ ] =new int[12];
        day_in_month[0]=31;
        day_in_month[1]=28 ;
        day_in_month[2]=31;
        day_in_month[3]=30;
        day_in_month[4]=31;
        day_in_month[5]=30;
        day_in_month[6]=31;
        day_in_month[7]=31;
        day_in_month[8]=30;
```

```
            day_in_month[9]=31;
            day_in_month[10]=30;
            day_in_month[11]=31;
            for (int i =0 ; i<12 ; i++)
                System.out.println ( " Month " + (i+1) + " has " + day_in_month[i] + "days ");
        }
    }
```

២.១ ការកំណត់តម្លៃដំបូងទៅឲ្យArray

ទម្រង់នៃការកំណត់តម្លៃដំបូងទៅឲ្យ array មួយវិមាត្រ៖

```
data-type array-name [ ] = {value1, value2, value3,……,valueN};
```

តាមរយៈឧទាហរណ៍ខាងលើយើងអាចសរសេរ៖

```
class DemoArray{
    public static void main(String[] agr){
        int day_in_month[ ] ={31,28, 31,30,31,30,31,31,30,31,30,31};
        for (int i =0 ; i<12 ; i++)
            System.out.println ( " Month " + (i+1) + " has " + day_in_month[i] + "days ");
    }
}
```

៣. Array ពីរវិមាត្រ

នៅក្នុង java programming យើងអាចប្រកាស array ពីរវិមាត្រដូចខាងក្រោម៖

```
data-type array-name[ ][ ] = new data-type[row][column] ;
```

ឧទាហរណ៍៖

```
class DemoArrayTwoD{
    public static void main (String [] agr){
        int myTwoArray[ ][ ] =new int [4][5];
        int i , j , k =1;
        for (i=0 ; i<4 ; i++)
            for (j=0 ; j<5 ; j++){
                myTwoArray[i][j] = (i + j) *k;
                k++;
            }
        for (i=0 ; i<4 ; i++)
            for (j=0 ; j<5 ; j++)
                System.ou.println ("myTwoArray[" +i+ "][" +j+ "] "+ myTwoArray[i][j]);
    }
}
```

ឧទាហរណ៍៖ បង្ហាញពីការបង្កើត array ពីរវិមាត្រ ដែលចំនួន column មិនស្មើគ្នា

```
class DemoArrayTwoD2{
        public static void main (String [] agr){
                int myTwoArray[ ][ ] =new int [4][ ];
                myTwoArray[0] =new int [1];
                myTwoArray[1] =new int [2];
                myTwoArray[2] =new int [3];
                myTwoArray[3] =new int [4];
                int i , j , k =0;
                for (i=0 ; i<4 ; i++)
                        for (j=0 ; j<i+1 ; j++){
                                myTwoArray[i][j] = k ;
                                k++;
                        }
                for (i=0 ; i<4 ; i++){
                     for (j=0 ; j<i+1 ; j++)
                          System.ou.print( myTwoArray[i][j] + " ");
                     System.out.println();
                }
        }
}
```

End

មេរៀនទី ៥

ការប្រើប្រាស់ Object នៅក្នុង Java

Object in Java

១. Classes

Classes គឺជា Template មួយសម្រាប់កំណត់ទម្រង់របស់ Object ។ នៅក្នុង classes អាចផ្ទុកទៅដោយ Methods (functions) , Variable , Initialization code ។ Classes មាន Members ពីរគឺ Methods និង Data (variable) ។ Data ឬ Variable នៅក្នុង class មួយគេហៅថា Instance variables ។ Methods និង Variables នៃ Class មួយត្រូវស្ថិតនៅក្នុងសញ្ញា {...} ។ យើងប្រកាស class មួយដោយប្រើ keyword ***class***។

ទម្រង់ទូទៅរបស់ Class ៖

```
class class_name {
        type instance_variable1 ;
        type instance_variable2 ;
        ……..…
        type instance_variablen ;
        type method_name1 (parameter-list){
                //body of method
        }
        type method_name2 (parameter-list){
                //body of method
        }
        ……..…
        type method_nameN (parameter-list){
                //body of method
        }
}
```

ឧទាហរណ៍៖ យើងបង្កើត class មួយឈ្មោះ Vehicle ដែលមាន Variables ចំនួន ៣ គឺ passengers, fuelcap និង mpg ដែលមានប្រភេទទិន្នន័យជា integer ។

```
class Vehicle {
        int passengers ; // number of passengers
        int fuelcap ;  // fuel capacity in gallons
        int mpg ;  //fuel consumption in mile per gallon
    }
```

ដើម្បីបង្កើត Object មួយចេញពី class Vehicleយើងត្រូវសរសេរដូចខាងក្រោម៖

```
Vehicle minibus;
minibus = new Vehicle();
```

ចំពោះ statement (Vehicle minibus) មិនមានន័យថាយើងបង្កើត Object Vehicle នោះទេ។ ផ្ទុយទៅវិញវាបង្កើត Variable មួយដែល refer ទៅ Object ប្រភេទជា Vehicle ។ ដើម្បីបង្កើត Object Vehicle យើងត្រូវប្រើ Keyword ***new*** ។ បន្ទាប់ពីយើងប្រើ Keyword ***new*** ទើបយើងអាចប្រើប្រាស់ Data និង Method របស់វាបាន។

ដើម្បីប្រើ data និង methods របស់ class មួយ គឺយើងប្រើប្រាស់តាមរយៈឈ្មោះរបស់ variable ដែលយើងប្រកាសចេញពី class នោះដោយភ្ជាប់ជាមួយសញ្ញា(.) និងឈ្មោះ data ឬ method ។

ឧទាហរណ៍៖ (Save ដាក់ឈ្មោះ DemoClass.java)

```
class Vehicle {
        int passengers ; // number of passengers
        int fuelcap ;  // fuel capacity in gallons
        int mpg ;  //fuel consumption in mile per gallon
    }

class DemoClass{
        public static void main(String[] agr){
                Vehicle minibus = new Vehicle();
                Vehicle minicar = new Vehicle();
                Int range1, range2;
                minibus.passengers = 12;
                minibus.fuelcap = 60 ;
                minibus.mpg = 10 ;
                minicar.passengers = 5;
                minicar.fuelcap = 45 ;
                minicar.mpg = 13 ;
                range1 = minibus.fuelcap * minibus.mpg ;
                range2 = minicar.fuelcap.minicar.mpg ;
                System.out.println("Minibus can carry " + minibus.passengers + " with the
                                range of " + range1 );
                System.out.println("Minicar can carry " + minicar.passengers + " with the
                                range of " + range2 );
        }
}
```

១.១ <u>អញ្ញាត Reference និងការកំណត់តម្លៃ</u>(Reference variable and Assignment)

យើងអាចកំណត់តម្លៃរបស់អញ្ញាតមួយដែលទទួលលក្ខណៈ object ទៅឲ្យអញ្ញាតមួយផ្សេងទៀត លក្ខណៈបែបនេះមានន័យថាអញ្ញាតដែលទទួល object នោះមានលក្ខណៈជាអញ្ញាត Reference ។

ឧទាហរណ៍៖

```
Vehicle car1=new Vehicle();
Vehicle car2=car1;
```

បើតាមយើងមើលទៅវាហាក់ដូចជា car1 និង car2 វាបញ្ជាក់ឲ្យ object ពីរផ្សេងគ្នា។ តែតាមពិតវាបញ្ជាក់ឲ្យ object តែមួយ។ ដូច្នេះការកំណត់តម្លៃទៅឲ្យ car1 ឬ car2គឺដូចគ្នា។

ឧទាហរណ៍៖

```
car1.passengers= 10;
```

បន្ទាប់មកយើងសរសេរដូចខាងក្រោម៖

```
System.out.println(car1.passengers);
System.out.println(car2.passengers);
```

ជាលទ្ធផលគឺវាបោះតម្លៃដូចគ្នាគឺ ១០ ។

២. <u>Method</u>

Method គឺជាធាតុមួយយ៉ាងសំខាន់នៅក្នុង java ព្រោះវាផ្តល់ឲ្យនូវអានុភាព និងភាពអាចផ្លាស់ប្តូរបានយ៉ាងច្រើន។ method មួយអាចមានឃ្លាមួយ ឬច្រើន។ method មួយត្រូវធ្វើការងារតែមួយមុខគត់ ហើយឈ្មោះរបស់វាប្រើសម្រាប់ហៅ method នោះមកប្រើវិញ។

ទម្រង់ទូទៅរបស់ Method ៖

```
return-type method-name(parameters-list){
    // body of methods
}
```

Return-type ៖ គឺវាជាបញ្ជាក់ប្រាប់ពីប្រភេទទិន្នន័យដែលទទួលបានពី method នោះ។

method-name ៖គឺជាឈ្មោះរបស់ method ដែលយើងប្រកាសើដើម្បីហៅវាមកប្រើប្រាស់។ ឈ្មោះរបស់វាគឺដាក់យ៉ាងណាឲ្យវាសមស្របទៅនឹងអ្វីដែលយើងចង់ធ្វើ ។

parameters-list ៖ គឺជាសៀរីនៃប្រភេទទិន្នន័យ និង Identifier ដែលវាញែកដាច់ពីគ្នាដោយសញ្ញា (,)។

ឧទាហរណ៍៖ យើងបន្ថែម ១ ទៅលើ class Vehicle គឺ Method ***range*** ប៉ុន្តែវាមាន return type ប្រភេទជា void ។

```
class Vehicle {
    int passengers ; // number of passengers
    int fuelcap ;  // fuel capacity in gallons
    int mpg ;  //fuel consumption in mile per gallon
```

```
        void range(){
            System.out.println("Range is " + fuelcap * mpg );
        }
    }
class DemoMethod{
        public static void main(String[] agr){
            Vehicle minibus = new Vehicle();
            Vehicle minicar = new Vehicle();
            Int range1, range2;
            minibus.passengers = 12;
            minibus.fuelcap = 60 ;
            minibus.mpg = 10 ;
            minicar.passengers = 5;
            minicar.fuelcap = 45 ;
            minicar.mpg = 13 ;
            System.out.println("Minibus can carry " + minibus.passengers );
            minibus.range();
            System.out.println("Minicar can carry " + minicar.passengers );
            minicar.range();
        }
}
```

នៅក្នុង method មួយយើងអាចផ្តល់តម្លៃឲ្យវា (method) បានតាមរយៈការប្រើប្រាស់ keyword ***return*** ។ ការប្រើប្រាស់ return មានពីររបៀបគឺ អាច return ដោយមាន value និង return ដោយគ្មាន value។

ចំពោះ method ដែលមាន return type ជា void យើងអាចបញ្ចប់ដំណើរការរបស់ method នោះភ្លាមៗដោយប្រើ keyword return ដោយគ្មាន value គឺ (return ;)។

ឧទាហរណ៍៖

```
void testVoidMethod(){
        for (int i=10 ; i>0 ; i--){
            if (i==3) return;
            System.out.print( i + " ");
        }
}
```

ចំពោះ method ទាំងឡាយណាដែលមាន return type ខុសពី void គឺយើងប្រើ return ដោយមាន value។ ឧទាហរណ៍យើងធ្វើការផ្លាស់ប្តូរ return type របស់ range ពី void ទៅជាប្រភេទ int វិញ។ code ខាងក្រោមបង្ហាញពីការប្រើប្រាស់របស់វា ។

```
class Vehicle {
        int passengers ; // number of passengers
        int fuelcap ;  // fuel capacity in gallons
        int mpg ;  //fuel consumption in mile per gallon
        int range(){
                return ( fuelcap * mpg );
        }
   }
class DemoReturnMethod{
        public static void main(String[] agr){
                Vehicle minibus = new Vehicle();
                Vehicle minicar = new Vehicle();
                Int range1, range2;
                minibus.passengers = 12;
                minibus.fuelcap = 60 ;
                minibus.mpg = 10 ;
                minicar.passengers = 5;
                minicar.fuelcap = 45 ;
                minicar.mpg = 13 ;
                System.out.println("Minibus can carry " + minibus.passengers + " with the
                                range of " + minibus.range() );
                System.out.println("Minicar can carry " + minicar.passengers + " with the
                                range of " + minicar.range() );
        }
}
```

២.១ ការប្រើ method ដែលមាន parameter

គេអាចបញ្ជូនតម្លៃមួយ ឬក៏ច្រើនទៅកាន់ method មួយនៅពេលដែលគេហៅវាមកប្រើប្រាស់ តម្លៃដែលយើងបញ្ជូនទៅនោះគេហៅថា argument ហើយអញ្ញាតដែលទទួលតម្លៃពី argument គេហៅថា parameter ។ parameter ត្រូវបានគេប្រកាសនៅពេលគេបង្កើត method ដោយដាក់នៅក្នុងសញ្ញា(...) បន្ទាប់ពីឈ្មោះរបស់ method ។

```
class MethodWithParamet{
        int addTwoNumber(int a , int b){
                return (a+b);
        }
        int minusTwoNumber(int a , int b){
                return (a-b);
        }
        int multTwoNumber(int a , int b){
                return (a*b);
        }
        int divTwoNumber(int a , int b){
                return (a/b);
        }
}
class DemoMethodParameter{
        public static void main (String[] agr){
        MethodWithParamet obj=new MethodWithParamet();
        int a , b ;
        a=Integer.parseInt(agr[0]);
        b=Integer.parseInt(agr[1]);
        System.out.println( a + " + " + b +" = " +obj.addTwoNumber(a,b));
        System.out.println( a + " - " + b +" = " +obj.minusTwoNumber(a,b));
        System.out.println( a + " * " + b +" = " +obj.multTwoNumber(a,b));
        System.out.println( a + " / " + b +" = " +obj.divTwoNumber(a,b));
        }
}
```

២.២ ការប្រើ *this* keyword

ការប្រើ keyword ***this*** វាមានសារៈសំខាន់ណាស់នៅក្នុងភាសា java ។ នៅពេលដែលនៅក្នុង class មួយដែលមាន variable ឈ្មោះដូចគ្នាច្រើន ប៉ុន្តែវាខុសគ្នាដោយសារ level (scope) ។

ឧទាហរណ៍៖

```
class Bird {
        int xPos, yPos;
                double fly ( int xPos, int yPos ) {
                        double distance = Math.sqrt( xPos*xPos + yPos*yPos );
```

```
                flap( distance );
                this.xPos = xPos;
                this.yPos = yPos;
                return distance;
            }
        }
```

នៅក្នុងឧទាហរណ៍នេះ ត្រង់ឃ្លា this.xPos និង this.yPos គឺវាសំដៅទៅលើ variable xPos និង yPos ដែលយើងប្រកាសនៅក្នុង class Bird មិនមែនជា parameter-list ដែលយើងប្រកាសនៅក្នុង method fly នោះទេ។ លក្ខណៈនេះហើយដែលយើងចាំបាច់ត្រូវប្រើ keyword ***this*** ដើម្បីញែកឲ្យដាច់ពីគ្នា។

២.៣ ការប្រើប្រាស់ Static Member

ជាធម្មតា member របស់ class មួយ ឬ របស់ object មួយ យើងអាចប្រើប្រាស់វាបាន (access) តាមរយៈឈ្មោះ object ដែលយើងប្រកាសវា ។ ប៉ុន្តែយើងក៏អាចប្រើប្រាស់វា (members) ដោយផ្ទាល់ដោយពុំចាំបាច់ឆ្លងកាត់ object ឡើយ គឺយើងប្រើប្រាស់ Static Member ។

ខាងក្រោមជាឧទាហរណ៍បង្ហាញពីការប្រើប្រាស់ static variable ដែលវាមាន shared៖

```
class VariableStatic{
    int x;
    static int y;
}
class DemoStatic{
    public static void main(String[] agr){
        VariableStatic obj1 = new VariableStatic();
        VariableStatic obj2 = new VariableStatic();
        obj1.x = 10 ;
        obj1.y = 12 ;
        obj2.x = 20 ;
        System.out.println(" X in obj1 and obj2 are different ");
        System.out.println(" obj1.x = " + obj1.x  + " obj2.x = " + obj2.x );
        System.out.println(" Y in obj1 and obj2 are the same ,because shared ");
        System.out.println(" obj1.y = " + obj1.y  + " obj2.y = " + obj2.y );
    }
}
```

ខាងក្រោមជាឧទាហរណ៍បង្ហាញពីការប្រើប្រាស់ static variable និង static method ៖

```
class VarMethStatic{
        static int x;
        static int y;
        static int add(){
                return x+y;
        }
}
class DemoStatic{
        public static void main(String[] agr){
                VarMethStatic.x = 10 ;
                VarMethStatic.y = 12 ;
                System.out.println(VarMethStatic.x + " + " + VarMethStatic.y + " = " +
                                VarMethStatic.add() ) ;
        }
}
```

<u>កំណត់ចំណាំ៖</u> ប្រការគួចងចាំចំពោះការប្រើប្រាស់ method មានលក្ខណៈ static

-វាអាចហៅប្រើបានតែ method ដែលមានលក្ខណៈ static ប៉ុណ្ណោះ

-វាអាចប្រើប្រាស់តែ data member(variable) ដែលមានលក្ខណៈ static ប៉ុណ្ណោះ

-វាមិនត្រូវមានការប្រើជាមួយ keyword ***this*** ឡើយ

ឧទាហរណ៍ class ខាងក្រោមនេះវាបង្ហាញពីការប្រើប្រាស់មិនត្រឹមត្រូវ

```
class ErrorStatic{
        int x ; // គឺជា instance variable ធម្មតា
        static int y ;  //ជា static variable
        static int add(){
                return x+y ;  //error won't compile
                // x មិនមែនជា static variable
        }
}
```

២.៤ <u>ការបញ្ជូនតម្លៃ Argument តាមរយៈ Value និង References</u>

នៅក្នុង java ក៏ដូចជានៅក្នុង programming ផ្សេងៗដែរការ passed argument មានពីរគឺ ដោយតម្លៃ និងដោយ reference ។ យើងធ្លាប់បានឃើញរួចមកហើយការ passes ដោយតម្លៃនៅក្នុង methods គឺ parameter-list របស់វាមានប្រភេទទិន្នន័យជា int , char, float...ជាដើម វាគឺជាប្រភេទ simple data

type។ ផ្ទុយទៅវិញការ passed ដោយ references គឺ parameter-list របស់វាមានប្រភេទទិន្នន័យជាប្រភេទ object ណាមួយ ជា array និងជា ប្រភេទទិន្នន័យ String ជាដើម។

ខាងក្រោមជាផ្នែកមួយនៃ code

```
.....
   int i = 0;
   SomeKindOfObject obj = new SomeKindOfObject( );
   myMethod( i, obj );
   ...
void myMethod(int j, SomeKindOfObject o) {
   ...
}
```

នៅពេលដែលយើងហៅ myMethod គឺវាមាន argument ពីរ ចំពោះ argument ទី១ គឺវា passed ដោយ value។នៅក្នុង code ខាងលើ i passed ដោយតម្លៃ ទោះបីជាយើងធ្វើការផ្លាស់ប្តូរតម្លៃ j នៅក្នុង myMethod យ៉ាងណាក៏ដោយ ក៏វាមិនមានការប្រែប្រួលតម្លៃ i ដែរ ។ ផ្ទុយទៅវិញចំពោះ obj ដែលផ្តល់ទៅឲ្យ parameter **o** នៅពេលដែលយើងធ្វើការផ្លាស់ប្តូរតម្លៃរបស់ o នោះវាធ្វើឲ្យមានការផ្លាស់ប្តូរតម្លៃរបស់ obj ផងដែរ។

២.៥ Methods Overloading

Methods Overloading គឺវាមានសមត្ថភាពអាចឲ្យយើងបង្កើត methods ដែលមានឈ្មោះដូចៗគ្នាជាច្រើននៅក្នុង Class តែមួយ។ ប៉ុន្តែវាមានការប្រើប្រាស់ខុសគ្នាទៅតាមអ្វីដែលយើងចង់បាន។

ច្បាប់នៃការ Overloading Methods ៖

- **ត្រូវតែមានឈ្មោះដូចគ្នា**
- **ខុសគ្នាដោយចំនួន parameter-list**
- **ខុសគ្នាដោយ parameter-type**
- **ប៉ុន្តែមិនមែនខុសគ្នាដោយសារreturn-type របស់ Methodsនោះទេ**

នៅក្នុង java ពេលវាហៅ method overloading មកប្រើវាប្រតិបត្តទៅលើ method ណាដែលមានប៉ារ៉ាម៉ែត្រត្រូវគ្នានឹង argument ដែលយើងហៅមកប្រើប្រាស់ (ទាំង data-type និង ចំនួន argument)។

ឧទាហរណ៍៖

```
class OverLoad{
int sum( int x , int y){
        return x+y ;
}
int sum( int x , int y, int z){
        return x+y+z ;
}
```

```
float sum( int x , float y){
        return x+y ;
}
float sum( float x , float y){
        return x+y ;
}
double sum( double x , float y){
        return x+y ;
}
……..…
}
class DemoOverload{
        public static void main(String[] agr){
        OverLoad obj = new OverLoad();
        System.out.println( obj.sum(10 , 20 ));
        System.out.println( obj.sum(10.0 , 20));
        System.out.println( obj.sum(10 , 20.5));
        System.out.println( obj.sum(0.5 , 20.5));
        }
}
```

២.៦ អំពី Recursive Method

គឺជា method មួយប្រភេទដែលនៅក្នុង body របស់វាអាចហៅឈ្មោះរបស់ method នោះមកប្រើវិញបាន ។ នៅក្នុង body របស់ method recursive គឺត្រូវមានលក្ខណៈគ្រឹះ និងលក្ខណៈទូទៅ ។ លក្ខណៈគ្រឹះ គឺជាចំនុចមួយ ដែលជាចំនុចបញ្ចប់នៃ recursive ដូច្នេះយើងត្រូវប្រើ if ដើម្បីត្រួតពិនិត្យ និងបញ្ចប់ recursive ។ ចំពោះលក្ខណៈទូទៅ គឺជាចំនុចមួយដែលវាហៅខ្លួនឯងមកប្រើប្រាស់វិញ។

ឧទាហរណ៍៖ យើងបង្កើត method មួយដែលមានលក្ខណៈ recursive

```
class Recursive {
        int factorial (int n){
                if (n==1) return 1;
                return factorial( n-1) * n;
        }
}
class DemoRecursive{
        public static void main (String [] agr){
                Recursive obj = new Recursive () ;
                System.out.println( " The Factorial of 6 is " + obj.factorial(6));
                System.out.println( " The Factorial of 3 is " + obj.factorial(3));
                System.out.println( " The Factorial of 5 is " + obj.factorial(5));
                System.out.println( " The Factorial of 7 is " + obj.factorial(7));
        }
}
```

៣. ការប្រើប្រាស់ Constructors

Constructor គឺជាបណ្តុំនៃ code ដែលគេប្រើសម្រាប់កំណត់តម្លៃតំបូងទៅឲ្យ object នៅពេលដែលយើងបង្កើតវាឡើង។ Constructor វាត្រូវតែមានឈ្មោះដូចទៅនឹងឈ្មោះរបស់ class ហើយវាស្រដៀងទៅនឹង method ដែរ ប៉ុន្តែវាពុំមានបញ្ជាក់ពីប្រភេទទិន្នន័យឡើយ។ ជាធម្មតាយើងប្រើ constructor ដើម្បីកំណត់តម្លៃដំបូងទៅឲ្យអញ្ញាតដែលកំណត់ដោយ class ឬធ្វើការណាមួយដែលតម្រូវឲ្យបង្កើត object មួយមានលក្ខណៈពេញលេញ។

នៅគ្រប់ classes ទាំងអស់តែងតែមាន constructor មួយ ទោះបីជាយើងបង្កើតវា ឬមិនបានបង្កើតវាក៏ដោយ ព្រោះនៅក្នុង Java វាបានបង្កើតនូវ constructor មួយដែលមានលក្ខណៈ default ដោយស្វ័យប្រវត្តិ ហើយវាបានកំណត់តម្លៃ សូន្យ ទៅឲ្យអញ្ញាតទាំងអស់នៅក្នុង class នោះ ។ ប៉ុន្តែនៅពេលដែលយើងបង្កើត constructor ដោយខ្លួនឯងហើយ ពេលនោះ constructor ដែលមានលក្ខណៈ default មិនអាចប្រើប្រាស់បានទៀតទេ។

ឧទាហរណ៍៖ បង្ហាញពីការប្រើប្រាស់ constructor នៅក្នុង class មួយ

```
class DemoConstructor{
        int x ;
        int  y ;
        DemoConstructor(){
                x=10;
                y=1 ;
        }
        int multiple (){
                return x* y ;
        }
}
class TextConstructor{
        public  static void main (String[] agr){
                DemoConstructor obj = new DemoConstructor();
                System.out.println(" Before assign X and Y value ");
                System.out.println(" X= " + obj.x);
                System.out.println(" Y= " + obj.y);
                System.out.println(" Multiple X and Y = " + obj.multiple() );
                obj.x=12;
                obj.y=4;
                System.out.println("After assign X and Y value ");
                System.out.println(" X= " + obj.x);
                System.out.println(" Y= " + obj.y);
                System.out.println(" Multiple X and Y = " + obj.multiple() );
        }
}
```

ឧទាហរណ៍ខាងលើ យើងឃើញថា constructor របស់ class DemoConstructor មិនមាន parameter ទេ ។ ប៉ុន្តែវាអាចមាន parameters មួយ ឬច្រើន ដូចគ្នាទៅនឹង methods ដែរ ។ ខាងក្រោមនេះបង្ហាញពីការប្រើ constructor ដែលមាន parameter។

```
class DemoConstructor{
        int x ;
        int  y ;
        DemoConstructor(int x , int  y){
                this.x=x;
                this.y=y ;
        }
        int multiple (){
                return x* y ;
        }
}
class TextConstructor{
        public  static void main (String[] agr){
                DemoConstructor obj1 = new DemoConstructor(1,0);
                DemoConstructor obj2 = new DemoConstructor(8,9);
                System.out.println(" Obj1: X and Y value ");
                System.out.println(" X= " + obj1.x);
                System.out.println(" Y= " + obj1.y);
                System.out.println(" Multiple X and Y = " + obj1.multiple() );
                obj2.x=12;
                obj2.y=4;
                System.out.println("Obj2 : X and Y value ");
                System.out.println(" X= " + obj2.x);
                System.out.println(" Y= " + obj2.y);
                System.out.println(" Multiple X and Y = " + obj2.multiple() );
        }
}
```

ឧទាហរណ៍ ខាងក្រោមបង្ហាញពីការបន្ថែម constructor មួយទៅលើ class Vehicle

```
class Vehicle {
        int passengers ; // number of passengers
        int fuelcap ;  // fuel capacity in gallons
        int mpg ;  //fuel consumption in mile per gallon
        Vehicle (int p , int f , int  m){
                passengers = p ;
                fuelcap = f ;
```

```
                mpg = m ;
        }
        int range(){
                return fuelcap * mpg ;
        }
        double fuelNeed( int miles) {
                return (double) miles / mpg;
        }
    }
class DemoClass{
        public static void main(String[] agr){
                Vehicle minibus = new Vehicle(12 , 10 , 23);
                Vehicle minicar = new Vehicle(2,14,12);
                double gallon ;
                int dist = 300 ;
                gallon =minibus.fuelNeed(dist);
                System.out.println("Minibus can carry " + minibus.passengers + " with the
                                range of " + range1 );
                System.out.println( "Minibus go to " + dist + " miles need " + gallon +
                                " gallon s of fuel);
                gallon = minicar.fuelNeed(dist);
                System.out.println("Minicar can carry " + minicar.passengers + " with the
                                range of " + range2 );
                System.out.println( "Minicar go to " + dist + " miles need " + gallon +
                                " gallon s of fuel);
        }
}
```

៥. <u>អំពី Overloading Constructors</u>

នៅក្នុង class មួយមិនមែនមានតែ constructor តែមួយទេ ។ នៅក្នុង class មួយអាចមាន constructors ជាច្រើនវាតម្រូវទៅតាមការប្រើប្រាស់របស់យើង កាលណាក្នុង class មួយមាន constructor ច្រើន គេហៅថា Overloaded Constructors ។

ឧទាហរណ៍ ៖

```
class OverConstruct {
        int x ;
        OverConstruct (){
                x=0;
        }
        OverConstruct (int i){
```

```
                x=i ;
        }
        OverConstruct (double d){
                x=(int) d;
        }
        OverConstruct (int i , double d){
                x=i + (int) d;
        }
        void printX(){
                System.out.println( " The value of X = " + x);
        }
}
class DemoOverConst{
        public static void main(String[] agr){
                OverConstruct obj1 = new OverConstuct();
                OverConstruct obj2 = new OverConstuct(12);
                OverConstruct obj3 = new OverConstuct(15.67);
                OverConstruct obj4 = new OverConstuct(12 , 23.8);
                System.out.println( " Here the values of each x value in each objects ");
                Obj1.printX();
                Obj2.printX();
                Obj3.printX();
                Obj4.printX();
        }
}
```

- ផលប្រយោជន៍នៃការប្រើប្រាស់ Overloading Constructors គឺអាចផ្តល់តម្លៃតំបូងទៅឲ្យ object មួយតាមរយៈ Object មួយដែលមានតម្លៃស្រាប់ គឺ pass object ទៅឲ្យ object ។

```
class Summation{
        int sum ;
        Summation (int num){
                Sum = 0;
                for( int i =1 ; i <=num ; ++i ){
                sum+=i ;
                }
        }
        Summation (Summation obj){
        sum=obj.sum;
        }
}
```

```
class Demo{
    public static void main(String[] agr){
        Summation obj1 = new Summation (12);
        Summation obj2 = new Summation (obj1);
        System.out.println( " Obj1.sum = " + obj1.sum);
        System.out.println( " Obj2.sum = " + obj2.sum);
    }
}
```

៥. <u>អំពី Inner Class</u>

នៅក្នុង Java version 1.0 ពុំមានលក្ខណៈ class នៅក្នុង class ឡើយ ។ វាចាប់ផ្តើមមាននៅក្នុង Java version 1.1 ។ ទំហំរបស់ class មួយនៅក្នុង class មួយទៀតត្រូវបានកំណត់ដោយ class ខាងក្រៅរបស់វា ។ Inner class នោះអាចចូលប្រើ Members ទាំងអស់របស់ outer class រួមទាំង member ដែលមានលក្ខណៈ private ផងដែរ។ ប៉ុន្តែ outer class ពុំមានសិទ្ធិប្រើប្រាស់ member របស់ Inner class ណាមួយបានឡើយ ។

Inner class មានពីរប្រភេទគឺ static និង non-static ។ ប៉ុន្តែ Inner class ដែនមានលក្ខណៈ static ពុំសូវប្រើទេ ដោយសារវាមានភាពតឹងរឹងក្នុងការចូលប្រើ member នៃ class នោះ។ ចំណែកឯ Inner class ដែលមានលក្ខណៈ non-static វាអាចប្រើដោយផ្ទាល់នូវអញ្ញាត និង members របស់ outer class ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើ inner class ដើម្បីគណនាតម្លៃផ្សេងៗរបស់ outer class

```
class Outer{
    int nums[ ];
    Outer( int n[ ]){
        nums = n ;
    }
    void Analyze(){
        Inner inObject = new Inner();
        System.out.println( "Minimum : " + inObject.min());
        System.out.println( "Maximum : " + inObject.max());
        System.out.println( "Average : " + inObject.avg());
    }
    class Inner{
        int min(){
            int m=nums[0];
            for (int i =1 ; i <nums.length ; i++)
                if (nums[i] <m )
                m = nums[i];
            return m;
        }
```

```
                int max(){
                        int m=nums[0];
                        for (int i =1 ; i <nums.length ; i++)
                                if (nums[i] >m )
                                m = nums[i];
                        return m;
                }
                int avg(){
                        int s=0;
                        for (int i =1 ; i <nums.length ; i++)
                                s+=nums[i];
                        return s / nums.length;
                }
        }
}
class DemoInnerClas{
        public static void main(Srting[] agr){
                int myArray [ ] ={1,3,4,5,6,7,9,65,32};
                Outer outObject = new Outer(myArray);
                outObject.Analyze();
        }
}
```

<u>កំណត់ចំណាំ</u> ៖ គេអាចកំណត់ Inner class នៅក្នុង block ណាមួយក៏បាន ។ ឧទាហរណ៍ យើងអាចកំណត់ Inner class នៅក្នុង method មួយ ឬនៅក្នុង loop ។

ខាងក្រោមជាឧទាហរណ៍ដែលបង្ហាញពីការបង្កើត Inner class នៅក្នុង for loop ៖

```
class Outer {
        int outer_x = 100;
        void test(){
                for (int i =0 ; i<10 ;i++){
                        class Inner{
                                void display(){
                                        System.out.println( "Display outer_x = " + outer_x );
                                }
                        }
                        Inner inObject = new Inner();
                        inObject.display();
                }
        }
}
```

```
class DemoInnerClass{
        public static void main (String[] agr){
                Outer outObject = new Outer();
                outObject.test();
        }
}
```

<u>កំណត់ចំណាំ</u>៖ កាលណានៅ outer class ពុំបានបង្កើត object មួយចេញពី inner class ទេ នោះគេអាចប្រើ member របស់ inner class តាមកំរិតដូចបង្ហាញខាងក្រោម៖

```
 class A{
        int x, y ;
        int sum(){
                return (x+y);
        }
        class B{
                int z ;
                int mySum(){
                        return( x+z+y);
                }
        }
}
class DemoAB{
        public static void main(String[] agr){
                A obj1= new A();
                A.B obj2 = obj1.new B();
                obj1.x = 4;
                obj1.y = 8;
                obj2.z = 9;
                System.out.println("Sum = " + obj2.mySum());
        }
}
```

៦. <u>កម្រិតនៃការចូលប្រើប្រាស់member របស់ class</u>

នៅក្នុង java កម្រិតនៃការចូលទៅប្រើប្រាស់ members របស់ class មួយមានបីកម្រិតទៅតាម private protected និង public។ ត្រង់នេះហើយដែនបញ្ជាក់ពីលក្ខណៈ encapsulation ដែលវាមានសារៈសំខាន់ពីរយ៉ាងគឺ ទី១ វាភ្ជាប់ data ជាមួយនឹង code ដែលប្រើប្រាស់វា និងទី២ វាផ្តល់នូវមធ្យោបាយដែលអាចចូលទៅប្រើប្រាស់ member តាមកម្រិត assessment របស់វា។

-កាលណា member មួយមានលក្ខណៈ private នោះអ្នកដែលអាចប្រើ member នោះមានតែ methods ទាំងឡាយណាដែលមាននៅក្នុង class នោះតែប៉ុណ្ណោះ។

-កាលណា member មួយមានលក្ខណៈ public នោះមានន័យថាអ្នកណាក៏អាចប្រើប្រាស់វាបានដែរ គឺទាំងខាងក្រៅ class និងទាំងខាងក្នុង class ។

-ចំពោះ protected វិញវាអាចប្រើប្រាស់បានតែនៅក្នុង class ខ្លួនឯង នៅក្នុង subclasses ឬក៏ជាមួយ package class នោះស្ថិតនៅ។ ចំណាំថា subclasses នៅក្នុង package ផ្សេងអាច access បានតែ protected fields នៅក្នុង class វាផ្ទាល់ ឬក៏ជាមួយ object ដទៃទៀតដែលជា subclasses ប៉ុន្តែវាមិនអាច access protected fields ដែល instance ចេញពី superclass នោះបានទេ។

កាលដែលវាមានលក្ខណៈបែបនេះ វាជាផ្នែកសំខាន់មួយនៃភាសាសំនេរកម្មវិធី ព្រោះវាជួយការពារពីការមិនប្រើប្រាស់មិនត្រឹមត្រូវរបស់ object មួយ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើប្រាស់ public និង private members ៖

```
class  PrivatePublic{
        private x; // private access
        public y ; // public access
        int z ;     // default access
        void setX(int x){
                this.x=x;
        }
        int getX(){
                return x;
        }
}
classs DemoPrivatePublic{
        public static void main(String[] agr){
                PrivatePublic obj = new PrivatePublic();
                obj.x =10 ;
                obj.setX(20);
                obj.y=30;
                obj.z=40;
                System.out.println(" X value = " + obj.getX());
                System.out.println(" Y value = " + obj.y);
                System.out.println(" Z value = " + obj.z);
        }
}
```

End

មេរៀនទី ៦

អំពី *Inheritance*

Inheritance **គឺជាផ្នែកមួយនៃ** Object Oriented Programming **។ ការប្រើ** Inheritance **គឺអាចឲ្យយើងបង្កើត** Class **គំរូមួយ សម្រាប់ឲ្យ** class **ដទៃទៀតទទួលលក្ខណៈពីវា ដោយអាចបន្ថែមលក្ខណៈផ្ទាល់ខ្លួនបាន។** Class **ដែលផ្តល់លក្ខណៈឲ្យទៅគេ ត្រូវបានហៅថា** Super Class **ហើយ** Class **ដែលទទួលលក្ខណៈពីគេត្រូវបានហៅថា** Sub Class**។**

១. <u>**សញ្ញាណនៃ Inheritance**</u>

ដើម្បីឲ្យ Class **មួយទទួលលក្ខណៈ**(Inheritance) **ពី** Class **មួយទៀត គេត្រូវប្រើពាក្យ** extends **នៅពេលដែលគេបង្កើត**subclass **នោះ។**

ទម្រង់ទូទៅ៖

```
class SuperClass{
        data_members;
        methods_member;
        ……..…
}
class SubClass extends SuperClass{
        data_members;
        methods_members;
        ……..…
}
```

ឧទាហរណ៍៖

```
// Super Class
class Shape{
        double width;
        double height;
        void showWH(){
                System.out.println("Width = " + width + " and  Height = " + height);
        }
}
// Sub Class
class Triangle extends Shape{
        String style;
        double area(){
                return (width * height) / 2;
        }
        void showStyle(){
                System.out.println(" The Style of Triangle is " + style );
        }
}
```

```
class Demo{
        public static void main(String[] agr){
                Triangle obj = new Triangle ();
                obj.width = 4.0 ;
                obj.height = 5.0 ;
                obj.style = " isosceles ";
                System.out.println(" All Information of Object ");
                obj.showStyle();
                obj.showWH();
                System.out.println(" The Area = "+obj.area());
        }
}
```

ចំណាំ៖ ទោះបីជា Subclass **អាចទទួលយកលក្ខណៈទាំងអស់របស់** Superclass **ក៏ដោយ ក៏វាមិនអាចប្រើប្រាស់** members **ទាំងអស់នៃ** superclass **ដែលបានប្រកាសជា** private **បានឡើយ។ ក៏ប៉ុន្តែ ដើម្បីអាចប្រើប្រាស់** members **ទាំងអស់របស់** superclass **ដែលមានលក្ខណៈ** private **គេអាចប្រើវាតាមរយៈ** method **។**

ឧទាហរណ៍៖ យើងអនុវត្តទៅលើ class **ខាងលើ ដោយកំណត់** data member (width **និង** height) **ឲ្យមានលក្ខណៈ** private **វិញ យើងត្រូវសរសេរដូចខាងក្រោម៖**

```
// Super Class
class Shape{
        private double width;
        private double height;
        double getWidth(){
                return width;
        }
        double getHeight(){
                return height;
        }
        Void setWidth( double w){
                width=w;
        }
        Void setHeight( double h){
                height=h;
        }
        void showWH(){
                System.out.println("Width = " + width + " and  Height = " + height);
        }
}
```

```
// Sub Class
class Triangle extends Shape{
        String style;
        double area(){
                return (getWidth() * getHeight()) / 2;
        }
        void showStyle(){
                System.out.println(" The Style of Triangle is " + style );
        }
}
class Demo{
        public static void main(String[] agr){
                Triangle obj = new Triangle ();
                obj.setWidth = 4.0 ;
                obj.setHeight = 5.0 ;
                obj.style = " isosceles ";
                System.out.println(" All Information of Object ");
                obj.showStyle();
                obj.showWH();
                System.out.println(" The Area = "+obj.area());
        }
}
```

២. អំពី Constructor និង Inheritance

Super Class និង Sub Class វាមាន Constructor រៀងៗខ្លួនរបស់វា ដោយ constructor របស់ superclass វាកំណត់តម្លៃទៅឲ្យmember របស់វា ហើយ constructor របស់ subclass វាកំណត់តម្លៃទៅឲ្យ member របស់វា។ មូលហេតុដែលបណ្តាលឲ្យវាត្រូវមាន constructor រៀងៗខ្លួន ពីព្រោះ constructor របស់ superclass មិនអាចកំណត់តម្លៃទៅឲ្យmembers របស់ subclass បានទេ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការបង្កើតconstructor ទៅលើ subclass Triangle ៖

