ជំពូកទី១

ទិដ្ឋភាពទូទៅនៃ JAVA

១. ប្រវត្តិនៃ JAVA:

ដោយសារតែមានការចូលរួមចំនែកនៃការវិវត្ត microprocessor នាពេលបច្ចុប្បន្នកាល បានធ្វើឱ្យមានការអភិវឌ្ឍន៍ខាង Personal Computer ដែលមានចំនួនកើនឡើងរហូតដល់រាប់លានគ្រឿងទូទាំងពិភពលោក។ Personal Computers មានឥទ្ធិពលលើមនុស្សយ៉ាងខ្លាំង ទាំងការគ្រប់គ្រង និងការធ្វើអាជីវកម្ម។

មនុស្សជាច្រើនមានជំនឿថា នៅពេលខាងមុខផ្នែកដែលសំខាន់របស់ microprocessor ដែលមានឥទ្ធិពលខ្លាំងគឺ បរិក្ខាអេឡិចត្រូនិកដែលមានលក្ខណៈឆ្លាត។ ដោយយល់ឃើញបែបនេះទើបក្រុមហ៊ុន Sun Microsystems បានផ្តល់ថវិការសំរាប់គំរោងស្រាវជ្រាវក្នុងក្រុមហ៊ុនដែលមានឈ្មោះថា Green នាឆ្នាំ 1991។ គំរោងនេះបានប្រើភាសាមួយដែលមានមូលដ្ឋានលើការអភិវឌ្ឍន៍ភាសា C និង C^{++} ហើយបង្កើតឡើងដោយលោក James Gosling។ ដំបូងភាសានេះមានឈ្មោះថា Oak។ តែឈ្មោះនេះមានគេប្រើរួចហើយនៅក្នុង Programming Language នោះ វាក៏ត្រូវបានប្តូរមកជា **Java** វិញតាមសំណើរបស់អ្នកធ្វើការនៅក្នុងក្រុមហ៊ុន។ បន្ទាប់មកវាក៏មានឈ្មោះនេះជាប់រហូតមក។

២. Version របស់ Java:

ការផ្សព្វផ្សាយ Version សំខាន់ៗរបស់ Java មាន:

-ឆ្នាំ 1995: Java 1.0 ជា Version មួយដែលមានលក្ខណៈប្រសើរសំរាប់ប្រើនៅលើ World Wide Web ដែលបានបង្ហាញនូវអនុភាពសំរាប់ពង្រីកនៅក្នុង Programming Language ផ្សេងៗទៀត។

-ឆ្នាំ 1997: Java 1.1 ដែលបង្កើនដល់ផភាសានូវមធ្យោបាយដ៏សំបូរបែបសំរាប់បង្កើត និងប្រើ User Interface។

-ឆ្នាំ 1998: Java 2 ដែលបង្កើនដល់ផភាសានូវមធ្យោបាយដ៏សំបូរបែប ច្រើនជាង Programming Language ដ៏ទៃទៀត។

៣. លក្ខណៈរបស់ Java:

Java មានលក្ខណៈដូចខាក្រោម:

-Simple: អ្នកបង្កើត Java បានលុបបំបាត់ចោលនូវលក្ខណៈមិនចាំបាច់មួយចំនួនរបស់ Programming Language ជាន់ខ្ពស់។ ដូចជា Java មិនប្រើ Pointer, Structure ឬ Unions, Templates, Header files ឬ Multiple inheritance ជាដើម។

-Object-Oriented: ដូចនឹង C^{++} ដែរ Java ប្រើ Classes ដើម្បីរៀបចំ Code អោយទៅជាសំណុំមួយត្រឹមត្រូវ។ នៅពេលតំណើរការកម្មវិធីនោះ កម្មវិធីមួយបង្កើតនូវ Object តាមរយៈ Classes។ Classes របស់ Java អាច

ទទួលលក្ខណៈពី Classes ដ៏ទៃទៀតបាន ក៏ប៉ុន្តែ Class មួយមិនអាចទទួលលក្ខណៈពី Classes ច្រើនបានទេ។

