ជំពូកទី២

ប្រភេទទិន្នន័យ, អញ្ញាត្តិ, និង Arrays

Java គឺជាភាសាមានលក្ខណៈកំណត់ប្រភេទទិន្នន័យច្បាស់លាស់។ កត្តានេះហើយគឺជា ផ្នែកមួយរបស់ Java ដែលធ្វើអោយមានភាពរឹងមាំ និងមានសុវត្ថិភាព។ រាល់អញ្ញាត្តិ, កន្សោមលេខ និង រាល់ប្រភេទទិន្នន័យសុទ្ធតែត្រូវកំណត់ជាក់លាក់។ ការកំណត់តំលៃទាំងអស់ មិនថាមានលក្ខណៈបញ្ជាក់អោយឃើញនៅខាងក្រៅ ឬតាមរយៈ parameter បញ្ជូនទៅក្នុង method ពេលហៅយកមកប្រើ សុទ្ធតែត្រូវបានពិនិត្យចំពោះភាពត្រូវគ្នានៃប្រភេទទិន្នន័យ។ ប្រភេទទិន្នន័យណាមួយមិនត្រូវគ្នានឹងមាន Error កើតឡើង ហើយត្រូវតែកែបានរួចរាល់មុននឹង compiler បញ្ចាក់ការបំលែង Class។

១. ប្រភេទទិន្នន័យទំរង់ងាយ(*Simple Types*):

Java មានប្រភេទទិន្នន័យទំរង់ងាយចំនួន 8 : byte, short, int, long, char, float, double និង boolean។ ប្រភេទទិន្នន័យទាំងនេះចែកចេញជាបួនក្រុម:

-Integers: ក្រុមនេះរួមមាន byte, short, int និង long ដែលជាចំនួនគត់។

-Floating-point numbers: ក្រុមនេះរួមមាន float និង double ដែលជាចំនួនទស្សភាគ។

-Characters: ក្រុមនេះមានតែ char ដែលតាងអោយនិមិត្តសញ្ញា នៅក្នុងអក្សរ ដូចជាតួអក្សរ និងលេខជាដើម។

-Boolean: ក្រុមនេះរួមមាន boolean ដែលជាប្រភេទទិន្នន័យពិសេស តាងអោយតំលៃពិត និងមិនពិត។

១.១ ប្រភេទទិន្នន័យចំនួនគត់ (Integers)

Java កំណត់ប្រភេទទិន្នន័យចំនួនគត់ជា 4 ប្រភេទ: byte, short, int និង long។ ប្រភេទទិន្នន័យទាំងនេះ មានតំលៃជាចំនួនវិជ្ជមាន អវិជ្ជមាន និង មានសញ្ញា។

ទំហំនៃប្រភេទទិន្នន័យចំនួនគត់ មិនត្រូវបានគិតដូចទៅនិងបរិមានផ្ទុកដែលវាប្រើ ប៉ុន្តែចាត់ទុកដូចលក្ខណៈដែលវាកំណត់អោយអញ្ញាត្តិ និងកន្សោមលេខនូវប្រភេទទិន្នន័យនោះ។

ទំហំនិងតំលៃដែនកំណត់នៃប្រភេទទិន្នន័យចនួនគត់ខុសគ្នាឆ្ងាយណាស់ ដូចបានបង្ហាញខាងក្រោមនេះ:

ឈ្មោះ	ទំហំ	តំលៃដែនកំណត់
long	64	ពី -9,223,372,036,854,775,808 ដល់ 9,223, 372,036,854,775,807
int	32	ពី -2,147,483,648 ដល់ 2,147,483,647
short	18	ពី -32,768 ដល់ 32,767
byte	8	ពី -128 ដល់ 127

ឧទាហរណ៍ខងក្រោមនេះជាកម្មវិធីគណនាចំនួន miles ដែលពន្លឺធ្វើដំណើរក្នុងរយៈពេលកំណត់មួយ។

```
//Compute distance light travels using long variables.
class Light{
      public static void main(String Args[]){
            int lightspeed;
            long days;
            long seconds;
            long distance;
            //approximate speed of light in miles per second
            lightspeed=186000;
            days=1000; //specify number of days here
            seconds=days * 24 * 60 * 60;//convert to seconds
            distance=lightspeed * seconds;//compute distance

            system.out.print("In " + days);
            system.out.print(" days light will travel about ");
            system.out.print(distance + " miles.");
      }
}
```