```
// Super Class
class Shape{
        private double width;
        private double height;
        double getWidth(){
                return width;
        }
        double getHeight(){
                return height;
        }
```

```
        Void setWidth( double w){
            width=w;
        }
        Void setHeight( double h){
            height=h;
        }
        void showWH(){
            System.out.println("Width = " + width + " and  Height = " + height);
        }
    }
    // Sub Class
    class Triangle extends Shape{
        String style;
        Triangle(String s, double w, double h){
            style = s ;
            setWidth(w);
            setHeight(h);
        }
        double area(){
            return (getWidth() * getHeight()) / 2;
        }
        void showStyle(){
            System.out.println(" The Style of Triangle is " + style );
        }
    }
    class Demo{
        public static void main(String[] agr){
            Triangle obj = new Triangle ("isosceles " , 5.0, 8.0);
            System.out.println(" All Information of Object ");
            obj.showStyle();
            obj.showWH();
            System.out.println(" The Area = "+obj.area());
        }
    }
```

៣. <u>ការហៅ Constructor របស់ superclass</u>

Subclass **មួយអាចហៅ** constructor **មួយកំណត់ដោយ** Superclass **តាមរយៈការប្រើ** keyword ***super*** **។ ទម្រង់នៃការប្រើប្រាស់ដូចខាងក្រោម៖**

```
super (parameter-list );
```

ក្នុងនោះ parameter-list **បញ្ជាក់នូវ** parameter **ណាមួយដែលត្រូវគ្នានឹង** constructor **នៅក្នុង** superclass ។ super() **ត្រូវតែស្ថិតនៅឃ្លាទីមួយជានិច្ចដើម្បីប្រតិបត្តិការនៅក្នុង**constructor **នៃ** subclass។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើប្រាស់keyword super **នៅក្នុង** subclass

```
class Shape{
        private double width;
        private double height;
        Shape( double w, double h){
                width = w;
                height = h ;
        }
        double getWidth(){
                return width;
        }
        double getHeight(){
                return height;
        }
        Void setWidth( double w){
                width=w;
        }
        Void setHeight( double h){
                height=h;
        }
        void showWH(){
                System.out.println("Width = " + width + " and  Height = " + height);
        }
}
// Sub Class
class Triangle extends Shape{
        private String style;
        Triangle(String s, double w, double h){
                super (w , h) ; // call supper class constructor
                style = s ;
        }
        double area(){
                return (getWidth() * getHeight()) / 2;
        }
        void showStyle(){
                System.out.println(" The Style of Triangle is " + style );
        }
}
class Demo{
```

```
        public static void main(String[] agr){
            Triangle obj = new Triangle ("isosceles " , 5.0, 8.0);
            System.out.println(" All Information of Object ");
            obj.showStyle();
            obj.showWH();
            System.out.println(" The Area = "+obj.area());
        }
    }
```

៤. <u>ការចូលទៅប្រើ member របស់ superclass</u>

Super **ត្រូវបានគេប្រើប្រាស់ស្រដៀងគ្នាទៅនឹង** this **ដែរគ្រាន់តែវាត្រូវបានគេប្រើតែនៅក្នុង** sub class **ដើម្បីហៅ** member **របស់** super class **។**

```
    class A{
        int i ;
    }
    class B extends A{
        int i ;
        B (int a , int b){
            super.i=a ;
            i = b ;
        }
        void show(){
            System.out.println("i  in super class : " + super.i);
            System.out.println(" i in sub class : " + i );
        }
    }
    class DemoSuper{
        public static void main(String[] agr){
            B obj = new B(4,5);
            Obj.show();
        }
    }
```

៥. <u>Superclass References និង Object នៃ Subclass</u>

អញ្ជាតនៃ superclass **ដែលមានលក្ខណៈ** reference **អាចកំណត់តម្លៃទៅឲ្យអញ្ជាតដែលមានលក្ខណៈ** reference **នៃ** subclass **ទាំងឡាយណាដែលទទួលលក្ខណៈពី**superclass **នោះ ។**

```
    class X{
        int a;
        X(int i){
            a=i;
        }
```

```
}
class Y extends X{
        int b;
        Y(int i, int j){
        super(j);
        b=i;
        }
}
class DemoSupperRef{
        public static void main(String[] agr){
                X x = new X(10);
                X x1;
                Y y = new Y(7,8);
                x1=x; // the same type
                System.out.println("x1.a " + x1.a);
                x1 = y ; // ok Y is derived from X;
                System.out.println("x1.a "+x1.a);
                x1.a=10 ;
                //x1.b=2; //error
        }
}
```

៦. <u>អំពី method Overriding</u>

នៅក្នុង java **កាលណានៅក្នុង** subclass **មាន** method **និងប្រភេទទិន្នន័យដូចគ្នាទៅនឹង** method **របស់** superclass **ពេលនោះ** method **នោះវាត្រូវបានគេហៅថា** Overriding method **ទៅលើ** method **នៅក្នុង** superclass **។**

```
class A{
        int i , j ;
        A( int a, int b){
                i=a;
                j=b;
        }
        void show(){
                System.out.println(" i = " + i);
                System.out.println(" j = " + j);
        }
}
class B extends A{
        int k;
        B(int a, int b, int c){
                super(a,b);
```

```
            k=c;
        }
        void show(){
            System.out.println(" k : " + k);
        }
    }
    class DemoOverriding {
        public static void main(String[] agr){
        B obj = new B(6,4,7);
        obj.show(); // this method show in class B;
        }
    }
```

<u>ចំណាំ</u>៖ Overriding method **លុះត្រាតែវាមានឈ្មោះ និងប្រភេទទិន្នន័យដូចគ្នា(ទាំងឈ្មោះ ចំនួន** parameter **និង** parameter type)**។ ប្រសិនបើវាខុសពីនេះវាក្លាយជា**Overloaded method **វិញ។**

```
    class A{
        int i , j ;
        A( int a, int b){
            i=a;
            j=b;
        }
        void show(){
            System.out.println(" i = " + i);
            System.out.println(" j = " + j);
        }
    }
    class B extends A{
        int k;
        B(int a, int b, int c){
            super(a,b);
            k=c;
        }
        void show(String msg){
            System.out.println(msg + k);
        }
    }
    class DemoOverriding {
        public static void main(String[] agr){
        B obj = new B(6,4,7);
        obj.show(); // this method show in class A
        obj.show("k: "); // this method in class B
        }
    }
```

៧. <u>Overridden methods ផ្តល់នូវលក្ខណៈ Polymorphism</u>

ការ Overridden method **បង្កើតនូវមូលដ្ឋានដ៏មានឥទ្ធិពលមួយរបស់** java **គឺ** dynamic method dispatch **វាគឺជា** mechanism **មួយដែលការហៅយកមកប្រើចំពោះ** method overridden **មួយត្រូវបានដោះស្រាយនៅក្នុងពេលដំណើរការ ជាជាងដោះស្រាយនៅពេលបំលែង** code **។ វាមានសារសំខាន់ពីព្រោះនេះជារបៀបដែល**java **អនុវត្តនូវលក្ខណៈ**polymorphism **ក្នុងពេលដំណើរការ។**

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើdynamic method dispatch

```
class Sup{
	void who(){
		System.out.println(" who in Sup ");
	}
}
class Sup1 extends Sup{
	void who(){
		System.out.println(" who in Sup1 ");
	}
}
class Sup2 extends Sup{
	void who(){
		System.out.println(" who in Sup2 ");
	}
}

class DemoDynamicDispatch{
	public static void main(String[] agr){
		Sup obj = new Sup();
		Sup1 obj1 = new Sup1();
		Sup2 obj2 = new Sup2();
		Sup objRef;
		objRef = obj;
		objRef.who();
		objRef = obj1;
		objRef.who();
		objRef = obj2;
		objRef.who();
	}
}
```

៨. <u>ការប្រើ abstract classes</u>

ជូនកាលយើងបង្កើត superclass ដើម្បីគ្រាន់តែកំនត់ទម្រង់ទូទៅឲ្យ subclass ហើយទុកឲ្យ subclass នីមួយៗបំពេញលក្ខណៈរបស់វាបន្ថែម។class បែបនេះកំណត់ methods ដើម (the nature of the methods) ដែល subclass ត្រូវតែអនុវត្តប្រើប៉ុន្តែមិនផ្តល់នូវការប្រើmethod ខ្លួនវាមួយ ឬច្រើនបាន។

Abstract method ត្រូវបានបង្កើតឡើងដោយប្រើពាក្យabstract ។ ដូច្នេះ abstract method គ្មាន body ទេ ហើយមិនត្រូវប្រើដោយ superclass ឡើយ ។ ហេតុនេះនៅក្នុង subclass របស់វាត្រូវមាន method overridden បានន័យថាគេមិនអាចប្រើmethod ដែលកំណត់នៅក្នុងsuperclass ឡើយ។

ទម្រង់ទូទៅរបស់វា៖

Abstract type name (parameter list);

ពាក្យ abstract នេះប្រើបានតែជាមួយ method ធម្មតាតែប៉ុណ្ណោះ វាមិនអាចប្រើជាមួយ static method ឬជាមួយ constructor បានឡើយ។

Class មួយដែលមាន abstract method មួយ ឬក៏ច្រើនត្រូវតែប្រើពាក្យ abstract នៅពីមុខឈ្មោះ class របស់វា។ ដោយ abstract class ពុំមានការអនុវត្តពេញលេញ ហេតុនេះហើយយើងមិនអាចបង្កើត object ចេញពីវាដោយប្រើ keyword ***new*** បានឡើយ បើយើងប្រើវា វានឹង error កើតឡើងនៅពេលដែលយើង compile វា។

កាលណា subclass មួយទទួលលក្ខណៈ abstract ពីclass មួយនោះវាត្រូវតែប្រើគ្រប់ abstract methods ទាំងអស់របស់ superclass ។ បើមិនដូច្នោះទេ subclass ត្រូវតែបញ្ជាក់លក្ខណៈជា abstract ដែរ។

ឧទាហរណ៍៖

```
abstract class Shape{
        private double width;
        private double height;
        private String name;
        Shape(){
                width=height=0.0;
                name= "null";
        }
        Shape(double w, double h, String n){
                width=w;
                height=h;
                name=n;
        }
        Shape(double x, String n){
                width=height=x;
                name=n;
        }
        Shape(Shape obj){
                width=obj.width;
                height=obj.height;
```

```
            name=obj.name;
        }
        double getWidth(){
            return width;
        }
        double getHeight(){
            return height;
        }
        Void setWidth( double w){
            width=w;
        }
        Void setHeight( double h){
            height=h;
        }
        String getName(){
            return name;
        }
        void show(){
            System.out.println("Width = " +width+ " Height = " + height);
        }
        abstract double area();
}

class Triangle extends Shape{
        private String style;
        Triangle (){
            super();
            style = "null";
        }
        Triangle (double w, double h, String s){
            super(w,h, "Triangle");
            style=s;
        }
        Triangle (double x){
            super(x, "Triangle");
            style = "isosceles" ;
        }

        Triangle (Triangle obj){
            super(obj);
```

```
                style=obj.style;
        }
        double area(){
                return getWidth() * getHeight()/2;
        }
        void styleShow(){
                System.out.println( "Triangle is " + style);
        }
}
class Rectangle extends Shape{
        Rectangle(){
                super();
        }
        Rectangle(double w , double h){
                super (w,h, "Rectangle");
        }
        Rectangle(double x){
                super(x, "Rectangle");
        }
        Rectangle(Rectangle obj){
                super(obj);
        }
        boolean isSquare(){
                if (getWidth()==getHeight())
                        return true;
                return false;
        }
        double area(){
                return getWidth() * getHeight();
        }
}
class DemoAbstract{
        public static void main (String[] agr){
                Shape shapeObj [ ] = new Shape [4];
                shapeObj[0]= new Triangle( "Right" ,9.0,11.0);
                shapeObj[1]= new Rectangle(11);
                shapeObj[2]= new Rectangle(12,5);
                for (int i =0 ; i<shapeObj.length ; i++){
                        System.out.println( " Object shape is " + shapeObj[i].getName());
                        System.out.println( "Area is " + shapeObj[i].area());
```

```
                System.out.println();
            }
        }
    }
```

៩. <u>ការប្រើប្រាស់ keyword *final*</u>

Keyword final **ត្រូវបានគេប្រើសម្រាប់ការពារពីការ** overriding methods **ការពារមិនឲ្យមាមលក្ខណៈ:** Inheritance **និងការកំណត់តម្លៃថេរទៅឲ្យ**variable **មួយ។**

ចំពោះការកំណត់តម្លៃថេរ៖បើសិនជា variable **មួយ ជា** variable **ផ្ទុកតម្លៃថេរ** (constant variable) variable **នោះត្រូវតាងដោយអក្សរធំទាំងអស់ គឺវាងាយស្រួលក្នុងការចងចាំ និងដឹងថាវាជា** variable **ដែលមិនអាចប្តូរតម្លៃបានទេនៅពេលដំណើរកម្មវិធីឧទាហរណ៍ដូចជាតម្លៃ**(PI =3,14)**ជាដើម។**

ឧទាហរណ៍ទី១៖

```
class A{
    final void myMethod(){
        System.out.println( " This is the Final Methods ");
    }
}
class B extends A{
    void myMethod(){ // error occur
        System.out.println( "This method will error  ");
    }
}
```

ឧទាហរណ៍ទី ២៖

```
final class A{
     void myMethod(){
        System.out.println( " This is the Method in Final Class ");
    }
}
class B extends A{  // error B can't extends from A, because A is FINAL
    void myMethod(){
        System.out.println( "This method in class B  ");
    }
}
```

ឧទាហរណ៍ទី ៣៖

```
class MessageError{
```

```
        final int OUT_ERROR=0;
        final int IN_ERROR=1;
        final int DISKET_ERROR=2;
        final int INDEX_ERROR=3;
        String[] sms = { "Output Error " , "Input Error" , "Disk Full " , "Index out of bound ");
        String getMessage(int i){
                if (i>=0 & i<sms.length)
                        return sms[i];
                else
                        return " Invalid code! ";
        }
}
class Demo{
        public static void main(String[] agr){
                MessageError sms = new MessageError();
                System.out.println(sms.getMessage(IN_ERROR));
                System.out.println(sms.getMessage(OUT_ERROR));
                System.out.println(sms.getMessage(INDEX_ERROR));
        }
}
```

End

មេរៀនទី ៧

អំពី *Package* និង *Interface*

១. Package

Package គឺជាបណ្តុំនៃ classes ដែលមានទំនាក់ទំនងគ្នា វាជួយយើងឲ្យងាយស្រួលចងចាំ code របស់យើង និងផ្តល់នូវស្រទាប់ encapsulation ។ យើងបង្កើត package មានគោលបំណងពីរយ៉ាងគឺ ទី ១ វាផ្តល់នូវ mechanism មួយដែលធ្វើឲ្យមានទំនាក់ទំនងគ្នារវាងផ្នែកផ្សេងៗនៃកម្មវិធី ដើម្បីរៀបជាក្រុមៗ ។ ទី ២ ផ្តល់នូវ mechanism សម្រាប់ត្រួតពិនិត្យអំពីការចូលប្រើប្រាស់ ដោយវាអាចផ្តាច់មិនឲ្យចូលទៅប្រើប្រាស់នូវ class មួយដោយ code ដែលនៅខាងក្រៅ package ។ បានជាយើងប្រើ package ព្រោះនៅក្នុងទីតាំងតែមួយ java វាមិនអនុញ្ញាតឲ្យមាន class ពីរដែលមានឈ្មោះដូចគ្នាបានឡើយ នៅក្នុង package មួយក៏មិនអាចមាន class ដែលមានឈ្មោះដូចគ្នាបានដែរ។

ដើម្បីបង្កើតនូវ package យើងត្រូវប្រើ keyword ***package*** នៅខាងលើគេនៃបណ្តុំ classes ដែលយើងចង់ដាក់នៅក្នុង package នោះ។

ទម្រង់ទូទៅ៖

```
package package_name;
```

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការបង្កើត package មួយឈ្មោះ PackageBook ៖

```
package PackageBook;
class Book{
        private String title;
        private String author;
        private String category;
        private String publicDate;
        Book(String t, String c ,String a ,String p ){
                title = t ;
                author = a ;
                category = c ;
                publicDate = p ;
        }
        void showBookDetail(){
                System.out.println(" Book title = " + title);
                System.out.println(" Book Category = " + category);
                System.out.println("Author  by = " + author);
                System.out.println(" Public on date   = " +publicDate);
        }
}
```

```
class Demo{
    public static void main(String[] agr){
        Book objBook[ ] = new Book [4];
        objBook[0] = new Book( " Visual Basic 6.0 ", "IT" , "Jonh", "12 June 1998 ");
        objBook[1] = new Book( " Data Structure", "IT" , "Koney", "12 October 1998");
        objBook[2] = new Book( " C++ Programming ", "IT" , "Smith", "20 May 1985");
        objBook[3] = new Book( " Java Programming", "IT" , "Kharo", "12 june 1998");
        for (int i =0 ; i < objBook.length ; i++)
            objBook[i].showBookdetail();
    }
}
```

យើងត្រូវដាក់ file នេះដោយ save នៅក្នុង directory មួយឈ្មោះ PackageBook ។ បន្ទាប់មក យើង compile វា (ត្រូវប្រាកដថា class ដែលកើតចេញពី file ខាងលើនេះពិតជាស្ថិតនៅក្នុង directory PackageBook) ។ បន្ទាប់មកចូរសាកល្បង Run command ដូចខាងក្រោម៖

```
C:\jdk1.7\bin > java PackageBook.Demo
```

១.១ ការប្រើប្រាស់ member របស់ Package

ដើម្បីយល់ពីការប្រើប្រាស់ member របស់ package គឺវាទាក់ទងនឹងការប្រកាស keyword private protected public ឬ default ។

ខាងក្រោមនេះជាតារាងបង្ហាញពីកម្រិតដែលអាចប្រើប្រាស់បាននៃ member របស់វា៖

Private Member	Default Member	Protected Member	Public Member	កម្រិតប្រើបាន
បាន	បាន	បាន	បាន	កម្រិតប្រើបានក្នុង class ដូចគ្នា
មិនបាន	បាន	បាន	បាន	កម្រិតប្រើបានក្នុង class ដូចគ្នា ចំពោះ subclass
មិនបាន	បាន	បាន	បាន	កម្រិតប្រើបានក្នុង package ដូចគ្នា ចំពោះ non-subclass
មិនបាន	មិនបាន	បាន	បាន	កម្រិតប្រើបានក្នុង package ផ្សេងគ្នា ចំពោះ subclass
មិនបាន	មិនបាន	មិនបាន	បាន	កម្រិតប្រើបានក្នុង package ផ្សេងគ្នា ចំពោះ non-subclass

ចំពោះ member ដែលបញ្ជាក់ដោយ keyword ***private*** អាចប្រើបានតែចំពោះ members ដទៃទៀតដែលស្ថិតនៅក្នុង class របស់វាតែប៉ុណ្ណោះ។

ចំពោះ member ដែលបញ្ជាក់ដោយ keyword ***default*** ឬ មិនមានបញ្ជាក់ពី keyword ណាមួយ គឺវាអាចប្រើបានតែចំពោះនៅក្នុង Package ខ្លួនឯងតែប៉ុណ្ណោះ វាមិនអាចប្រើឆ្លង package (ខាងក្រៅ package) បានឡើយ។

ចំពោះ member ដែលបញ្ជាក់ដោយ keyword ***protected*** អាចប្រើបានតែនៅក្នុង package របស់វា និងគ្រប់ subclasses ទាំងអស់ រួមទាំង subclasses ដែលស្ថិតនៅក្នុង package ផ្សេងផងដែរ។

ចំពោះ member ដែលបញ្ជាក់ដោយ keyword ***public*** អាចប្រើបានគ្រប់ទីកន្លែងទាំងអស់។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើប្រាស់របស់package ៖ ដោយឧទាហរណ៍ខាងលើ យើងបានដាក់ package និង class ដែលយើងប្រើប្រាស់វានៅក្នុង package តែមួយ ដូច្នេះវាពុំមានបញ្ហាអ្វីទេចំពោះការចូលទៅប្រើប្រាស់ member របស់វា។ ប៉ុន្តែនៅពេលដែលយើងដាក់វានៅក្នុង package ផ្សេងគ្នាវិញវានឹងមានបញ្ហាដោយមិនអាចប្រើប្រាស់បានឡើយ ដោយយើងត្រូវកែប្រែនូវចំនុចមួយចំនួន

-ទី១ ត្រូវតែប្រកាស class Book ជា public ដើម្បីឲ្យវាអាចប្រើ Book ពីខាងក្រៅ package PackageBook បាន។

-ទី ២ ត្រូវ constructor របស់វាក៏ត្រូវដាក់ជា public ដែរ

-ទី ៣ methods ទាំងអស់របស់វាក៏ត្រូវដាក់ជា public ដែរ

```
package PackageBook;
public class Book{
        private String title;
        private String author;
        private String category;
        private String publicDate;
        public Book(String t, String c ,String a ,String p ){
                title = t ;
                author = a ;
                category = c ;
                publicDate = p ;
        }
        public void showBookDetail(){
                System.out.println(" Book title = " + title);
                System.out.println(" Book Category = " + category);
                System.out.println("Author  by = " + author);
                System.out.println(" Public on date   = " +publicDate);
        }
}
```

ខាងក្រោមនេះជា package មួយទៀតដែលហៅវាមកប្រើ

```
package PBook;
class Demo{
        public static void main(String[] agr){
                PackageBook.Book objBook[ ] = new PackageBook.Book [4];
```

```
            objBook[0] = new PackageBook.Book( " Visual Basic 6.0 ", "IT" , "Jonh", "12
        June 1998 ");
            objBook[1] = new PackageBook.Book( " Data Structure", "IT" , "Koney", "12
        October 1998");
            objBook[2] = new PackageBook.Book( " C++ Programming ", "IT" , "Smith",
        "20 May 1985");
            objBook[3] = new PackageBook.Book( " Java Programming", "IT" , "Kharo",
        "12 june 1998");
            for (int i =0 ; i < objBook.length ; i++)
                objBook[i].showBookdetail();
        }
    }
```

១.២ ការប្រើប្រាស់ member ដែលមានលក្ខណៈ protected

ចំពោះ protected member វាអាចប្រើបានតែនៅក្នុង class និង subclass ប៉ុន្តែវាមិនអាច access ពីខាងក្រៅ package បានឡើយ។

ខាងក្រោមនេះជាឧទាហរណ៍ដែលបង្ហាញពីការប្រើ member មានលក្ខណៈ protected ដោយយើងធ្វើការផ្លាស់ប្តូរទៅលើ member របស់ class ខាងលើ ឲ្យមានលក្ខណៈជា protected វិញ។

```
package PackageBook;
public class Book{
    protected String title;
    protected String author;
    protected String category;
    protected String publicDate;
    public Book(String t, String c ,String a ,String p ){
        title = t ;
        author = a ;
        category = c ;
        publicDate = p ;
    }
    public void showBookDetail(){
        System.out.println(" Book title = " + title);
        System.out.println(" Book Category = " + category);
        System.out.println("Author  by = " + author);
        System.out.println(" Public on date   = " +publicDate);
    }
}
```

ខាងក្រោមនេះជា package មួយទៀតដែលហៅវាមកប្រើ

```
package PBook;
class Book1 extends PackageBook.Book{
        private String publisher;
        pubic Book1(String t, String c ,String a ,String pd, String ps ){
                super ( t , c , a , pd );
                publisher = ps ;
        }
        public void showBookDetail(){
                super.showBookDetail();
                System.out.println(publisher);
        }
        public String getPublisher(){
                return publisher;
        }
        public void setPublisher(String ps){
                publisher = ps ;
        }
        public String getTitle(){
                return title;
        }
        public void setTitle(String t){
                title = t ;
        }
        public String getAuthor(){
                return author;
        }
        public void setAuthor(String a){
                author = a ;
        }
        public String getCategory(){
                return category;
        }
        public void setCategory(String c){
                category = c ;
        }
        public String getPublicDate(){
                return publicDate;
        }
```

```
        public void setPublicDate(String p){
            publicDate = p ;
        }
    }
    class DemoProtected{
        public static void main(String[] agr){
            Book1 objBook[ ] = new Book1 [4];
            objBook[0] = new Book1( " Visual Basic 6.0 ", "IT" , "Jonh", "12 June 1998 ",
        "IIC");
            objBook[1] = new Book1( " Data Structure", "IT" , "Koney", "12 October
        1998","IIC");
            objBook[2] = new Book1( " C++ Programming ", "IT" , "Smith", "20 May
        1985");
            objBook[3] = new Book1( " Java Programming", "IT" , "Kharo", "12 june
        1998","RUPP");
            for (int i =0 ; i < objBook.length ; i++)
                objBook[i].showBookdetail();
            //find by publisher
            System.out.println("Show alls book that publish by IIC");
            for (int i=0 ; i< objBook.length ; i++)
                if (objBook[i].getPublisher () == "IIC")
                    System.out.println(objBook[i].getTitle());
        }
    }
```

១.៣ <u>ការប្រើប្រាស់ class នៅក្នុង Package</u>

ដើម្បីយក classes នៅក្នុង package មួយមកប្រើ យើងត្រូវប្រាប់ពីឈ្មោះ package និង class ដែលរបស់វា ដូចបង្ហាញនៅក្នុងឧទាហរណ៍ត្រង់ចំនុច **(១.១)** ប៉ុន្តែការដែលប្រើបែបនេះវាមាន ភាពស្មុគ្រស្មាញបន្តិច។ យើងមានវិធីមួយទៀតដែលកាត់បន្ថយការសរសេរច្រំដែលៗ និងភាពស្មុគ្រស្មាញនេះ ដោយជំនួសនូវពាក្យ ***import*** វិញ។

ទម្រង់ទូទៅរបស់ Import ៖

import package_name.class_name;

ខាងក្រោមនេះជាឧទាហរណ៍ដែលនៅក្នុងចំនុច (១.១) ដោយយើងប្រើនូវពាក្យ import ជំនួសវិញ

```
package PBook;
import PackageBook.*;
class Demo{
    public static void main(String[] agr){
        Book objBook[ ] = new Book [4];
        objBook[0] = new Book( " Visual Basic 6.0 ", "IT" , "Jonh", "12 June 1998 ");
        objBook[1] = new Book( " Data Structure", "IT" , "Koney", "12 October 1998");
```

```
                objBook[2] = new Book( " C++ Programming ", "IT" , "Smith", "20 May 1985");
                objBook[3] = new Book( " Java Programming", "IT" , "Kharo", "12 june 1998");
                for (int i =0 ; i < objBook.length ; i++)
                        objBook[i].showBookdetail();
        }
}
```

២. Interface

យើងបានសិក្សាម្តងរួចមកហើយ អំពី abstract method ។ ដែលនៅក្នុង super class បានកំណត់នូវអ្វីដែលត្រូវធ្វើ តែមិនបានកំណត់នូវរបៀបធ្វើឡើយ ដោយទុកលទ្ធភាពទៅឲ្យ subclass ជាអ្នកកំណត់ថាតើ methods នីមួយៗធ្វើអ្វីខ្លះ ធ្វើរបៀបណា? ។ ហេតុទាំងនេះគេថា abstract method ជាអ្នកកំណត់ interface ទៅឲ្យ method ។

ទម្រង់ទូទៅរបស់ interface មួយ៖

```
access interface interface_name{
        return_type methods_name (parameter_list);
        ……
        Data_type final_variable_name=value;
}
```

កាលណា access មិនបានដាក់នោះវានឹងចាប់យកជា default ហើយ interface នេះប្រើបានតែនៅក្នុង package របស់វាតែប៉ុណ្ណោះ។ តែបើយើងប្រកាសវាជា public នោះវាអាចប្រើបាននៅគ្រប់ទីកន្លែង ប៉ុន្តែត្រូវដាក់វានៅក្នុង file មួយដែលមានឈ្មោះដូចគ្នា។

ចំពោះ variable ដែលប្រកាសនៅក្នុង interface មិនចាំបាច់បញ្ជាក់ពី access ឡើយ ដោយវាអាចជា public , final និង static **ប៉ុន្តែត្រូវតែកំណត់តម្លៃដំបូង**(នាំឲ្យវាជាតម្លៃថេរ)។ methods និង variables ទាំងអស់សុទ្ធតែជា public តែយើងមិនបាច់សរសេរឡើយ។

ឧទាហរណ៍ទី ១ ៖ បង្កើត interface មួយសម្រាប់បង្កើត series នៃលេខ

```
public interface Series{
        int getNext(); // show the next number
        void reset(); // reset
        void setStart(int x); // set the number to be start
}
```

ឧទាហរណ៍ទី ២ ៖ បង្កើត interface Driveable

```
interface Driveable {
        boolean startEngine( );
        void stopEngine( );
        float accelerate( float acc );
        boolean turn( String direction );
}
```

២.១ <u>វិធីនៃការយក interface ទៅប្រើ</u>

ដើម្បីប្រើប្រាស់ interface នៅក្នុង class មួយយើងត្រូវប្រើពាក្យ implements នៅពីក្រោយ class ដែលយើងទើបបង្កើត។ ប្រសិនបើយើងត្រូវការប្រើ interfaces ច្រើននៅក្នុង class តែមួយយើងត្រូវផ្តាច់ពីគ្នាដោយសញ្ញា (,)។

ទម្រង់ទូទៅនៃ class មួយដែល implements ទៅលើ Interface ៖

```
access class class_name extends superclass implements interface{
        //body
}
```

ពាក្យ extends យើងប្រើក៏បាន មិនប្រើក៏បាន ហើយ access ជា public ឬក៏មិនប្រើ។

ឧទាហរណ៍បង្ហាញពីការប្រើ interface ទាំងពីរខាងលើ៖

ឧទាហរណ៍ទី ១ ៖

```
class ByTwos implements Series{
        int start;
        int val ;
        ByTwos(){
                start =0 ;
                Val = 0 ;
        }
        pubic int getNext(){
                val+=2;
                return val;
        }
        public void reset(){
                start = 0;
                val = 0;
        }
        public void setStart( int x){
                start = x;
                val = x;
        }
}
```

ឧទាហរណ៍ទី ២ ៖

```
class Automobile implements Driveable {
   public boolean startEngine( ) {
        System.out.println(" The machine is running now ");
   }
   public void stopEngine( ) {
      System.out.println(" The machine is stopped now ");
   }
```

```
    public float accelerate( float acc ) {
        System.out.println(" The car speed add  " + acc + " more! Please drive carefully! ");
    }
    public boolean turn( String direction ) {
        System.out.println(" Now, the car turn " +  dir );
    }
}
```

<u>កំណត់ចំណាំ៖</u>

នៅក្នុង class នីមួយៗ យើងអាចបន្ថែម data member និងmethod ប៉ុន្មានទៀតក៏បានដែរ។

ខាងក្រោមបង្ហាញពីការប្រើប្រាស់របស់វា

```
class DemoSeries{
        public static void main(String[] agr){
                ByTwos obj= new ByTwos();
                for (int i=0 ; i< 10 ; i++)
                        System.out.print ( obj.getNext() + "  ");
                System.out.println();
                System.out.println(" Resetting the Value ");
                obj.reset();
                for (int i=0 ; i< 10 ; i++)
                        System.out.print ( obj.getNext() + "  ");
                System.out.println();
                System.out.println(" Setting the start value ");
                obj.setStart(23);
                for (int i=0 ; i< 10 ; i++)
                        System.out.print ( obj.getNext() + "  ");
        }
}
class DemAutomobile{
        public static void main(String[] agr){
                Automobile obj =new Automobile();
                obj. startEngine( );
                obj.stopEngine( );
                obj.accelerate( 30.0 );
                obj.turn( "right" );
        }
}
```

២.២ ការប្រើ variable នៅក្នុង Interface

ជាទូទៅកម្មវិធីជាច្រើនដែលបានប្រើប្រាស់តម្លៃថេរ ដើម្បីធ្វើការកំណត់ និងរៀបរាប់ពីរបស់ផ្សេងៗ ដូចជាទំហំរបស់ array តម្លៃពិសេសណាមួយ តែដោយសារកម្មវិធីទាំងនោះវាទាញយក resource ផ្សេងៗពីគ្នា ដូច្នេះដើម្បី អាចឲ្យប្រើបាននូវតម្លៃទាំងនោះនៅលើ resources នីមួយៗ គេប្រើ variable របស់ interface ក្នុងការដោះស្រាយបញ្ហានេះ។ គេត្រូវបង្កើតនូវ Interface មួយដែលផ្ទុកនូវ តម្លៃថេរទាំងនោះ ដោយមិនមាន method ណាមួយឡើយ។

ឧទាហរណ៍៖

```
interface MyConst{
        int MAX=15;
        int MIN = 1;
        String SMSERROR = " Index Out of Bound ";
}
class DemoConst implements MyConst{
        public static void main(String[] agr){
                int num[] = new int[MAX];
                for (int i=MIN; i<23 ; i++)
                        if (i>MAX)
                                System.out.println(SMSERROR);
                        else
                                num[i] = i * i;
                                System.out.println(" num ["+ i + "] = " + num[i]);
        }
}
```

២.៣ ការពង្រីកលក្ខណៈរបស់Interface

Interface មួយអាចពង្រីកខ្លួនរបស់វាបានដោយប្រើ keyword extends ។ កាលណា class មួយប្រើ interface មួយដែលទទួលលក្ខណៈពី interface មួយទៀតនោះវាត្រូវតែប្រើ methods ទាំងអស់ដែលមាននៅក្នុង Interface ដើម។

ឧទាហរណ៍៖

```
interface A{
        void method1();
        void method2();
}
interface B extends A{
        void method3();
}
class MyClass implement B{
        public void method1(){
                System.out.println("This is implement of  Method 1 ");
```

```
        }
        public void method2(){
            System.out.println("This is implement of  Method 2 ");
        }
        public void method3(){
            System.out.println("This is implement of  Method 3 ");
        }
    }
    class Demo{
        public static void main(String[] agr){
            MyClass obj=new MyClass();
            Obj.method1();
            Obj.method2();
            Obj.method3();
        }
    }
```

ចូរសាកល្បងលុប នូវ method1 ឬ method2 នៅក្នុងClass Myclass ដោយមិន Implement តើមានបញ្ហាអ្វីកើតឡើង?

End

មេរៀនទី ៨

ការប្រើ *Exception*

១. Exception

Exception គឺជា error ដែលកើតឡើងនៅពេលដែលយើងដំណើរការកម្មវិធី។ការប្រើ Exception វាមានប្រយោជន៍ដូចតទៅ៖ វាធ្វើការដោយស្វ័យប្រវត្តនូវ code ដើម្បីដោះស្រាយ error ជាច្រើនដែលពីមុនត្រូវបញ្ចូលដោយខ្លួនឯងទៅក្នុងកម្មវិធី។ការប្រើ exception រៀបចំឡើងវិញនូវ Error ដោយអនុញ្ញាតឲ្យកម្មវិធីកំណត់នូវ block នៃ code (exception handler) ដែលត្រូវអនុវត្តដោយស្វ័យប្រវត្តនៅពេលដែលមាន error កើតឡើង។ ម្យ៉ាងទៀត java អាចឲ្យយើងកំណត់នូវ exception គំរូសម្រាប់ error នៃកម្មវិធីដូចជា ការចែកចំនួនមួយនឹងសូន្យឬការរកឯកសារមិនឃើញជាដើម។

Exception classes ទាំងអស់មានប្រភពចេញពី class មួយឈ្មោះថា Throwable ។ Throwable មាន subclass ពីរគឺ Exception និង Error។

Exception ជាប្រភេទ Error ដែលទាក់ទងនឹង error ដែលកើតឡើងនៅក្នុង Java Virtual Machine ខ្លួនវា។

Error ជាលទ្ធផលពីសកម្មភាពកម្មវិធី ត្រូវបានតាងដោយពពួក subclass នៃ Exception ។ ឧទាហរណ៍ដូចជា ការចែកមួយចំនួននឹងសូន្យ ការប្រើហួសដែនកំណត់របស់ Array និង file error ជាដើម។

២. មូលដ្ឋាននៃការប្រើ Exception

ដើម្បីគ្រប់គ្រង Exception យើងប្រើ keyword មួយចំនួនដូចជា try, catch, throw, throws និង finally ជាដើម។ code ដែលត្រូវត្រួតពិនិត្យ exception ត្រូវផ្ទុកនៅក្នុង try block ព្រោះបើសិនជាមាន exception កើតឡើងវានឹងចាប់គ្រវែងចោល។វាចាប់ exception ដោយប្រើ catch។ ដើម្បីគ្រវែងចោលនូវ exception មួយ តាមលក្ខណៈធម្មតា គេប្រើ keyword throw។ ក្នុងករណីខ្លះ exception មួយដែលបានចាប់គ្រវែងចោលពី method គេប្រើ keyword throws ។ ចំពោះ code ដែលជានិច្ចកាលតែងតែប្រតិបត្តការនោះត្រូវដាក់នៅក្នុងfinally block ទោះជាគ្មានការកើតឡើងនូវerror ក្នុង try block ក៏ដោយ។

៣. ការប្រើ try - catch

Try និង catch ត្រូវតែប្រើជាមួយគ្នា មិនអាចប្រើ try ដោយគ្មាន catch ឬប្រើ catch ដោយគ្មាន try បានឡើយ។

ទម្រង់ទូទៅរបស់try catch៖

```
try {
        //block of code to monitor for errors
}catch(exception_Type1 exobj){
        // handler for exception type1
} catch(exception_Type2 exobj){
        // handler for exception type2
}
```

ឧទាហរណ៍ទី ១៖

```
class DemoException {
	public static void main(String[] agr){
		int num[ ] = new int[4];
		try{
			System.out.println( "Before Exception is generate ");
			for (int i=0 ; i< 9 ; i++){
				num[i] = i * 2 ;
				System.out.println( "num [" + i + "]= " + num[i]);
			 }
		}catch (ArrayIndexOutOfBoundsException ex){
			System.out.println("Hey Index out of bound sir! ");
		}
		System.out.println( "After catch execute ");
	}
}
```

ឧទាហរណ៍ទី ២៖

```
import java.util.*;
public class DemoHandlerException{
	public static void main(String[] agr){
		Scanner sc=new Scanner(System.in);
		boolean cont=true;
		do{
			System.out.println( "Please input an Integer Number ");
			int num=sc.nextInt();
			System.out.println(" The number is " + num);
			cont =false;
		}catch (InputMismatchException ex){
			System.out.println(" Try again sir! Yr input not Integer ");
			sc.nextLine();
		}
	}while(cont);
}
```

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើcatch ច្រើននៅក្នុង try តែមួយ

```
class DemoMultiCatch{
	public static void main(String[] agr){
		int num[ ] ={ 4,8,16,32,64,128};
		int divide[ ] ={ 2,0,5,0,8};
		for(int i=0 ; i<num.length;i++){
```

```
                    try{
                        System.out.println(num[i] + " / " + divide[i] + "=" +num[i]
                                /divide[i]);
                    }catch(ArithmeticException ex){
                        System.out.println("Can not divide by Zero ! ");
                    }catch(ArrayIndexOutOfBoundsException ex){
                        System.out.println("No match element found ! ");
                    }
                }
            }
        }
```

៤. <u>ការប្រើ try នៅក្នុង try</u>

វាមានលក្ខណៈស្រដៀងគ្នាទៅនឹងការប្រើ if **នៅក្នុង** if **ដែរ។ គឺ** try **នៅផ្នែកខាងក្រៅវាសម្រាប់ចាប់យក** error **ដែលមានលក្ខណៈទូទៅនិង** try **ខាងក្នុងសម្រាប់ចាប់យក** error **លំអិតជាង។**

```
class DemoTryInTry{
    public static void main(String[] agr){
        int num[ ] ={ 4,8,16,32,64,128};
        int divide[ ] ={ 2,0,5,0,8};
        try{
            for(int i=0 ; i<num.length;i++){
                try{
                    System.out.println(num[i] + " / " + divide[i] + "=" +num[i]
                            /divide[i]);
                }catch(ArithmeticException ex){
                    System.out.println("Can not divide by Zero ! ");
                }
            }
        }catch(ArrayIndexOutOfBoundsException ex){
            System.out.println("No match element found ! ");
        }
    }
}
```

៥. <u>ការគ្រវែងចោលនូវ Exception</u>

គេអាចធ្វើការគ្រវែងចោលនូវ exception **មួយតាមលក្ខណៈធម្មតាបាន ដោយប្រើ** keyword throw **បាន។**

ទម្រង់ទូទៅរបស់វាគឺ៖

throw exceptionObject;

ដែល exceptionObject **គឺជា** Object **នៃ** Exception class **មួយបានទទួលលក្ខណៈពី** throwable**។**