-Statically Typed: គ្រប់ Objects ទាំងអស់ដែលបានប្រើនៅក្នុងកម្មវិធីមួយ ត្រូវតែប្រកាសមុននឹងប្រើ។ លក្ខណៈនេះអាចធ្វើអោយ Compiler របស់ Java រកឃើញនូវទីតាំង និងប្រាប់អោយដឹងនូវប្រភេទទិន្នន័យដែលមិនត្រូវគ្នា។

-Compiled: មុននឹងយើងអាចតំណើរការកម្មវិធីមួយដែលសរសេរឡើងដោយភាសា Java បាននោះ លុះត្រាតែយើង Compile វាតាមរយៈ Compiler របស់ Java ។ ការធ្វើ Compilation នេះ យើងនឹងទទួលបាននូវឯកសារប្រភេទ Byte-Code ដែលមានលក្ខណៈស្រដៀងទៅនឹងឯកសារប្រភេទ Machine-Code ដែរ ហើយវាអាចតំណើរការនៅក្រោមប្រព័ន្ធ Computer មួយដែលមាន Interpreter របស់ Java។ Interpreter នេះអានឯកសារប្រភេទ Byte-Code ហើយបកប្រែពាក្យបញ្ជាប្រភេទ Byte-Code ទៅជាពាក្យបញ្ជាប្រភេទ Machine-Code ដែលអាចតំណើរការផ្ទាល់នៅលើម៉ាស៊ីនបាន។ ហេតុនេះយើងអាចនិយាយបានថា Java ជាភាសាមួយ ដែលមានលក្ខណៈ Compiled និង Interpreted។

-Architecture Neutral and Portable: ដោយសារតែកម្មវិធីរបស់ Java ត្រូវបាន Compiled ជាទំរង់ Byte-Code ដែលមានលក្ខណៈមិនអាស្រ័យទំរង់ខាងក្នុងរបស់ Computer នោះ Java អាចតំណើរការលើប្រព័ន្ធ Computer មួយ ក៏ដូចទៅលើប្រព័ន្ធ Computer ដែលប្រើ Java Virtual Machine ដ៏ទៃទៀតដែរ។ ដោយសារលក្ខណៈដែល Java Compile ទៅជា Byte-Code ដែលមានលក្ខណៈមិនអាស្រ័យទៅនឹងទំរង់ខាងក្នុងរបស់ Computer ទូទៅនោះ គឺជាផ្នែកធំមួយនៃ Portable។

-Multithread: Java មានលក្ខណៈ Multithread ដែលអាចអោយកម្មវិធីអនុវត្តការងារច្រើនក្នុងពេលតែមួយ។ ឧទាហរណ៍Multithread អាចសំដែងរូបភាពមួយនៅលើអេក្រង់បាន កំឡុងពេលដែលធ្វើការបញ្ចូលទិន្នន័យពី Keyboard ទៅក្នុង Main thread។ គ្រប់កម្មវិធីទាំងអស់មាន Thread យ៉ាងតិចមួយដែលតំណាងអោយតំណើរការនៃកម្មវិធី។

-Garbage Collected: Java មានលក្ខណៈអាចលុបនូវរាល់ Objects ដែលមិនត្រូវបានប្រើបន្តដោយស្វ័យប្រវត្ត ចេញពីក្នុង Memory។ លក្ខណៈនេះជួយសំរួលដល់អ្នកសរសេរកម្មវិធី ក្នុងការគ្រប់គ្រង Memory។

-Robust: ដោយសារ Java ប្រើប្រាស់ Interpreter ដើម្បីត្រួតពិនិត្យទៅគ្រប់កម្មវិធីរបស់ Java ដូច្នេះកម្មវិធីរបស់ Java មិនប៉ះពាល់ទៅដល់ប្រព័ន្ធ Computer ឡើយ។ Java ប្រើប្រាស់ Exception ដែលប្រើសំរាប់ចាប់យកនូវ Error ដែលកើតមានឡើងដើម្បីជៀសវាងនូវផលវិបាកមួយចំនួនដល់ប្រព័ន្ធ Computer។