```
class DemoThrow{
        public static void main(String[] agr){
                try{
                        System.out.println("Before Throw ");
                        throw new ArithmeticException();
                }catch(ArithmeticException ex){
                        System.out.println("Exception caught ");
                }
                System.out.println("After try/catch block ");
        }
}
```

៦. <u>ការប្រើ keyword *finally*</u>

ក្នុង java យើងប្រើ try catch ដើម្បីចាប់ error និងបង្ខំវាឲ្យចាកចេញពីបណ្តុំនៃ code ក៏ប៉ុន្តែនៅក្នុងបណ្តុំនៃ code នោះពេលខ្លះយើងបានបើក file ឬ connection ណាមួយ តែមិនទាន់បានបិទនៅឡើយ ស្រាប់តែមាន error ដូច្នេះនៅគ្រប់នៃភាសាសរសេរកម្មវិធីទាំងអស់ទាមទារឲ្យយើងត្រូវតែបិទ file ឬក៏ connectpion នោះវិញ។ ករណីនេះ java បានផ្តល់នូវមធ្យោបាយងាយស្រួលមួយសម្រាប់ការប្រើប្រាស់វាដែរគឺតាមរយៈការប្រើfinally ។

ទម្រង់ទូទៅនៃការប្រើfinally ៖

```
try{
        //block of code to monitor for ERROR
}catch(ExceptionType1 ex){
        //handler for exception  type 1
}catch(ExceptionType2 ex){
        //handler for exception  type 2
}
finally{
        //finally code
}
```

Finally code នឹងដំណើរការនៅពេលដែលដំណើរការបានចាកចេញផុតពីtry catch block ។

```
public class TestFinally {
   public static void generateException(int gen){
        int t;
        int num[] = new int[2];
        System.out.println("Receiving " + gen);
        try{
                switch(gen){
                        case 0 :
                                t=10; break;
                        case 1:
```

```
                        num[4]=4; break;
                    case 2 :
                        return ;
                }
            }catch(ArithmeticException ex){
                System.out.println( "Can not divide by zero!");
                return;
            }catch (ArrayIndexOutOfBoundsException ex){
                System.out.println("No matching element found.");
            }finally{
                System.out.println("Leaving try");
            }
        }
    }
    class DemoFinally{
        public static void main (String[] args) {
            for(int i=0 ;i<3;i++){
                TestFinally.generateException(i);
                System.out.println();
            }
        }
    }
```

៧. <u>ការប្រើ keyword *throws*</u>

យើងអាចប្រើ keyword **throws នៅផ្នែកខាងក្រោយនៃការប្រកាស** method **ដើម្បីឲ្យវារាយនូវប្រភេទនៃ** exception **ដែល** method **អាចចាប់គ្រវែងចោលបាន។ វាមានសារៈសំខាន់សម្រាប់គ្រប់** exceptions **ទាំងអស់លើកលែងតែពួក**error **និង** RuntimeException **និង** subclass **របស់វា។**

ទម្រង់ទូទៅនៃការប្រើthrows keyword **៖**

```
Return-type method-name (parameters-list) throws exception-list{
    // body of method
}
```

យើងអាចឆ្ងល់ថាពីមុនមកហេតុអ្វីបានជាយើងមិនប្រើthrows **ដែលត្រូវគ្រវែងចោលនូវ** exception **មកក្រៅ** method **ព្រោះ** exception **ទាំងនោះគឺជា** subclass **នៃ** error or RuntimeException **ដែលមិនចាំបាច់បញ្ជាក់ឃ្លា** throws **ឡើយ។ តែចំពោះ** Exception **ផ្សេងទៀតចាំបាច់ត្រូវតែប្រកាស បើមិនដូច្នេះទេវានឹងមាន** error **កើតឡើងនៅពេលដែលយើង**compile **។**

```
// demo use throws
class DemoThrows{
    public static char prompt(String [] str) throws java.io.IOException{
```

```
                System.out.println(str + " :  ");
                return (char) System.in.read();
        }
        public static void main(String[] agr){
                char ch;
                try{
                        ch= prompt("Enter a letter ");
                }catch(java.io.IOException ex){
                        System.out.println("I/O exception occurred. ");
                        ch = 'X';
                }
                System.out.println("You press  " + ch );
        }
}
```

៨. <u>ការបង្កើត Exception Subclass</u>

ថ្វីត្បិតតែ java **បានផ្តល់នូវ** Exception **ជាច្រើន ប៉ុន្តែវានៅតែមិនគ្រប់គ្រាន់សម្រាប់ចាប់គ្រប់វែង ចោលនូវ** error **ដែលកើតឡើងនៅពេលដែលយើងប្រើប្រាស់ដែរ។ដូច្នេះយើងអាចបង្កើត** exception **ទៅ តាម** error **នៅក្នុងកម្មវិធីរបស់យើង ។ ការបង្កើតនេះ គ្រាន់តែកំណត់លក្ខណៈទៅឲ្យ** subclass **នៃ** exception (**ដែលជា** subclass **នៃ** throwable)**នោះជាការស្រេច ។**

Subclass **មិនចាំបាច់អនុវត្ត ឬ ប្រើអ្វីផ្សេងទៀតឡើយ បានន័យថាវាអនុញ្ញាតឲ្យប្រើបានដូច** exception **ដែរ។ តាមពិត** Exception **ពុំបានកំណត់នូវ** methods **អ្វីសម្រាប់ខ្លួនវាឡើយ តែវាអាចប្រើ** method **ផ្សេងៗបាន ដោយសារវាទទួលលក្ខណៈពី** throwable **ប៉ុណ្ណោះ។ ហេតុនេះគ្រប់** exceptions **ទាំងអស់រួមទាំង** exceptions **ដែលគេបង្កើតសុទ្ធតែមាន**methods **កំណត់ដោយ** throwable **។**

ឧទាហរណ៍៖

```
class NonIntResultException extends Exception{
        int n;
        int d;
        NonIntResultException (int i , int j){
                n=i;
                d=j;
        }
        public String toString(){
                return "Result of " + n + " / " + d + " is non-integer. ";
        }
}
class DemoException{
        public static void main(String[] agr){
                int num[] ={4,8,16,32,64,128,256};
                int denom[] = {2,0,4,6,0,8};
                for(int i=0;i<num.length;i++){
```

```
                try{
                    if ((num[i]%2)!=0) throw new NonIntResultException(num[i],denom[i];
                    System.out.println(num[i] + "/" + denom[i] + " is " + num[i] / denom[i]);
                }catch(ArithmeticException ex){
                    System.out.println("Can not divide by Zero! ");
                }catch(ArrayIndexOutOfBoundsException ex){
                    System.out.println("No match element found ");
                }catch(NonIntResultException ex){
                    System.out.println(ex);
                }
            }
        }
    }
```

៨. Exception ដែលមានស្រាប់

Java **បានកំណត់នូវ** Exception class **ជាច្រើននៅក្នុង** package **រួមមួយឈ្មោះ** java.lang។ Exceptions **ដែលប្រើជាទូទៅ ភាគច្រើនជា** subclass **នៃពពួក** RuntimeException ។ **លើសពីនេះទៅទៀតវាមិនចាំបាច់បញ្ជូលឃ្លា** throws **ដែលត្រូវរាយឈ្មោះ** exception **ទៅឲ្យ** methods **ណាមួយឡើយ។ ពពួក** exception **ទាំងនេះត្រូវបានហៅថា** Unchecked Exceptions **ព្រោះ** compiler **មិនធ្វើការត្រួតពិនិត្យទៅលើ** method **ថាតើវាបានប្រើ ឬចាប់គ្រែងចោលនូវ** exceptions **ដែរឬទេ។**

ខាងក្រោមនេះជា Unchecked Exception **នៅក្នុង** java.lang **៖**

Exception Type	ខ្លឹមសារ
ArithmeticExceptin	កំហុសបែបនព្វន្តលេខ ដូចជាចែកមួយចំនួននឹងសូន្យ.....
ArrayIndexOutOfBoundsException	Index របស់ array ក្រៅដែនកំណត់
ArrayStoreException	ការកំណត់ធាតុ Array មួយមានប្រភេទទិន្នន័យមិនស៊ីគ្នា
ClassCastException	ការប្រើលក្ខណៈ cast មិនត្រឹមត្រូវ
IllegalArgumentException	ការប្រើ argument មិនត្រឹមត្រូវ
IllegalMonitorStateException	ប្រតិបត្តិការត្រួតពិនិត្យមិនត្រឹមត្រូវ ដូចជាការរងចាំទៅលើ thread មិនឃាំងទុក
IllegalStateException	មជ្ឈដ្ឋាន ឬការអនុវត្តមិនត្រឹមត្រូវ
IllegalThreadStateException	ការប្រតិបត្តិដែលស្នើឡើងមិនត្រូវគ្នានឹងសភាពលក្ខណៈរបស់ thread កំពុងប្រើ
IndexOutOfBoundsException	Index របស់ទិន្នន័យប្រភេទអ្វីមួយនៅក្រៅដែនកំណត់
NegativeArraySizeException	Array បានបង្កើតនូវទំហំតម្លៃអវិជ្ជមាន
NullPointerException	ការប្រើមិនបានត្រឹមត្រូវនៃការបញ្ជាក់ឲ្យ object មួយដែលពុំមានសោះ
NumberFormatException	ការបំលែង string មួយឲ្យទៅជាចំនួនលេខមិនត្រឹមត្រូវ
SecurityException	ព្យាយាមបំពានទៅលើផ្នែកសុវត្ថភាព
StringIndexOutOfBounds	Index របស់ទិន្នន័យប្រភេទ string នៅក្រៅដែនកំណត់
UnsupportedOperationException	ជួបនឹងប្រតិបត្តិការមួយដែលមិនស្គាល់

ខាងក្រោមនេះជា checked Exception **នៅក្នុង** java.lang **៖**

Exception Type	ខ្លឹមសា
ClassNotFoundException	រក class មិនឃើញ
CloneNotSupportedException	ព្យាយាមធ្វើឲ្យដូច object មួយដែលមិនបានប្រើ Cloneable interface
IllegalAccessedException	ការចូលទៅប្រើ class មួយត្រូវបានបដិសេធ
InstantiationException	ព្យាយាមបង្កើតនូវ object នៃ abstract class ឬ interface មួយ
InterruptedException	Thread មួយត្រូវបានបង្អាក់ដោយ thread មួយទៀត
NoSuchFieldException	អញ្ញាតមួយ(field)ដែលស្នើឡើងនោះពុំមានសោះ
NoSuchMethodException	Method ដែលស្នើឡើងនោះពុំមានសោះ

End

មេរៀនទី ៩

String និង Collections

១. ទម្រង់ទូទៅនៃការប្រើ String

String គឺជា data type មួយប្រភេទដែលមានលក្ខណៈខុសពី Simple data type ។ វាគឺជា class មួយ ដែលមានទម្រង់នៃការប្រើប្រាស់ដូចខាងក្រោម៖

```
String s = new String();
String s= new String( char chars[]);
String s = new String(char chars[], int startIndex, int numChar);
String s = new String (String strObj);
String s = new String(byte asciiChars[]);
String s = new String(byte asciiChars[], int startIndex,int numChars);
```

ឧទាហរណ៍៖

```
class DemoString{
        public static void main(String[] agr){
                char ch[]={´C´, ´H´, ´A´, ´R´, ´A´, ´C´, ´T´, ´E´, ´R´};
                byte ascii[] = { 65,66,67,68};
                String s1 = new String();
                String s11 =  "MY STRING";
                String s2 = new String(ch);
                String s3 = new String(ch , 3,8);
                String s4 = new String(s2);
                String s5 = new String(ascii);
                String s6 = new String(ascii,1,3);
                s1= "CHARACTER";
                System.out.println(s1);
                System.out.println(s11);
                System.out.println(s2);
                System.out.println(s3);
                System.out.println(s4);
                System.out.println(s5);
                System.out.println(s6);
        }
}
```

១.១ ការតភ្ជាប់ String

នៅក្នុង java វាមិនអនុញ្ញាតឲ្យយើងប្រើសញ្ញានព្វន្តជាមួយ object នៃ String បានឡើយ លើកលែងតែសញ្ញា (+) ដែលនៅក្នុង java វាដើរតួជាសញ្ញាតភ្ជាប់ String កាលណាអង្គម្ខាងមានប្រភេទទិន្នន័យជា String ។ យើងអាចតភ្ជាប់ String ជាមួយ String, String ជាមួយប្រភេទទិន្នន័យផ្សេងៗបាន។

ឧទាហរណ៍៖

```
class DemoConcatenate{
        public static void main(String[] agr){
                int age = 29;
                String defString="STRING : A class used to represent textual information." +
                                "The String class includes many methods for operating " +
                                "on string objects. Java overloads the + operator for " +
                                "string concatenation. ";
                String myName = " My name is BUN SEIHAK. I'm " + age + " years old.";
                System.out.println(defString);
                System.out.println(myName);
                System.out.println("============================" );
                System.out.println("Very important! Be careful! ");
                String number4 = "four : " + 2 + 2;
                System.out.println("The result: " + number4);
                String number44 = "four : " + (2 + 2);
                System.out.println("The result: " + number44);
        }
}
```

១.២ ការប្រើអនុគមន៍ផ្សេងៗរបស់String Object

Java បានផ្តល់នូវ methods ដ៏ច្រើន សម្រាប់ប្រើប្រាស់នៅក្នុង String class ។

១.២.១ អនុគមន៍ length()

Method ***length()*** ត្រូវបានគេប្រើសម្រាប់ផ្តល់ចំនួន characters នៅក្នុង String មួយ ដោយរាប់បញ្ចូលទាំង ដកឃ្លាផងដែរ។

ឧទាហរណ៍៖

```
char ch[]={'C', 'H', 'A', 'R', 'A', 'C', 'T', 'E', 'R'};
 str = new String(ch);
System.out.println( " This string have length : "+ str.length() + " digit(s)." );
```

ឬ យើងអាចសរសេរដូចខាងក្រោម៖

```
System.out.println("My name is Seihak".length());
```

១.២.២ ការប្រើ toString()

Java វានឹងប្រើប្រាស់ method **valueOf()** ដើម្បីបំលែងប្រភេទទិន្នន័យទៅជា String ក្នុងការភ្ជាប់ String នោះ។ ដែលជា method Overloaded មួយក្នុងចំណោម method ទាំងឡាយ

ដែលកំណត់ដោយ String class ។ ដោយ valueOf() វាបានហៅ method toString() របស់object នោះមកប្រើ។

ដើម្បីយល់ពីលក្ខណៈនេះ ចូរពិនិត្យមើលឧទាហរណ៍ខាងក្រោម៖

```
class Box{
        double width, height, depth;
        Box(double w, double h, double d){
                width= w ;
                height= h ;
                depth = d ;
        }
        public String toString(){
                return "Dimensions are " +width+ " by "+ depth + " by "+ height + "." ;
        }
}
class DemoToString{
        public static void main(String[] agr){
                Box b = new Box(11,23,12);
                String s = "Box b " + b;
                System.out.println(b);
                System.out.println(s);
        }
}
```

១.២.៣ ការប្រើ charAt()

គេប្រើ charAt() ដើម្បីទាញយក character ណាមួយចេញពី String នៅទីតាំងណាមួយដែលយើងបានកំណត់។

ទម្រង់ទូទៅរបស់ charAt()៖

```
char charAt(int where);
```

ដែល where គឺជា index ដែលយើងចង់យកចេញពី String មួយ ហើយ index នេះដាច់ខាតត្រូវតែជាចំនួន **វិជ្ជមាន**។

ឧទាហរណ៍៖

```
char ch;
ch = "My name is Seihak".charAt(1); // វាផ្តល់តម្លៃ y ទៅឲ្យ ch
ch = "My name is Seihak".charAt(9); // វាផ្តល់តម្លៃ s ទៅឲ្យ ch
ch = "My name is Seihak".charAt(11); // វាផ្តល់តម្លៃ S ទៅឲ្យ ch
```

១.២.៤ ការប្រើ getChars()

យើងឃើញថា method charAt() គឺវាអាចទាញយកបានតែមួយcharacter ប៉ុណ្ណោះចេញពី String ណាមួយ តែវាមិនអាចទាញយកតួអក្សរជាច្រើន ឬពាក្យណាមួយចេញពី String ណា

មួយបានឡើយ។ដើម្បីទាញយកតួអក្សរជាច្រើនឬពាក្យណាមួយយើងអាចប្រើ method getChars() ។

ទូទៅរបស់ getChars();

void getChars(int sourceStart, int sourceEnd, char target[],int targetStart);

ដែល sourceStart សម្រាប់កំណត់ index ចាប់ផ្តើមរបស់substring, sourceEnd សម្រាប់កំណត់ឲ្យប index ខាងចុងនៃ substring តែទិន្នន័យដែលទទួលបានគឺ ពី sourceStart ដល់ sourceEnd-1។ ហើយ target ជាអ្នកទទួលទុកនូវអក្សរទាំងនោះ។ index នៅក្នុង target នឹងត្រូវចម្លងដាក់នោះ គឺត្រូវឆ្លងតាម targetStart ។

ឧទាហរណ៍៖

```
class DemogetChars{
        public static void main(String[] agr){
                String str = " This is a Demo of the getChars method.";
                int start=10 ;
                int end = 14 ;
                char ch[] = new char[end - start];
                str.getChars(start,end,ch,0);
                System.out.println(ch);
        }
}
```

១.២.៥ ការប្រើ getBytes()

Method getBytes() ត្រូវបានគេប្រើសម្រាប់ជួយឲ្យគេអាចផ្ទុក តួអក្សរជា byte នៅក្នុង array ដោយវាបំលែងលក្ខណៈពីតួអក្សរទៅជា byte ។

ទម្រង់ទូទៅរបស់ getBytes()៖

```
byte[] getBytes();
```

ឧទាហរណ៍៖

```
class DemoGetBytes{
        public static void main (String[] agr){
                String str = "Now we demo about how to use getBytes method";
                byte b[] = s.getBytes();
                for(int i=0 ; i<b.length ; i++)
                System.out.println((char) b[i]);
        }
}
```

១.២.៦ ការប្រើ toCharArray()

Method toCharArray() ត្រូវបានគេប្រើសម្រាប់បំបែកពី String មួយ ដើម្បីយកធាតុមួយនីមួយៗនៅក្នុង String នោះទៅដាក់នៅក្នុង array មួយវិញ។

ទម្រង់របស់ toCharArray() ៖

char[] toCharArray();

ឧទាហរណ៍៖

```
class DemotoCharArray{
        public static void main(String[] agr){
                String str= new String("This is demo of toCharArray method.");
                char ch[];
                ch = str.toCharArray();
                for(int i=0 ; i<str.length() ; i++)
                        System.out.println(ch[i]);
        }
}
```

១.២.៧ ការប្រើ valueOf()

Method valueOf() ត្រូវបានគេប្រើបំលែងប្រភេទទិន្នន័យផ្សេងៗទៅជា String ។ វាមានទម្រង់ដូចខាងក្រោម៖

static String valueOf(double num)

static String valueOf(float num)

static String valueOf(Object ob)

static String valueOf(char chars[])

១.២.៨ ការប្រើ toLowerCase()

Method toLowerCase() ត្រូវបានគេប្រើសម្រាប់បំលែងពីទម្រង់អក្សរធំ ទៅជាអក្សរតូចទាំងអស់។

១.២.៩ ការប្រើ toUpperCase()

Method toUpperCase() ត្រូវបានគេប្រើសម្រាប់បំលែងពីទម្រង់អក្សរតូច ទៅជាអក្សរធំទាំងអស់។

ឧទាហរណ៍៖ ខាងក្រោមនេះបង្ហាញពីការប្រើ toLowerCase() និង toUpperCase()

```
class DemoUCaseLCase{
        public static void main(String[] agr){
                String strLCase = "LOWER Case".toLowerCase();
                String strUCase = "UPPER Case".toUpperCase();
                System.out.println(strLCase);
                System.out.println(strUCase);
        }
}
```

១.៣ ការប្រើអនុគមន៍ទាក់ទងនឹងការប្រៀបធៀបString

វិធី Java បានផ្តល់នូវ methods ជាច្រើនដែលអាចឲ្យយើងធ្វើការប្រៀបធៀប String និងអាចបង្កើត substring ចេញពី String មួយបាន។ substring គឺជាបំនែក String មួយនៃ String ដើម។

១.៣.១ <u>ការប្រើ equals() និងequalsIgnoreCase()</u>

គេប្រើ method equals() ដើម្បីធ្វើការប្រៀបធៀប String ពីរ ថាវាស្មើគ្នាឬក៏អត់។ វានឹងផ្តល់តម្លៃ true កាលណា String ទាំងពីរនោះមានតួអក្សរដូចគ្នាទាំងអស់ តាមលំដាប់លំដោយ បើមានការខុសតួអក្សរនៅលំដាប់ណាមួយនោះ វានឹងផ្តល់តម្លៃ false ។ ហើយ method equals() វាមានការប្រកាន់ តួអក្សរតូច និងអក្សរធំ គឺខុសគ្នា។

ទម្រង់របស់វា៖

boolean equals(String str)

str ជា object នៃ String ដែលត្រូវធ្វើការប្រៀបធៀបជាមួយនឹង String មួយទៀត។

ចំពោះ method equalsIgnoreCase() មានការប្រើប្រាស់ដូច method equals() ដែរ គឺគ្រាន់តែវាខុសគ្នាត្រង់ថា វាមិនមានលក្ខណៈខុសគ្នាទេរវាងអក្សរធំ និងអក្សរតូច។

ទម្រង់របស់វា៖

boolean equalsIgnoreCase(String str)

ឧទាហរណ៍៖

```
class DemoEqual{
        public static void main(String[] agr){
                String str1 = "Welcome CUS";
                String str2 = "Welcome CUS";
                String str3 = "Welccome CUS";
                String str4 = "WELCOME CUS";
                System.out.println(str1 +" equals "+ str2 + "=" + str1.equals(str2));
                System.out.println(str1 +" equals "+ str4 + "=" + str1.equals(str4));
                System.out.println(str1 +" equals "+ str3 + "=" + str1.equals(str3));
                System.out.println(str1 +" equalsIgnoreCase "+ str4 + "=" +
                                        str1.equalsIgnoreCase(str4));
        }
}
```

<u>ចំណាំ៖</u> រវាងការប្រើ equals() និង == មានការខុសគ្នាខាងក្រោមជាឧទាហរណ៍បង្ហាញពីការខុសគ្នារវាង equals() និង ==

```
class DemoEqual{
        public static void main(String[] agr){
                String str1 = "Testing";
                String str2 = new String(str1);
        System.out.println( str1 + " equals " + str2 + " = " + str1.equals(str2));
```

```
        System.out.println( str1 + " == " + str2 + " = " + (str1==str2));
        }
    }
```

ជាលទ្ធផលវាបាន៖

Testing equals Testing = true

Testing == Testing = false

មូលហេតុ equals វាធ្វើការប្រៀបធៀបតួអក្សរនៅក្នុង Object នៃ String ចំណែកឯ == វាធ្វើការប្រៀបធៀបថា object ទាំងពីរជា object ដូចគ្នាដែរឬទេ? ។

១.៣.២ <u>ការប្រើ regionMatches()</u>

គេប្រើ regionMatches() ដើម្បីធ្វើការប្រៀបធៀប substring នៅត្រង់ទីតាំងកំណត់មួយរបស់ String មួយ ជាមួយនឹង substring នៅត្រង់ទីតាំងកំណត់នៅក្នុង String មួយទៀត។

ទម្រង់របស់ regionMatches() មាន៖

```
boolean  regionMatches(int startIndex, String str2, int str2StartIndex, int numChars)

boolean  regionMatches(boolean ignoreCase, int startIndex, String str2, int
                        str2StartIndex, int numChars)
```

ដែល startIndex សម្រាប់កំណត់ទីតាំងចាប់ផ្តើមនៃ String ទី១ , str2 ជា String ទី២ ដែលត្រូវយកមកប្រៀបធៀប, str2StartIndex សម្រាប់កំណត់ទីតាំងចាប់ផ្តើមនៃ String ទី២, numChars ជាចំនួន ឬប្រវែងនៃ substring ដែលត្រូវធ្វើការប្រៀបធៀប និង ignoreCase បើដាក់ true មានន័យថាវាមិនប្រកាន់អក្សរតូច ឬ ធំឡើយ។

១.៣.៣ <u>ការប្រើ startsWith, endsWith()</u>

គេប្រើ method startsWith() សម្រាប់កំណត់នូវ ពាក្យ ឬ String ដែលចាប់ផ្តើមជាមួយនឹងពាក្យ ...នៅក្នុង String ដែលគេចង់រក។ ចំណែក endsWith() វិញ សម្រាប់រកពាក្យនៅក្នុង String ដែលបញ្ចប់ដោយពាក្យ...នៅក្នុង String ដែលគេចង់រក។

ទម្រង់ទូទៅរបស់ startsWith() និង endsWith()

```
boolean startsWith(String str)
boolean startsWith(String str, int startIndex)
boolean endsWith(String str)
```

ដែល str គឺជា String ដែលត្រូវផ្ទៀងផ្ទាត់រក។ startIndex សម្រាប់កំណត់ index ដែលត្រូវចាប់ផ្តើមរក។

ឧទាហរណ៍៖

```
"Football".startsWith("Foo"); //  វានឹងផ្តល់តម្លៃ true
"Football".endsWith("ball"); //  វានឹងផ្តល់តម្លៃ true
"Football".startsWith("ball",4); //  វានឹងផ្តល់តម្លៃ true
```

១.៣.៤ <u>ការប្រើ compareTo()</u>

Method compareTo() ត្រូវបានគេប្រើដើម្បីធ្វើការប្រៀបធៀបរវាង String ពីរ ថាតើ String មួយណាធំ មួយណាតូច ឬក៏ String ទាំងពីរស្មើគ្នា។

ទម្រង់ទូទៅរបស់វា៖

int compareTo(String str)

str គឺជា String ដែលត្រូវប្រៀបធៀបជាមួយ String ដែលកំពុងប្រើ។

លទ្ធផលរបស់វា៖

-បើតូចជាងសូន្យ មានន័យថា String ដែលកំពុងប្រើតូចជាង String str

-បើធំជាងសូន្យ មានន័យថា String ដែលកំពុងប្រើធំជាង String str

-បើស្មើសូន្យ មានន័យថា String ទាំងពីរស្មើគ្នា

ឧទាហរណ៍៖

```
class SortString{
        static String number[] ={"one","two" ,"three" ,"three" ,"four" ,"five"
                                  ,"six" ,"seven" ,"eight" ,"nine" ,"ten"};
        public static void main(String[] agr){
                for (int i=0;i<number.length;i++){
                        for(int j=i+1;j<number.length;j++){
                                if number[j].compareTo(number[i])<0){
                                        String temp=number[i];
                                        number[i]=number[j];
                                        number[j]=temp;
                                }
                        }
                        System.out.println(number[i]);
                }
        }
}
```

១.៤ <u>ការរុករកនៅក្នុង String</u>

String class បានផ្តល់នូវ methods ពីរ ដែលអាចឲ្យយើងរុករកនៅក្នុង String ចំពោះតួអក្សរដែលបានកំណត់ ឬ substring ណាមួយ។ methods ទាំងពីរនោះគឺ៖

-indexOf() រកតួអក្សរ ឬ substring ដែលកើតឡើងមុនគេបង្អស់នៅក្នុង String ដែលយើងចង់រក។

-lastIndexOf() រកតួអក្សរ ឬ substring ដែលកើតឡើងនៅខាងក្រោយគេបង្អស់នៅក្នុង String ដែលយើងចង់រក។

Methods ទាំងពីរនេះ វានឹងផ្តល់មកវិញនូវលេខ index ដែលតួអក្សរ ឬ substring នោះបានជួបប្រទះ តែបើសិនជារកមិនឃើញវិញ វានឹងផ្តល់តម្លៃ -1 វិញ ដែលបញ្ជាក់ថារកមិនឃើញ។

ទម្រង់របស់វាមាន៖

int indexOf(int ch)

int lastIndexOf(int ch)

int indexOf(String str)

int lastIndexOf(String str)

int indexOf(int ch, int startIndex)

int lastIndexOf(int ch , int startIndex)

int indexOf(String str, int startIndex)

int lastIndexOf(String str , int startIndex)

- ch ជាតួអក្សរដែលយើងចង់រក
- str ជា substring ដែលយើងចង់រក
- startIndex ជាចំនុចដែលយើងចាប់ផ្តើមរក។ ចំពោះ indexOf() វាចាប់ផ្តើមពីទីតាំង startIndex រហូតដល់ចុងបញ្ចប់នៃ String រីឯ lastIndexOf() វិញវាចាប់ផ្តើមពី startIndex ត្រឡប់មកក្រោយរហូតដល់ index សូន្យ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើប្រាស់ methods ទាំងពីរ៖

```
class DemoIndexOf{
    public static void main(String[] agr){
            String str=″CLASSPATH : The directory path specifying the location of ‟ +
                    ‟compiled Java class files on the local system.‟;
            System.out.println(str);
            System.out.println(‟indexOf(e)=‟+str.indexOf(‛e‛));
            System.out.println(‟lastIndexOf(e)=‟+str.lastIndexOf(‛e‛));
            System.out.println(‟indexOf(the)=‟+str.indexOf(‟the‟));
            System.out.println(‟lastIndexOf(the)=‟+str.lastIndexOf(‟the‟));
            System.out.println(‟indexOf(t,14)=‟+str.indexOf(‛t‛,14));
            System.out.println(‟lastIndexOf(t,14)=‟+str.lastIndexOf(‛t‛,60));
            System.out.println(‟indexOf(the,14)=‟+str.indexOf( ‟the‟,14));
            System.out.println(‟lastIndexOf(the,14)=‟+str.lastIndexOf( ‟the‟,60));
    }
}
```

១.៥ ការផ្លាស់ប្តូរតម្លៃនៅក្នុង String

នៅក្នុង java មានការពិបាkក្នុងការផ្លាស់ប្តូរ ឬកែប្រែនៅក្នុង String មួយ គឺវាត្រូវតែចម្លងចូលទៅក្នុង StringBuffer ឬប្រើ method មួយចំនួនដូចខាងក្រោម៖

១.៥.១ ការប្រើ substring()

Substring ត្រូវបានគេប្រើដើម្បីដកយកផ្នែកមួយនៃ String ដើម។ ដែលមានទម្រង់ដូចខាងក្រោម៖

String substring(int startIndex)

String substring(int startIndex , int endIndex)

-startIndex សម្រាប់កំណត់ចំនុចចាប់ផ្តើមនៃ substring ដែលត្រូវដកយកមក

-endIndex សម្រាប់កំណត់ថាយកមកដល់ត្រឹម endIndex នេះ

ចំពោះទម្រង់ទី១ ដែលគ្មាន endIndex មានន័យថា substring ដែលយកចេញមកចាប់ពី startIndex រហូតដល់ចុងបញ្ចប់នៃ String ដើម។

ឧទាហរណ៍៖

```
class StringReplace{
        public static void main(String[] agr){
                String original = "This is a test. This is too";
                String search = "is";
                String sub = "was";
                String result="";
                int i;
                do {
                        System.out.println(original);
                        i = original.indexOf(search);
                        if ( i != -1 ){
                                result=original.substring(0,i);
                                result=result + sub ;
                                result=result + original.substring(i + search.lengtj());
                                original=result;
                        }
                }while(i!=-1);
        }
}
```

១.៥.២ ការប្រើ concat()

គេប្រើ concat() method ដើម្បីភ្ជាប់រវាង String ពីរ។

ទម្រង់របស់វា៖

String concat(String str)

ឧទាហរណ៍៖

```
String str1 = "Welcome ";
String str2 = "CUS" ;
String str3 = str1.concat(str2);
str1=str1.concat("CUS");
```

វាដូចគ្នានឹងការសរសេរ

```
str3 = str1 + "CUS";
```

១.៥.៣ ការប្រើ replace()

Method replace() ត្រូវបានគេប្រើសម្រាប់ជំនួសឲ្យតួអក្សរណាមួយនៅក្នុង String ទាំងមូលដោយតួអក្សរណាមួយផ្សេងទៀត។

ទម្រង់ទូទៅរបស់វា៖

String replace(char original, char replacement)

ឧទាហរណ៍៖

String str = "Welcome to cus.".replace('c','i');

ពេលនោះ str = "Weliome to ius.";

១.៥.៤ ការប្រើ trim()

យើងប្រើ trim() method ដើម្បីកាត់តំរឹម space ពីខាងមុខ និងខាងក្រោយនៃ string មួយ។

ទម្រង់ទូទៅ៖

String trim()

ឧទាហរណ៍៖

String str = " Welcome to CUS ".trim();

```
import java.io.*;
class DemoTrim{
    public static void main(String[] agr){
    BufferReader br = new BufferReader(new InputStreamReader(System.in));
        String str;
        System.out.println("Please Enter 'stop' to Quit program!");
        System.out.println("Enter state : ");
        do{
            str=br.readLine();
            str = str.trim();
            if(str.equals("Illinois"))
            System.out.println("Capital is Springfiels");
            else if(str.equals("Missouri"))
            System.out.println("Capital is Jefferson City");
            else if(str.equals("California"))
            System.out.println("Capital is Sacramento");
            else if(str.equals("Washington"))
            System.out.println("Capital is Olympiaa");
        }while (!str.equals("stop"));
    }
}
```

២. អំពី StringBuffer

StringBuffer គឺជាពពួក class នៃ String ដែលផ្តល់នូវមុខងារនៃ Strings ជាច្រើន។ដូចយើងបានដឹងហើយថា String វាតាងឲ្យតួអក្សរមួយខ្សែ ហើយប្រEងរបស់វាមានកំណត់។ ហេតុនេះហើយបានជាយើងប្រើ StringBuffer ព្រោះវាតាងឲ្យតួអក្សរមួយខ្សែដែលអាចសរសេរបាន អាចពង្រីកបាន អាចឲ្យគេ

បញ្ចូលតួអក្សរ និង substring នៅផ្នែកកណ្តាល ឬខាងចុងបាន ហើយទំហំរបស់ StringBuffer វាពង្រីកទំហំដោយស្វ័យប្រវត្តិ។

Constructors របស់ StringBuffer មាន៖

StringBuffer()

StringBuffer(int size)

StringBuffer(String str)

២.១ ការប្រើ length() និង capacity()

គេប្រើ method length() ដើម្បីដឹងពីប្រវែងរបស់ StringBuffer មួយ ហើយដើម្បីដឹងពីចំណុះសរុបរបស់ (ទំហំ) របស់ StringBuffer យើងត្រូវប្រើ method capacity() ។

ទម្រង់ទូទៅ៖

int length()

int capacity()

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើប្រាស់ Methods ទាំងពីរ៖

```
class DemoStringBuffer{
    public static void main(String[] agr){
        StringBuffer sb=new StringBuffer("Welcome StringBuffer");
        System.out.println("buffer : "+ sb);
        System.out.println("length : "+ sb.length());
        System.out.println("capacity : "+ sb.capacity());
    }
}
```

២.២ ការប្រើ ensureCapacity()

បើសិនជាយើងចង់ត្រៀមលក្ខណៈបង្កើតទីតាំងនូវទំហំមួយតាមចំនួនតួរអក្សរ បន្ទាប់ពី String Buffer បានបង្កើតឡើងហើយនោះ យើងត្រូវប្រើ ensureCapacity() method ។ វាមានសារៈសំខាន់ បើយើងចង់ដឹងមុនថាគេនឹងបន្ថែមនូវ string តូចៗជាច្រើនទៅឲ្យ StringBuffer ។

ទម្រង់ទូទៅ៖

void ensureCapacity(int capacity)

២.៣ ការប្រើ setLength()

គេប្រើ method setLength() ដើម្បីកំណត់ប្រវែង buffer នៅក្នុង Object StringBuffer។

ទម្រង់ទូទៅ៖

void setLength(int len)

កាលណាយើងបង្កើនទំហំ buffer តួអក្សរ null ត្រូវបន្ថែមនៅខាងចុង buffer ដែលមានរួចហើយ។ កាលណាបើយើង setLength() របស់ buffer មួយមានតម្លៃតូចជាង length ដែលកំពុងប្រើ ពេលនោះវានឹងបាត់បង់តួអក្សរ។

២.៤ ការប្រើ charAt() និង setCharAt()

គេប្រើ method charAt() ដើម្បីទទួលបាននូវតួអក្សរណាមួយ នៅ index ណាមួយ។ យើងក៏អាចកំណត់តួអក្សរណាមួយទៅឲ្យ StringBuffer នៅទីតាំងណាមួយបានដែរ ដោយយើងប្រើ method setCharAt() ។

ទម្រង់ទូទៅ៖

char charAt(int where)

void setCharAt(int where , char ch)

ដែល where ត្រូវតែជាចំនួនវិជ្ជមាន។

ឧទាហរណ៍៖

```
class DemoCharAt{
        public static void main(String[] agr){
                StringBuffer sb=new StringBuffer("Welcome CUS.");
                System.out.println("buffer before : " + sb);
                System.out.println("charAt(1) : " + sb.charAt(1));
                sb.setCharAt(1,'Y');
                sb.setLength(5);
                System.out.println("buffer after : " + sb);
                System.out.println("charAt(1) : " + sb.charAt(1));
        }
}
```

២.៥ ការប្រើ getChars()

គេប្រើ method getChars() ដើម្បីចម្លងទិន្នន័យនៅក្នុង StringBuffer មួយទៅដាក់ក្នុង array មួយ។ វាមានទម្រង់ទូទៅ៖

void getChars(int sourceStart, int sourceEnd,char target[], int targetStart);

២.៦ ការប្រើ append()

យើងប្រើ method append() សម្រាប់ភ្ជាប់តំណាង string នៃប្រភេទទិន្នន័យណាមួយទៅខាងចុង object នៃ StringBuffer ។

ទម្រង់ទូទៅរបស់វា៖

StringBuffer append(String str)

StringBuffer append(int num)

StringBuffer append(Object obj)

ឧទាហរណ៍៖

```
class DemoAppend{
        public static void main(String[] agr){
                String str;
                int x = 54;
                StringBuffer sb = new StringBuffer(65);
                str=sb.append("x = ").append(x).append("!").toString();
```

```
            System.out.println(str);
        }
    }
```

២.៧ ការប្រើ insert()

Method insert() ប្រើសម្រាប់បញ្ចូល String មួយទៅក្នុង String មួយទៀត។ ដូចទៅនឹង method append() ដែរ វាហៅ String.valueOf() មកប្រើ ដើម្បីទទួលបាននូវ string តំណាងឲ្យតម្លៃដែលបានប្រើជាមួយនឹងវានោះ។ ពេលនោះ string នេះត្រូវបានបញ្ចូលទៅក្នុង objects នៃ StringBuffer ។

ទម្រង់ទូទៅ៖

StringBuffer insert(int index, String str)

StringBuffer insert(int index, char ch)

StringBuffer insert(int index, Object obj)

ត្រង់ index បញ្ជាក់ពីទីតាំងដែលត្រូវបញ្ចូល String ទៅក្នុង object នៃ StringBuffer ។

```
class DemoInsert{
    public static void main(String[] agr){
        StringBuffer sb = new StringBuffer("I java programming!")
        sb.insert(2, " like ");
        System.out.println(sb);
    }
}
```

២.៨ ការប្រើ reverse()

យើងប្រើ method reverse() ដើម្បីត្រឡប់តួអក្សរនៅក្នុង String មួយ។

ទម្រង់របស់វា៖

StringBuffer reverse()

ឧទាហរណ៍៖

```
class  DemoReverse{
    public static void main(String[] agr){
        StringBuffer sb = new StringBuffer("esreveR");
        System.out.println(sb);
        sb.reverse();
        System.out.println(sb);
    }
}
```

២.៩ ការប្រើ delete() និង deleteCharAt()

ទាំង delete() និង deleteCharAt() គឺយើងប្រើសម្រាប់លុបតួអក្សរនៅក្នុង StringBuffer។

ទម្រង់ទូទៅ៖

StringBuffer delete(int startIndex, int endIndex)

StringBuffer deleteCharAt(int loc)

ឧទាហរណ៍៖

```
class DemoDelete{
    public static void main(String[] agr){
        StringBuffer sb=new StringBuffer("Welcome CUS!");
        System.out.println("Before delete :" + sb);
        sb.delete(4,7);
        System.out.println("After delete (4,7) :" + sb);
        sb.deleteCharAt(2);
        System.out.println("After deleteCharAt(2) :" + sb);
    }
}
```

២.១០ ការប្រើ replace()

យើងប្រើ method replace() ដើម្បីជំនួសតួអក្សរ ឬ substring មួយដោយ តួអក្សរ ឬ substring ថ្មី នៅក្នុង StringBuffer។

ទម្រង់ទូទៅ៖

StringBuffer replace(int startIndex, int endIndex, String str)

ឧទាហរណ៍៖

```
class DemoReplace{
    public static void main(String[] agr){
        StringBuffer sb = new StringBuffer(""Welcome CUS!");
        System.out.println("Before replace: "+sb);
        sb.replace(8,10,"SOK");
        System.out.println("After replace: "+sb);
    }
}
```

២.១១ ការប្រើ substring()

យើងក៏អាចប្រើ substring() ជាមួយ StringBuffer ដូចនឹងការប្រើជាមួយ String ដែរ។

៣. StringTokenizer

ដំណើរការជារឿយៗមានការវិភាគទ្រង់ទ្រាយអក្សរដែលបានបញ្ចូល ការវិភាគគឺបំបែកអក្សរទៅជាសំណុំបំណែកតូចៗដាច់ពីគ្នា ឬ tokens ដែលលំដាប់អក្សរអាចធ្វើឲ្យមានអត្ថន័យទៅតាមចំណង់របស់គេ ។ StringTokenizer class បានផ្តល់នូវជំហានទី១ ក្នុងការវិភាគនេះហៅថា lexer(lexical analyzer) ឬ scanner ។ StringTokenizer អនុវត្តតាម Enumeration interface។ ហេតុនេះ អក្សរដែលយើងវាយបញ្ចូលនោះ គេអាចធ្វើការរាប់ tokens ដោយឡែកៗពីគ្នា ដែលមានក្នុងវាដោយប្រើ StringTokenizer។

ដើម្បីប្រើប្រាស់ StringTokenizer គេត្រូវពង្រាក់ពីអក្សរដែលត្រូវបញ្ចូល និងអក្សរដែលមាន delimiters (ដែនកំណត់) ជាតួអក្សរដែលខ័ណ្ឌ tokens ។ តួអក្សរនីមួយៗនៅក្នុង delimiters ត្រូវគិតដែនកំណត់ឲ្យបានត្រឹមត្រូវ ឧទាហរណ៍ ",;:" កំណត់ delimiter នូវសញ្ញា (,) សញ្ញា (;) និងសញ្ញា (:)។ តាមធម្មតាការកំណត់ delimiters មានតួអក្សរដូចជា space , tab, newline និង carriage return ជាដើម។

ទម្រង់ constructors របស់ StringTokenizer ៖

StringTokenizer(String str)

StringTokenizer(String str, String delimiters)

StringTokenizer(String str,String delimiters, boolean delimAsToken)

- str ជា String ដែលត្រូវបំបែក
- delimAsToken បើយើងដាក់ true នោះ delimiters ត្រូវឲ្យតម្លៃជា tokens ដែរនៅពេល String ត្រូវបានវិភាគ។ ផ្ទុយទៅវិញ delimiter មិនឲ្យតម្លៃទេ។

យើងប្រើ method nextToken() ដើម្បីទាញយក tokens តគ្នា។ method hasMoreToken() ប្រើសម្រាប់ចង់ដឹងថា វាមាន tokens សម្រាប់ទាញយកទៀត ឬក៏អត់ វានឹង return true ប្រសិនបើមាន។ ដោយសារ StringTokenizer អនុវត្តតាម Enumeration ពេលនេះ method hasMoreElements() និង nextElement() ក៏ត្រូវបានប្រើដែរ ហើយវាអនុវត្តដូចគ្នាទៅនឹង hasMoreTokens() និង nextToken() ដែរ។

ខាងក្រោមនេះជាឧទាហរណ៍មួយបង្កើតនូវ StringTokenizer សម្រាប់ញែក"key = value " ជាគូ។ សំណុំនៃគូ "key = value " តគ្នាត្រូវញែកដោយសញ្ញា (;)។

```
import java.util.StringTokenizer;
class DemoStringTokenizer{
        static String str = "title=Java: The Complete Reference;"+
                            "author=Naughton and Schildt; "+
                            "publisher=Osborne/McGraw-Hill;"+
                            "copyright=1999";
        public static void main(String[] agr){
                StringTokenizer st = new StringTokenizer(str,"=;");
                while(st.hasMoreTokens()){
                        String key =st.nextToken();
                        String value=st.nextToken();
                        System.out.println(key + "\\t"+ value);
                }
        }
}
```

លទ្ធផលដែលទទួលបាន៖

```
title Java: The Complete Reference
author Naughton and Schildt
publisher Osborne/McGraw-Hill
copyright 1999
```

ខាងក្រោមនេះជាតារាងបង្ហាញពី method របស់ StringTokenizer

Method Name	អត្ថន័យ
int countTokens()	ផ្តល់នូវចំនួន token
boolean hasMoreTokens()	ឲ្យតម្លៃ true បើសិនជាមាន token មួយឬច្រើនក្នុងString
boolean hasMoreElements()	ឲ្យតម្លៃ true បើសិនជាមាន token មួយឬច្រើនក្នុងString
Object nextElement()	ផ្តល់តម្លៃ token នៅបន្ទាប់ជា object មួយ
String nextToken()	ផ្តល់តម្លៃ token នៅបន្ទាប់ជា String មួយ
String nextToken(String delimiters)	ឲ្យតម្លៃ token នៅបន្ទាប់ជា String មួយ ហើយកំណត់ delimiters string ទៅឲ្យ delimiters នោះ

៥. Collection

ពេលនេះយើងគួរតែស្គាល់ពី collection interfaces ហើយនិងត្រៀមលក្ខណៈដើម្បីពិនិត្យមើលថា classes ធម្មតានិងការប្រើ collection Interfaces ។ class មួយចំនួនផ្តល់នូវការអនុវត្តពេញលេញ ដែលអាចប្រើបានតែម្តង ឯ classes មួយចំនួនទៀតមានលក្ខណៈជា abstract class ផ្តល់នូវការអនុវត្តជាគម្រោងដែលត្រូវបានប្រើជាចំនុចចាប់ផ្តើមសម្រាប់បង្កើតពពួក collections ដែលគ្មានលក្ខណៈជា abstract class នោះ។ វាពុំមាន collection classes ដែលមានលក្ខណៈជា synchronized ឡើយ ប៉ុន្តែយើងនឹងឃើញនៅពេលក្រោយគឺវាអាចមានប្រើជាលក្ខណៈ synchronized ។

ឈ្មោះរបស់ Class	ការអធិប្បាយ
AbstractCollection	ភាគច្រើនប្រើ Collection interface
AbstractList	ទទួលលក្ខណៈពី AbstractCollection ហើយច្រើនប្រើ List interface
AbstractSequentialList	ទទួលលក្ខណៈពី AbstractList សម្រាប់ប្រើជាមួយcollection ដែលប្រើលក្ខណៈបន្តគ្នា គ្រាន់ជាងប្រើលក្ខណៈមិនតាមលំដាប់លំដោយនៃធាតុរបស់វា
LinkedList	ប្រើ linked list ដោយទទួលលក្ខណៈពី AbstractSequentialList
ArrayList	ប្រើ dynamic array ដោយទទួលលក្ខណៈពី AbstractList
AbstractSet	ទទួលលក្ខណៈពី AbstractCollection និងភាគច្រើនប្រើ Set interface
HashSet	ទទួលលក្ខណៈពី AbstractSet សម្រាប់ប្រើ hash table
TreeSet	ទទួលលក្ខណៈពី AbstractSet ដែលប្រើ set ផ្ទុកក្នុង tree មួយ

៥.១ ArrayList

ArrayList class ទទួលលក្ខណៈពី AbstractList ហើយប្រើ List interface ។ ArrayList ផ្តល់នូវ dynamic arrays ដែលអាចពង្រីកតាមតម្រូវការ ព្រោះក្នុង java ជាធម្មតាគឺមានប្រវែងថេរ។ បន្ទាប់ពី array ត្រូវបានបង្កើតឡើងនោះ គេមិនអាចពង្រីក ឬបង្រួមបានទេ ហើយគេត្រូវតែដឹងជាមុននូវចំនួនធាតុដែលត្រូវផ្ទុកក្នុង array។ ប៉ុន្តែពេលខ្លះគេមិនអាចដឹងពីទំហំរបស់ array បានទេ ដើម្បីប្រើសភាពបែបនេះគម្រោងការ collection បានកំណត់នូវ ArrayList ។ ArrayList គឺជាអញ្ជាត array មួយដែលមានប្រវែងនៃ

object references វាអាចបន្ថែម ឬបន្ថយបានតាមតម្រូវការ។ ArrayList ពេលបង្កើតត្រូវតែកំណត់ទំហំតំបូង ហើយនៅពេលដែលយើងប្រើហួសវានឹងបង្កើតដោយស្វ័យប្រវត្ត ហើយនៅពេលយើងលុបធាតុនោះ វានឹងបន្ថយទំហំមកវិញ។

Constructor របស់ ArrayList មានដូចខាងក្រោម៖

ArrayList()

ArrayList(Collection c)

ArrayList(int capacity)

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើប្រាស់ ArrayList ៖

```
import java.util.*;
class DemoArrayList{
	public static void main(String[] agr){
		//Create an array list
		ArrayList al=new ArrayList();
		System.out.println("Initial size of al : "+al.size());
	//add element to array list
		al.add("C");
		al.add("A");
		al.add("M");
		al.add("B");
		al.add("O");
		al.add("D");
		al.add("I");
		al.add("A");
		al.add(1,"C1");
		System.out.println("Size of al after add Element :" + al.size());
		//display the array list
		System.out.println("Contents of al :" +al);
		//to remove elements from the array list
		al.remove("A");
		al.remove(2);
		System.out.println("Size of al after deleted : " + al.size());
		System.out.println("Contents of al :" + al);
	}
}
```

ជាលទ្ធផលយើងទទួលបាន៖

```
Initial size of al : 0
Size of al after add Element : 9
Contents of al :[C, C1, A, M, B, O, D, I, A]
```

```
Size of al after deleted : 7
Contents of al :[C, C1, B, O, D, I, A]
```

ទោះបីជាចំណុះរបស់ ArrayList object កើនឡើងដោយស្វ័យប្រវត្តតាមដែល objects មានក្នុងវាក៏ដោយ គេអាចបង្កើនចំណុះ ArrayList តាមលក្ខណៈធម្មតាដោយហៅ ensureCapacity() method មកប្រើ។ គេអាចធ្វើបែបនេះ នៅពេលដែលគេដឹងជាមុននូវអ្វីដែលគេត្រូវផ្ទុកនូវចំនួន items ក្នុង collection ដែលកំពុងមាន។ តាមរយៈការបង្កើននូវចំណុះម្តងៗនៅត្រង់ចំនុចចាប់ផ្តើម គេអាចរក្សាទុកនូវទីតាំងជាច្រើនដែលប្រើជាថ្មីសម្រាប់ពេលក្រោយ។

ទម្រង់របស់ensureCapacity()៖

void ensureCapacity(int cap)

ដែល cap ជាចំណុះថ្មី។

ប្រសិនបើយើងចង់បន្តយទំហំរបស់ array list ទៅតាមចំនួនរបស់ items ដែលវាកំពុងមាន យើងត្រូវប្រើ method trimToSize()។

ទម្រង់របស់ trimToSize()៖

void trimToSize()

៥.២ <u>ដើម្បីទទួល Array មួយពី ArrayList</u>

គេអាចបំលែងពី ArrayList ទៅជា Array មួយយ៉ាងពិតប្រាកដ ដោយប្រើ method toArray() ។ ហេតុផលមួយចំនួនដែលគេចង់បំលែងពី collection ទៅជា Array គឺ៖

-ដើម្បីទទួលបានរយៈពេលដំណើរការលឿនចំពោះការប្រតិបត្តិពិតប្រាកដ

-ដើម្បីបញ្ជូន array មួយទៅឲ្យ method ដែលមិនបានធ្វើ overloaded សម្រាប់ទទួលយកនូវ collection មួយ។

-ដើម្បីបញ្ចូលនូវ code ដែលមានមូលដ្ឋានលើ collection ហើយមានសភាពថ្មីទៅលើ code ចាស់ដែលវាមិនទាន់ស្គាល់នូវ collection ។

ឧទាហរណ៍៖

```
import java.util.*;
class ArrayListToArray{
    public static void main(String[] agr){
        //Create an array list
        ArrayList al=new ArrayList();
        //add element to array list
        al.add(new Integer(1));
        al.add(new Integer(2));
        al.add(new Integer(3));
        al.add(new Integer(4));
        System.out.println("Contents of al : "+ al);
        //get array
```

```
                Object ia[]=al.toArray();
                int sum=0;
                //sum the array
                for(int i=0; i<ia.length;i++)
                        sum+=((Integer) ia[i]).intValue();
                System.out.println("Sum is = " + sum);
                }
        }
```

ជាលទ្ធផលយើងទទួលបាន៖

```
Contents of al : [1, 2, 3, 4]
Sum is = 10
```

៥.៣ អំពី LinkList

LinkedList class វាទទួលលក្ខណៈពី AbstractSequentialList ហើយអនុវត្តន៍ List interface ។ ទម្រង់ Constructor របស់វា៖

LinkedList()

LinkedList(Collection c)

ខាងក្រោមនេះជា methods មួយចំនួនដែលប្រើជាមួយ LinkedList ៖

```
void add(Object obj)
void add(int index , Object obj)
void addFirst(Object obj)
void addLast(Object obj)
void set(Object obj)
Object get(int index)
Object getFirst()
Object getLast()
Object removeFirst()
Object removeLast()
```

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើ LinkedList ៖

```
import java.util.*;
class DemoLinkedList{
        public static void main(String[] agr){
                LinkedList ll = new LinkedList();
                ll.add("H");
                ll.add("E");
                ll.add("L");
                ll.add("L");
                ll.add("O");
```

```
                ll.addLast(" ");
                ll.addLast("Z");
                ll.addFirst ("I");
                ll.add(1," ");
                System.out.println("Original contents of ll : "+ll);
                ll.remove("Z");
                ll.remove(1);
                System.out.println("Contents of ll after deleted : "+ll);
                ll.removeFirst();
                ll.removeLast();
                System.out.println("Contents of ll after removeFirst and Last: "+ll);
                Object val=ll.get(2);
                ll.set(2,(String) val + "Changed");
                System.out.println("ll after changed : " + ll);
        }
}
```

លទ្ធផលដែលទទួលបាន៖

```
Original contents of ll : [I,  , H, E, L, L, O,  , Z]
Contents of ll after deleted : [I, H, E, L, L, O,  ]
Contents of ll after removeFirst and Last: [H, E, L, L, O]
ll after changed : [H, E, LChanged, L, O]
```

៥.៤ ការចូលប្រើ Collection មួយតាមរយៈ Iterator

Iterator អាចឲ្យយើងធ្វើសកម្មភាពដដែលៗឆ្លងកាត់ collection មួយក្នុងការទទួលធាតុ ឬលុបធាតុណាមួយ។ ListIterator ទទួលលក្ខណៈពី Iterator ដែលអាចឲ្យឆ្លងកាត់ list តាមទិសដៅ ហើយធ្វើការកែប្រែធាតុ។

ខាងក្រោមជា methods របស់ Iterator interface

Method Name	អធិប្បាយ
boolean hasNext()	ផ្តល់នូវតម្លៃ true ប្រសិនបើនៅមានធាតុទៀត
Object next()	ឲ្យតម្លៃជាធាតុដែលនៅបន្ទាប់ ប្រើ NoSuchElementException បើសិនជាវាពុំមានធាតុនៅបន្ទាប់
void remove()	លុបនូវធាតុដែលកំពុងប្រើ ប្រើ IllegalStateException បើសិនជាវាមានការព្យាយាមហៅ remove() មកប្រើដែលមិនមានធាតុនៅខាងមុខ និងហៅ next() មកប្រើ

ខាងក្រោមជា methods របស់ ListIterator

Method Name	អធិប្បាយ
void add(Object obj)	បញ្ចូល object ទៅក្នុងlist ខាងមុខធាតុដែលនឹងឲ្យតម្លៃតាមរយៈ

	ការហៅ nex() មកប្រើ។
boolean hasNext()	ឲ្យតម្លៃ true កាលណាមានធាតុនៅបន្ទាប់
Boolean hasPrevious()	ឲ្យតម្លៃ true កាលណាមានធាតុនៅខាងមុខ
Object next()	ឲ្យតម្លៃជាធាតុដែលនៅបន្ទាប់ ប្រើ NoSuchElementException បើសិនជាវាពុំមានធាតុនៅបន្ទាប់
int nextIndex()	ឲ្យតម្លៃជា index នៃធាតុដែលនៅបន្ទាប់ បើសិនវាពុំមានធាតុនៅបន្ទាប់ទេ នោះវាឲ្យទំហំនៃ List
Object previous()	ឲ្យតម្លៃជាធាតុនៅខាងមុខ ប្រើ NoSuchElementException បើសិនជាវាពុំមានធាតុនៅមុខទេ
int previousIndex()	ឲ្យតម្លៃជា index នៃធាតុដែលនៅខាងមុខបើសិនវាពុំមានធាតុនៅខាងមុខទេនោះវាឲ្យតម្លៃ -1
void remove()	លុបនូវធាតុដែលកំពុងប្រើ ប្រើ IllegalStateException បើសិនជាវាមានការព្យាយាមហៅ remove() មកប្រើមុខ next() ឬ previous() ត្រូវបានប្រើ
void set(Object obj)	ផ្តល់object ទៅឲ្យធាតុដែលកំពុងប្រើ វាជាធាតុចុងក្រោយឲ្យតម្លៃដោយហៅ next() ឬ previous() មកប្រើ

៤.៥ ការប្រើ Iteractor

មុននឹងយើងអាចចូលប្រើ collection មួយតាមរយៈ iterator យើងត្រូវតែមាន collection class មួយសិន ដោយវាផ្តល់នូវ iterator() method ដែលអាចឲ្យតម្លៃជា iterator មួយចំពោះការចាប់ផ្តើម collection។ ជាទូទៅដើម្បីប្រើ iterator មួយសម្រាប់ដំណើរការដដែលៗតាមរយៈតម្លៃនៃ collection ដោយអនុវត្តតាមជំហានដូចខាងក្រោម៖

១- ទទួលបាន iterator មួយសម្រាប់ចាប់ផ្តើមនៃ collection តាមរយៈនៃការហៅ iterator របស់ collection មកប្រើ

២- ប្រើ loop ដែលមានការប្រើ hasNext() រហូតដល់ hasNext() មានតម្លៃ false

៣- នៅក្នុង loop យើងទទួលបានធាតុនីមួយៗតាមរយៈការហៅ next() មកប្រើ

ចំពោះ collection ដែលប្រើ List យើងអាចមាន iterator មួយដោយហៅ ListIterator មកប្រើ។ ដូចបានពន្យល់រួចមកហើយថា list iterator ផ្តល់នូវលទ្ធភាពក្នុងការចូលប្រើ collection ទាំងទិសដៅទៅមុខ និងទៅក្រោយបាន ថែមទាំងអាចកែប្រែធាតុបានទៀតផង។

ឧទាហរណ៍៖

```
import java.util.*;
class DemoIterator{
        public static void main(String[] agr){
                // create array list
                ArrayList al=new ArrayList();
                // add element to array list
```

```
                al.add("C");
                al.add("A");
                al.add("M");
                al.add("B");
                al.add("O");
                al.add("D");
                al.add("I");
                al.add("A");
                //use iterator to Display contents of array list
                System.out.print("Original contents of al : ");
                Iterator itr = al.iterator();
                while(itr.hasNext()){
                        Object element = itr.next();
                        System.out.print(element+ " ");
                }
                System.out.println();
                // modify Object being iterated
                ListIterator litr = al.listIterator();
                while(litr.hasNext()){
                        Object element=litr.next();
                        litr.set(element + "+");
                }
                System.out.print("Midified contents of al : ");
                itr = al.iterator();
                while(itr.hasNext()){
                        Object element=itr.next();
                        System.out.print(element+ " ");
                }
                System.out.println();
                //now display the list backwards
                System.out.print("Modified list backwards : ");
                while(litr.hasPrevious()){
                        Object element = litr.previous();
                        System.out.print(element+ " ");
                }
                System.out.println();
        }
}
```

លទ្ធផលទទួលបាន៖

Original contents of al : C A M B O D I A

```
Midified contents of al : C+ A+ M+ B+ O+ D+ I+ A+
Modified list backwards : A+ I+ D+ O+ B+ M+ A+ C+
```

៥.៦ អំពី Comparator

ទាំង TreeSet និង TreeMap សុទ្ធតែផ្ទុកធាតុតាមលំដាប់ វាជា comparator ដែលបង្កើតនូវមធ្យោបាយ "រៀបតាមលំដាប់" យ៉ាងជាក់លាក់ ។ បើសិនជាយើងចង់រៀបតាមលំដាប់ផ្សេងវិញ ពេលនោះយើងត្រូវកំណត់នូវ object នៃ comparator នៅពេលដែលយើងបង្កើតនូវសំណុំ ឬ map ។ ការធ្វើបែបនេះផ្តល់ឲ្យយើងនូវលទ្ធភាពដើម្បីដឹកនាំទៅរកវិធីផ្ទុកធាតុក្នុង collection ដែលបានរៀប និង map។

Comparator interface មាន methods ពីរគឺ compare() និង equals()។

ទម្រង់ទូទៅ៖

int compare(Object obj1,Object obj2)

obj1 និង obj2 គឺជា Object ដែលត្រូវធ្វើការប្រៀបធៀប។ វាផ្តល់តម្លៃ 0 កាលណា Object ទាំងពីរស្មើគ្នា, តម្លៃអវិជ្ជមាន កាលណា obj1 ធំជាង obj2 និងវិជ្ជមានកាលណា obj2 ធំជាង obj1 វិញ។ method នេះយើងអាចប្រើ ClassCastException បើប្រភេទ Object ទាំងពីរមិនត្រូវគ្នាក្នុងការប្រៀបធៀប

Method equals() វាធ្វើការប្រៀបធៀប object មួយស្មើនឹងcomparator ដែលកំពុងប្រើឬទេ។

boolean equals(Object obj)

វាផ្តល់តម្លៃ true បើសិនជាobj និង object ដែលកំពុងប្រើគឺជា Comparator object និងប្រើលំដាប់ដូចគ្នា។

ឧទាហរណ៍៖

```
import java.util.*;
class MyCompare implements Comparator{
    public int compare(Object obj1,Object obj2){
        String str1,str2;
        str1=(String) obj1;
        str2=(String) obj2;
        return str2.compareTo(str1);
    }
}
class DemoCompare{
    public static void main(String[] agr){
        TreeSet ts=new TreeSet(new MyCompare());
        ts.add("C");
        ts.add("A");
        ts.add("M");
        ts.add("B");
        ts.add("O");
        ts.add("D");
        ts.add("I");
```

```
                ts.add("A");
                //get an iterator
                Iterator itr = ts.iterator();
                while(itr.hasNext()){
                        Object element = itr.next();
                        System.out.print(element + " ");
                }
                System.out.println();
        }
}
```

លទ្ធផល៖

O M I D C B A

សូមពិនិត្យឲ្យបានល្អិតល្អន់ទៅលើ MyCompare ដែលប្រើ Comparator និង Overridden ទៅលើ method compare() ។ ដែលក្នុងនោះវាមានប្រើ method compareTo() របស់ String Class ដើម្បីធ្វើការប្រៀបធៀប String ពីរ ដោយយើងធ្វើការត្រឡប់ដោយយក parameter ទី២ មក compareTo() នឹង parameter ទី១ ជាហេតុធ្វើឲ្យវាតំរៀបបញ្ច្រាសមកវិញ។

កម្មវិធីខាងក្រោមបង្ហាញភាពជាក់ស្តែង របស់កម្មវិធី TreeMap ពីផ្នែកខាងដើមដែលផ្តុកទឹកប្រាក់បញ្ញើរ។ នៅក្នុងកំណែថ្មីមុននេះគណនីត្រូវរៀបតាមឈ្មោះ ក៏ប៉ុន្តែការរៀបតាមលំដាប់ចាប់ផ្តើមដោយនាមខ្លួន ។ កម្មវិធីខាងក្រោមនេះរៀបលំដាប់នូវ accounts តាមនាមត្រកូល។ ដើម្បីធ្វើបែបនេះ យើងប្រើ comparator ធ្វើការប្រៀបធៀបនាមត្រកូលរបស់ accounts នីមួយៗ។

ឧទាហរណ៍៖

```
//user a comparator to sort accounts by last name
import java.util.*;
//compare last whole words in two strings
class TreeMapCom implements Comparator{
        //Overide compare() method
        public int compare(Object obj1,Object obj2){
                int i,j,k;
                String str1,str2;
                str1=(String)obj1;
                str2=(String)obj2;
                //find index of beginning of last name
                i=str1.lastIndexOf(' ');
                j=str2.lastIndexOf(' ');
                k=str1.substring(i).compareTo(str2.substring(j));
                if(k==0) // last name match, check entire name
                        return str1.compareTo(str2);
                else
                        return k;
        }
```

```
        //no need to overide equals() method
}
class DemoTreeMap{
        public static void main(String[] agr){
                //create tree map
                TreeMap tm=new TreeMap(new TreeMapCom());
                //put element to the map
                tm.put("Choeun Net",new Double(3456.43));
                tm.put("Lim Navuth",new Double(5432.12));
                tm.put("Lim Sothea",new Double(9876.54));
                tm.put("Phan Thenghay",new Double(8765.43));
                tm.put("Lai Chansok",new Double(7654.32));
                tm.put("Srorn Seiha",new Double(9753.13));
                tm.put("You Hengadom",new Double(7531.31));
                //get a set of the entires
                Set set=tm.entrySet();
                //get an iterator
                Iterator itr = set.iterator();
                //display elements
                while(itr.hasNext()){
                        Map.Entry me=(Map.Entry)itr.next();
                        System.out.print(me.getKey() + ": ");
                        System.out.println(me.getValue());
                }
                System.out.println();
                //deposit 1111 into Srorn Seiha's account
                double balance =((Double)tm.get("Srorn Seiha")).doubleValue();
                tm.put("Srorn Seiha",new Double(balance + 1111));
                System.out.println("Srorn Seiha's new balance" + tm.get("Srorn Seiha"));
        }
}
```

លទ្ធផលទទួលបាន៖

```
Lai Chansok: 7654.32
You Hengadom: 7531.31
Lim Navuth: 5432.12
Choeun Net: 3456.43
Srorn Seiha: 9753.13
Lim Sothea: 9876.54
Phan Thenghay: 8765.43
Srorn Seiha's new balance10864.13
```

៩.៧ អំពី Arrays

Java 2 បានបន្ថែម class ថ្មីមួយទៀតទៅក្នុង java.util គឺ Arrays class វាផ្តល់នូវ methods ផ្សេងៗ ដែលមានសារៈសំខាន់នៅពេលយើងធ្វើការជាមួយ Arrays។ ទោះបីជា methods ទាំងនេះមិនមែនជាផ្នែកមួយនៃពពួក collection framework ក៏ដោយ ប៉ុន្តែវាជាស្ពានចម្លងចន្លោះរវាងពពួក collections និង array។

Methods ដែលកំណត់ដោយ Arrays class មាន៖

- Method asList() ឲ្យតម្លៃជា List ដែលត្រូវបានស្គាល់ដោយ Array ។ គេនិយាយម្យ៉ាងទៀតថា List និង Array បញ្ជាក់ឲ្យទីតាំងដូចគ្នា។ទម្រង់របស់វា៖

static List asList(Object[] array) // array គឺជាarray ផ្ទុកទិន្នន័យ

- Method binarySearch() ប្រើសម្រាប់តម្លៃដែលបានកំណត់ វាត្រូវបានប្រើជាមួយ array ដែលបានតំរៀបរួចជាស្រេច។ វានឹងផ្តល់នូវលេខ index នៃធាតុដែលយើង search ឃើញ។ តែបើsearch មិនឃើញវានឹងផ្តល់តម្លៃអវិជ្ជមាន។

ទម្រង់ទូទៅរបស់វា៖

static int binarySearch(byte[] array , byte value)
static int binarySearch(char[] array , char value)
static int binarySearch(double[] array , double value)
static int binarySearch(float[] array , float value)
static int binarySearch(int[] array , int value)
static int binarySearch(long[] array , long value)
static int binarySearch(short[] array , short value)
static int binarySearch(Object[] array , Object value)
static int binarySearch(Object[] array , Object value, Comparator c)

- Method equals() ប្រើសម្រាប់ប្រៀបធៀប array ពីរ បើសិនជា Array ទាំងពីរស្មើគ្នា វានឹងផ្តល់តម្លៃ true ផ្ទុយទៅវិញ វានឹងផ្តល់តម្លៃ false។

static boolean equals(boolean array1[] , boolean array2[])
static boolean equals(byte array1[] , byte array2[])
static boolean equals(char array1[] , char array2[])
static boolean equals(double array1[] , double array2[])
static boolean equals(float array1[] , float array2[])
static boolean equals(int array1[] , int array2[])
static boolean equals(long array1[] ,long array2[])
static boolean equals(short array1[] , short array2[])
static boolean equals(Object array1[] , Object array2[])

- Method fill() ប្រើសម្រាប់កំណត់តម្លៃមួយទៅឲ្យធាតុទាំងអស់នៅក្នុង array ។ យើងអាចនិយាយម្យ៉ាងទៀតថា វាបំពេញឲ្យ array នូវតម្លៃកំណត់មួយ ។ fill() មានពីរទម្រង់

ទម្រង់ទី១៖

```
static void fill(boolean[] array , boolean value)
static void fill(byte[] array , byte value)
static void fill(char[] array , char value)
static void fill(double[] array , double value)
static void fill(float[] array , float value)
static void fill(int[] array , int value)
static void fill(long[] array , long value)
static void fill(short[] array ,short value)
static void fill(Object[] array , Object value)
```

ទម្រង់ទី២៖

```
static void fill(boolean[] array ,int start, int end, boolean value)
static void fill(byte[] array , int start, int end, byte value)
static void fill(char[] array , int start, int end, char value)
static void fill(double[] array , int start, int end, double value)
static void fill(float[] array , int start, int end,float value)
static void fill(int[] array , int start, int end, int value)
static void fill(long[] array , int start, int end, long value)
static void fill(short[] array , int start, int end, short value)
static void fill(Object[] array , int start, int end, Object value)
```

methods ទាំងអស់នេះអាចប្រើជាមួយ Exception ប្រភេទ IllegalArgumentException នៅពេលដែល start ធំជាង end ឬក៏ ArrayIndexOutOfBoundsException កាលណា start ឬ end នៅក្រៅដែនកំណត់។

- Method sort() ត្រូវបានគេប្រើសម្រាប់តំរៀបទិន្នន័យក្នុង array តាមលំដាប់កើន។ វាមានពីរ ទម្រង់។

ទម្រង់ទី១៖ រៀបតាមលំដាប់ក្នុង array ទាំងមូល

```
static void sort(byte[] array)
static void sort(char[] array)
static void sort(double[] array)
static void sort(float[] array)
static void sort(int[] array)
static void sort(long[] array)
static void sort(short[] array)
static void sort(Object[] array)
```

static void sort(Object[] array, Comparator c)

ទម្រង់ទី២៖ រៀបតាមលំដាប់ក្នុង array ចន្លោះ index ពី start ដល់ end-1

methods ទាំងអស់នេះអាចប្រើជាមួយ Exception ប្រភេទ **static void sort(byte[] array, int start, int end)**

static void sort(char[] array, int start, int end)

static void sort(double[] array, int start, int end)

static void sort(float[] array, int start, int end)

static void sort(int[] array, int start, int end)

static void sort(long[] array, int start, int end)

static void sort(short[] array, int start, int end)

static void sort(Object[] array, int start, int end)

static void sort(Object[] array, int start, int end, Comparator c)

IllegalArgumentException នៅពេលដែល start ធំជាង end ឬក៏ ArrayIndexOutOfBounds-Exception កាលណា start ឬ end នៅក្រៅដែនកំណត់។

<u>ចំណាំ</u> ៖ គ្រប់ទម្រង់ទាំងអស់ ដែលមាន Comparator parameter អាចប្រើ ClassCastException កាលណាធាតុនៃ array ដែលត្រូវរៀបតាមលំដាប់នោះមិនអាចធ្វើការប្រៀបធៀបបាន។

ឧទាហរណ៍៖

```
import java.util.*;
class DemoArray{
        public static void main(String[] agr){
                int[] array=new int[10];
                for (int i=0 ; i<10 ; i++)
                        array[i]= -5*i ;
                //display  then sort and display again
                System.out.print("Original contents : ");
                display(array);
                Arrays.sort(array);
                System.out.print("Sorted : ");
                display(array);
                //fill and display
                Arrays.fill(array,2,6,-1);
                System.out.print("After fill() : ");
                display(array);
                //sort and diplay
                Arrays.sort(array);
                System.out.print("after sort again: ");
                display(array);
```

```
        //binary search for -15
        System.out.print("the value -15 in location : ");
        int index=Arrays.binarySearch(array,-15);
        System.out.println(index);

    }
    static void display(int[] array){
        for(int i=0;i<array.length;i++)
            System.out.print(array[i] + " ");
            System.out.println();
    }
}
```

លទ្ធផល៖

```
Original contents : 0 -5 -10 -15 -20 -25 -30 -35 -40 -45
Sorted : -45 -40 -35 -30 -25 -20 -15 -10 -5 0
After fill() : -45 -40 -1 -1 -1 -1 -15 -10 -5 0
after sort again: -45 -40 -15 -10 -5 -1 -1 -1 -1 0
the value -15 in location : 2
```

៥.៨ អំពី Vector

Vector វាមានលក្ខណៈដូចជា dynamic array ហើយវាស្រដៀងទៅនឹង ArrayList ដែរ ក៏ប៉ុន្តែវាមានភាពខុសគ្នាពីរគឺ Vector មានលក្ខណៈ synchronized ហើយនិងមាន methods ជាច្រើនដែលមិនមែនជាចំណែកនៃពួក collection។ ជាមួយ Java 2 នេះ Vector ទទួលលក្ខណៈពី AbstractList និងអនុវត្តតាម List interface ហេតុនេះវាមានលក្ខណៈត្រូវគ្នានឹង collections យ៉ាងពេញលេញ។

ទម្រង់ Constructors របស់ Vector៖

Vector( )

Vector(int size)

Vector(int size, int incr)

Vector(Collection c)

ចំពោះ vector កាលណាយើងមិនបានកំណត់តម្លៃតំបូងដូចក្នុងករណី constructor ទី១ វាឲ្យតម្លៃជា default គឺ ១០។ ចំពោះ incr ជាកំណើនទំហំរបស់ Vector រាល់ពេលដែល Vector នោះប្តូរទំហំ។ ហើយ constructor ទី៤ គឺបង្កើតនូវ vector មួយដែលមានធាតុនៃ collection វាត្រូវបានបន្ថែមឡើងដោយ Java 2 ។

គ្រប់ vectors ទាំងអស់ចាប់ផ្តើមឡើងដោយមានចំណុះតំបូង បន្ទាប់ពីចំណុះដំបូងនេះត្រូវបានអស់(ប្រើមកដល់ទំហំនេះ) ហើយនៅពេលដែលគេចង់ផ្ទុក Object នៅក្នុង vector នោះទៀត vector នឹងបង្កើតទីតាំងសម្រាប់ Object នោះ ដោយបូកទំហំសម្រាប់ object នោះដោយស្វ័យប្រវត្តិ។ តាមរយៈការបង្កើតទីតាំងលើសពីតម្រូវការ នោះ vector នឹងបន្ថយដោយស្វ័យប្រវត្តិដែរ។ បរិមាណនៃការបង្កើតទីតាំង

បន្ថែមត្រូវបានកំណត់នៅក្នុង (incr) នៅពេលយើងបង្កើត vector ។ បើមិនបញ្ជាក់ទេវានឹងកើនឡើងទ្វេតាមរយៈដំណើរបង្កើតទីតាំងនីមួយៗ។

Vector បានបង្កើត data members ដែលមានលក្ខណៈ protected ដូចខាងក្រោម៖

int capacityIncrement;

int elementCount;

Object elementData[];

តម្លៃកំណើនផ្ទុកក្នុង capacityIncrement, ចំនួនធាតុផ្ទុកក្នុង elementCount និងarray ដែលផ្ទុក vector ដាក់នៅក្នុង elementData។

Methods របស់ Vector ដែលអាចប្រើបានបន្ថែមទៅលើ methods ដែលមានក្នុងពពួក collections មានដូចខាងក្រោម៖

Methods Name	អធិប្បាយ
final void addElement(Object element)	បញ្ចូល element ថ្មីទៅក្នុង vector
final int capacity()	ប្រាប់ពីទំហំ(ចំណុះ) នៃ vector
Object clone()	ឲ្យតម្លៃស្ទួននៃ vector ដែលកំពុងប្រើ
final boolean contains(Object element)	ផ្តល់តម្លៃ true ប្រសិនបើ element នោះមាននៅក្នុងVector
final void copyInto(Object array[])	ចម្លងធាតុនៅក្នុង vector ទៅឲ្យ array
final Object elementAt(int index)	ផ្តល់ធាតុនៅត្រង់ index នោះ
final Enumeration elements()	ឲ្យតម្លៃជារបាប់នៃធាតុក្នុង vector
final ensureCapacity(int size)	កំណត់ចំណុះអប្បបរមានៃ vector ដោយ size
final Object firstElement()	ផ្តល់ធាតុទី១ នៅក្នុង vector
final int indexOf(Object element)	ផ្តល់តម្លៃ index របស់ element បើសិនជាមានក្នុង vector និងតម្លៃ -1 កាលណាគ្មាន element នៅក្នុង vector
final int indexOf(Object element,int start)	ផ្តល់តម្លៃ index នៅត្រង់ element ឬបន្ទាប់ start ផ្តល់តម្លៃ (-1) បើសិនជាមិនមាននៅក្នុង vector
final void insertElement(Object element, int index)	បញ្ចូល element ទៅក្នុង Vector នៅត្រង់ index
final boolean isEmpty()	ផ្តល់តម្លៃ true បើសិនជា vector គ្មានធាតុ
final Object lastElement()	ផ្តលតម្លៃ element ដែលនៅចុងក្រោយគេ
final int lastIndexOf(Object element)	ផ្តល់តម្លៃជា index នៅផ្នែកខាងចុងនៃ element បើសិនជា element មិនមាននៅក្នុង vector វាផ្តល់តម្លៃ -1
final int lastIndexOf(Object element, int start)	ផ្តល់តម្លៃជា index នៅផ្នែកខាងចុងនៃ element មុន start បើសិនជា element មិនមាននៅក្នុង vector វាផ្តល់តម្លៃ -1
final void removeAllElements()	លុបធាតុទាំងអស់ចេញពី vector
final boolean removeElement(Object	លុប element ចេញពី vector ហើយវាផ្តល់តម្លៃ true បើ

element)	សិនជាលុបបានជោគជ័យ
final void removeElementAt(int index)	លុប element នៅត្រង់ index
final void setElementAt(Object element, int index)	បញ្ចូល element ចូលក្នុង vector ត្រង់ index
final void setSize(int size)	ផ្តល់ទំហំ(ចំណុះ) size ទៅឲ្យ vector បើទំហំថ្មីតូចជាងទំហំចាស់ ពេលនោះបាត់បង់ធាតុ តែបើធំជាង ពេលនោះធាតុ null ត្រូវបានដាក់បញ្ចូល
final int size()	ផ្តល់តម្លៃជាចំនួនធាតុដែលកំពុងមាននៅក្នុង vector
String toString()	ឲ្យតម្លៃជា string ដែលសមមូលនឹង vector
final void trimToSize()	កំណត់ទំហំរបស់ vector ស្មើនឹងចំនួនធាតុដែលវាកំពុងផ្ទុក

ឧទាហរណ៍៖

```
import java.util.*;
class DemoVector{
        public static void main(String[] agr){
                //initial size of vector is 3 and increment by 2
                Vector v =new Vector(3,2);
                System.out.println("Initial size: " + v.size());
                System.out.println("Initial capacity: " + v.capacity());
                v.addElement(new Integer(1));
                v.addElement(new Integer(2));
                v.addElement(new Integer(3));
                v.addElement(new Integer(4));
                System.out.println("Capacity after added 4 integer Number :" + v.capacity());
                v.addElement(new Double(4.5));
                System.out.println("Current Capacity : " + v.capacity());
                v.addElement(new Double(6.08));
                v.addElement(new Integer(9));
                System.out.println("Current Capacity : " + v.capacity());
                v.addElement(new Float(9.4));
                v.addElement(new Integer(11));
                System.out.println("Current Capacity : " + v.capacity());
                System.out.println("First element : " +(Integer)v.firstElement() );
                System.out.println("Last element : " +(Integer)v.lastElement() );
                if (v.contains(new Integer(3)))
                System.out.println("Vector contain 3");
                //enumerate the elements in the vector
                Enumeration vEnum=v.elements();
```