-Secured: ដោយសារតែ Java មិនប្រើ Pointer ដូច្នេះកម្មវិធីរបស់ Java មិនឆ្លង Virus នៅពេលតំណើរការកម្មវិធី Applet។

-Extensible: Java អាចអោយប្រើបាននូវ Native methods ដែលជា Function ដែលសរសេរដោយ C^{++} programming ។ លក្ខណៈនេះធ្វើអោយអ្នកសរសេរ Functions ដែលអាចប្រតិបត្តិបានលឿនជាងការសរសេរ

Functions នៅលើ Java។

៤. Java Applications និង Java Applets:

គេអាចបង្កើតកម្មវិធី Java បានតាមពីរបៀប: Java Applications និង Java Applet។

-Java Application: គឺជាកម្មវិធីមួយដែលអាចតំណើរការលើ Computer មួយក្រោមប្រព័ន្ធ Computer នោះ។

-Java Applet: គឺជាកម្មវិធីមួយរបស់ Java ដែលអាចបញ្ជូនទៅលើ Internet ហើយអាចតំណើរបាននៅលើ Web Browser ណាក៏បាន អោយតែវា Support Java។ វាអាចប្រើបាននូវរូបភាព, សំលេង, ឬ វីដេអូ។

៥. សំនេរកម្មវិធីបែបវត្ថុ: (*Object-Oriented Programming*)

គ្រប់កម្មវិធីរបស់ Java ទាំងអស់ជាកម្មវិធីសំនេរបែបវត្ថុ។ កម្មវិធីបែបវត្ថុជាចំនុចសំខាន់របស់ Java ដូច្នេះមុននឹងសរសេរកម្មវិធី Java មួយ យើងត្រូវដឹងពីគោលការណ៍មួយចំនួនសិន។

៥.១ លំនាំសំនេរកម្មវិធី (Two Paradigms)

គ្រប់កម្មវិធីកុំព្យូទ័រទាំងអស់ រួមមានពីរធាតុគឺ Code និង Data។ *process-oriented model* មានលក្ខណៈសំគាល់នូវកម្មវិធីមួយ ដូចទៅស៊េរីនៃជំហានបន្តបន្ទាប់គ្នា (បាននំយថា Code)។ ភាសាសរសេរកម្មវិធីបែបនេះមាន Pascal និង C។ *object-oriented programming* សំរាប់គ្រប់គ្រងភាពសាំញ៉ាំ។ សំនេរកម្មវិធីបែបវត្ថុ រៀបចំកម្មវិធីឡើងជុំវិញ Data របស់វា (បាននំយថា Objects) ហើយនិងការបង្កើត Interfaces យ៉ាងល្អអោយ Data នោះ។ កម្មវិធីរបៀបនេះមានលក្ខណៈសំគាល់ដូចទៅនឹង Data ត្រួតពិនិត្យការចូលទៅប្រើ Code នោះ។

៥.២ លក្ខណៈអរូបី (Abstraction)

Data ដែលបានមកពីលំនាំ *process-oriented* អាចបំបែករូបរាងបានទៅជា Objects តាមលក្ខណៈអរូបីដើម្បីងាយស្រួលក្នុងការគ្រប់គ្រងនូវភាពសាំញ៉ាំ។ មាននំយថា Data ទាំងមូលអាចបង្កជារូបរាងដើមដោយការប្រមូលផ្តុំនៃ Objects ទាំងនោះ។ ហេតុនេះ Object និមួយៗ មានលក្ខណៈមួយរបស់ខ្លួន។

៦. គោលការណ៍ទាំងបីនៃសំនេរកម្មវិធីបែបវត្ថុ: (*The three OOP principles*)

គ្រប់ភាសាសំនេរកម្មវិធីបែបវត្ថុទាំងអស់ ផ្តល់នូវ mechanisms ដែលជួយអោយយើងអនុវត្តលំនាំ object-oriented បាន។ Mechanisms ទាំងនេះរួមមាន: encapsulation, inheritance, និង ploymorphysim។