```
                System.out.println("\nElements in vector:");
                while (vEnum.hasMoreElements())
                        System.out.print(vEnum.nextElement()+ " ");
                System.out.println();
        }
}
```

លទ្ធផលគឺ៖

```
Initial size: 0
Initial capacity: 3
Capacity after added 4 integer Number :5
Current Capacity : 5
Current Capacity : 7
Current Capacity : 9
First element : 1
Last element : 11
Vector contain 3

Elements in vector:
1 2 3 4 4.5 6.08 9 9.4 11
```

នៅក្នុង java 2 យើងអាចប្រើ iterator ជាមួយ Vector បានដែរ គេប្រើ iterator ជំនួសឲ្យការប្រើ enumeration ។

```
//use an iterator to display contents
Iterator itr=v.iterator();
System.out.println("\nElements in vector:");
while (itr.hasNext())
        System.out.print(itr.next()+ " ");
System.out.println();
```

៥. <u>អំពីការប្រើប្រាស់ Hashtable</u>

Hashtable class ស្ថិតនៅក្នុង java.util ហើយជា subclass មួយដែលទទួលលក្ខណៈពី abstract class មួយឈ្មោះ Dictionary ។ ទោះជាយ៉ាងណាក៏ដោយ Java 2 បានបង្កើតជាថ្មីនូវ Hashtable ដោយឲ្យវាអនុវត្តតាម Map interface ។ ដូច្នេះ Hashtable បានបញ្ចូលទៅក្នុងពពួក collections ។

Hashtable ផ្ទុកនូវតម្លៃ key/value ជាគូនៅក្នុងតារាង "hash"។ នៅពេលប្រើ Hashtable គេបញ្ជាក់នូវ object ដែលត្រូវប្រើជា key និងតម្លៃដែលគេចង់ភ្ជាប់ទៅនឹង key នោះ។ ពេលនោះ key ត្រូវធ្វើឲ្យទៅជា hash ហើយ hash code ដែលជាលទ្ធផលត្រូវបានប្រើជា index ដែលតម្លៃនោះផ្ទុកក្នុងតារាង។

Hash table អាចផ្ទុកតែ Objects ប៉ុណ្ណោះ ដែលសរសេរ Overriden លើ hashCode() method និង equals() method ហើយត្រូវបានកំណត់ដោយ object ។ hashCode() ត្រូវគណនា និងឲ្យតម្លៃជា hash

code សម្រាប់ object។ តាមពិត equals() ធ្វើការប្រៀបធៀប objects ពីរ ។ Java ផ្តល់នូវលក្ខណៈងាយស្រួលជាច្រើន ដោយវាមាន classes ដែលមានស្រាប់ ប្រើនូវ hashCode() ។ ឧទាហរណ៍ប្រភេទទិន្នន័យភាគច្រើនប្រើ String object ជា Key ។ String ប្រើទាំង hashCode() និង equals()។

ទម្រង់ Constructors របស់ Hashtable ៖

Hashtable()

Hashtable(int size)

Hashtable(int size, float fillRatio)

Hashtable(Map m)

ទម្រង់ទី១ ជា default constructor ។ទី២ បង្កើត hash table ដែលកំណត់ទំហំតម្លៃតំបូងគឺ size។ ទី៣ បង្កើតនូវ hash table ដូចនឹងទម្រង់ទី២ ដែរ តែវាមានចំនួនសមាមាត្រដែលត្រូវបំពេញ ។ ចំនួនសមាមាត្រត្រូវស្ថិតនៅចន្លោះ 0.0 និង 1.0 ហើយវាកំណត់នូវរបៀបដែល hash table ពេញមុននឹងត្រូវប្តូរទំហំ ។ ជាពិសេសនៅពេលចំនួនធាតុធំជាងចំណុះនៃ hash table គុណនឹងលេខសមាមាត្រដែលត្រូវបំពេញរបស់វា ពេលនោះ hash table នឹងត្រូវពង្រីក។ ប្រសិនបើយើងមិនបានបញ្ជាក់នូវចំនួនសមាមាត្រទេវានឹងប្រើ 0.75 ។ទម្រង់ចុងក្រោយបង្អស់ បង្កើតនូវ hash table មួយដែលកំណត់តម្លៃតំបូងដោយធាតុ Map m ទម្រង់នេះត្រូវបានបញ្ចូលដោយ java 2 ។

Methods ដែលអាចប្រើជាមួយ Hashtable ៖

ឈ្មោះ method	អធិប្បាយ
void clear()	Clear hash table
Object clone()	Clone Object ពីរដូចគ្នា
boolean contains(Object value)	ឲ្យតម្លៃ true បើសិនជា hash table មានផ្ទុក object value
boolean containsKey(Object key)	ឲ្យតម្លៃ true បើសិនជា hash table មានផ្ទុក object key នោះ
boolean containsValue(Object value)	ឲ្យតម្លៃ true បើសិនជា hash table មានផ្ទុក object value
Enumeration elements()	ឲ្យតម្លៃជាចំនួនរបាប់នៃតម្លៃដែលផ្ទុកក្នុង hash table
Object get(Object key)	ផ្តល់មកវិញនូវ Object ដែលត្រូវនឹង key បើគ្មានផ្តល់ null
boolean isEmpty()	ផ្តល់តម្លៃ true បើសិនជា hash table ទទេ
Enumeration keys()	ឲ្យតម្លៃជាចំនួនរបាប់នៃ key ដែលផ្ទុកក្នុង hash table
Object put(Object key, Object value)	បញ្ចូល key និង value ទៅក្នុង hash table
void rehash()	បង្កើនទំហំរបស់ hash table ហើយរៀបដាក់ជាថ្មីគ្រប់ keys ទាំងអស់របស់វាក្នុង hash table
Object remove(Object key)	លុបតម្លៃចេញពី hash table បើ key វាត្រូវគ្នា
int size()	ផ្តល់តម្លៃជាចំនួននៃការបញ្ចូលក្នុង hash table
String toString()	ឲ្យតម្លៃជា string ដែលសមមូលនឹង hash table

ឧទាហរណ៍៖

```
import java.util.*;
class DemoHashTable1{
    public static void main(String[] agr){
```

```
                Hashtable balance = new Hashtable();
                Enumeration names;
                String str;
                double bal;
                balance.put("Choeun Net",new Double(3456.43));
                balance.put("Lim Navuth",new Double(5432.12));
                balance.put("Lim Sothea",new Double(9876.54));
                balance.put("Phan Thenghay",new Double(8765.43));
                balance.put("Lai Chansok",new Double(7654.32));
                balance.put("Srorn Seiha",new Double(9753.13));
                balance.put("You Hengadom",new Double(7531.31));
                //show all balance in hash table
                names=balance.keys();
                while(names.hasMoreElements()){
                        str=(String) names.nextElement();
                        System.out.println(str + ": "+ balance.get(str));
                }
                System.out.println();
                //deposit 1111 into Srorn Seiha's account
                bal=((Double)balance.get("Srorn Seiha")).doubleValue();
                balance.put("Srorn Seiha",new Double(bal + 1111));
                System.out.println("Srorn Seiha's new balance" +
balance.get("Srorn Seiha"));
        }
}
```

លទ្ធផលគឺ៖

```
Lai Chansok: 7654.32
Lim Navuth: 5432.12
Srorn Seiha: 9753.13
Lim Sothea: 9876.54
You Hengadom: 7531.31
Phan Thenghay: 8765.43
Choeun Net: 3456.43

Srorn Seiha's new balance 10864.13
```

ចំណុចសំខាន់មួយគឺដូចទៅនឹង map class ដែរ hash table មិនអាចប្រើ iterators បានដោយផ្ទាល់ឡើយ ។ ហេតុនេះកម្មវិធីមុននេះប្រើ Enumeration ដើម្បីបង្ហាញតម្លៃនៃចំនួនទឹកប្រាក់។ ទោះជាយ៉ាងណាក៏ដោយ យើងក៏អាចប្រើ iterators បានដែរ។ ដើម្បីធ្វើដូច្នេះបាន យើងប្រើ method មួយក្នុងចំណោម methods របស់ពពួក collection ដែលកំណត់ដោយ Map ដូចជា entrySet() ឬ keySet()។

ឧទាហរណ៍៖

```
import java.util.*;
class DemoHashTable2{
	public static void main(String[] agr){
		Hashtable balance = new Hashtable();
		String str;
		double bal;
		balance.put("Choeun Net",new Double(3456.43));
		balance.put("Lim Navuth",new Double(5432.12));
		balance.put("Lim Sothea",new Double(9876.54));
		balance.put("Phan Thenghay",new Double(8765.43));
		balance.put("Lai Chansok",new Double(7654.32));
		balance.put("Srorn Seiha",new Double(9753.13));
		balance.put("You Hengadom",new Double(7531.31));
		//show all balance in hash table
		Set set=balance.keySet();
		//get iterator
		Iterator itr = set.iterator();
		while(itr.hasNext()){
			str=(String) itr.next();
			System.out.println(str + ": "+ balance.get(str));
		}
		System.out.println();
		//deposit 1111 into Srorn Seiha's account
		bal=((Double)balance.get("Srorn Seiha")).doubleValue();
		balance.put("Srorn Seiha",new Double(bal + 1111));
		System.out.println("Srorn Seiha's new balance" + balance.get("Srorn
Seiha"));
	}
}
```

លទ្ធផលគឺ៖

```
	Lai Chansok: 7654.32
	Lim Navuth: 5432.12
	Srorn Seiha: 9753.13
	Lim Sothea: 9876.54
	You Hengadom: 7531.31
	Phan Thenghay: 8765.43
	Choeun Net: 3456.43
	Srorn Seiha's new balance 10864.13
```

End

មេរៀនទី ១០

អំពីការប្រើប្រាស់ *Events*

១. អំពី Events

ការប្រើប្រាស់ event គឺដើម្បីឲ្យមានទំនាក់ទំនងជាមួយ applets និង components ផ្សេងៗទៀត។ Events ភាគច្រើនដែលគេប្រើនោះ គឺវាកើតឡើងដោយសារ mouse, keyboard និង controls ផ្សេងៗទៀត។ គេអាចប្រើ Events ទាំងអស់នៅក្នុង Java 2 តាមរយៈ Package មួយឈ្មោះ java.awt.event ។ យើងអាចប្រើ Event តាមលំនាំរបស់ Java 2 ដែលគេហៅថា delegation event model ជាជាងប្រើតាមលំនាំ Java version 1.0 ព្រោះតាមលំនាំ Java 2 នេះវាមានលក្ខណៈទូទៅ ច្បាស់លាស់ដែលអាចបង្កើត និងដំណើរការ Event បានយ៉ាងល្អ ពោលគឺ source បង្កើតនូវ event ហើយបញ្ជូន event នេះទៅកាន់ listeners មួយ ឬច្រើន។ listener រងចាំទទួល event ពេលទទួលបានហើយវាដំណើរការ event រួចបញ្ជូនលទ្ធផលត្រឡប់មកវិញ។

Java អាចដំណើរការ event ដោយមិនប្រើលំនាំ delegation event model បាន ដោយវាទទួលលក្ខណៈពី AWT component តែទោះជាយ៉ាងណាក៏វាពុំមានលក្ខណៈល្អប្រសើរដូចលំនាំ delegation event model ដែរ។

Event គឺជា object មួយដែលរៀបរាប់ពីបម្រែបម្រួលសភាពលក្ខណៈនៅក្នុង Source ។ វាកើតឡើងដោយសារអំពើរបស់ user ទៅលើសមាសធាតុនៅក្នុង graphical user interface ។ event វាអាចកើតឡើងពេលចុច button , បញ្ចូលតាម keyboard , ជ្រើសរើស items , ការចុចលើ mouse , timer ហួសកំណត់, counter ហួសតម្លៃ , software/ hardware មានបញ្ហាកើតឡើង ឬការប្រតិបត្តចប់សព្វគ្រប់ជាដើម។

២. អំពី Event Sources

Source គឺជា Object មួយបង្កើតនូវ event , វាអាចបង្កើតនូវ event មួយ ឬក៏ច្រើនបាន។ Source ត្រូវតែកត់ត្រានូវ listeners ដើម្បីឲ្យ listeners ទទួលសញ្ញាកំនត់ត្រាអំពីប្រភេទ event ដែលបានបញ្ជាក់។ ប្រភេទ event នីមួយៗ មាន methods កំណត់ត្រារបស់ខ្លួនរៀងៗខ្លួន។

ទម្រង់ទូទៅ៖

public void add*Type*Listener(*Type*Listener el)

ដែល *type* គឺជាឈ្មោះរបស់ event, el គឺ event listener ។

ឧទាហរណ៍ method ដែលកត់ត្រានូវ keyboard event listener គឺ addKeyListener(), method កត់ត្រានូវ mouse motion listener គឺ addMouseMotionListener()។ នៅពេល event មួយកើតឡើងនោះ គ្រប់ listeners ទាំងអស់ដែលបានកត់ត្រា ត្រូវកត់សំគាល់ និងទទួលនូវការចម្លងនៃ event object ដែលលក្ខណៈនេះត្រូវបានគេហៅថាmulticasting the event ។

Source ខ្លះអនុញ្ញាតឲ្យ listener តែមួយគត់សម្រាប់តក់ត្រា។ទម្រង់ទូទៅរបស់ method នេះគឺ៖

public void add*Type*Listener(*Type*Listener el) throws java.util.TooManyListenersException

កាលណា event **មួយកើតឡើង** listener **ដែលបានកត់ត្រា ត្រូវបានកត់សំគាល់ លក្ខណៈនេះ ត្រូវបានហៅថា** unicasting the event ។

Source **ត្រូវតែផ្តល់នូវវិធីមួយដែលអាចឲ្យ**listener **មួយលុបបំបាត់នូវកំណត់ត្រាចំពោះ**event **ណា មួយដែលបានបញ្ជាក់។ទម្រង់ទូទៅរបស់វា៖**

public void remove *Type*Listener(*Type*Listener el)

ឧទាហរណ៍ ដើម្បីលុបបំបាត់នូវkeyboard listener **គឺ** removeKeyListener()។

៣. <u>អំពី Event Listeners</u>

Listener **គឺជា** object **មួយដែលត្រូវបានកំណត់នៅពេល**event **មួយបានកើតឡើង ។ ទី១ វាត្រូវ តែកត់ត្រាទុកនូវ**source **មួយឬច្រើន ដើម្បីទទួលសញ្ញាកំណត់ត្រាប្រភេទ** events **ដែលបានបញ្ជាក់។ទី២ វាត្រូវប្រើ** methods **ដើម្បីទទួល និងដំណើរការសញ្ញាកំណត់ត្រាទាំងនេះ។**

Methods **ដែលទទួល** events **និងដំណើរការ** events **ត្រូវបានកំណត់នៅក្នុងសំនុំ** interface **ដែល ស្ថិតនៅក្នុង** java.awt.event ។

ឧទាហរណ៍៖ MouseMotionListener interface **មាន** methods **ពីរសម្រាប់ទទួលសញ្ញាកំណត់ត្រា នៅពេល** mouse **ត្រូវបានផ្លាស់ទី** (move) **ឬចុចសង្កត់ហើយរំកិល**(drag)។

៤. <u>អំពី Event Class</u>

Class **តាងឲ្យ** events **គឺជាស្នូលនៃការប្រើប្រាស់**event ។ **លំដាប់ថ្នាក់របស់** java event class **គឺ** event object **ដែលស្ថិតនៅក្នុង**java.util **វាជា** superclass **នៃ** event classes **ទាំងអស់។**

ទម្រង់ constructor **របស់វា៖**

EventObject(Object src) // src **គឺជា** object **ដែលបង្កើតនូវ**event **នេះ។**

EventObject **មាន** methods **ចំនួនពីរ គឺ** getSource() **ឲ្យតម្លៃជា** source **របស់** event **និង** toString() **ដែលឲ្យតម្លៃជា** String **សមមូលនឹង** event ។ **ដែលមានទម្រង់ទូទៅ៖**

Object getSourcce()

String toString()

Class **ឈ្មោះ** AWTEvent **មាននៅក្នុង** package java.awt **ហើយវាជា** subclass **នៃ** EventObject **ក៏ប៉ុន្តែវាជា** superclass **នៃ** event classes **ទាំងអស់ដែលមានមូលដ្ឋាននៅលើ** AWT (**ដោយផ្ទាល់ ឬដោយប្រយោល) ហើយប្រើនៅក្នុងលំនាំ** delegation event model ។ getID() method **របស់វាត្រូវបាន ប្រើសម្រាប់កំណត់ប្រភេទនៃ**event ។ **ទម្រង់របស់វាគឺ** int getID()។

តារាងខាងក្រោមនេះបង្ហាញនូវ event class **សំខាន់ៗ និងការអធិប្បាយដោយសង្ខេប នៅពេល ដែលវាត្រូវបានបង្កើតឡើង។**

<u>Event class</u>	<u>អធិប្បាយដោយសង្ខេប</u>
ActionEvent	កើតឡើងនៅពេលគេចុច button ឬ list item ត្រូវបានDouble-Click ឬ menu item ត្រូវបានជ្រើសរើស
AdjustmentEvent	កើតឡើងនៅពេលដែលយើងប្រើ scroll bar
ComponentEvent	កើតឡើងនៅពេល component មួយត្រូវបានបិទបាំង បង្ហាញឲ្យឃើញមកវិញ

	ផ្លាស់ទី ឬប្តូរទំហំ
ContainerEvent	កើតឡើងនៅពេល component មួយត្រូវបានបន្ថែមចូល ឬលុបចេញពី container
FocusEvent	កើតឡើងនៅពេល component មួយទទួល ឬបាត់បង់ keyboard focus
InputEvent	ជា abstract super class ចំពោះគ្រប់ events classes ទាំងអស់ដែលមានលក្ខណៈ input របស់ component
ItemEvent	កើតឡើងនៅពេល checkBox ឬ list item ត្រូវបានចុច ការជ្រើសយក item មួយនៃ Choice, ជ្រើសយក ឬលុបបំបាត់ការជ្រើសយក menu item ដែលមានលក្ខណៈ checkbox
KeyEvent	កើតឡើងនៅពេលមានការបញ្ចូល ឬបានទទួលពី keyboard
MouseEvent	កើតឡើងគ្រប់សកម្មភាពនីមួយៗរបស់ mouse
TextEvent	កើតឡើងនៅពេលតម្លៃរបស់ text area ឬ text field មានការប្រែប្រួល
WindowEvent	កើតឡើងនៅពេលដែល window មួយកំពុងមានសកម្មភាព ឬមិនមានសកម្មភាព ការបើក ការបិទ ការបង្រួម ការពង្រីក...

៤.១ អំពី ActionEvent

ActionEvent class **មានបួនតម្លៃថេរ ដែលជាចំនួនគត់ត្រូវបានប្រើសម្រាប់សំគាល់សភាពប្រែប្រួលណាមួយដែលទាក់ទងទៅនឹង** action event : ALT_MASK, CTRL_MASK, META_MASK **និង** SHIFT_MASK **។ លើសពីនេះវាក៏មានតម្លៃថេរមួយគឺ** ACTION_PERFORMED **ដែលត្រូវបានប្រើសម្រាប់សំគាល់** action event **។**

ActionEvent class **មាន**constructor **ចំនួនពីរ៖**

ActionEvent (Object src, int type, String cmd)

ActionEvent (Object src, int type, String cmd, int modifiers)

ក្នុងនេះ scr **ជា** reference **សម្រាប់** object **ដែលបានបង្កើត** event **នេះ** , type **បញ្ជាក់ពីប្រភេទ** event, cmd **ជាអក្សរនៃពាក្យបញ្ជារបស់វានិង** modifiers **ប្រាប់នូវ** key **ណាមួយបានចុច ឬផ្លាស់ប្តូរ** (ALT, CTRL, META, SHIFT)**។**

គេអាចទទួលបានឈ្មោះពាក្យបញ្ជាសម្រាប់ប្រើជាមួយ ActionEvent Object **តាមរយៈ** getAction-Command() **ដែលមានទម្រង់ដូចខាងក្រោម៖**

String getActionCommand()

getMidifiers() method **នឹងឲ្យតម្លៃមួយសម្រាប់ប្រាប់នូវ**keys **ណាដែលមានសភាពប្រែប្រួល**(ALT, CTRL, META, SHIFT) **ត្រូវបានចុច នៅពេល** event **បានកើតឡើង ។**

int getModifiers()

៤.២ អំពី AdjustmentEvent

មានប្រភេទចំនួន ៥ adjustment event **។** AdjustmentEvent **មានចំនួនថេរជាចំនួនគត់សម្រាប់សំគាល់** events **ដូចបង្ហាញក្នុងតារាងខាងក្រោម៖**

BLOCK_DECREMENT	ពេលយើងចុចខាងក្នុង scroll bar ដើម្បីបន្ថយតម្លៃ
BLOCK_INCREMENT	ពេលយើងចុចខាងក្នុង scroll bar ដើម្បីបង្កើនតម្លៃ
TRACK	នៅពេលយើង Slider ត្រូវចុចបង្ហូស
UNIT_DECREMENT	ចំនុចខាងចុងនៃ Scroll bar ត្រូវបានចុចបន្ថយតម្លៃរបស់វា
UNIT_INCREMENT	ចំនុចខាងចុងនៃ Scroll bar ត្រូវបានចុចបង្កើនតម្លៃរបស់វា

លើសពីនេះទៅទៀតវាមានចំនួនថេរមួយជាចំនួនគត់ គឺ ADJUSTMENT_VALUE_CHANGED **ដែលបង្ហាញថាការប្រែប្រួលបានកើតឡើង។**

Constructor **របស់វា៖**

AdjustmentEvent(Adjustable, src, int id, int type, int data)

-src **គឺជា** reference **នៃ** Object **ដែលបានបង្កើត** event

-id **សមមូលនឹង** ADJUSTMENT_VALUE_CHANGED

-**ប្រភេទនៃ** event **ត្រូវបានបញ្ជាក់តាមរយៈ**type

-**ទិន្នន័យដែលមានទំនាក់ទំនងជាមួយគ្នាគឺ**data

getAdjustable() **ឲ្យតម្លៃជា** object **ដែលបានបង្កើតនូវ**event **។ ទម្រង់របស់វាគឺ៖**

Adjustable getAdjustable()

ប្រភេទនៃ Adjustment event **អាចទទួលបានតាមរយៈ:** Adjustable getAdjustmentType() **។ វាឲ្យតម្លៃជាចំនួនថេរដែលមាននៅក្នុង** AdjustmentEvent **។ ទម្រង់របស់វា៖**

getAdjustmentType()

បរិមាណនៃតម្រូវការអាចទទួលបានតាមរយៈ:getValue() **។ ទម្រង់របស់វា៖**

getValue()

៤.៣ <u>អំពី ComponentEvent</u>

ComponentEvent **កើតឡើងនៅពេលដែល ទំហំ ទីតាំង ឬភាពមើលឃើញរបស់** component **មួយមានការផ្លាស់ប្តូរ ។ វាមានប្រភេទ** component event **ចំនួន ៤ ដែលជាចំនួនថេរសម្រាប់សំគាល់** events **នីមួយៗដូចខាងក្រោម៖**

COMPONENT_HIDDEN	នៅពេលដែល component មួយត្រូវបានបិតបាំង
COMPONENT_MOVED	នៅពេលដែល component មួយត្រូវបានផ្លាស់ទី
COMPONENT_RESIZED	នៅពេលដែល component មួយត្រូវបានប្តូរទំហំ
COMPONENT_SHOWN	នៅពេលដែល component មួយត្រូវបានបង្ហាញឲ្យឃើញវិញ

៤.៤ <u>អំពី ContainerEvent</u>

ContainerEvent **កើតឡើងនៅពេល** component **មួយត្រូវបានបន្ថែមចូលឬលុបចេញពី** container **។** ContainerEvent **មានចំនួនថេរជាចំនួនគត់ដែលអាចប្រើសម្រាប់សំគាល់**Container events **ដូចជា៖** COMPONENT_ADDED **និង** COMPONENT_REMOVED **។**

ContainerEvent **គឺជា** subclass **នៃ** ComponentEvent **មាន** Constructor **ដូចខាងក្រោម៖**

ContainerEvent(Component src , int type, Component comp)

-src ជា reference សម្រាប់ container ដែលបានបង្កើត event នេះ

-type បញ្ជាក់ពីប្រភេទ event

-comp គឺជា component មួយត្រូវបានបន្ថែមចូលឬលុបចេញពី container ។

យើងអាចទទួលបាន reference **សម្រាប់** container **ដែលបង្កើតនូវ** event **នេះដោយប្រើ** getContainer() method **ដែលមានទម្រង់ដូចខាងក្រោម៖**

Container getContainner()

getChild)() method **ឲ្យតម្លៃជា** reference **ចំពោះ** component **ដែលបានបន្ថែមចូលឬលុបចេញ។**

ទម្រង់ទូទៅ៖

Component getChild()

៤.៥ <u>អំពី FocusEvent</u>

FocusEvent **កើតឡើងនៅពេល** component **មួយទទួល ឬបាត់បង់** keyboard focus ។ events **ទាំងនោះត្រូវបានសំគាល់ដោយចំនួនថេរជាចំនួនគត់គឺ**FOCUS_GAINED **និង** FOCUS_LOST ។

Constructor **របស់** FocusEvent **៖**

FocusEvent(Component src, int type)

FocusEvent(Component src, int type, boolean *temporaryFlag*)

-src **និង** type **ដូចខាងលើ ចំពោះ** temporary **ត្រូវកំណត់តម្លៃ** true **បើសិន** focus event **ស្ថិតក្នុងសភាព** temporary ។ **ផ្ទុយទៅវិញត្រូវកំណត់តម្លៃ** false ។ **បានន័យថា** focus event **ដែលមានលក្ខណៈ** temporary **កើតឡើងដោយសារលទ្ធផលនៃការប្រតិបត្តលើ** user interface **ផ្សេងទៀត។ ឧទាហរណ៍ឧបមាថា** focus **កំពុងស្ថិតក្នុង** text field **ហើយប្រសិនបើជាអ្នកប្រើបានផ្លាស់ទី** mouse **ដើម្បីតម្រូវ** scrollbar **នោះ** focus **ត្រូវបាត់បង់សភាព** temporary ។

isTemporary() method **ប្រាប់ឲ្យដឹងនូវភាពបម្រែបម្រួល** focus **នេះកំពុងស្ថិតនៅក្នុងលក្ខណៈ** temporary **ដែរ ឬទេ។ ទម្រង់របស់វា៖**

boolean isTemporary()

វាឲ្យតម្លៃ true **កាលណាសភាពប្រែប្រួលកំពុងស្ថិតក្នុងលក្ខណៈ**temporary**។**

៤.៦ <u>អំពី InputEvent</u>

InputEvent **គឺជា** abstract class **ដែលជា** subclass **នៃ** ComponentEvent **និងជា** superclass **ចំពោះ** input event **របស់** component**។** subclass **របស់វាមាន** KeyEvent **និង** MouseEvent**។** InputEvent class **មាន ៨ ចំនួនថេរជាចំនួនគត់ ដែលប្រើដើម្បីទទួលព័ត៌មានអំពី** modifier **ណាមួយដែលមានទំនាក់ទំនងជាមួយនឹង** event ។ **ចំនួនថេរទាំងនោះមាន៖**

ALT_MASK	ALT_GRAPH_MASK	META_MASK
BUTTON1_MASK	BUTTON2_MASK	BUTTON3_MASK
SHIFT_MASK	CTRL_MASK	

យើងមាន methods **ប្រើសម្រាប់ផ្ទៀងផ្ទាត់ ប្រសិនបើ** modifiers **ទាំងនោះត្រូវបានចុចក្នុងខណៈដែល** event **នេះ ដូចខាងក្រោម៖**

boolean isAltDown()

boolean isAltGraphDown()

boolean isControlDown()

boolean isMetaDown()

boolean isShiftDown()

getModifiers() ឲ្យតម្លៃមួយដែលមានគ្រប់ modifier flags សម្រាប់ event នេះ។ វាមានទម្រង់ដូចខាងក្រោម៖

int getModifiers()

៤.៧ អំពី ItemEvent

ItemEvent កើតឡើងនៅពេល checkBox ឬ list item ត្រូវបានចុច ការជ្រើសយក item មួយនៃ Choice ជ្រើសយក ឬលុបបំបាត់ការជ្រើសយក menu item ដែលមានលក្ខណៈ checkBox ។ វាមាន item events ពីរប្រភេទដែលសំគាល់ដោយចំនួនថេរ និងចំនួនថេរមួយដែលផ្តល់សញ្ញាប្រែប្រួលនៃសភាពលក្ខណៈ គឺ៖

- DESELECTED និង SELECTED
- ITEM_STATE_CHANGED

Constructor របស់ ItemEvent ៖

ItemEvent(ItemSelectable src, int type, Object entry, int state)

- src ជា reference សម្រាប់ component ដែលបានបង្កើត event ជាក់ស្តែងវាអាចជាធាតុរបស់ Choice ឬ list ។
- type បញ្ជាក់ឲ្យប្រភេទ Event
- item ដែលបង្កើតនូវ item event ត្រូវបានបញ្ជូនតាមរយៈ entry
- state បញ្ជាក់ប្រាប់ពីសភាពលក្ខណៈកំពុងមាននៅលើitem

យើងអាចប្រើ getItem() ដើម្បីទទួលបាននូវ reference ចំពោះ item ដែលបង្កើតឡើងនូវ event។

ទម្រង់របស់វា៖

Object getItem()

getItemSelectable() សម្រាប់ទទួលយកនូវ reference ចំពោះ ItemSelectable object ដែលបានកើតមាន event ។

ទម្រង់របស់វា៖

ItemSelectable getItemSelectable()

getStateChange() សម្រាប់ឲ្យតម្លៃជាសភាពប្រែប្រួលSELECTED ឬ DESELECTED ។

ទម្រង់របស់វា៖

int getStateChange()

៤.៨ អំពី KeyEvent

កើតឡើងនៅពេលមានការបញ្ចូល ឬបានទទួលពី keyboard ។ វាមានចំនួនថេរ ៣ សម្រាប់

សំគាល់ប្រភេទ event **របស់វាគឺ** KEY_PRESSED, KEY_RELEASED **និង** KEY_TYPED **។ វាមានចំនួនថេរមួយចំនួនទៀត ដែលកំណត់ដោយ** KeyEvent **មានដូចជា៖** VK_0 **ដល់** VK_9, VK_A **ដល់** VK_Z, VK_a **ដល់** VK_z...**ដែលមាន** ASCII **ត្រូវគ្នានឹងលេខ និងអក្សរ។ ហើយក៏មានចំនួនថេរមួយចំនួនទៀតដែរ ដូចជា៖**

VK_ENTER	VK_ESCAPE	VK_CANCEL	VK_UP
VK_DOWN	VK_LEFT	VK_RIGHT	VK_PAGE_DOWN
VK_PAGE_UP	VK_SHIFT	VK_ALT	VK_CONTROL

ដែល VK **បញ្ជាក់ឲ្យ** virtual key code **ហើយមានលក្ខណៈមិនអាស្រ័យនឹង** modifiers **ណាមួយឡើយ ដូចជា** control, shift **ឬ** alt **។**

វាមាន Constructor **ដូចខាងក្រោម៖**

KeyEvent(Component src, int type, long when, int modifiers, int code)

KeyEvent(Component src, int type, long when, int modifiers, int code, char ch)

- src **ជា** reference **សម្រាប់** component **ដែលបានបង្កើត** event

-type **បញ្ជាក់ពីប្រភេទនៃ** event

-when **បញ្ជាក់ពីពេលវេលាដែល**key **បានចុចបញ្ចូន**

-modifiers **ប្រាប់នូវ** modifiers **ណាដែលបានចុចនៅពេល** key event **នេះកើតឡើង**

-code **ជាកន្លែងដែលត្រូវបញ្ចូន**virtual key code

-ch **ជាកន្លែងដែល តួអក្សរដែលសមមូលនឹងត្រូវបញ្ចូន ប្រសិនបើពុំមានតួអក្សរត្រឹមត្រូវទេ ពេលនោះ** ch **មានតម្លៃជា** CHAR_UNDEFINED **ចំពោះ** KEY_TYPED events **វិញ** code **នឹងមានតម្លៃជា** VK_UNDEFINED **។**

KeyEvent **មាន** methods **ជាច្រើន ប៉ុន្តែ** methods **ដែលយើងប្រើញឹកញាប់ជាគេគឺ**getKeyChar() **និង** getKeyCode()**។**

Constructor **របស់** methods **ទាំងពីរគឺ៖**

char getKeyChar()

int getKeyCode()

បើវាប្រើតួអក្សរមិនត្រឹមត្រូវនោះ getKeyChar() **ឲ្យតម្លៃជា** CHAR_UNDEFINED **។ កាលណា** KEY_TYPED event **កើតឡើង ពេលនោះ** getKeyCode() **ឲ្យតម្លៃជា** VK_UNDEFINED **។**

៤.៩ <u>អំពី MouseEvent</u>

ក្នុងនោះវាមានចំនួនថេរដែលប្រើប្រាស់នៅក្នុង MouseEvent **ដូចខាងក្រោម៖**

MOUSE_CLICKED	កើតឡើងនៅពេលដែលយើង click
MOUSE_DRAGGED	កើតឡើងនៅពេលដែលយើងចុចអូស mouse
MOUSE_ENTERED	កើតឡើងនៅពេលដែល mouse pointer អូសកាត់ component
MOUSE_EXITED	កើតឡើងនៅពេល mouse pointer រំកិលផុតពី Component
MOUSE_MOVED	កើតឡើងនៅពេលដែលយើងផ្លាស់ទី mouse pointer
MOUSE_PRESSED	កើតឡើងនៅពេលដែល mouse ត្រូវបានយើងចុចជាប់
MOUSE_RELEASED	កើតឡើងនៅពេលដែលយើងព្រលែង mouse ពីការចុច

Constructor របស់ MouseEvent ៖

MouseEvent(Component src, int type, long when, int modifiers, int x, int y,int clicks, boolean triggersPopup)

-src ជា reference សម្រាប់ component ដែលបានបង្កើត event

-type បញ្ជាក់ពីប្រភេទនៃ event

-when បញ្ជាក់ពីពេលវេលាដែល mouse បានចុចបញ្ជូន

-modifiers ប្រាប់នូវ modifiers ណាដែលបានចុច នៅពេល mouse event នេះកើតឡើង

- x និង y ប្រាប់ពីកូអរដោនេ នៅពេលដែល mouse event កើតឡើងនៅត្រង់ណា

-clicks ប្រាប់ពីចំនួនដងនៃការclick

-triggersPopup ប្រាប់ឲ្យដឹងថា បើសិនជា event នេះនាំឲ្យមាន popup-menu លេចឡើងលើ platform មានតម្លៃ true ផ្ទុយទៅវិញត្រូវដាក់តម្លៃfalse ។

Methods ដែលប្រើញឹកញាប់ជាងគេគឺgetX() និង getY()។ទម្រង់របស់វា៖

```
int getX()
int getY()
```

យើងអាចប្រើ getPoint() ដើម្បីទទួលយកបាននូវកូអរដោនេរបស់mouse បាន។

```
Point getPoint()
```

យើងប្រើ translatePoint() ដើម្បីបំលាស់ប្តូរទីតាំងនៃevent ។

```
void translatePoint(int x, int y)
```

យើងប្រើ getClickCount() ដើម្បីទទួលបាននូវចំនួនដងនៃការclick

```
int getClickCount()
```

យើងប្រើ isPopupTrigger() ដើម្បីផ្ទៀងផ្ទាត់ថាតើ event នេះមាន popup menu ដែរ ឬទេ។

```
boolean isPopupTrigger()
```

៥.១០ អំពី TextEvent

TextEvent បង្កើតឡើងដោយ text field និង text area កាលណាតួអក្សរត្រូវបានបញ្ចូលដោយ user ឬដោយកម្មវិធី។ ចំនួនថេរដែលប្រើជាមួយTextEvent គឺ TEXT_VALUE_CHANGED ។

ទម្រង់ Constructor របស់ TextEvent ៖

```
TextEvent(Object src, int type)
```

៥.១១ អំពី WindowEvent

WindowEvent មានចំនួនថេរដែលប្រើសម្រាប់សំគាល់event នេះដូចខាងក្រោម៖

WINDOW_ACTIVATED	កើតឡើងនៅពេលដែល window កំពុងមានសកម្មភាពលើវា
WINDOW_CLOSED	កើតឡើងនៅពេលដែល window ត្រូវបានបិទ
WINDOW_CLOSING	កើតឡើងនៅពេលដែល window កំពុងត្រូវបានចុចបិទ
WINDOW_DEACTIVATED	កើតឡើងនៅពេលដែល window ពុំមានសកម្មភាពលើវា

WINDOW_DEICONIFIED	កើតឡើងនៅពេលដែល window ត្រូវបាន maximize វិញ
WINDOW_ICONIFIED	កើតឡើងនៅពេលដែល window ត្រូវបាន minimize
WINDOW_OPENED	កើតឡើងនៅពេលដែល window ត្រូវបានបើក

ទម្រង់ Constructor **របស់** WindowEvent **៖**

WindowEvent(Window src, int type)

Method **ដែលប្រើញឹកញាប់គឺ**getWindow() **វាឲ្យតម្លៃជា** object **នៃ** window **ដែលបង្កើតនូវ** event

Window getWindow()

៥. <u>អំពី Source នៃ Events</u>

ខាងក្រោមជា តារាងបង្ហាញពី components **សម្រាប់** user interface **ដែលអាចបង្កើត** event **។ លើសពីនេះ ក៏នៅមាន** component **មួយចំនួនទៀតដែលអាចបង្កើតនូវ** event **បានដែរដូចជា** applet **ជាដើម។**

Event Source	អធិប្បាយ
Button	បង្កើតនូវ action events ពេល button មួយត្រូវបានចុច
CheckBox	បង្កើតនូវ item events ពេល Checkbox ត្រូវបានជ្រើសយក ឬលុបបំបាត់ការជ្រើសយក
Choice	បង្កើតនូវ item events ពេល Choice ត្រូវបានផ្លាស់ប្តូរ
List	បង្កើតនូវ Action events ពេល item មួយត្រូវបានចុចពីរដងស្នួន ហើយបង្កើតនូវ item events ពេល item មួយត្រូវបានជ្រើសយក ឬលុបបំបាត់ការជ្រើសយក
MenuItem	បង្កើតនូវ Action events ពេល menu item មួយត្រូវបានជ្រើសរើសយក ហើយបង្កើតនូវ item events ពេល menu item ដែលមានលក្ខណៈ checkbox មួយត្រូវបានជ្រើសយក ឬលុបបំបាត់ការជ្រើសយក
Scollbar	បង្កើតនូវ Adjustment events នៅពេល scrollbar ត្រូវបានប្រើ
TextComponent	បង្កើតនូវ text event កាលណាអ្នកប្រើបានបញ្ចូលតួអក្សរ
Window	បង្កើតនូវ window event កាលណា window មួយស្ថិតក្នុងសភាពកំពុងមានសកម្មភាព, មិនមានសកម្មភាព បើក បិទ minimize maximize ឬ ចាកចេញ

៦. <u>អំពី Event Listener Interfaces</u>

ដូចយើងបានដឹងរួចមកហើយថាdelegation event model **មានពីរផ្នែកគឺ** sources **និង** listeners**។** Listeners **ត្រូវបានបង្កើតឡើងដោយ** interface **មួយឬច្រើន ដែលស្ថិតនៅក្នុង** package java.awt.event**។ កាលណា** event **មួយកើតឡើង** event source **ប្រើ** method **ដែលសមស្របកំណត់ដោយ** listener **និង ផ្តល់នូវ** event object **ជា** argument **របស់វា។**

ខាងក្រោមនេះជាតារាងបង្ហាញពីឈ្មោះlistener **ដែលប្រើញឹកញាប់៖**

Interface	អធិប្បាយ
ActionListener	មាន method មួយសម្រាប់ទទួល Action events
AdjustmentListener	មាន method មួយសម្រាប់ទទួល Adjustment events
ComponentListener	មាន methods បួន សម្រាប់សំគាល់នៅពេល component មួយត្រូវបាន

	បិទបាំង ផ្លាស់ទី ប្តូរទំហំ ឬបង្ហាញមកវិញ
ContainerListener	មាន methods ពីរ សម្រាប់សំគាល់នៅពេល component មួយត្រូវបានបន្ថែម ឬដកចេញ
FocusListener	មាន methods ពីរសម្រាប់សំគាល់នៅពេលដែល component មួយទទួល ឬ បាត់បង់ focus
ItemListener	មាន method មួយ ដើម្បីដឹងបាននៅពេលដែល item មួយបានប្រែប្រួល
KeyListener	មាន methods បី ដើម្បីដឹងថា key ចុចជាប់ ដកដៃមកវិញ key បានវាយ
MouseListener	មាន methods ប្រាំ ដើម្បីដឹងពីសកម្មភាព mouse ចុច , ចុចសង្កត់ជាប់ ព្រលែងពីការចុច រំកិលចូលទៅ component ឬចាកចេញពី component
MouseMotionListener	មាន method ពីរ ដើម្បីដឹងនៅពេល mouse ត្រូវបានចុចអូស ឬផ្លាស់ទី
TextListener	មាន method មួយដើម្បីបានតម្លៃរបស់ text បានប្រែប្រួល
WindowListener	មាន methods ប្រាំពីរ ដើម្បីបានសកម្មភាព window ទាំងអស់

៦.១ អំពី ActionListener interface

Interface **មាន** method **មួយឈ្មោះ** actionPerformed() **ដែលត្រូវបានប្រើនៅពេល** action event **កើតឡើង ។**

ទម្រង់របស់វា៖