៦.១ គេលការណ៍ Encapsulation

គេអាចនិយាយបានថា Encapsulation គឺជា wrapper មួយដែលអាចការពារ Code និង Data ពីការប្រើប្រាស់របស់ Code ផ្សេងទៀតដែលនៅខាងក្រៅ wrapper នោះ។ ការចូលទៅកាន់ Code និង Data ខាងក្នុង wrapper គឺត្រូវត្រួតពិនិត្យឆ្លងតាមរយ: Interface ដែលបានកំណត់យ៉ាងម៉ត់ចត់។

នៅក្នុង Java មូលដ្ឋានគ្រឹះនៃ Encapsulation គឺជា Class។ Class មួយកំណត់ទំរង់ និងចរិកលក្ខណៈ (Code និង Data) ដែលត្រូវប្រើរួមតាមរយៈការបង្កើត Objects។ Object និមួយៗដែលបានមកពី Class មានទំរង់

និងចរិកលក្ខណៈដូចអ្វីដែលបានកំណត់នៅក្នុង Class នោះ។ ដូច្នេះ Class គឺជាពុម្ពសំរាប់បង្កើត Object។

នៅពេលបង្កើត Class មួយ យើងនឹងកំណត់ Code និង Data ដែលបង្កើត Class នោះ។ ធាតុបង្ក Class ទាំងនោះហៅថា Members របស់ Class។ ដោយឡែក Data កំណត់ដោយ Class ត្រូវចាត់ទុកដូចជាអញ្ញាត្តិ។ Code ដែលប្រតិបត្តិលើ data នោះត្រូវចាត់ទុកដូចជា Method។

ដោយសារវត្ថុបំណងនៃ Class មួយគឺបន្តយភាពសាំញ៉ាំ ធ្វើអោយវាមាន mechanisms សំរាប់បិទបាំងភាពសាំញ៉ាំនៃការប្រើប្រាស់នៅខាងក្នុង Class។ Method ឬ អញ្ញាត្តិនិមួយៗអាចមានកំណត់សំគាល់ជា private ឬ public។ លក្ខណៈ public របស់ Class តាងអោយអ្វីៗដែលអ្នកប្រើប្រាស់ខាងក្រៅនៃ Class អាចចូលប្រើបាន។ Method និង Data ដែលមានលក្ខណៈ privatet អាចចូលប្រើបានដោយ code ដែលជា members របស់ Class ប៉ុណ្ណោះ។ ដូច្នេះ Code ណាមួយដែលមិនមែនជា member របស់ Class មិនចូលប្រើ methods ឬ អញ្ញាត្តិដែលមានលក្ខណៈ private បានទេ។

១.២ គោលការណ៍ Inheritance

Inheritance គឺជាតំណើរដែល Object មួយស្គាល់លក្ខណៈរបស់ Object មួយទៀត។ វាអាចទទួលលក្ខណៈទូទៅពីលំដាប់ថ្នាក់ Class ដែលជាប្រភពដើមរបស់វា។ដូច្នេះហើយ Inheritance mechanism អាចធ្វើអោយ Object មួយទៅជា Object ដែលមានលក្ខណៈកាន់តែលំអិត។

Inheritance មានសកម្មភាពជាមួយ encapsulation ផងដែរ។ ប្រសិនបើ Class មួយប្រមូលផ្តុំលក្ខណៈខ្លះពេលនោះ subclass របស់វានឹងមានលក្ខណៈដូចគ្នា ហើយបូកបន្ថែមលើសលក្ខណៈទាំងឡាយណាដែលជាលក្ខណៈលំអិត។ នេះជាលក្ខណៈលំអិតមួយ ដែលអាចអោយកម្មវិធីបែបវត្ថុ បង្កើនភាពសាំញ៉ាំបន្តិចម្តងៗ ជាជាងភាពសាំញ៉ាំកើនឡើងច្រើនតែម្តង។ Subclass ថ្មីមួយទទួលលក្ខណៈទាំងអស់ពី Class ទាំងឡាយដែលជាប្រភពដើម។

១.៣ គោលការណ៍ Polymorphysm

Polymorphysm (មកពីពាក្យក្រិចមានន័យថា “ច្រើនទំរង់”) គឺជាលក្ខណៈមួយដែលអាចអោយ Interface មួយ អាចប្រើសំរាប់ធ្វើសកម្មភាពអោយ Class ទូទៅមួយ។