```
void actionPerformed(ActionEvent ae)
```

៦.២ អំពី AdjustmentListener interface

Interface **មាន** method **មួយឈ្មោះ** adjustmentValueChanged() **ដែលត្រូវបានប្រើនៅពេល** adjustment event **កើតឡើង ។**

ទម្រង់របស់វា៖

```
void adjustmentValueChanged(AdjustmentEvent ae)
```

៦.៣ អំពី ComponentListener interface

Interface **មាន** methods **ចំនួនបួនដែលត្រូវបានប្រើនៅពេល**component **មួយត្រូវប្តូរទំហំ ផ្លាស់ទី បង្ហាញ ឬបិទបាំង ។**

ទម្រង់ទូទៅនៃ methods **ទាំងនោះ៖**

```
void componentResized(ComponentEvent ce)
void componentMoved(ComponentEvent ce)
void componentShown(ComponentEvent ce)
void componentHidden(ComponentEvent ce)
```

៦.៤ អំពី ContainerListener interface

Interface **មាន** methods **ចំនួន ២ នៅពេលដែលត្រូវការ** add **ធាតុចូល** container **យើងប្រើ** componentAdded **និងនៅពេលដែលត្រូវការលុប យើងប្រើ** componentRemoved()**។**

ទំរង់ទូទៅ៖

```
void componentAdded(ContainerEvent ce)
void componentRemoved(ContainerEvent ce)
```

៦.៥ <u>អំពី FocusListener interface</u>

Interface **មាន** methods **ចំនួន ២ ។ នៅពេល** component **មួយមាន** focus **របស់** keyboard **គេប្រើ** focusGained() **ហើយនៅពេល** component **មួយបាត់** focus **របស់** keyboard **គេប្រើ** focusLost()**។**

ទម្រង់ទូទៅ៖

```
void focusGained(FocusEvent fe)
void focusLost(FocusEvent fe)
```

៦.៦ <u>អំពី ItemListener interface</u>

Interface **មាន** method **មួយឈ្មោះ** itemStateChanged() **ដែលត្រូវបានប្រើនៅពេលសភាពលក្ខណៈរបស់** item **មួយផ្លាស់ប្តូរ ។**

ទម្រង់របស់វា៖

```
void itemStateChanged(ItemEvent ie)
```

៦.៧ <u>អំពី KeyListener interface</u>

Interface **មាន** methods **ចំនួន ៣ ។** keyPressed() **និង** keyReleased() **ប្រើនៅពេល** key **មួយត្រូវបានចុចសង្កត់និងព្រលែង។** keyTyped() **ត្រូវបានប្រើនៅពេលតួអក្សរមួយត្រូវបានវាយបញ្ចូល។**

ទម្រង់ទូទៅ៖

```
void keyPressed(KeyEvent ke)
void keyReleased (KeyEvent ke)
void keyTyped(KeyEvent ke)
```

៦.៨ <u>អំពី MouseListener interface</u>

Interface **មាន** methods **ចំនួន ៥ គឺ** mouseClicked(), mouseEntered(), mouseExited() mousePressed() **និង** mouseReleased() **។**

ទម្រង់ទូទៅ៖

```
void mouseClicked(MouseEvent me)
void mouseEntered(MouseEvent me)
void mouseExited(MouseEvent me)
void mousePressed(MouseEvent me)
void mouseReleased(MouseEvent me)
```

៦.៩ <u>អំពី MouseMotionListener interface</u>

Interface **មាន** methods **ចំនួន ២ ។** mouseDragged() **ត្រូវបានប្រើនៅពេលដែលគេចុចសង្កត់** mouse **ហើយរំកិលទៅដាក់ទីណាមួយនិង** mouseMoved() **ប្រើនៅពេលដែលយើងរំកិល**mouse**។**

ទម្រង់ទូទៅ៖

```
void mouseDragged(MouseEvent me)
void mouseMoved(MouseEvent me)
```

៦.១០ អំពី TextListener interface

Interface **មាន** methods **ចំនួន១ គឺ** textValueChanged()**។**

ទម្រង់ទូទៅ៖

```
void textValueChanged(TextEvent te)
```

៦.១១ អំពី WindowListener interface

Interface **មាន** methods **ចំនួន៧។** windowActivated(), windowDeactivated(), windowClosed(), windowOpened(), windowIconified(), windowDeiconified()**និង** windowClosing() **។**

ទម្រង់ទូទៅ៖

```
void windowActivated(WindowEvent we)
void windowDeactivated(WindowEvent we)
void windowClosed(WindowEvent we)
void windowClosing(WindowEvent we)
void windowIconified(WindowEvent we)
void windowDeiconified(WindowEvent we)
void windowOpened(WindowEvent we)
```

៧. ការប្រើប្រាស់ Delegation Event Model

ដើម្បីអនុវត្តតាមលំនាំdelegation event model **យើងត្រូវអនុវត្តនូវជំហានទាំងពីរដូចខាងក្រោម៖**

១- អនុវត្តនូវ interface **ដែលត្រូវគ្នានឹង** listener **ដូចនេះវានឹងទទួលបាននូវប្រភេទ** event **ដែលត្រូវការចាំបាច់**

២- ប្រើ code **ដើម្បីកត់ត្រា និងមិនកត់ត្រា(បើសិនជាត្រូវការ)នូវ** listener **ដូចទៅនឹងអ្នកទទួលកំណត់ត្រា** event

គួរចងចាំថា source **មួយអាចបង្កើតនូវ** event **ច្រើនប្រភេទ។** event **នីមួយៗត្រូវតែកត់ត្រាទុកដោយឡែកពីគ្នា។** object **មួយអាចកត់ត្រាទុកដើម្បីទទួល** events **ច្រើនប្រភេទបានដែរ ប៉ុន្តែវាត្រូវតែប្រើគ្រប់** interfaces **ដែលជាតម្រូវការ ដើម្បីទទួលបាន** events **ទាំងនេះ។**

៨. ការប្រើប្រាស់ Mouse Event

ដើម្បីអាចប្រើ mouse event **បាន យើងត្រូវតែអនុវត្តនូវ** interfaces **ឈ្មោះ** MouseListener **និង** MouseMotionListener **។**

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
class DemoMouseEvent extends Frame implements MouseListener,MouseMotionListener{
    String sms="";
```

```
String s="-";
String str="Mouse Events";
int mouseX=20,mouseY=20;
DemoMouseEvent(){
        super ("Demo Mouse Events!");
        setSize(300,250);
        addMouseListener(this);
        addMouseMotionListener(this);
        setVisible(true);
}
public void mouseClicked(MouseEvent me){
        sms="Mouse clicked,at x= " + me.getX() + " and Y = " + me.getY();
        setTitle(sms);
}
public void mouseEntered(MouseEvent me){
        mouseX=0;
        mouseY=10;
        sms="Mouse has Enetered";
        setTitle(sms);
}
public void mouseExited(MouseEvent me){
        mouseX=0;
        mouseY=10;
        sms="Mouse has Exited";
        setTitle(sms);
}
public void mousePressed(MouseEvent me){
        mouseX=me.getX();
        mouseY=me.getY();
        sms="Mouse has Pressed Down";
        setTitle(sms);
}
public void mouseReleased(MouseEvent me){
        mouseX=me.getX();
        mouseY=me.getY();
        sms="Mouse has Up";
        setTitle(sms);
}
public void mouseDragged(MouseEvent me){
        mouseX=me.getX();
```

```
            mouseY=me.getY();
            s="*";
            str="Mouse dragging at: " + mouseX + " , " + mouseY;
            repaint();
        }
        public void mouseMoved(MouseEvent me){
            str="Mouse has moved to " + me.getX() + " , "+me.getY();
            repaint();
        }
        // display sms in applet window at current x, y
        public void paint(Graphics g){
            g.drawString(s,mouseX,mouseY);
            g.drawString(str,mouseX,mouseY);
        }
        public static void main(String[] agr){
            new DemoMouseEvent();
        }
    }
```

ចូរពិនិត្យមើលឧទាហរណ៍ខាងលើ យើងឃើញថា class DemoMouseEvent ទទួលលក្ខណៈពី Frame class ហើយ implements ទៅលើ interface ពីរ គឺ MouseListener និង MouseMotionListener។ Frame គឺជា source ហើយក៏ជា listener សម្រាប់ event ទាំងនេះ។ វាអាចដំណើរការបានដោយសារ Component ជាអ្នកផ្តល់នូវ addMouseListener() និង addMouseMotionListener() method ហើយក៏ជា superclass របស់ Frame ដែរ។

នៅខាងក្នុង Constructor របស់ DemoMouseEvent កត់ត្រាខ្លួនវាជា listener សម្រាប់ mouse events ។ វាធ្វើឡើងដោយប្រើ addMouseListener() និង addMouseMotionListener() ។

ទម្រង់របស់វា៖

synchronized void addMouseListener(MouseListener ml)

synchronized void addMouseMotionListener(MouseMotionListener mml)

បន្ទាប់មក Frame អនុវត្តរាល់ methods ទាំងអស់កំណត់ដោយ MouseListener និង Mouse-MotionListener interface ។ Methods ទាំងនេះជាអ្នកប្រើ events ចំពោះ mouse events ប្រភេទផ្សេងៗ។

៩. <u>អំពី Adapter Classes</u>

Java 2 បានផ្តល់នូវលក្ខណៈពិសេសមួយដែលហៅថា Adapter class ដែលអាចជួយសម្រួលនូវការច្នៃប្រឌិតនៃការប្រើប្រាស់ event តាមស្ថានភាពជាក់ស្តែង ។ adapter class ផ្តល់នូវការអនុវត្តទទេនៃរាល់ methods ទាំងអស់មានក្នុង event listener interface ។ adapter class មានសារៈសំខាន់ណាស់នៅពេលដែលយើងចង់ទទួលនិងដំណើរការ events តែមួយចំនួនប៉ុណ្ណោះដែលប្រើដោយevent listener interface ពិសេស។ គេអាចកំណត់ class ថ្មីមួយដើម្បីធ្វើសកម្មភាពដូចនឹង event listener តាមរយៈការទទួលលក្ខណៈពី adapter class ហើយប្រើតែ events ដែលគេត្រូវការប៉ុណ្ណោះ។

ឧទាហរណ៍ MouseMotionListener **មាន** methods **ពីរគឺ** mouseDragged() **និង** mouseMoved()។ **សញ្ញាណនៃ** methods **ទទេ ទាំងនេះ គឺដូចទៅនឹងអ្វីដែលបានកំណត់នៅក្នុង** MouseMotionListener interface **ដែរ បើសិនជាយើងត្រូវការតែ** mousseDragged event **ទេ គឺយើងគ្រាន់តែទទួលលក្ខណៈពី** MouseMotionAdapter **ហើយប្រើតែ** method mouseDragged() **ជាការស្រេច។ វាអាចកាត់បន្ថយការសរសេរន្លៃ** mouseMoved() **បាន។**

តារាងខាងក្រោមបង្ហាញពី Adapter class និង interface ដែលត្រូវគ្នា

Adapter Class	Listener interface
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
class DemoAdapter extends Frame{
        DemoAdapter(){
                super("Demo adapter class.");
                setSize(300,300);
                addMouseListener(new MyMouseAdapter(this));
                addMouseMotionListener(new MyMouseMotionAdapter(this));
                setVisible(true);
        }
        public static void main(String[] agr){
                new DemoAdapter();
        }
}
class MyMouseAdapter extends MouseAdapter{
        DemoAdapter demoAdapter;
        public MyMouseAdapter (DemoAdapter demoAdapter){
                this.demoAdapter = demoAdapter;
        }
        public void mouseClicked(MouseEvent me){
                demoAdapter.setTitle("Mouse has been clicked.");
        }
}
```

```
class MyMouseMotionAdapter extends MouseMotionAdapter{
        DemoAdapter demoAdapter;
        public MyMouseMotionAdapter(DemoAdapter demoAdapter){
                this.demoAdapter=demoAdapter;
        }
        public void mouseDragged(MouseEvent me){
                demoAdapter.setTitle("Mouse dragged");
        }
}
```

១០. អំពី Adapter Inner Classes

នៅក្នុងករណីនេះ យើងនឹងលើកយកលក្ខណៈ inner class **ដែលយើងបានសិក្សាម្តងរួចមកហើយមកប្រើជាមួយនឹង** Adapter class **ដើម្បីជួយសម្រួលដល់ការសរសេរ**code **នៅពេលដែលយើងប្រើ** event adapter class **។**

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
class DemoAdapterInner extends Frame{
        DemoAdapterInner(){
                super("Demo Adapter inner!");
                setSize(400,300);
                addMouseListener(new MyMouseAdapter());
                addMouseMotionListener(new MyMouseMotionAdapter());
                setVisible(true);
        }
        public static void main(String[] agr){
                new DemoAdapterInner();
        }
        class MyMouseAdapter extends MouseAdapter{
                public void mouseClicked(MouseEvent me){
                        setTitle("Mouse clicked!");
                }
        }
        class MyMouseMotionAdapter extends MouseMotionAdapter{
                public void mouseDragged(MouseEvent me){
                        setTitle("Mouse Dragged!");
                }
        }
}
```

១១. អំពី Anonymous Adapter Inner Classes

Anonymous Adapter Inner Classes ជា class ដែលមិនមានការប្រើឈ្មោះ។ វាជួយសម្រួលដល់ការសរសេរ code នៃការប្រើប្រាស់ event ។

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
class DemoAnonymousAdapterInner extends Frame{
        DemoAnonymousAdapterInner(){
                super("Demo Anonymous Adapter Inner");
                setSize(400,350);
                addMouseListener(new MouseAdapter(){
                        public void mouseClicked(MouseEvent me){
                                setTitle("Mouse clicked!!!!");
                        }
                });
                addMouseMotionListener(new MouseMotionAdapter(){
                        public void mouseDragged(MouseEvent me){
                                setTitle("Mouse dragged!!!");
                        }
                });
                setVisible(true);
        }
        public static void main(String[] agr){
        new DemoAnonymousAdapterInner();
        }
}
```

ចូរសង្កេត វាមាន class តែមួយគត់នៅក្នុងកម្មវិធីនេះគឺ DemoAnonymousAdapterInner ។ ដោយ constructor របស់វាបានហៅ method addMouseListener() មកប្រើ។ argument របស់វាគឺជាកន្សោមមួយដែលបានកំណត់និងបង្កើតនូវ object នៃ anonymous inner class ។

ទម្រង់ new MouseAdapter(){...} បញ្ជាក់ប្រាប់ទៅ compiler ថា code នៅចន្លោះសញ្ញា { } កំណត់នូវ anonymous inner class ។ លើសពីនេះ class នោះទទួលលក្ខន:ពី MouseAdapter។ class ថ្មីនេះមិនមានឈ្មោះទេក៏ប៉ុន្តែវាបង្កើត object ដោយស្វ័យប្រវត្តនៅពេលកន្សោមនោះចាប់ប្រតិបត្តការ។ ដោយសារ anonymous inner class ត្រូវបានកំនត់នៅក្នុង code នៃ DemoAnonymousAdapterInner នោះវាមានសិទ្ធចូលប្រើប្រាស់អញ្ជាត និង methods ទាំងអស់នៅក្នុង code នៃ class នោះ។ ហេតុនេះវាអាចហៅ method setTitle() មកប្រើប្រាស់ដោយផ្ទាល់បាន។

End

មេរៀនទី ១១

AWT Controls, Layout Managers និង Menus

១. <u>អំពី AWT Classes</u>

AWT class វាជា package ដ៏ធំមួយដែលស្ថិតនៅក្នុង package java.awt ។ គោលបំណងសំខាន់របស់ AWT គឺផ្តល់នូវលក្ខណៈដែលប្រើលើ applet windows ហើយវាក៏អាចប្រើសម្រាប់បង្កើត windows ដែលមានលក្ខណៈ stand-alone ផងដែរ និងអាចដំណើរrrការនៅក្នុង GUI ដូចជា windows។

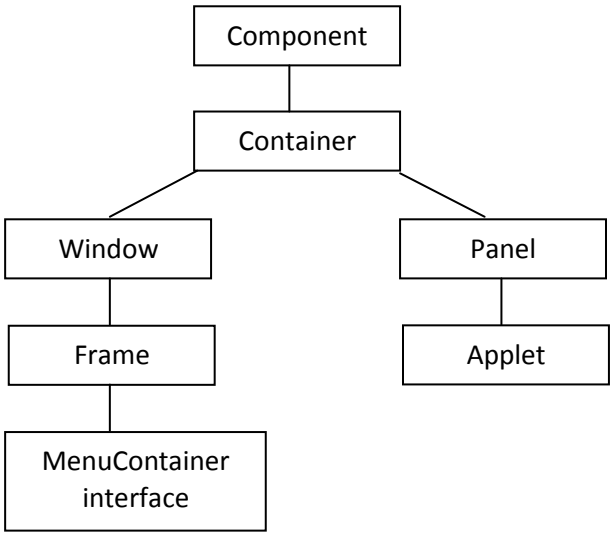
ខាងក្រោមនេះជាតារាងដែលបង្ហាញពីAWT classes

Class Name	អធិប្បាយ
AWTEvent	ផ្ទុកនូវ AWT events
AWTEventMulticaster	នាំយក events ទៅឲ្យ listeners ជាច្រើន
BoderLayout	ជាLayout ដែលមានប្រាំទីតាំង North, South, East, West និង Center សម្រាប់ដាក់ Component
Button	គឺជា Command Button
Canvas	គឺជា window ទទេ
CardLayout	ជា Layout មួយប្រភេទមានលក្ខណៈដូច Card
Checkbox	ប្រើសម្រាប់បង្កើត check box
CheckboxGroup	ប្រើសម្រាប់បង្កើត check box ជាក្រុម (Option Button)
CheckboxMenuItem	ប្រើសម្រាប់បង្កើត menu item ដែលមានលក្ខណៈ check box
Choice	ប្រើសម្រាប់បង្កើត list ដែលមានលក្ខណៈ pop-up
Color	ប្រើសម្រាប់បង្កើតនូវ object Color
Component	ជា superclass ដែលមានលក្ខណៈអរូបីសម្រាប់ AWT component ផ្សេងៗប្រើ
Container	ជា subclass នៃ component ដែលអាចផ្ទុក components ផ្សេងៗទៀត
Cursor	នាំមកជាមួយនូវ cursor ដែលមានលក្ខណៈជា bitmapped
Dialog	ប្រើសម្រាប់បង្កើត dialog window
Dimension	ប្រើសម្រាប់កំណត់ទំហំនៃ object មាន width និង height
Event	ផ្ទុកនូវ events ទាំងឡាយ
EventQueue	Events បន្តគ្នា
FileDialog	ប្រើសម្រាប់នូវ File dialog box
FlowLayout	ជា Layout មួយប្រភេទដែលរៀប component ពីឆ្វេងទៅស្តាំ ពីលើចុះក្រោម
Font	ប្រើសម្រាប់បង្កើត object នៃ font
FontMetrics	ប្រើសម្រាប់ប្រាប់ពីព័ត៌មានផ្សេងៗដែលទាក់ទងនឹង font
Frame	បង្កើតនូវ window ទូទៅមួយដែលមាន title bar, ជ្រុងប្តូរទំហំ និង menu bar
Graphics	ផ្តល់នូវមធ្យោបាយដែលអាចឲ្យយើងធ្វើការដែលទាក់ទងនឹង graphics
GraphicsDevice	ពណ៌នាពីឧបករណ៍ graphic ដូចជា អេក្រង់ ឬ printer
GraphicsEnvironment	ពណ៌នានូវការប្រមូលផ្តុំ fonts និង objects នៃ graphics device ដែលអាចប្រើបាន

GridBagConstraints	កំណត់លក្ខណៈកំរិតផ្សេងៗដែលទាក់ទងទៅនឹង GridBagLayout class
GridBagLayout	ជា Layout មួយប្រភេទដែលមានការតំរៀប components ទៅតាមការកំរិតដែលត្រូវបញ្ជាក់ដោយ GridBagConstraints
GridLayout	ជា Layout មួយប្រភេទដែលរៀប component ទៅតាមលក្ខណៈជា Grid
Image	សម្រាប់រូបភាព
Insets	សម្រាប់ការកំណត់ជាយខាងនៃ container មួយ
Label	សម្រាប់បង្កើត label
List	សម្រាប់បង្កើត list
MediaTracker	សម្រាប់គ្រប់គ្រងនូវ Object media
Menu	សម្រាប់បង្កើត Menu
MenuBar	សម្រាប់បង្កើត Menu bar
MenuComponent	ជា abstract class ដែលប្រើដោយ menu classes ផ្សេងៗ
MenuItem	សម្រាប់បង្កើត menu item
MenuShortcut	សម្រាប់បង្កើតនូវ shortcut key សម្រាប់ menu នីមួយៗ
Panel	ជា subclass នៃ container ដែលមានលក្ខណៈរូបី
Point	ផ្ទុកនូវកូអរដោនេ x និង y
Polygon	សម្រាប់បង្កើត Polygon
PopupMenu	សម្រាប់បង្កើត pop-up menu
PrintJob	ជា abstract class ដែលប្រើសម្រាប់ការងារបោះពុម្ព
Rectangle	សម្រាប់បង្កើត Rectangle
Scrollbar	សម្រាប់បង្កើត Scroll bar
ScrollPane	ជាcontainer ដែលផ្តល់នូវ scroll bar តាមទិសដៅដេក ឬឈរ សម្រាប់ component មួយផ្សេងទៀត
SystemColor	ផ្ទុកពណ៌របស់ GUI widgets ដូចជា windows, scroll bar …
TextArea	សម្រាប់បង្កើត Text box ដែលអាចសរសេរបានច្រើនបន្ទាត់
TextComponent	គឺជា superclass នៃ text area និង text field
TextField	សម្រាប់បង្កើត Text box ដែលអាចសរសេរបានតែមួយបន្ទាត់
Toolkit	ជា abstract class ដែលប្រើដោយ AWT
Window	សម្រាប់បង្កើត window មួយដោយគ្មាន ស៊ុម គ្មាន menu bar និងគ្មាន title

២. <u>អំពី Component</u>

Component គឺជា Abstract class មួយដែលនាំមកនូវលក្ខណៈទាំងអស់របស់ Component ដែលមើលឃើញ។ វាបង្កើតឲ្យមាន method ដែលមានលក្ខណៈ public ចំនួនរាប់រយ ហើយ methods ទាំងនេះទទួលបន្ទុកនូវការគ្រប់គ្រងនូវ events ដូចជាការបញ្ចូលតាម mouse និង Keyboard ការកំណត់ទីតាំងនិងទំហំ windows និងការបង្ហាញព័ត៌មានសារជាថ្មីនៅលើ window។ object នៃ component ទទួលបន្ទុកចងចាំពណ៌នៃ background, foreground និង font អក្សរដែលកំពុងប្រើប្រាស់។



២.១ អំពី Container

Container គឺជា subclass នៃ Component វាអាចឲ្យ Object នៃ component ផ្សេងទៀតដាក់បញ្ចូលក្នុងវាបាន។ Container មួយទទួលបន្ទុកសម្រាប់រៀប components ទាំងឡាយទៅតាមទីតាំងដែលកំណត់ដោយ Layout management ។

២.២ អំពី Panel

Panel គឺជា subclass នៃ Container ហើយវាក៏ជា superclass នៃ Applet ។ Panel គឺជា window ដែលគ្មាន title bar, menu bar ឬ ស៊ុមឡើយ។ លក្ខណៈនេះដែលគេមិនអាចមើលឃើញរបស់ទាំងអស់នេះនៅពេល applet មួយកំពុងដំណើរការលើ browser ។ កាលណាយើងដំណើរការ applet មួយដោយប្រើ applet viewer ពេលនោះ applet viewer វាផ្តល់នូវ title bar និង ស៊ុម។

- **add()** ប្រើសម្រាប់ add Components ផ្សេងៗចូលទៅក្នុង object panel
- **setLocation()** សម្រាប់កំណត់ទីតាំង Component នៅក្នុង panel
- **setSize()** សម្រាប់កំណត់ទំហំរបស់ Component នៅក្នុង panel
- **setBounds()** សម្រាប់កំណត់ top និង left, កំពស់ និង បណ្តោយ របស់ component

២.៣ អំពី Window

Window class បង្កើតនូវ window ថ្នាក់កំពូល វាពុំស្ថិតនៅក្នុង object ណាមួយឡើយ វាឈរផ្ទាល់នៅលើ destop ។ ជាទូទៅយើងមិនបង្កើត object នៃ window ដោយផ្ទាល់ទេ គឺយើងប្រើ subclass របស់ window តាមរយៈ Frame ។

២.៤ អំពី Frame

Frame គឺជា subclass នៃ windows ដែលមាន title bar, menu bar, ស៊ុម និងជ្រុងដែលអាចប្តូរទំហំ។ កាលណាយើងបង្កើត Object នៃ Frame នៅក្នុង applet មួយ ពេលនោះវានឹងបង្ហាញអក្សរ "Warning : Applet Window" អក្សរនេះប្រាប់ដល់អ្នកប្រើប្រាស់ឲ្យដឹងថា window ដែលយើងមើលឃើញនេះបង្កើតឡើងដោយ Applet មិនមែនតាមរយៈដំណើរការកម្មវិធីនៃ computer ឡើយ។ នៅពេលដែល Frame window មួយបានបង្កើតឡើងតាមរយៈកម្មវិធី វាមានលក្ខណៈជា window ធម្មតាប្រសើរជាង Applet ។

Constructor និង methods ដែលទាក់ទងនឹង Frame

Frame()	បង្កើត window មួយដែលមិនមាន title
Frame(String title)	បង្កើត window មួយដែលមាន title
void setSize(int newWidth,int newHeight) void setSize(Dimension newSize)	ប្រើសម្រាប់កំណត់ទំហំរបស់ window ព្រោះយើងមិនអាចកំណត់ទំហំរបស់ window នៅក្នុងពេលដែលយើងកំពុងបង្កើតវាបានទេ
Dimension getSize()	ប្រើសម្រាប់ទទួលយកនូវទំហំរបស់ window កំពុងប្រើប្រាស់
void setVisible(boolean visibleFlag)	សម្រាប់បង្ហាញឲ្យឃើញ (true) និងលាក់(hide)មិនឲ្យឃើញ(false)
void setTitle(String newTitle)	ប្រើសម្រាប់កំណត់ title ទៅឲ្យ window
ដើម្បីប្រើ event បិទ window យើងត្រូវប្រើ method windowClosing() របស់ WindowListener interface	

ឧទាហរណ៍៖ ខាងក្រោមនេះជាកម្មវិធីមួយដែលបង្កើត Frame Window មួយដែលឆ្លើយតបទៅនឹងការចុច mouse និងការចុច key។

```
import java.awt.*;
import java.awt.event.*;
class DemoAppWin extends Frame{
        String keysms="";
        String mousesms="";
        int mouseX=30,mouseY=30;
        DemoAppWin(){
                addKeyListener(new MyKeyAdapter(this));
                addMouseListener(new MyMouseAdapter(this));
                addWindowListener(new MyWindowAdapter());
        }
        public void paint(Graphics g){
                g.drawString(keysms,10,40);
                g.drawString(mousesms,mouseX,mouseY);
        }
        public static void main(String[] agr){
                DemoAppWin appwin=new DemoAppWin();
                appwin.setSize(new Dimension(300,200));
                appwin.setTitle("Welcome!");
                appwin.setVisible(true);
        }
}
```

```
class MyKeyAdapter extends KeyAdapter{
        DemoAppWin appWindow;
        public MyKeyAdapter(DemoAppWin appWindow){
                this.appWindow=appWindow;
        }
        public void keyTyped(KeyEvent ke){
                appWindow.keysms+=ke.getKeyChar();
                appWindow.repaint();
        };
}
class MyMouseAdapter extends MouseAdapter{
        DemoAppWin appWindow;
        public MyMouseAdapter(DemoAppWin appWindow){
                this.appWindow=appWindow;
        }
        public void mousePressed(MouseEvent me){
                appWindow.mouseX=me.getX();
                appWindow.mouseY=me.getY();
                appWindow.mousesms="Mouse down at" + appWindow.mouseX + ", " +
appWindow.mouseY;
                appWindow.repaint();
        }
}
class MyWindowAdapter extends WindowAdapter{
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
}
```

២.៥ អំពី Canvas

Canvas គឺជាប្រភេទ window មួយ ដែលមិនមែនជាចំណែកនៃលំដាប់ថ្នាក់របស់ applet window ឬ frame window ទេ តែវានាំមកនូវ window ទទេមួយដែលអាចគូរអ្វីៗបាននៅលើវា។

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
class MyCan{
        public static void main(String[] agr){
                Frame f = new Frame("Paint Example");
                f.add(new MainCanvas(),BorderLayout.CENTER);
                f.setBounds(200,300,60,90);
```

```
                f.show();
                f.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent we){
                                System.exit(0);
                        }
                });
        }
}
class MainCanvas extends Canvas{
        int x,y;
        int lastX=0,lastY=0;
        MainCanvas(){
                addMouseListener(new MouseAdapter(){
                        public void mousePressed(MouseEvent me){
                                x=me.getX();
                                y=me.getY();
                                lastX=x;
                                lastY=y;
                        }
                });
                addMouseMotionListener(new MouseMotionAdapter(){
                        public void mouseDragged(MouseEvent me){
                                Graphics g=getGraphics();
                                x=me.getX();
                                y=me.getY();
                                g.drawLine(lastX,lastY,x,y);
                                lastX=x;
                                lastY=y;
                        }
                });
        }
        public void paint(Graphics g){
                setBackground(Color.yellow);
                g.setColor(Color.blue);
                g.drawOval(0,0,getSize().width-1,getSize().height-1);
        }
}
```

៣. អំពី Controls

AWT បានផ្តល់នូវ Control ជាច្រើនដូចខាងក្រោម៖

Labels	សម្រាប់ដាក់ឈ្មោះឲ្យ Component ផ្សេងៗ
Push Buttons	សម្រាប់បង្កើត button
Check Boxes	សម្រាប់បង្កើត CheckBox
Choice lists	សម្រាប់បង្កើត Combobox
Lists	សម្រាប់បង្កើត listbox
Scroll bars	សម្រាប់បង្កើត Scrollbar
Text editing	សម្រាប់បង្កើត text field និង text area

Methods ដែលប្រើមាន៖

- Component add(Component compObject) ប្រើសម្រាប់ add control ចូលក្នុង window
- void remove(Component compObject) ប្រើសម្រាប់ លុប control ណាមួយចេញពី window
- void removeAll() សម្រាប់លុប controls ទាំងអស់ពី window

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
class DemoContainerEvent extends Frame implements ContainerListener, ActionListener{
        DemoContainerEvent(){
                super(" Demo Container Event");
                //create button
                Button bt=new Button("New Button");
                //listen for event
                bt.addActionListener(this);
                addContainerListener(this);
                //Layout component
                setLayout(new FlowLayout());
                add(bt);
                addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent we){
                                System.exit(0);
                        }
                });
                setBounds(250,210,350,300);
                setVisible(true);
        }
        //container event handler methods
        public void componentAdded(ContainerEvent ce){
```

```
                System.out.println("Component added: " + ce.getChild());
                System.out.println("There are now "+
                                ce.getContainer().getComponentCount()+ " children." );
        }
        public void componentRemoved(ContainerEvent ce){
                System.out.println("Component removed : " + ce.getChild());
                System.out.println("There are now "+
                                ce.getContainer().getComponentCount()+ " children." );
        }
        int count;
        //Action handler method
        public void actionPerformed(ActionEvent ae){
                if (ae.getActionCommand().equals("New Button")){
                        //add button
                        Button bt=new Button("Removed me" + (count++));
                        //Listen for event
                        bt.addActionListener(this);
                        add(bt);
                        bt.setVisible(true);
                        //relayout container
                        validate();
                }else if (ae.getActionCommand().startsWith("Removed me")){
                        remove((Component) ae.getSource());
                        validate();
                }
        }
        public static void main (String[] args) {
                new DemoContainerEvent();
        }
}
```

៣.១ <u>ការប្រើ Label</u>

ទម្រង់ Constructor ៖

Label()

Label(String str)

Label(String str, int how)

- str ជា Caption របស់ Label
- how សម្រាប់តម្រឹមអក្សរ(alignment) មាន label.LEFT, label.RIGHT និង label.CENTER

មាន methods មួយចំនួន៖

- **void setText()** សម្រាប់ផ្តល់ caption ទៅឲ្យ label

- **String getText()** សម្រាប់ទទួលយក caption ពី label
- **void setAlignment(int how)** សម្រាប់កំណត់ alignment ទៅឲ្យ label
- **int getAlignment()** សម្រាប់ទទួលយក alignment ពី label

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoLabel  extends Frame{
        Label one,two,three;
   DemoLabel(){
      super("Demo Label");
        addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent we){
                        System.exit(0);
                    }
                });
      setLayout(new FlowLayout());
      setSize(300,250);
      one=new Label("one");
      two=new Label("two");
      three=new Label("three");
      add(one);
      add(two);
      add(three);
        setVisible(true);
   }
   public static void main(String[] args) {
      new DemoLabel();
   }
}
```

៣.២ ការប្រើ Button

ទម្រង់ Constructor ៖

Button()

Button(String str) // str ជា Caption ដែលត្រូវដាក់នៅលើ button នោះ

មាន methods មួយចំនួនដែលប្រើជាមួយវា៖

void setLabel(String str) សម្រាប់ប្តូរ Caption នៅលើ button

String getLabel()សម្រាប់ទាញយក Caption ពី button មកវិញ

String getActionCommand() ដើម្បីទទួលយក label នៅលើ button នៅពេលដែល button នោះទទួលនូវ Action ណាមួយ

getSource() សម្រាប់ទាញយក object នៃ button មកប្រៀបធៀបគ្នា

ឧទាហរណ៍ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoButton extends Frame implements ActionListener {
    String sms="";
    Button yes,no,maybe;
    Label lb=new Label();
  public DemoButton() {
      super("Demo button");
      addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                  System.exit(0);
            }
      });
      setLayout(null);
      setSize(300,300);
      setVisible(true);
      yes=new Button("Yes");
      no=new Button("No");
      maybe=new Button("Undicided");
      yes.setBounds(100,40,40,30);
      no.setBounds(100,80,40,30);
      maybe.setBounds(80,120,70,30);
      add(yes);
      add(no);
      add(maybe);
      lb.setBounds(100,160,240,30);
      add(lb);
      yes.addActionListener(this);
      no.addActionListener(this);
      maybe.addActionListener(this);
      setVisible(true);
  }
  public void actionPerformed(ActionEvent ae){
      String str = ae.getActionCommand();
      if(str.equals("Yes")){
            sms="You pressed Yes.";
      }else if(str.equals("No")){
```

```
                sms="You pressed No.";
            }else{
                sms="You pressed Undicided.";
            }
            lb.setText(sms);
        }
        public static void main(String[] args) {
            new DemoButton();

        }
    }
```

ឧទាហរណ៍ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoButton1 extends Frame implements ActionListener {
        String sms="";
        Button btList[]=new Button[3];
        Label lb=new Label();
    public DemoButton1() {
            super("Demo Button List");
            addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                    System.exit(0);
                }
            });
            setLayout(null);
            setSize(300,300);
            Button yes=new Button("Yes");
            Button no=new Button("No");
            Button maybe=new Button("Undicided");
            yes.setBounds(100,40,40,30);
            no.setBounds(100,80,40,30);
            maybe.setBounds(80,120,70,30);
            //stored references to buttons as addes
            btList[0]=(Button) add(yes);
            btList[1]=(Button)add(no);
            btList[2]=(Button)add(maybe);
            lb.setBounds(100,160,240,30);
            add(lb);
```

```
            for(int i=0;i<3;i++)
                    btList[i].addActionListener(this);
            setVisible(true);
        }
        public void actionPerformed(ActionEvent ae){
            for(int i=0;i<3;i++){
                    if(ae.getSource()==btList[i])
                            sms="You pressed " + btList[i].getLabel();
            }
            lb.setText(sms);
        }
        public static void main(String[] args) {
           new DemoButton();

        }
    }
```

៣.៣ <u>ការប្រើ CheckBox</u>

ទម្រង់ Constructor ៖

Checkbox()

Checkbox(String str)

Checkbox(String str,boolean on)

Checkbox(String str,boolean on, CheckboxGroup cbGroup)

Checkbox(String str, CheckboxGroup cbGroup,boolean on)

- str ជា Caption ដាក់លើ Check box
- on ជាតម្លៃឲ្យវា checked ឬមិន checked (true/false)
- cbGroup ស្ថិតនៅក្នុង Group ណាមួយ បើមិនស្ថិតនៅក្នុងក្រុមណាមួយទេត្រូវដាក់ null

methods ដែលប្រើជាមួយវាមាន៖

boolean getState()

void setState(boolean on)

String getLabel()

void setLabel(String str)

ឧទាហរណ៍ ទី១ ៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoCheckBox extends Frame implements ItemListener {
    String sms="";
    Checkbox win95,win98,win2000,winXP,winVista;
    Label lb=new Label();
```

```
public DemoCheckBox() {
    super("Demo Check Box");
    addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
            System.exit(0);
        }
    });
    setLayout(new FlowLayout());
    setSize(600,300);
    win95=new Checkbox("Window 95",null,true);
    win98=new Checkbox("Window 98");
    win2000=new Checkbox("Window 2000");
        winXP=new Checkbox("Window XP Service page..");
        winVista=new Checkbox("Window Vista ultimate");
        add(win95);
        add(win98);
        add(win2000);
        add(winXP);
        add(winVista);
        win95.addItemListener(this);
        win98.addItemListener(this);
        win2000.addItemListener(this);
        winXP.addItemListener(this);
        winVista.addItemListener(this);
        setVisible(true);
}
public void itemStateChanged(ItemEvent ie){
    repaint();
}
public void paint(Graphics g){
    sms="Current state:";
    g.drawString(sms,6,80);
    sms="Window 95: " + win95.getState();
    g.drawString(sms,6,100);
    sms="Window 98: " + win98.getState();
    g.drawString(sms,6,120);
    sms="Window 2000: " + win2000.getState();
    g.drawString(sms,6,140);
    sms="Window XP: " + winXP.getState();
    g.drawString(sms,6,160);
```

```
            sms="Window Vista: " + winVista.getState();
            g.drawString(sms,6,180);
        }
        public static void main(String[] args) {
            new DemoCheckBox();

        }
    }
```

ឧទាហរណ៍ ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoColorCheckBox extends Frame implements ItemListener {
    Checkbox red, green,blue;
    Canvas canvas;
    public DemoColorCheckBox() {
            super("Demo Color check box");
            setLayout(new FlowLayout());
            setSize(300,200);
            red=new Checkbox("Red");
            green=new Checkbox("Green");
            blue=new Checkbox("Blue");
            red.addItemListener(this);
            green.addItemListener(this);
            blue.addItemListener(this);
            canvas =new Canvas();
            canvas.setBackground(Color.black);
            canvas.setSize(100,130);
            add(red);
            add(green);
            add(blue);
            add(canvas);
            setVisible(true);
            addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent we){
                            System.exit(0);
                    }
            });
    }
```

```
    public void itemStateChanged(ItemEvent ie){
        int rgb=0;
        if(red.getState()) rgb=0xff0000;
        if(green.getState()) rgb|=0x00ff00;
        if(blue.getState()) rgb|=0x0000ff;
        Color color=new Color(rgb);
        canvas.setBackground(color);
        canvas.repaint();
    }
    public static void main(String[] args) {
      new DemoColorCheckBox();
    }
}
```