និយាយទូទៅ ទស្សនៈ Polymorphysm ជាធម្មតាសំដែងដោយ “Interface មួយមាន Methods ច្រើន”។ នេះមានន័យថា វាអាចរៀបចំគំរោង Interface មួយមានលក្ខណៈទូទៅសំរាប់សកម្មភាពនៃក្រុមមានទំនាក់ទំនងគ្នា។ លក្ខណៈនេះជួយកាត់បន្ថយភាពសាំញ៉ាំ ដោយអនុញ្ញាត្តិអោយ Interface ដូចគ្នា ប្រើសំរាប់កំណត់សកម្មភាពនៃ Class មួយ។

៧. ការចាប់ផ្តើមតំណើរការកម្មវិធី: (*Starting the program*)

ចូរចាប់ផ្តើមដោយបំលែង និងតំណើរការកម្មវិធីតូចមួយជាគំរូ ដូចបានបង្ហាញខាងក្រោមនេះ:

```
/*
     This is a simple Java program.
     call this file "Example.java" */

class Example{
     //Your program begins with a call to main()
     public static void main(String Args[]){
          System.out.println("This is a simple Java program");
     }
}
```

៧.១ ការបញ្ចូលកម្មវិធី (*Entering the program*)

រឿងសំខាន់ទីមួយ ដែលយើងត្រូវដឹងពី Java នោះ គឺជាឈ្មោះដែលយើងផ្តល់អោយឯកសារដើម។ នៅក្នុងឧទាហរណ៍នេះឈ្មោះឯកសារដើមគួរតែជា Example.java។

នៅក្នុង Java ឯកសារដើម ជាឯកសារបែបអត្ថបទដែលមានការកំណត់ Class មួយ ឬច្រើន។ Compiler របស់ Java ត្រូវការឯកសារដើមដែលមាន Extension **.java**។ តាមធម្មតាឈ្មោះរបស់ Class គួរតែស៊ីគ្នាជាមួយឈ្មោះរបស់ឯកសារដើម ដែលផ្ទុកកម្មវិធីនោះ។ យើងត្រូវចាំថា អក្សរតូចធំរបស់ឈ្មោះឯកសារ ត្រូវស៊ីគ្នានឹងឈ្មោះរបស់ Class ព្រោះភាសា Java មានលក្ខណៈប្រកាន់ណាស់។

៧.២ ការបំលែង និងតំណើរការកម្មវិធី (*Compiling and running the program*)

ដើម្បីបំលែងកម្មវិធី Example យើងប្រតិបត្តិដោយប្រើ compiler ឈ្មោះ **javac** រួចបញ្ជាក់ឈ្មោះឯកសារដើម នៅលើ command-line ដូចបង្ហាញខាងក្រោមនេះ:

c:\jdk1.3\javac Example.java ↵

Compiler ឈ្មោះ javac បង្កើត File មួយហៅថា Example.class ដែលផ្ទុកជាទំរង់ Byte-Code នៃកម្មវិធី។ Java Byte-Code គឺតំណាងកំរិតមធ្យមនៃកម្មវិធី ដែលផ្ទុក Instructions ហើយ Java Interpreter នឹងអាចប្រតិបត្តិការ។ ហេតុនេះលទ្ធផលនៃ javac មិនមែនជា Code ដែលអាចប្រតិបត្តិការដោយផ្ទាល់នោះទេ។

ដើម្បីតំណើរការកម្មវិធី ជាធម្មតាត្រូវប្រើ java interpreter ឈ្មោះ **Java**។ គេត្រូវប្រើឈ្មោះរបស់ Class ពី Example នៅលើ command-line ដូចបង្ហាញខាងក្រោមនេះ:

c:\jdk1.3\bin\java Exmaple ↵

ក្រោយពីតំណើរការកម្មវិធីចប់ គេនឹងទទួលបានលទ្ធផលដូចខាងក្រោមនេះ:

This is a simple Java program