៣.៤ <u>ការប្រើ CheckboxGroup</u>

CheckboxGroup គឺជា Option button មានន័យថាក្នុងមួយក្រុមរបស់វាយើងអាចជ្រើសរើសបានតែមួយគត់។

ទម្រង់ Constructor ៖

ChecboxGroup()

Methods ដែលប្រើជាមួយវា៖

-Checkbox getSelectedCheckbox() សម្រាប់ទទួលយក option ណាមួយដែលគេបានជ្រើសរើស

-void setSelectedCheckbox(Checkbox where) ឲ្យ select Checkbox ណាមួយ

ឧទាហរណ៍ ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoCheckBoxGroup extends Frame implements ItemListener {
    String sms="";
    Checkbox win95,win98,win2000,winXP,winVista;
    CheckboxGroup chkg;
  public DemoCheckBoxGroup() {
      super("Demo Check Box Group ");
      addWindowListener(new WindowAdapter(){
          public void windowClosing(WindowEvent we){
              System.exit(0);
          }
      });
      setLayout(new FlowLayout());
      setSize(600,200);
      chkg=new CheckboxGroup();
      win95=new Checkbox("Window 95",chkg,true);
```

```
            win98=new Checkbox("Window 98",chkg,false);
            win2000=new Checkbox("Window 2000",chkg,false);
                    winXP=new Checkbox("Window XP Service page..",chkg,false);
                    winVista=new Checkbox("Window Vista ultimate",chkg,false);
                    add(win95);
                    add(win98);
                    add(win2000);
                    add(winXP);
                    add(winVista);
                    win95.addItemListener(this);
                    win98.addItemListener(this);
                    win2000.addItemListener(this);
                    winXP.addItemListener(this);
                    winVista.addItemListener(this);
                    setVisible(true);
    }
    public void itemStateChanged(ItemEvent ie){
            repaint();
    }
    public void paint(Graphics g){
            sms="Current state: ";
            sms+=chkg.getSelectedCheckbox().getLabel();
            g.drawString(sms,100,100);
    }

    public static void main(String[] args) {
       new DemoCheckBoxGroup();

    }
}
```

ឧទាហរណ៍ ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoColorCheckGroup extends Frame implements ItemListener{
    Checkbox red, green,blue;
    Canvas canvas;
            CheckboxGroup chkg;
    public DemoColorCheckGroup() {
            super("Demo Color check box group");
```

```
        setLayout(new FlowLayout());
        setSize(300,200);
        chkg=new CheckboxGroup();
        red=new Checkbox("Red",chkg,true);
        green=new Checkbox("Green",chkg,true);
        blue=new Checkbox("Blue",chkg,true);
        red.addItemListener(this);
        green.addItemListener(this);
        blue.addItemListener(this);
        canvas =new Canvas();
        canvas.setBackground(Color.red);
        canvas.setSize(100,130);
        add(red);
        add(green);
        add(blue);
        add(canvas);
        setVisible(true);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
    }
        public void itemStateChanged(ItemEvent ie){
                Color color;
        Checkbox chk= chkg.getSelectedCheckbox();
        if(chk==red)
                color=Color.red;
        else if(chk==green)
                color=Color.green;
        else
                color=Color.blue;
        canvas.setBackground(color);
        canvas.repaint();
    }
    public static void main(String[] args) {
       new DemoColorCheckGroup();
    }
}
```

៣.៥ <u>ការប្រើ Choice</u>

Choice គឺជា Combobox ។ ទម្រង់ Constructor របស់វា៖

Choice()

Methods ដែលប្រើជាមួយវា៖

void addItem(String name) //សម្រាប់ add ធាតុចូលក្នុង combo box

void add(String name) //សម្រាប់ add ធាតុចូលក្នុង combo box

String getSelectedItem()//ផ្តល់ item ដែលបាន select

int getSelectedIndex()//ផ្តល់ index ដែល item នោះបាន select

int getItemcount() //ផ្តល់នូវចំនួន item ទាំងអស់

void select(int index) // ឲ្យធាតុនៅ index ណាមួយ select

void select(String name) //ឲ្យ select ធាតុនៅក្នុង combobox ដែលមានតម្លៃស្មើ name

String getItem(int index) // ទទួលយក item នៅ index ណាមួយ

ឧទាហរណ៍ ទី ១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoChoice extends Frame implements ItemListener {
        Choice os, browser;
        String sms="";
    public DemoChoice() {
        super("Demo using Choice");
        setLayout(new FlowLayout());
        setSize(300,200);
        os=new Choice();
        browser=new Choice();
        os.add("Window 95");
        os.add("Window 98");
        os.add("Window 2000");
        os.add("Window XP");
        os.add("Window Vista");
        browser.add("NetScape 1.1");
        browser.add("NetScape 2.x");
        browser.add("NetScape 3.x");
        browser.add("NetScape 4.x");
        browser.add("Internet Explorer 2.0");
        browser.add("Internet Explorer 3.0");
        browser.add("Internet Explorer 4.0");
        browser.select("Internet Explorer 4.0");
        add(os);
        add(browser);
        os.addItemListener(this);
        browser.addItemListener(this);
        setVisible(true);
```

```
            addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                    System.exit(0);
                }
            });
        }
        public void itemStateChanged(ItemEvent ie){
            repaint();
        }
        public void paint(Graphics g){
            sms="Curent OS: ";
            sms+=os.getSelectedItem();
            g.drawString(sms, 60,100);
            sms="Current Browser : ";
            sms+=browser.getSelectedItem();
            g.drawString(sms, 60,130);
        }
        public static void main(String[] args) {
          new DemoChoice();
        }
    }
```

ឧទាហរណ៍ ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoFontShow extends Frame implements ActionListener,ItemListener {
        Canvas cv;
        Choice fontChoice;
        Choice pointChoice;
        Checkbox bold;
        Checkbox italic;
        int smallestPoint=8;
        int largestPoint=72;
        int canvasWidth=300;
        int canvasHeight=100;
        String fontName;
        int fontStyle;
        int pointSize;
    public DemoFontShow() {
        cv= new Canvas();
        cv.setSize(canvasWidth,canvasHeight);
```

```
        add("Center",cv);
        Panel p=new Panel();
        createFontChoice();
        p.add(fontChoice);
        createPointChoice();
        p.add(pointChoice);
        bold=new Checkbox("Bold");
        italic=new Checkbox("Italic");
        Button quit=new Button("Quit");
        p.add(bold);
        p.add(italic);
        p.add(quit) ;
        bold.addItemListener(this);
        italic.addItemListener(this);
        quit.addActionListener(this);
        add("South",p);
        pack();
        showString();
        setVisible(true);
    }
    private void createFontChoice(){
        fontChoice=new Choice();
        fontChoice.addItemListener(this);
        GraphicsEnvironment ge= GraphicsEnvironment.getLocalGraphicsEnvironment();
        String fontList[];
        fontList=ge.getAvailableFontFamilyNames();
        for (int i=0;i<fontList.length;i++)
                fontChoice.add(fontList[i]);
        fontName=fontList[0];
    }
    private void createPointChoice(){
        pointChoice =new Choice();
        pointChoice.addItemListener(this);
        for(int i=smallestPoint;i<largestPoint;i+=2)
                pointChoice.add(i+" pt");
        pointChoice.select("20 pt");
        pointSize=20;
    }
    private void showString(){
        Font font=new Font(fontName,fontStyle,pointSize);
```

```
            Graphics g= cv.getGraphics();
            cv.paint(g);
            g.setFont(font);
            g.setColor(Color.black);
            g.drawString(fontName,5,canvasHeight/2);
        }
        public void itemStateChanged(ItemEvent ie){
            fontName=fontChoice.getSelectedItem();
            if(bold.getState() && italic.getState())
                    fontStyle=Font.BOLD|Font.ITALIC;
            else if(bold.getState())
                    fontStyle=Font.BOLD;
            else if(italic.getState())
                    fontStyle=Font.ITALIC;
            else
                    fontStyle=Font.PLAIN;
                    pointSize=pointChoice.getSelectedIndex();
                    pointSize+=smallestPoint;
                    showString();
        }
        public void actionPerformed(ActionEvent ae){
            System.exit(0);
        }
        public static void main(String[] args) {
          new DemoFontShow();
        }
    }
```

៣.៦ ការប្រើ List

ទម្រង់ Constructor ៖

List()

List(int numRows)

List(int numRows, boolean multipleSelect)

Methods ដែលប្រើជាមួយ៖

void add(String name)

void add(String name, int index)

String getSelectedItem()

int getSelectedIndex()

String[] getSelectedItems()

int[] getSelectedIndex()

int getItemCount()

void select(int index)

String getItem(int index)

ឧទាហរណ៍ ទី ១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoList extends Frame implements ActionListener {
        List os, browser;
        String sms="";
    public DemoList() {
        super("Demo using List");
        setLayout(new FlowLayout());
        setSize(300,300);
        os=new List(6,false);
        browser=new List(6,false);
        os.add("Window 95");
        os.add("Window 98");
        os.add("Window 2000");
        os.add("Window XP");
        os.add("Window Vista");
        browser.add("NetScape 1.1");
        browser.add("NetScape 2.x");
        browser.add("NetScape 3.x");
        browser.add("NetScape 4.x");
        browser.add("Internet Explorer 2.0");
        browser.add("Internet Explorer 3.0");
        browser.add("Internet Explorer 4.0");
        browser.select(1);
        add(os);
        add(browser);
        os.addActionListener(this);
        browser.addActionListener(this);
        setVisible(true);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
    }
```

```
    public void paint(Graphics g)  {
        int[] index;
        sms="Current OS : ";
        index=os.getSelectedIndexes();
        for(int i=0 ;i<index.length;i++)
            sms+=os.getItem(index[i]) + " " ;
        g.drawString(sms,20,200);
        sms="Current Browser : ";
        sms+=browser.getSelectedItem();
        g.drawString(sms,20,220);
    }
    public void actionPerformed(ActionEvent ae){
        repaint();
    }
    public static void main(String[] args) {
       new DemoList();
    }
}
```

ឧទាហរណ៍ ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoCursorList extends Frame implements Runnable,ItemListener {
    List ls=new List();
    String[] cursor={"DEFAULT_CURSOR","CROSSHAIR_CURSOR",
                     "TEXT_CURSOR","WAIT_CURSOR",
                     "SW_RESIZE_CURSOR","SE_RESIZE_CURSOR",
                     "NW_RESIZE_CURSOR","NE_RESIZE_CURSOR",
                     "N_RESIZE_CURSOR","S_RESIZE_CURSOR",
                     "W_RESIZE_CURSOR","E_RESIZE_CURSOR",
                     "HAND_CURSOR","MOVE_CURSOR"};
    public DemoCursorList() {
        super("Demo user cursor!");
        for(int i=0;i<cursor.length;i++)
            ls.add(cursor[i]);
        add(ls,BorderLayout.CENTER);
        ls.addItemListener(this);
        (new Thread(this)).start();
        setBounds(100,100,200,200);
        setVisible(true);
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
    }
```

```
        public void itemStateChanged(ItemEvent ie){
            setCursor(Cursor.getPredefinedCursor(ls.getSelectedIndex()));
        }
        public void run(){
            while(true){
                try{
                    Cursor cursor=getCursor();
                    setCursor(Cursor.getDefaultCursor());
                    Thread.sleep(2000);
                    setCursor(cursor);
                    Thread.sleep(10000);
                }catch(Exception e){}
            }
        }
        public static void main(String[] args) {
           new DemoCursorList();
        }
    }
```

៣.៧ <u>ការប្រើ Scrollbar</u>

ទម្រង់ Constructor ៖

Scrollbar()

Scrollbar(int style)

Scrollbar(int style, int initialValue, int thumbSize, int min, int max)

- style មាន Scrollbar.VERTICAL និង Scrollbar.HORIZONTAL
- initialValue កំណាត់តម្លៃដំបូង
- thumSize កំណាត់ចំនួនឯកតាតាងឲ្យកំពស់នៃ thum
- min, max តម្លៃអតិប្បរិមា និងអប្បបរមា

Methods ដែលប្រើជាមួយ៖

```
void setValue(int initialValue, int thumbSize,int min,int max)
int setValues(int initialValue,int thumbSize, int min, int max)
int getValue()
void setValue(int newValue)
int getMinimum()
int getMaximum()
void setUnitIncrement(int newIncr)
void setBlockIncrement(int newIncr)
```

ដើម្បីដំណើរការ event របស់ scroll bar បានយើងត្រូវប្រើ AdjustmentListener interface។ ដើម្បីដឹងនូវប្រភេទនៃ adjustment យើងត្រូវប្រើ getAdjustmentType()។ adjustment event មាន៖

BLOCK_DECREMENT នៅពេល event នៃ page-down កើតឡើង
BLOCK_INCREMENT នៅពេល event នៃ page-up កើតឡើង
TRACK នៅពេល event នៃ TRACK កើតឡើង
UNIT_DECREMENT នៅពេល button នៃសញ្ញាព្រួញចុះក្រោមបានចុច
UNIT_INCREMENT នៅពេល button នៃសញ្ញាព្រួញឡើងលើបានចុច

ឧទាហរណ៍ ទី ១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoScrollBar extends Frame implements AdjustmentListener,
                                                    MouseMotionListener {
   String sms="";
   Scrollbar vertScrollBar,horzScrollBar;
   Dimension d;
   public DemoScrollBar() {
        super("Demo using Scroll bar");
        setLayout(new FlowLayout());
        setSize(400,400);
        d=getSize();
        vertScrollBar=new Scrollbar(Scrollbar.VERTICAL,0,1,0,d.height);
        horzScrollBar=new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,d.width);
        add(vertScrollBar);
        add(horzScrollBar);
        vertScrollBar.addAdjustmentListener(this);
        horzScrollBar.addAdjustmentListener(this);
        addMouseMotionListener(this);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        setVisible(true);
   }
   public void adjustmentValueChanged(AdjustmentEvent ae){
        repaint();
   }
   public void mouseDragged(MouseEvent me){
        int x=me.getX();
        int y=me.getY();
        vertScrollBar.setValue(y);
```

```
            horzScrollBar.setValue(x);
            repaint();
        }
        public void mouseMoved(MouseEvent me){}
        public void paint(Graphics g){
            sms="Vertical : " + vertScrollBar.getValue();
            sms+=", horizontal : " + horzScrollBar.getValue();
            g.drawString(sms,10,168);
            g.drawString("*", horzScrollBar.getValue(),vertScrollBar.getValue());
        }

        public static void main(String[] args) {
           new DemoScrollBar();
        }
    }
```

ឧទាហរណ៍ ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoColorScrollBar extends Frame implements AdjustmentListener {
    Scrollbar rsb,gsb,bsb;
    Canvas cv;
    public DemoColorScrollBar() {
            super("Demo Scroll bar to change color.");
            setLayout(new FlowLayout());
            setSize(400,200);
            Label lblR=new Label("Red",Label.RIGHT);
            Label lblG=new Label("Green",Label.RIGHT);
            Label lblB=new Label("Blue",Label.RIGHT);
            rsb=new Scrollbar(Scrollbar.HORIZONTAL,255,5,0,255);
            gsb=new Scrollbar(Scrollbar.HORIZONTAL,0,5,0,255);
            bsb=new Scrollbar(Scrollbar.HORIZONTAL,0,5,0,255);
            rsb.addAdjustmentListener(this);
            gsb.addAdjustmentListener(this);
            bsb.addAdjustmentListener(this);
            add(lblR);add(rsb);
            add(lblG);add(gsb);
            add(lblB);add(bsb);
            cv=new Canvas();
            cv.setSize(68,68);
```

```
            cv.setBackground(new Color(255,0,0));
            add(cv);
            addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                    System.exit(0);
                }
            });
            setVisible(true);
        }
        public void adjustmentValueChanged(AdjustmentEvent ae){
            int r=rsb.getValue();
            int g=gsb.getValue();
            int b=bsb.getValue();
            Color c=new Color(r,g,b);
            cv.setBackground(c);
            cv.repaint();
        }
        public static void main(String[] args) {
           new DemoColorScrollBar();
        }
    }
```

៣.៨ ការប្រើ TextField

ទម្រង់ Constructor៖

TextField()

TextField(int numChars)

TextField(String str)

TextField(String str ,int numChars)

Methods ដែលប្រើជាមួយវា៖

String getText()

void setText(String str)

String getSelectedText()

void select(int startIndex, int endIndex)

boolean isEditable()

void setEditable(boolean canEdit)

void setEchoChar(char ch)

boolean echoCharIsSet()

char getEchoChar()

ឧទាហរណ៍ ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoTextField extends Frame implements ActionListener {
   TextField name,pass;
   public DemoTextField() {
        super("Demo using Text Field!");
        setLayout(new FlowLayout());
        setSize(468,168);
        Label lblName=new Label("Name : ",Label.RIGHT);
        Label lblPass=new Label("Password : ",Label.RIGHT);
        name=new TextField("enter your name",20);
        pass=new TextField("enter password ",12);
        pass.setEchoChar('*');
        add(lblName);
        add(name);
        add(lblPass);
        add(pass);
        name.addActionListener(this);
        pass.addActionListener(this);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        setVisible(true);
   }
   public void actionPerformed(ActionEvent ae){
        repaint();
   }
   public void paint(Graphics g){
        g.drawString("Name : " +name.getText(),16,88);
        g.drawString("Selected text in name :"+ name.getSelectedText(),16,108);
        g.drawString("Passwoord : "+ pass.getText(),16,128);
   }
   public static void main(String[] args) {
      new DemoTextField();
   }
}
```

ឧទាហរណ៍ ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoTextField1 extends Frame {
   TextField tf1,tf2;
   String        name="";
   Font font;
   public DemoTextField1() {
        setLayout(new FlowLayout());
        add(new Label("First Name: "));
        tf1=new TextField("surasl",12);
        add(tf1);
        add(new Label("Last Name: "));
        tf2=new TextField("Bun",20);
        font=new Font("Limon s1",Font.PLAIN,12);
        tf2.setFont(font);
        add(tf2);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        tf2.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent ae){
                        name=tf2.getText()+ " " + tf1.getText();
                }
        });
        setBounds(100,100,300,200);
   }
   public void paint(Graphics g){
        g.setFont(font);
        g.drawString(name,268,168);
   }
   public static void main(String[] args) {
     new DemoTextField1().setVisible(true);
   }
}
```

៣.៩ ការប្រើ TextArea

ទម្រង់ Constructor៖

TextArea()

TextArea(int numlines, int numChars)

TextArea(String str)

TextArea(String str, int numlines, int numChars)

TextArea(String str, int numlines, int numChars,int sBars)

- sBars មាន៖

SCROLLBARS_BOTH SCROLLBARS_HORIZONTAL_ONLY

SCROLLBARS_NONE SCROLLBARS_VERTICAL_ONLY

Methods ដែលប្រើជាមួយវា៖

String getText()

void setText(String str)

String getSelectedText()

void select(int startIndex, int endIndex)

boolean isEditable()

void setEditable(boolean canEdit)

void append(String str)

void insert(String str, int index)

void replaceRange(String str, int startIndex, int endIndex)

ឧទាហរណ៍ ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoTextArea extends Frame{
    public DemoTextArea() {
            super("Demo Using Text area.");
        setLayout(new FlowLayout());
        setSize(500,400);
        String val="Exstends :A keyword used in a class declaration \n"+
                        " to specify the superclass of the class being defined.\n"+
                        " The class being defined has access to all the public \n"+
                        "and protected variables and methods of the superclass \n"+
                        "(or, if the class being defined is in the same package,\n"+
                        " it has access to all non-private variables and methods).\n"+
                        " If a class definition omits the extends clause, its \n"+
                        "superclass is taken to be java.lang.Object. ";
        TextArea textarea=new TextArea(val,20,50);
        add(textarea);
```

```
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        setVisible(true);
    }
    public static void main(String[] args) {
       new DemoTextArea();
    }
}
```

ឧទាហរណ៍ ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoTextFieldEvent extends Frame implements ActionListener, TextListener {
        TextArea ta;
        TextField tf;
    public DemoTextFieldEvent() {
        super("Demo Text field Event.");
        setLayout(new FlowLayout());
        setSize(400,300);
        tf=new TextField(20);
        tf.addActionListener(this);
        tf.addTextListener(this);
        add (tf);
        ta=new TextArea(10,20);
        add(ta);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        setVisible(true);
    }
    public void actionPerformed(ActionEvent ae){
        ta.append("Action Event : "+ ae.getActionCommand()+ "\n");
        tf.setText("");
    }
    public void textValueChanged(TextEvent te){
        ta.append("Text Event :" +tf.getText() + "\n");
    }
```

```
    public static void main(String[] args) {
        new DemoTextFieldEvent();
    }
}
```

៤. ការប្រើ Layout Managers

យើងប្រើ Layout Manager ដើម្បីធ្វើការគ្រប់គ្រង និងរៀបចំ components ដោយស្វ័យប្រវត្ត នៅក្នុង window មួយ។ យើងកំណត់ layout manager ដោយប្រើ method setLayout() ដែលមានទម្រង់ទូទៅដូចខាងក្រោម៖

void setLayout(LayoutManger layoutObject)

Layout Managers មាន Layout ដូចជា FlowLayout, BorderLayout, GrifLayout,CardLayout, GridBagLayout, BoxLayout,SpringLayout ជាដើម។

Methods ដែលអាចប្រើជាមួយ Layout មាន៖ setBound(), minimumLayoutSize(), preferredLayoutSize(), getPreferredSize, getMinimumSize() ។

៤.១ FlowLayout

FlowLayout គឺជា layout មួយប្រភេទដែលរៀប components ពីគែមខាងលើពីខាងឆ្វេងទៅស្តាំ និងពីលើចុះក្រោម។

ទម្រង់ Constructor ៖

FlowLayout()

FlowLayout(int how)

FlowLayout(int how,int horz,int vert)

ទម្រង់ទី១ រៀប component នៅចំកណ្តាល ហើយទុកចន្លោះ ៥ pixels ពី component មួយទៅ component មួយទៀត។

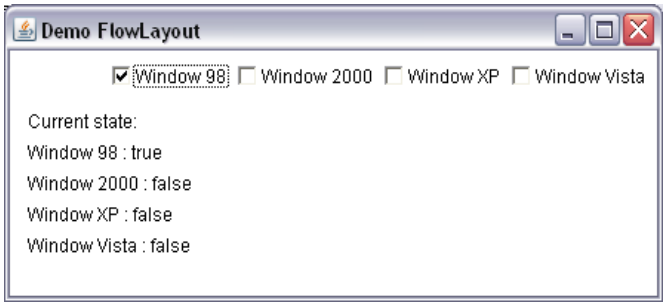
ទម្រង់ទី២ អាចឲ្យយើងកំណត់នូវប្រភេទ alignment ដល់ជួររៀប components។ ដែល how អាចជា៖ FlowLayout.LEFT, FlowLayout.CENTER និង FlowLayout.RIGHT។

ទី៣ អាចឲ្យយើងកំណត់គំលាតរវាង component មួយទៅ component មួយទៀតតាមជួរដេក ឬជួរឈរ (horz,vert)

ឧទាហរណ៍ ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoFlowLayout extends Frame implements ItemListener {
        String sms="";
        Checkbox win98,win2000,winXP,winVista;
    public DemoFlowLayout() {
        super("Demo FlowLayout");
        setSize(450,300);
```

```
        setLayout(new FlowLayout(FlowLayout.RIGHT ));
        win98=new Checkbox("Window 98",null,true);
        win2000=new Checkbox("Window 2000");
        winXP=new Checkbox("Window XP");
        winVista=new Checkbox("Window Vista");
        add(win98);
        add(win2000);
        add(winXP);
        add(winVista);
        win98.addItemListener(this);
        win2000.addItemListener(this);
        winXP.addItemListener(this);
        winVista.addItemListener(this);
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        setVisible(true);
    }
    public void itemStateChanged(ItemEvent ie){
        repaint();
    }
    public  void paint(Graphics g){
        sms="Current state: ";
        g.drawString(sms,16,80);
        sms="Window 98 : " + win98.getState();
        g.drawString(sms,16,100);
        sms="Window 2000 : " + win2000.getState();
        g.drawString(sms,16,120);
        sms="Window XP : " + winXP.getState();
        g.drawString(sms,16,140);
        sms="Window Vista : " + winVista.getState();
        g.drawString(sms,16,160);
    }
    public static void main(String[] args) {
       new DemoFlowLayout();
    }
}
```



ឧទាហរណ៍ ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoFlowLayout1 {
   public static void main(String[] args) {
      Frame fr= new Frame("Demo Flow Layout :");
      fr.setLayout(new FlowLayout());
      fr.add(new Button("Red"));
      fr.add(new Button("Green"));
      fr.add(new Button("Blue"));
      List ls=new List();
      for(int i=0;i<args.length;i++)
        ls.add(args[i]);
      fr.add(ls);
      fr.add(new Checkbox("Pick me",true));
      fr.add(new Label("Enter your name: "));
      fr.add(new TextField(2));
      fr.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
      });
      fr.pack();
      fr.setVisible(true);
   }
}
```

៥.២ BorderLayout

BorderLayout គឺជា Layout មួយដែលចែកផ្ទាំង window ជា ៥ ផ្នែកគឺ north, south, east, west, និង center។

ទម្រង់ Constructor៖

BorderLayout()

BorderLayout(int horz, int vert)

-horz កំណត់គំលាតទិសដៅឈរពី component មួយទៅ component មួយទៀត

-vert កំណត់គំលាតទិសដៅដេកពី component មួយទៅ component មួយទៀត

ចំនួនថេរសម្រាប់កំណត់ទីតាំងរបស់BorderLayout មាន៖

BorerLayout.CENTER

BorerLayout.SOUTH

BorerLayout.NORTH

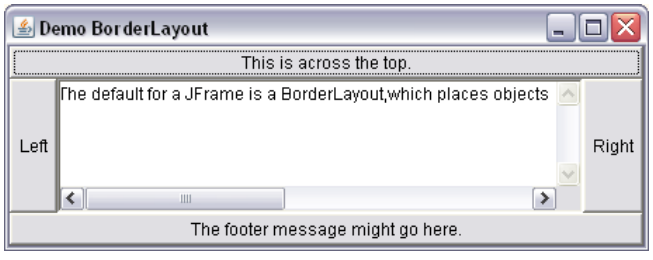
BorerLayout.EAST

BorerLayout.WEST

Methods ដែលប្រើជាមួយវា៖ add(Component compObj, Object region), getLayout(), setHgap(), setVgap(), getHgap(), getVgap()។

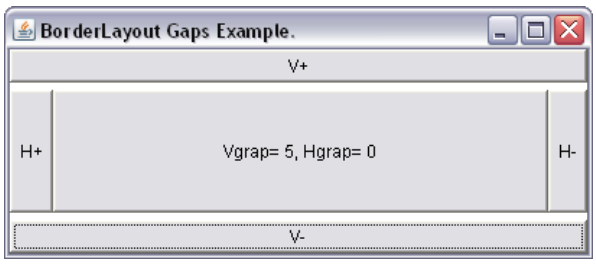
ឧទាហរណ៍ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoBorderLayout extends Frame {
   public DemoBorderLayout() {
        super("Demo BorderLayout");
        setLayout(new BorderLayout());
        setSize(400,200);
        setVisible(true);
        add(new Button("This is across the top."),BorderLayout.NORTH);
        add(new Button("The footer message might go here."),BorderLayout.SOUTH);
        add(new Button("Right"),BorderLayout.EAST);
        add(new Button("Left"),BorderLayout.WEST);
        String sms="The default for a JFrame is a BorderLayout,"+
                        "which places objects at specific locations "+
                        "within the window, such as NORTH, SOUTH, and CENTER.";
        add(new TextArea(sms),BorderLayout.CENTER);
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
      });
        show();
   }
   public static void main(String[] args) {
      new DemoBorderLayout();
   }
}
```



ឧទាហរណ៍ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoBorderLayout1 extends Frame implements ActionListener{
   Button status;
   BorderLayout layout;
   public DemoBorderLayout1() {
        super("BorderLayout Gaps Example.");
        layout=(BorderLayout)getLayout();
        Button b;
        add(b=new Button("V+"),BorderLayout.NORTH);
        b.addActionListener(this);
        add(b=new Button("H+"),BorderLayout.WEST);
        b.addActionListener(this);
        status=new Button("Vgap="+layout.getVgap()+", Hgap= " + layout.getHgap());
        add(status,BorderLayout.CENTER);
        add(b=new Button("H-"),BorderLayout.EAST);
        b.addActionListener(this);
        add(b=new Button("V-"),BorderLayout.SOUTH);
        b.addActionListener(this);
        setBounds(100,200,350,250);
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
      });
      setVisible(true);
   }
   public void actionPerformed(ActionEvent ae){
        String what=ae.getActionCommand();
        if ("H+".equals(what))
                layout.setHgap(layout.getHgap()+5);
        else if ("H-".equals(what))
                layout.setHgap(Math.max(0,layout.getHgap()-5));
        else if ("V+".equals(what))
                layout.setVgap(layout.getVgap()+5);
        else if ("V-".equals(what))
                layout.setVgap(Math.max(0,layout.getVgap()-5));
        status.setLabel("Vgrap= " +layout.getVgap()+", Hgrap= " + layout.getHgap());
        invalidate();
        validate();
   }
   public static void main(String[] args) {
      new DemoBorderLayout1();
   }
}
```



៤.៣ ការប្រើ Insets

ជួនកាលយើងចង់ទុកឲ្យវាមានទំហំតូចមួយនៅចន្លោះរវាង container ដែលផ្ទុក Component ហើយនិង window ដែលផ្ទុក container នោះ។ ដូច្នេះយើងត្រូវ overridden ទៅលើ method Inset()។ ដែល Insets ចែកជា ៤ ផ្នែកគឺ top, bottom, left, និងright។

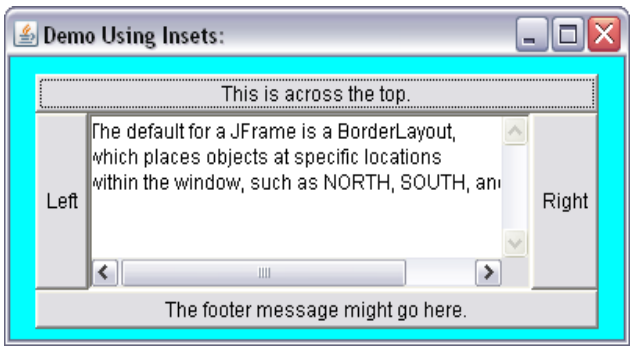
ទម្រង់ Constructor របស់វា៖

nsets (int top, int left, int bottom, int right)

យើងក៏អាចប្រើ getInsets() ដើម្បីទទួលចន្លោះទាំងនោះផងដែរ។

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoInsets extends Frame {
   public DemoInsets() {
        super("Demo Using Insets:");
        setSize(400,200);
        setBackground(Color.cyan);
        setLayout(new BorderLayout());
        setVisible(true);
        add(new Button("This is across the top."),BorderLayout.NORTH);
        add(new Button("The footer message might go here."),BorderLayout.SOUTH);
        add(new Button("Right"),BorderLayout.EAST);
        add(new Button("Left"),BorderLayout.WEST);
        String sms="The default for a JFrame is a BorderLayout,\n"+
                        "which places objects at specific locations \n"+
                        "within the window, such as NORTH, SOUTH, and CENTER.";
        add(new TextArea(sms),BorderLayout.CENTER);
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
      });
   }
   public Insets getInsets(){
        return new Insets(40,20,10,20);
   }
   public static void main(String[] args) {
        new DemoInsets();
   }
}
```



៥.៥ GridLayout

GridLayout គឺជា layout មួយប្រភេទដែលមានលក្ខណៈជា Grid ដែលផ្គុក component តាមជួរឈរ និងជួរដេក។

ទម្រង់ Constructor ៖

GridLayout()

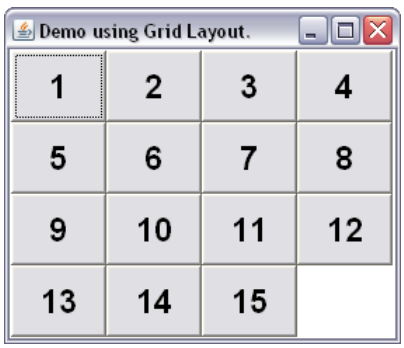
GridLayout(int numRows, int numColumns)

GridLayout(int numRows, int numColumns,int horz, int vert)

-horz និងver សម្រាប់កំណត់គំលាត ពី Component មួយទៅ Component មួយទៀត តាមជួរដេក និងតាមជួរឈរ។ បើដាក់ numRows=0 គឺមានន័យថា យើងអាចរៀប Component តាមជួរ Columns មិនកំណត់ តែបើដាក់ numColumns =0 វិញ យើងអាចរៀប Component តាមជួរដេកមិនកំណត់។

ឧទាហរណ៍ ទី១៖

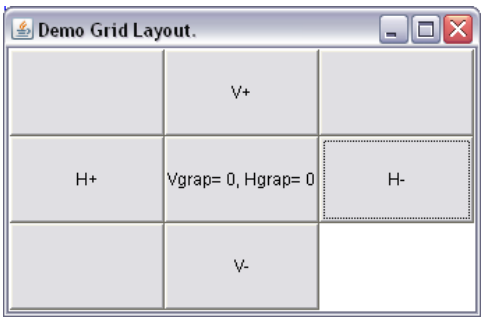
```
import java.awt.*;
import java.awt.event.*;
public class DemoGridLayout extends Frame {
        static final int n=4;
    public DemoGridLayout() {
        super ("Demo using Grid Layout.");
        setSize(300,300);
        setLayout(new GridLayout(n,n));
        setFont(new Font("ScansSerif",Font.BOLD,24));
        for(int i=0;i<n;i++){
                for(int j=0;j<n;j++){
                        int k=i*n+j;
                        if (k>0)
                                add(new Button(""+k));
                }
        }
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
       });
        setVisible(true);
    }
    public static void main(String[] args) {
      new DemoGridLayout();
    }
}
```



ឧទាហរណ៍ ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoGridLayout1 extends Frame implements ActionListener {
   GridLayout glayout=new GridLayout(3,3);
   Button status,b;
   public DemoGridLayout1() {
        super("Demo Grid Layout.");
        setLayout(glayout);
        add(b=new Button());
        add(b=new Button("V+"));
        b.addActionListener(this);
        add(b=new Button());
        add(b=new Button("H+"));
        b.addActionListener(this);
        status=new Button("Vgap= " +glayout.getVgap()+", Hgap= "+ glayout.getHgap());
        add(status);
        add(b=new Button("H-"));
        b.addActionListener(this);
        add(b=new Button());
        add(b=new Button("V-"));
        b.addActionListener(this);
        setBounds(100,200,300,250);
        setVisible(true);
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
      });
   }
   public void actionPerformed(ActionEvent ae){
        String what=ae.getActionCommand();
        if ("H+".equals(what))
                glayout.setHgap(glayout.getHgap()+5);
        else if ("H-".equals(what))
                glayout.setHgap(Math.max(0,glayout.getHgap()-5));
        else if ("V+".equals(what))
                glayout.setVgap(glayout.getVgap()+5);
        else if ("V-".equals(what))
```

```
                glayout.setVgap(Math.max(0,glayout.getVgap()-5));
            status.setLabel("Vgrap= " +glayout.getVgap()+", Hgrap= " + glayout.getHgap());
            invalidate();
            validate();
        }
        public static void main(String[] args) {
            new DemoGridLayout1();
        }
    }
```



៤.៥ CardLayout

CardLayout គឺជា Layout មួយប្រភេទដែលអាចផ្ទុកនូវ layouts ខុសៗគ្នាជាច្រើន តាមលក្ខណៈដូចសន្លឹកបៀរនៅក្នុងហ្វូ។

ទម្រង់ Constructors៖

CardLayout()

CardLayout(int horz, int vert)

-horz និង vert កំណត់គំលាតពី Component មួយទៅ component មួយទៀត

ការប្រើ CardLayout តម្រូវឲ្យធ្វើការងារជាច្រើនជាង Layout ផ្សេងទៀត ព្រោះសន្លឹកបៀរមួយសន្លឹកត្រូវដាក់នៅក្នុង object នៃ Panel មួយ ហើយសន្លឹកបៀរមួយហ្វូត្រូវផ្ទុកនៅក្នុង object នៃ Panel មួយផ្សេងទៀត(កំណត់យកជា CardLayout)។ បន្ទាប់មក បញ្ចូល panel នៃសន្លឹកបៀរនីមួយៗ ចូលទៅក្នុង panel ដែលកំណត់យកជា CardLayout បន្ទាប់មក យើងបញ្ចូល panel នោះទៅក្នុង applet ។ ហើយជាទូទៅយើងប្រើ button ដើម្បីជ្រើសរើស សន្លឹកបៀរណាមួយពីក្នុងហ្វូ។

Methods ដែលប្រើជាមួយវា៖

```
void add(Component panelObj, Object name)
void first(Container deck)//ហៅសន្លឹកបៀរទី១ មកបង្ហាញ
void last(Container deck) //ហៅសន្លឹកបៀរចុងក្រោយមកបង្ហាញ
void next(Container deck) //ហៅសន្លឹកបៀរបន្ទាប់មកបង្ហាញ
void previous(Container deck) //ហៅសន្លឹកបៀរខាងមុខមកបង្ហាញ
void show(Container deck, String cardName)//បង្ហាញសន្លឹកណាមួយតាម cardName
```

-deck សំដៅលើ container ជាធម្មតាគឺជា panel ដែលផ្ទុកសន្លឹកបៀរ

-cardName ឈ្មោះរបស់សន្លឹកបៀរ

ឧទាហរណ៍ ទី១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoCardLayout extends Frame implements ActionListener,MouseListener {
   Checkbox win2000,win2003,winXP,winVista,linux,suse;
```

```
Panel osCard;
CardLayout cardLo;
Button win,other;
public DemoCardLayout() {
      super("Demo Card Layout.");
      setLayout(new FlowLayout());
      setSize(300,200);
      win =new Button("Windows");
      other=new Button("Other");
      add(win);
      add(other);
      cardLo=new CardLayout();
      osCard=new Panel();
      osCard.setLayout(cardLo);
      win2000=new Checkbox("Window 2000",null,true);
      win2003=new Checkbox("Window 2003 Server");
      winXP=new Checkbox("Window XP Gold");
      winVista=new Checkbox("Window Vista Ultimate");
      linux=new Checkbox("Linux Open Source");
      suse=new Checkbox("Suse Open Source");
      Panel winPan=new Panel();
      winPan.add(win2000);
      winPan.add(win2003);
      winPan.add(winXP);
      winPan.add(winVista);
      Panel otherPan=new Panel();
      otherPan.add(linux);
      otherPan.add(suse);
      osCard.add(winPan,"Windows");
      osCard.add(otherPan,"Others");
      add(osCard);
      win.addActionListener(this);
      other.addActionListener(this);
      addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent we){
            System.exit(0);
      }
    });
      setVisible(true);
}
```

```
    public void mousePressed(MouseEvent me){
        cardLo.next(osCard);
    }
    public void mouseClicked(MouseEvent me){}
    public void mouseEntered(MouseEvent me){}
    public void mouseExited(MouseEvent me){}
    public void mouseReleased(MouseEvent me){}
    public void actionPerformed(ActionEvent ae){
        if(ae.getSource()==win)
            cardLo.show(osCard,"Windows");
        else
            cardLo.show(osCard,"Others");
    }
    public static void main(String[] args) {
       new DemoCardLayout();
    }
}
```

ឧទាហរណ៍ ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoCardLayout1  extends Frame implements ActionListener{
    Panel flow=new Panel();
    Panel card=new Panel();
    public DemoCardLayout1() {
        super("Demo set layout");
        flow.setLayout(new FlowLayout());
        card.setLayout(new CardLayout());
        addComp("Button",new Button("Button"));
        addComp("TextArea",new TextArea("TextArea"));
        addComp("List",new List());
        add(flow,BorderLayout.NORTH);
        add(card,BorderLayout.CENTER);
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
            System.exit(0);
        }
       });
       setBounds(100,100,200,200);
        setVisible(true);
    }
```

```
    void addComp(String label,Component cp){
        Button b=new Button(label);
        card.add(cp,label);
        flow.add(b);
        b.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae){
        String str=ae.getActionCommand();
        CardLayout cl=(CardLayout)card.getLayout();
        cl.show(card,str);
    }
    public static void main(String[] args) {
       new DemoCardLayout1();
    }
}
```

៥. ការបង្កើត Menu Bars និង Menus

Menu ត្រូវដាក់នៅលើ MenuBar ដូច្នេះយើងត្រូវបង្កើត MenuBar មុនបង្កើត Menu។ បន្ទាប់ពីបង្កើត Menu ហើយ យើងត្រូវ menuItem ។

ទម្រង់Constructor៖

MenuBar()

Menu()

Menu(String optionName)

MenuItem()

MenuItem(String itemName)

MenuItem(String itemName, MenuShortcut keyAccel)

យើងអាចបង្កើត menu item មានលក្ខណៈជា checkbox បានដែរ ដោយប្រើCheckboxMenuItem ទម្រង់ Constructor៖

CheckboxMenuItem()

CheckboxMenuItem(String itemName)

CheckboxMenuItem(String itemName, boolean on)

Methods ដែលប្រើជាមួយ៖

void setEnabled(boolean enabledFlag)

boolean isEnable(d)

void setLabel(String newName)

String getLabel()

boolean getState()

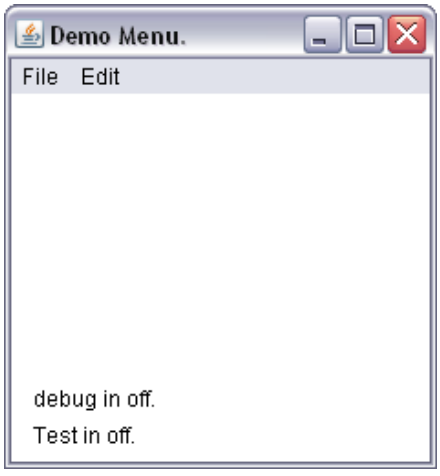
```
        void setState(boolean checked)
        MenuItem add(MenuItem item)
        Menu add(Menu menu) // បញ្ចូល menu ទៅក្នុង Menubar
```

ឧទាហរណ៍៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoMenu extends Frame {
   String sms="";
   CheckboxMenuItem debug,test;
   public DemoMenu(String title) {
        super(title);
        setSize(250,250);
        MenuBar mbar=new MenuBar();
        setMenuBar(mbar);
        Menu file = new Menu("File");
        MenuItem item1,item2,item3,item4,item5;
        file.add(item1=new MenuItem("New.."));
        file.add(item2=new MenuItem("Open."));
        file.add(item3=new MenuItem("Close"));
        file.add(item4=new MenuItem("-"));
        file.add(item5=new MenuItem("Quit."));
        mbar.add(file);
        Menu edit=new Menu("Edit");
        MenuItem item6,item7,item8;
        edit.add(item6=new MenuItem("Cut"));
        edit.add(item7=new MenuItem("Copy"));
        edit.add(item8=new MenuItem("Paste"));
        Menu format =new Menu("Format");
        MenuItem item9,item10,item11;
        format.add(item9=new MenuItem("Font"));
        format.add(item10=new MenuItem("Color"));
        format.add(item11=new MenuItem("Other"));
        edit.add(format);
        debug=new CheckboxMenuItem("Debug");
        edit.add(debug);
        test=new CheckboxMenuItem("Test");
        edit.add(test);
        mbar.add(edit);
        MyMenuHandler handler=new MyMenuHandler(this);
```

```
            item1.addActionListener(handler);
            item2.addActionListener(handler);
            item3.addActionListener(handler);
            item4.addActionListener(handler);
            item5.addActionListener(handler);
            item6.addActionListener(handler);
            item7.addActionListener(handler);
            item8.addActionListener(handler);
            item9.addActionListener(handler);
            item10.addActionListener(handler);
            item11.addActionListener(handler);
            debug.addItemListener(handler);
            test.addItemListener(handler);
            MyWindowAdapter adapter=new MyWindowAdapter(this);
            addWindowListener(adapter);
            setVisible(true);
      }
      public void paint(Graphics g){
            g.drawString(sms,16,200);
            if(debug.getState())
                  g.drawString("debug in on.",16,220);
            else
                  g.drawString("debug in off.",16,220);
            if(test.getState())
                  g.drawString("Test in on.",16,240);
            else
                  g.drawString("Test in off.",16,240);
      }
      public static void main(String[] args) {
         new DemoMenu("Demo Menu.");
      }
}
class MyWindowAdapter extends WindowAdapter{
            DemoMenu menu;
            public MyWindowAdapter(DemoMenu menu){
                  this.menu=menu;
            }
            public void windowClosing(WindowEvent we){
                  menu.setVisible(false);
            }
}
```

```
class MyMenuHandler implements ActionListener,ItemListener{
        DemoMenu menu;
        public MyMenuHandler(DemoMenu menu){
                this.menu=menu;
        }
        public void actionPerformed(ActionEvent ae){
                String sms="You Seleted : ";
                String str=(String) ae.getActionCommand();
                if(str.equals("New.."))
                        sms+="New.. ";
                else if (str.equals("Open."))
                        sms+="Open. ";
                else if (str.equals("Close"))
                        sms+="Close ";
                else if (str.equals("Quit."))
                        sms+="Quit.";
                else if (str.equals("Edit"))
                        sms+="Edit ";
                else if (str.equals("Cut"))
                        sms+="Cut.";
                else if (str.equals("Copy"))
                        sms+="Copy.";
                else if (str.equals("Paste"))
                        sms+="Paste.";
                else if (str.equals("Font"))
                        sms+="Font.";
                else if (str.equals("Color"))
                        sms+="Color.";
                else if (str.equals("Other"))
                        sms+="Other.";
                menu.sms=sms;
                menu.repaint();
        }
        public void itemStateChanged(ItemEvent ie){
                menu.repaint();
        }
}
```



៦. ការប្រើ Popup Menu និង Shortcut Menu

PopupMenu គឺជាផ្ទាំង Menu មួយដែលលេចឡើងនៅពេលដែលយើង ចុចលើ mouse ផ្នែកខាងស្តាំ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើ PopupMenu និង Shortcut Menu៖

ឧទាហរណ៍ ទី ១៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoPopUpMenu extends Frame implements ActionListener,MouseListener {
   PopupMenu pmenu=new PopupMenu();
   Label lblsms= new Label("Right click here");
   public DemoPopUpMenu() {
        super("Demp Pop-Up Menu");
        setLayout( new FlowLayout());
        setSize(300,150);
        DesignMenu();
        lblsms.setFont(new Font("TimesRoman",Font.BOLD,24));
        add(lblsms);
        lblsms.add(pmenu);
        lblsms.addMouseListener(this);
        pmenu.addActionListener(this);
        setVisible(true);
        addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
                System.exit(0);
        }
      });
   }
   public static void main(String[] args) {
      new DemoPopUpMenu();
   }
   public void DesignMenu(){
        String[] strMenuLabel={"Red","Green","Blue"};
        MenuItem item1=new MenuItem(strMenuLabel[0],
                                                newMenuShortcut(KeyEvent.VK_R));
        MenuItem item2=new MenuItem(strMenuLabel[1],
                                                new MenuShortcut(KeyEvent.VK_G));
        MenuItem item3=new MenuItem(strMenuLabel[2],
                                                new MenuShortcut(KeyEvent.VK_B));
        pmenu.add(item1);
```

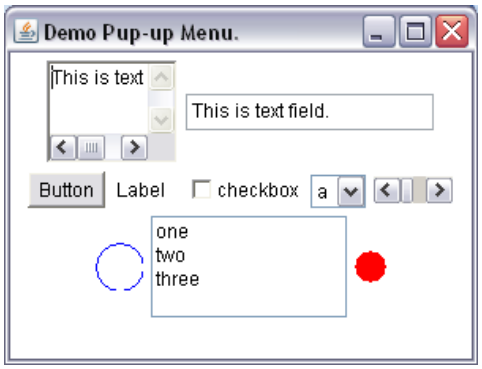
```
        pmenu.add(item2);
        pmenu.add(item3);
    }
    public void actionPerformed(ActionEvent ae){
        String strMenuItem=ae.getActionCommand();
        if (strMenuItem.equals("Red"))
            lblsms.setForeground(Color.red);
        else if (strMenuItem.equals("Green"))
            lblsms.setForeground(Color.green);
        else if (strMenuItem.equals("Blue"))
            lblsms.setForeground(Color.blue);
    }
    public void mousePressed(MouseEvent me){}
    public void mouseClicked(MouseEvent me){}
    public void mouseEntered(MouseEvent me){}
    public void mouseExited(MouseEvent me){}
    public void mouseReleased(MouseEvent me){
        if (me.isPopupTrigger())
            pmenu.show(lblsms,me.getX(),me.getY());
    }
}
```

ឧទាហរណ៍ ទី ២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoPopupMenu1 extends Frame {
    Component[] comp=new Component[11];
    PopupMenu[] pmenu=new PopupMenu[11];

    public DemoPopupMenu1() {
        super("Demo Pup-up Menu.");
        setLayout(new FlowLayout());
        comp[0]= new TextArea("This is text area.",3,9);
        comp[1]=new TextField("This is text field.");
        comp[2]=new Button("Button");
        comp[3]=new Label("Label");
        comp[4]=new Checkbox("checkbox");
        Choice cho=new Choice();
        cho.add("a");
        cho.add("b");
```

```
        cho.add("c");
        comp[5]=cho;
        comp[6]=new Scrollbar(Scrollbar.HORIZONTAL);
        Canvas ca=new MyCanvas();
        ca.setSize(30,30);
        comp[7]=ca;
        List ls=new List();
        comp[8]=ls;
        ls.add("one");
        ls.add("two");
        ls.add("three");
        Component lw=new MyComponent();
        comp[9]=lw;
        comp[10]=this;
        MouseListener ml=new MouseEventHandler();
        //add listener and popup menu to component
        for(int i=0;i<11;i++){
                comp[i].addMouseListener(ml);
                pmenu[i]=new PopupMenu("Popup"+i);
                pmenu[i].add("Popup item "+i);
                comp[i].add(pmenu[i]);
                if(i!=10) add(comp[i]);
        }
        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent we){
                        System.exit(0);
                }
        });
        setBounds(100,100,300,300);
        setVisible(true);
}
class MyCanvas extends Canvas{
        public void paint(Graphics g){
                g.setColor(Color.blue);
                g.drawOval(0,0,getSize().width,getSize().height);
        }
}
class MyComponent extends Component{
        public Dimension getPreferredSize(){
                return new Dimension(20,20);
        }
```

```
        public void paint(Graphics g){
            g.setColor(Color.red);
            g.fillOval(0,0,getSize().width,getSize().height);
        }
    }
    class MouseEventHandler extends MouseAdapter{
        int findComponent(Component cp){
            for(int i=0; i<11;i++){
                if (cp.equals(comp[i]))
                    return i;
            }
            return -1;
        }
        public void mouseReleased(MouseEvent me){
            if(me.isPopupTrigger()){
                int i=findComponent((Component)me.getSource());
                if (i>0)
                    pmenu[i].show(comp[i],me.getX(),me.getY());
                else
                    System.out.println("No Popup");
            }
        }
        public void mousePressed(MouseEvent me){
            if(me.isPopupTrigger()){
                int i=findComponent((Component)me.getSource());
                if (i>0)
                    pmenu[i].show(comp[i],me.getX(),me.getY());
                else
                    System.out.println("No Popup");
            }
        }
    }

    public static void main(String[] args) {
            new DemoPopupMenu1();
    }
}
```



៧. ការប្រើ Dialog Box

ទម្រង់ Constructor ៖

Dialog (Frame parentWindow, boolean mode)

Dialog (Frame parentWindow, String title, boolean mode)

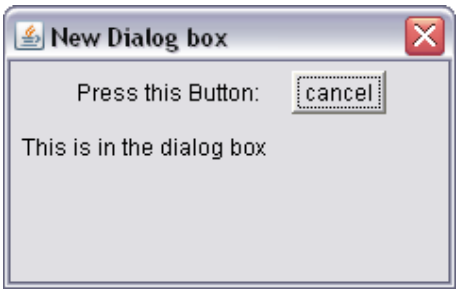
ឧទាហរណ៍ ទី១៖

```
import java.awt.*;
import java.awt.event.*;
class SampleDialog extends Dialog implements ActionListener {

   public SampleDialog(Frame parent,String title) {
            super(parent,title,false);
            setSize(250,250);
            setLayout(new FlowLayout());
            add(new Label("Press this Button:" ));
            Button b;
            add(b=new Button("cancel"));
            b.addActionListener(this);
   }
   public void actionPerformed(ActionEvent ae){
            dispose();
   }
   public void paint(Graphics g){
            g.drawString("This is in the dialog box",10,80 );
   }
}
public class DemoMenu2 extends Frame {
   String sms="";
   CheckboxMenuItem debug,test;
   public DemoMenu2(String title) {
            super(title);
            setSize(250,250);
            MenuBar mbar=new MenuBar();
            setMenuBar(mbar);

            Menu file = new Menu("File");
            MenuItem item1,item2,item3,item4,item5;
            file.add(item1=new MenuItem("New.."));
            file.add(item2=new MenuItem("Open."));
            file.add(item3=new MenuItem("Close"));
            file.add(item4=new MenuItem("-"));
```

```
        file.add(item5=new MenuItem("Quit."));
        mbar.add(file);

        Menu edit=new Menu("Edit");
        MenuItem item6,item7,item8;
        edit.add(item6=new MenuItem("Cut"));
        edit.add(item7=new MenuItem("Copy"));
        edit.add(item8=new MenuItem("Paste"));

        Menu format =new Menu("Format");
        MenuItem item9,item10,item11;
        format.add(item9=new MenuItem("Font"));
        format.add(item10=new MenuItem("Color"));
        format.add(item11=new MenuItem("Other"));

        edit.add(format);
        debug=new CheckboxMenuItem("Debug");
        edit.add(debug);
        test=new CheckboxMenuItem("Test");
        edit.add(test);
        mbar.add(edit);

        MyMenuHandler handler=new MyMenuHandler(this);
        item1.addActionListener(handler);
        item2.addActionListener(handler);
        item3.addActionListener(handler);
        item4.addActionListener(handler);
        item5.addActionListener(handler);
        item6.addActionListener(handler);
        item7.addActionListener(handler);
        item8.addActionListener(handler);
        item9.addActionListener(handler);
        item10.addActionListener(handler);
        item11.addActionListener(handler);

        debug.addItemListener(handler);
        test.addItemListener(handler);
        MyWindowAdapter adapter=new MyWindowAdapter(this);
        addWindowListener(adapter);
        setVisible(true);
    }
```

```
    public void paint(Graphics g){
            g.drawString(sms,16,200);
            if(debug.getState())
                g.drawString("debug in on.",16,220);
            else
                g.drawString("debug in off.",16,220);
            if(test.getState())
                g.drawString("Test in on.",16,240);
            else
                g.drawString("Test in off.",16,240);
    }
    public static void main(String[] args) {
        new DemoMenu2("Demo Menu.");
    }
}
class MyWindowAdapter extends WindowAdapter{
            DemoMenu2 menu;
            public MyWindowAdapter(DemoMenu2 menu){
                this.menu=menu;
            }
            public void windowClosing(WindowEvent we){
                menu.setVisible(false);
            }
}
class MyMenuHandler implements ActionListener,ItemListener{
            DemoMenu2 menu;
            public MyMenuHandler(DemoMenu2 menu){
                this.menu=menu;
            }
            public void actionPerformed(ActionEvent ae){
                String sms="You Seleted : ";
                String str=(String) ae.getActionCommand();
                if(str.equals("New..")){
                        sms+="New.. ";
                        SampleDialog d =new SampleDialog(menu,"New Dialog box");
                        d.setVisible(true);
                }
                else if (str.equals("Open."))
                        sms+="Open. ";
                else if (str.equals("Close"))
                        sms+="Close ";
```

```
                else if (str.equals("Quit."))
                        sms+="Quit.";
                else if (str.equals("Edit"))
                        sms+="Edit ";
                else if (str.equals("Cut"))
                        sms+="Cut.";
                else if (str.equals("Copy"))
                        sms+="Copy.";
                else if (str.equals("Paste"))
                        sms+="Paste.";
                else if (str.equals("Font"))
                        sms+="Font.";
                else if (str.equals("Color"))
                        sms+="Color.";
                else if (str.equals("Other"))
                        sms+="Other.";
                menu.sms=sms;
                menu.repaint();
        }
        public void itemStateChanged(ItemEvent ie){
                menu.repaint();
        }
}
```



ឧទាហរណ៍ ទី២៖

```
import java.awt.*;
import java.awt.event.*;
public class DemoColorDialog extends Frame implements ActionListener {
   Canvas cv;
   public DemoColorDialog() {
        super("Demo Color Dialog");
        setLayout(new FlowLayout());
        Button bt=new Button("Color Dialog" );
        bt.addActionListener(this);
        add(bt);
        cv=new Canvas();
        cv.setSize(50,50);
        cv.setBackground(Color.red);
        add(cv);
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
   }
```

```
    public void actionPerformed(ActionEvent ae){
            ColorDialog cd=new ColorDialog(this,"Color Dialog",true );
            cd.setBounds(300,200,180,200);
            cd.setVisible(true);
    }
    public static void main(String[] args) {
       DemoColorDialog dcd=new DemoColorDialog();
       dcd.setVisible(true);
       dcd.setBounds(100,200,150,120);
    }
}
class ColorDialog extends Dialog implements ActionListener, AdjustmentListener{
            DemoColorDialog parent;
            Scrollbar scR,scG,scB;
            ColorDialog(DemoColorDialog parent,String title, boolean mode){
                super(parent,title,mode);
                this.parent=parent;
                Color color=parent.cv.getBackground();
                int r=color.getRed();
                int g=color.getGreen();
                int b=color.getBlue();
                Panel p=new Panel();
                p.setLayout(new GridLayout(3,2,5,5));
                Label lblR=new Label("Red");
                Label lblG=new Label("Green");
                Label lblB=new Label("Blue");
                lblR.setAlignment(Label.RIGHT);
                lblG.setAlignment(Label.RIGHT);
                lblB.setAlignment(Label.RIGHT);
                p.add(lblR);
                p.add(lblG);
                p.add(lblB);
                scR=new Scrollbar(Scrollbar.HORIZONTAL,r,10,0,255);
                scG=new Scrollbar(Scrollbar.HORIZONTAL,g,10,0,255);
                scB=new Scrollbar(Scrollbar.HORIZONTAL,b,10,0,255);
                p.add(scR);
                p.add(scG);
                p.add(scB);
                scR.addAdjustmentListener(this);
                scG.addAdjustmentListener(this);
                scB.addAdjustmentListener(this);
```

```
                add(p,"Center");
                Panel ps=new Panel();
                Button ok=new Button("OK");
                ok.addActionListener(this);
                ps.add(ok);
                add(ps,"South");
                addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent we){
                                setVisible(false);
                        }
                });
                pack();
        }
        public      Insets getInsets(){
                return new Insets(40,20,20,20);
        }
        public void actionPerformed(ActionEvent ae){
                dispose();
        }
        public void adjustmentValueChanged(AdjustmentEvent ae){
                int r=scR.getValue();
                int g=scG.getValue();
                int b=scB.getValue();
                parent.cv.setBackground(new Color(r,g,b));
                parent.cv.repaint();
        }
}
```

៨. ការប្រើ FileDialog Box

ទម្រង់ Constructor ៖

FileDialog(Frame parent, String boxName)

FileDialog(Frame parent, String boxName, int how)

FileDialog(Frame parent)

```
import java.awt.*;
import java.awt.event.*;
public class DemoFileDialog {
   public static void main(String[] args) {
```

```
        Frame fr=new SampleFrame("Demo File Dialog.");
        fr.setVisible(true);
        fr.setSize(100,100);
        FileDialog fd=new FileDialog(fr,"File Dialog.");
        fd.setVisible(true);
    }
}
class SampleFrame extends Frame{
        SampleFrame(String title){
                super(title);
                MyWindowAdapter adapter=new MyWindowAdapter(this);
                addWindowListener(adapter);
        }
}
class MyWindowAdapter extends WindowAdapter{
        SampleFrame sampleframe;
        public MyWindowAdapter(SampleFrame sampleframe){
                this.sampleframe=sampleframe;
        }
        public void windowClosing(WindowEvent we){
                sampleframe.setVisible(false);
        }
}
```

End

មេរៀនទី ១២

អំពី *Applet*

១. <u>សេចក្តីផ្តើម</u>

Applet គឺជាកម្មវិធីតូចមួយ ដែលមានប្រសិទ្ធភាពក្នុងការប្រើប្រាស់នៅលើ Internet វាអាចបញ្ជូនទិន្នន័យលើ Internet ពី Computer មួយទៅ Computer ដទៃទៀតដោយប្រើ Applet Viewer ឬក៏ Web browser ដែលវា Supports ជាមួយ Java។ Applet វាអាចធ្វើការងារជាច្រើន ដូចជាធ្វើប្រមាណវិធីគណិត បង្ហាញរូបភាព សំលេង វីដេអូ ធ្វើជា Games បង្កើតរូបភាពមានចលនា ទទួលយកការបញ្ជូលរបស់ User និងឆ្លើយតបទៅ User វិញ។

២. <u>ភាពខុសគ្នារវាង Applet និង Applications</u>

ទោះបីជា Applet និង Stand-alone Applications ជាកម្មវិធីរបស់ Java ដូចគ្នាក៏ដោយ តែវានៅតែមានភាពខុសគ្នា។ Applet មិនមែនជាកម្មវិធីមួយពេញលេញទេ វាត្រូវបានគេសរសេរដើម្បីធ្វើការងារមួយចំនួនតូច ឬផ្នែកណាមួយប៉ុណ្ណោះ ព្រោះពេលគេសរសេរវានៅលើInternet វាមានលក្ខណៈ limitations និង restrictions ជាច្រើន។

-Applets វាមិនប្រើ main() method ដើម្បីដំណើរការ code របស់វាទេ។ នៅពេលដែល applet loaded វាហៅ methods របស់ applet class ដើម្បីដំណើរការ code របស់វាដោយស្វ័យប្រវត្តិ។

-Applets មិនអាច run ដោយខ្លួនឯងបានទេវាត្រូវការ run នៅក្នុង webpage ដោយប្រើ HTML tag

-Applets មិនអាច read ឬ write ចូលទៅក្នុង file នៅក្នុង local computer បានទេ

-Applets មិនអាចធ្វើការទំនាក់ទំនងជាមួយនឹងserver ដទៃនៅលើ network បានទេ

-Applets មិនអាចហៅកម្មវិធីផ្សេងៗឲ្យដំណើរការនៅក្នុងlocal computer បានទេ

-Applets ត្រូវបានហាមឃាត់មិនឲ្យប្រើlibraries ចេញពីភាសាផ្សេងៗដូចជាC ឬ C++ បានទេ

៣. <u>Methods ដែលប្រើប្រាស់</u>

វាមាន methods ចំនួន ៥ ដែលប្រើប្រាស់ ជាមួយ Applet គឺ init(), start(), stop(), destroy() និង paint() ។

៣.១ <u>init() method</u>

Method នេះត្រូវបានប្រើសម្រាប់ បង្កើត object ដែល applet ត្រូវការ, ផ្តល់តម្លៃ initial , Load រូបភាព ឬ ក៏ Font និងការផ្តល់ Color ជាដើម។ Method នេះវាកើតឡើងតែម្តងគត់នៅក្នុងlife cycle របស់ Applet ។

ទម្រង់ទូទៅរបស់វា៖

```
public void init (){
……..…..//Action
}
```

៣.២ start() method

Method នេះប្រើសម្រាប់ running applet ។ បន្ទាប់ពី init() បានដំណើរការរួច method start() នឹងដំណើរការដោយ automatic ។ method start() ក៏អាចកើតឡើងដងដែរ កាលណាស្ថានភាពរបស់ applet stopped ហើយត្រឡប់មកវិញ ឧទាហរណ៍នៅពេលដែលយើងបានចាកចេញពីទំព័រដែលផ្ទុក applet បណ្តោះអាសន្នទៅទំព័រផ្សេង ហើយត្រឡប់មកវិញ។ Method start() ខុសពី init() វាអាចត្រូវបានគេហៅមកប្រើច្រើនជាមួយដងនៅក្នុងlife cycle របស់ applet ។

ទម្រង់ទូទៅរបស់វា៖

```
public void start (){
……..//Action
}
```

៣.៣ stop() method

Method stop() កើតឡើងដោយស្វ័យប្រវត្តនៅពេលដែលយើងចាកចេញពីទំព័រដែលផ្ទុកapplet។ បើនៅក្នុង applet យើងមានប្រើ thread ពេលនោះវានឹងបញ្ឈប់thread នោះ។

ទម្រង់ទូទៅរបស់វា៖

```
public void stop (){
……..//Action
}
```

៣.៤ destroy() method

Method destroy() វាកើតឡើងនៅពេលដែលយើងបិទbrowser មានន័យថា applet ត្រូវបាន លុបចេញពី memory។ destroy() ក៏ដូចនឹង init() ដែរ គឺវាកើតឡើតតែម្តងគត់នៅក្នុងlife cycle របស់ applet។

ទម្រង់ទូទៅរបស់វា៖

```
public void destroy (){
……..//Action
}
```

៣.៥ paint() method

Method paint() ត្រូវបានគេប្រើសម្រាប់បង្ហាញនូវរាល់អ្វីៗទាំងអស់ដែលយើងមើលឃើញនៅលើ applet វានឹងកើតឡើងដោយស្វ័យប្រវត្តនៅពេលដែលapplet ស្ថិតនៅក្នុងស្ថានភាពrun (start) ។

ទម្រង់ទូទៅរបស់វា៖

```
public void paint (Graphics g){
……..//Display statements
}
```

៤. ជំហាននៃការសរសេរ

ដំបូងយើងត្រូវបង្កើត Java file មួយដែលត្រូវធ្វើការអ៊ីខ្លះជាមួយ applet សិន រួចហើយយើងត្រូវ compile java file នោះឲ្យទៅជា class file ជាចុងក្រោយត្រូវបង្កើត HTML file រួចហើយ add វាចូលទៅក្នុង applet tag។

៤.១ ទម្រង់ទូទៅរបស់ HTML

ចែកចេញជាបីផ្នែកធំៗគឺ៖

-Comment Section

-Head Section

-Body Section

```
<HTML>
        <!……..……
          ……..……..…..        } Comment Section
        >
        <HEAD>                  }
              //title tag        } Head Section
        </HEAD>                 }
        <BODY>                  }
        ……..…..                 } Body Section
        //applet tag            }
        </BODY>                 }
</HTML>
```

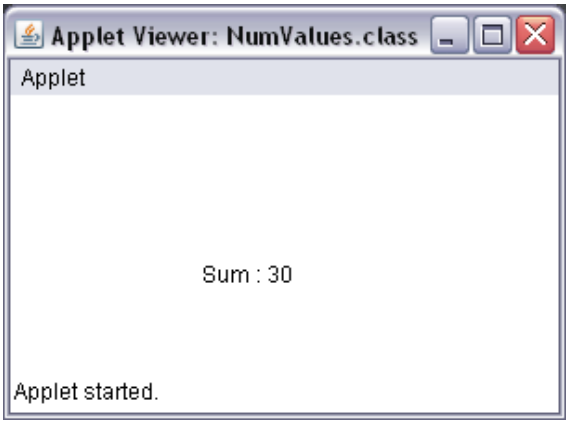
៤.២ ទម្រង់ទូទៅរបស់ Applet tag

```
<APPLET
        [ CODEBASE = codebase_URL ]
        [ ALT = alternate_text ]
        [ NAME = applet_instance_name ]
        WIDTH = pixels
        HEIGHT = pixels
        [ ALING = alignment ]
        [ VSPACE = pixels ]
        [ HSPACE = pixels ]
>
        [ < PARAM NAME = name1  VALUE = value1> ]
        [ < PARAM NAME = name2  VALUE = value2> ]
        ……..……..……..……..……..……..……
        [ text to be displayed in the absent of java ]
</APPLET>
```

៥. ឧទាហរណ៍បង្ហាញពីការប្រើប្រាស់

ឧទាហរណ៍ទី១៖

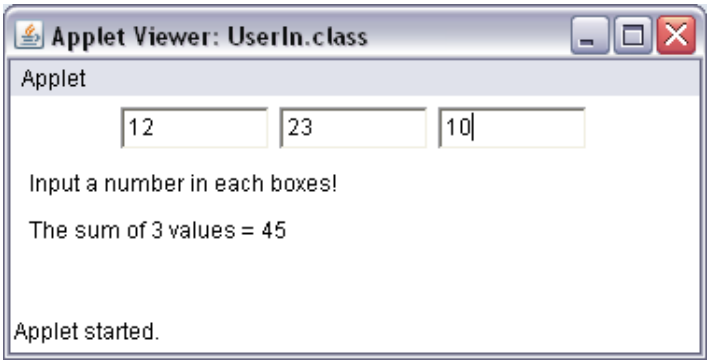
```
//  Save as NumValues.java
import java.awt.*;
import java.applet.*;
public class NumValues extends Applet{
        public void paint(Graphics g){
                int value1 = 10;
                int value2 = 20;
                int sum=value1 + value2;
                String str= " Sum : " + String.valueOf(sum);
                g.drawString(str,100,100);
        }
}
//  Save as  NumValues.HTML
<HTML>
        <APPLET
                CODE = NumValues.class
                WIDTH = 400
                HEIGHT= 250>
        </APPLET>
</HTML>
```



ឧទាហរណ៍ទី២៖

```
// Save as UserIn.java
import java.awt.*;
import java.applet.*;
public class UserIn extends Applet{
        TextField txt1,txt2,txt3;
        public void init(){
                txt1=new TextField(8);
                txt2=new TextField(8);
                txt3=new TextField(8);
                add(txt1);
                add(txt2);
                add(txt3);
                txt1.setText("0");
```

```
                txt2.setText("0");
                txt3.setText("0");
        }
        public void paint(Graphics g){
                int x=0,y=0,z=0,result=0;
                g.drawString("Input a number in each boxes!", 10,50);
                try{
                        x=Integer.parseInt(txt1.getText());
                        y=Integer.parseInt(txt2.getText());
                        z=Integer.parseInt(txt3.getText());
                }catch(Exception ex){}
                result= x+y+z;
                g.drawString("The sum of 3 values = " + result ,10,75);
        }
        public boolean action(Event event,Object obj){
                repaint();
                return true;
        }

}
// HTML file UserIn.HTML
<HTML>
   <APPLET
        CODE = UserIn.class
        WIDTH = 400
        HEIGHT= 250>
   </APPLET>
</HTML>
```



End

មេរៀនទី ១៣

អំពីការប្រើប្រាស់ *Graphics*

១. សេចក្តីផ្តើម

លក្ខណៈចំបងមួយរបស់ java គឺវាអាចឲ្យយើងធ្វើការគូសជា Graphics ផ្សេងៗ ។ យើងអាច គូសជាបន្ទាត់ ជាទម្រង់រាងផ្សេងៗ ជារូបភាព ជាអក្សរក្នុងទម្រង់ font និង style ផ្សេងៗ ពី Java applet និងអាចលាយពណ៌ផ្សេងៗចូលគ្នាដើម្បីបង្ហាញទៀតផង។

គ្រប់ Applet វាមានតំបន់អេក្រង់របស់វា (canvas) ហើយទំហំរបស់វាត្រូវបានកំណត់ដោយ attribute របស់ Applet tag ។ រាល់ការគូសត្រូវបានកំណត់ដោយកូអរដោនេ(x,y) ។

២. អំពី Graphics Class

នៅក្នុង Graphics class មានផ្នុកនូវ methods ជាច្រើនដែលអាចឲ្យយើងគូសរូបផ្សេងៗគ្នា។ ខាងក្រោមនេះជា methods ដែលមាននៅក្នុង Graphics class សម្រាប់គូសរូប និងអក្សរផ្សេងៗ៖

clearRect(int, int, Int, int) សម្រាប់លុបផ្ទៃរបស់ Rectangle

copyArea(int, int, int, int, int,int) សម្រាប់ copy ផ្ទៃមួយរបស់ canvas ទៅតំបន់មួយទៀត

drawArc(int, int, int, int, int, int) សម្រាប់គូស ខ្សែកោង ឬធ្នូ

drawLine(int, int, int, int) សម្រាប់គូស បន្ទាត់

drawOval(int, int, int, int) សម្រាប់គូសរង្វង់រាងពងក្រពើរង្វង់មូល

drawPolygon(int[],int[], int) សម្រាប់គូស polygon

drawRect(int, int, int, int) សម្រាប់គូស Rectangle

drawRoundRect(int, int, int, int, int, int)សម្រាប់គូស Rectangle ដែលបិតជ្រុងរាងមូល

drawstring(String, int, int) សម្រាប់គូស String

fillArc(int, int, int, int, int, int) ចាក់ពណ៌ ខ្សែកោង

fillOval(int, int, int, int)ចាក់ពណ៌រង្វង់

fillPolygon(int[],int[], int) ចាក់ពណ៌ polygon

fillRect(int, int, int, int) ចាក់ពណ៌ rectangle

fillRoundRect(int, int, int, int, int, int)ចាក់ពណ៌ Rectangle ដែលបិតជ្រុងរាងមូល

getColor() ទាញយកពណ៌

getFontMetrics() ទាញយក Font ដែលប្រើបាន

setColor(Color c) ផ្តល់ពណ៌

setFont(Font f) ផ្តល់ font

៣. ឧទាហរណ៍នៃការប្រើប្រាស់

```
// Save file នេះដាក់ឈ្មោះ LineRect.java
import java.awt.*;
import java.applet.*;
public class LineRect extends Applet {
```

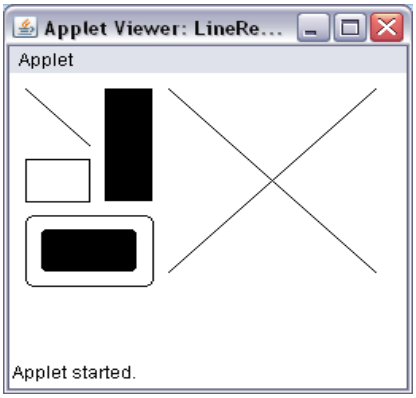
```
    public void paint(Graphics g) {
      g.drawLine(10,10,50,50);
      g.drawRect(10,60,40,30);
      g.fillRect(60,10,30,80);
      g.drawRoundRect(10,100,80,50,10,10);
      g.fillRoundRect(20,110,60,30,5,5);
      g.drawLine(100,10,230,140);
      g.drawLine(100,140,230,10);
    }
}
```

បន្ទាប់ត្រូវ compile file LineRect.java **ដើម្បីបង្កើត** LineRect.class **ហើយត្រូវបង្កើត** File HTML **មួយទៀតឈ្មោះ:** LineRect.html **ដូចខាងក្រោម៖**

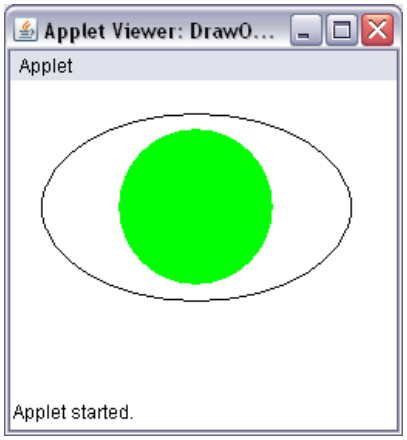
```
<APPLET
        CODE = LineRect.class
        WIDTH = 250
        HEIGHT= 200 >
</APPLET>
```

ដើម្បី Run File **នេះ តាម** Applet viewer **យើងត្រូវប្រើ** command **ដូចខាងក្រោម៖**

C:\jdk1.7\bin\appletviewer LineRect.html

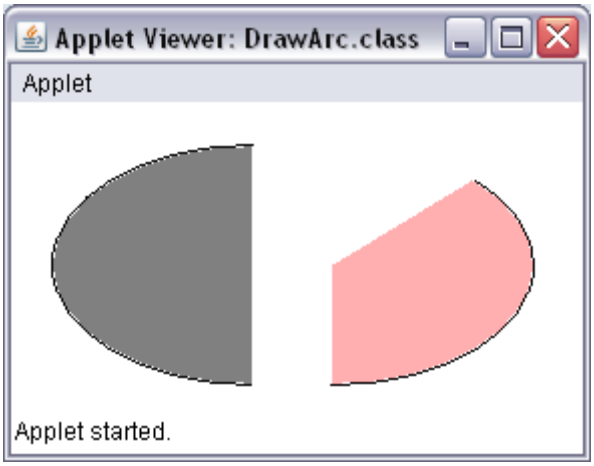



```
//DrawOval()
import java.awt.*;
import java.applet.*;
public class DrawOval extends Applet {

    public void paint(Graphics g) {
      g.drawOval(20,20,200,120);
      g.setColor(Color.green);
      g.fillOval(70,30,100,100);
    }
}
```

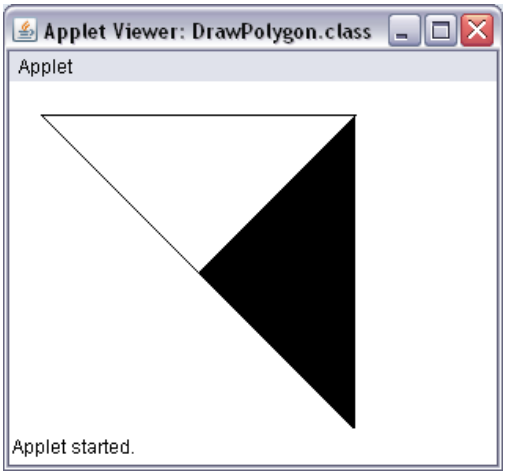



```
//DrawArc()
```

```
import java.awt.*;
import java.applet.*;
public class DrawArc extends Applet {
    public void paint(Graphics g) {
        g.drawArc(20,20,200,120,90,180);
        g.setColor(Color.gray);
        g.fillArc(21,21,199,119,90,180);
        g.setColor(Color.black);
        g.drawArc(60,20,200,120,45,-135);
        g.setColor(Color.pink);
        g.fillArc(61,21,199,119,45,-135);
    }
}
```




```
//DrawPolygon()
import java.awt.*;
import java.applet.*;
public class DrawPolygon extends Applet {
    public void paint(Graphics g) {
        int[] x ={20,120,220,20};
        int[] y ={20,120,20,20};
        int[] x1={120,220,220,120};
        int[] y1={120,20,220,120};
        g.drawPolygon(x,y,4);
        g.fillPolygon(x1,y1,4);
    }
}
```

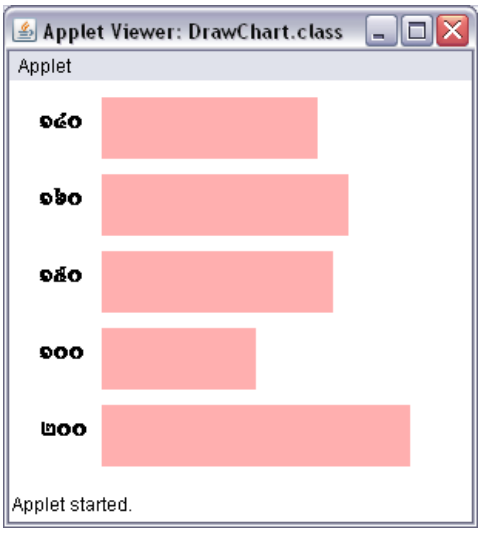


យើងក៏អាចប្រើ method addPoint(x,y) **ដូចខាងក្រោម៖**

```
Polygon polygon=new Polygon();
Polygon.addPoint(20,20);
Polygon.addPoint(120,120);
Polygon.addPoint(220,20);
Polygon.addPoint(20,20);
g.drawPolygon(polygon);
```

ខាងក្រោមនេះការបង្ហាញពីការ Chart **តាមទម្រង់ជួរដេក៖**

```
import java.awt.*;
import java.applet.*;
public class DrawChart extends Applet {
        int n=0;
        String[] label;
        int[] value;
        public void init(){
                try{
                        n=Integer.parseInt(getParameter("columns"));
                        label=new String[n];
                        value=new int[n];
                        for(int i=0;i<n;i++){
                                label[i]=getParameter("label"+(i+1));
                                value[i]=Integer.parseInt(getParameter("c"+(i+1)));
                        }
                }catch(NumberFormatException e){}
        }
    public void paint(Graphics g) {
         Font font=new Font("Limon R1",Font.BOLD,20);
      for(int i=0;i<n;i++){
                        g.setColor(Color.black);
                        g.setFont(font);
                        g.drawString(label[i],20,i*50+30);
                        g.setColor(Color.pink);
                        g.fillRect(60,i*50+10,value[i],40);
         }
    }
}
```



End