១.២ ប្រភេទទិន្នន័យចំនួនទសភាគ (Floating-point types)

ប្រភេទទិន្នន័យចំនួនទស្សភាគ គឺជាចំនួនពិត ដែលគេប្រើនៅពេលវាយតំលៃកន្សោមលេខ ដែលត្រូវការទស្សភាគ។ ឧទាហរណ៍ ការគណនាឬសការេ ស៊ីនុស កូស៊ីនុស ដែលលទ្ធផលត្រូវការតំលៃទស្សភាគជាក់លាក់។ វាមានប្រភេទទិន្នន័យចំនួនទស្សភាគចំនួនពីរគឺ float និង double។ ទំហំ និងតំលៃដែនកំណត់ត្រូវបានបង្ហាញដូចខាងក្រោម:

ឈ្មោះ	ទំហំ	តំលៃដែនកំណត់
double	64	ពី -1.79769313486231570E+308 ដល់ 1.79769313486231570E+308
int	32	ពី -3.40292347E+38 ដល់ -3.40292347E+38

ឧទាហរណ៍ខាងក្រោមជាកម្មវិធីមួយប្រើអញ្ញាត្តប្រភេទ double ដើម្បីគណនាក្រលាផ្ទៃរបស់រង្វង់មួយ:

```
//compute the area of a circle
class Area{
      public static void main(String Args[]){
            double pi, r, a;

            r=10.8;           //radius of circle
            pi=3.1416;        //pi, aprroximately
            a=pi * r * r;     //compute area

            System.out.println("Area of circle is " + a);
      }
}
```

១.៣ ប្រភេទទិន្នន័យជាតួអក្សរ (Characters)

នៅក្នុងភាសា Java ប្រភេទទិន្នន័យដែលប្រើសំរាប់ផ្ទុកអក្សរនោះ គឺជា Char។ Java ប្រើ Unicode ដើម្បីតាងអោយតួអក្សរ។ Unicode កំណត់អោយតួអក្សរអន្តរជាតិពេញលេញ ដែលអាចតាងអោយគ្រប់តួអក្សរទាំង

អស់មាននៅក្នុងភាសាមនុស្សជាតិ។ លក្ខណៈនេះវាត្រូវការទំហំ 16 bits។ ហេតុនេះនៅក្នុងភាសា Java គឺជាប្រភេទទិន្នន័យទំហំ 16 bits និងមានតំលៃដែនកំណត់ពី 0 ដល់ 65,536។ វាពុំមានតំលៃអវិជ្ជមាននោះទេ។ តួអក្សរធម្មតានៃ ASCII នៅតែមានតំលៃពីចន្លោះ 0 ដល់ 127 ដដែល ហើយតួអក្សរ ISO-Latin-1 មានតំលៃពីចន្លោះ 0 ដល់ 255។

កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការប្រើអញ្ញាត្តប្រភេទ Char:

```
//demonstrate char data type.
class CharDemo{
      public static void main(String Args[]){
            char chi, ch2;

            ch1=88;     //code for X
            ch2='y';
            System.out.print("ch1 and ch2: ");
            System.out.print(ch1 + "" +ch2);
      }
}
```

ចូរកំណត់សំគាល់ថា ch1 ត្រូវបានកំនត់តំលៃ 88 ដែលជាតំលៃ ASCII (និង Unicode) ហើយត្រូវគ្នានឹងតួ អក្សរ X។

ទោះបីជា char មិនមែនជាចំនួនគត់ក៏ដោយ ក្នុងករណីជាច្រើន គេអាចធ្វើប្រមាណវិធីជាមួយនឹងវាដូចចំនួនគត់ដែរ។ លក្ខណៈនេះ អាចអោយយើងបូកពីរតួអក្សរបញ្ចូលគ្នា ឬបង្កើនតំលៃនៃអញ្ញាត្តតួអក្សរមួយ។ ចូរសង្កេតកម្មវិធីខាងក្រោមនេះ:

```
//char variables behave like integers.
class CharDemo2{
      public static void main(String Args[]){
            char ch;

            ch='X';
            System.out.print("ch contains " + ch);
            ch++; //incensement ch
            System.out.print("ch now is " + ch);
      }
}
```

នៅក្នុងកម្មវិធីខាងលើ ch អោយតំលៃតំបូងគឺ X បន្ទាប់ ch បង្កើនមួយឯកតា។ ការបង្កើនតំលៃមួយឯកតានាំអោយ ch មានតំលៃជា Y ដែលជាតួអក្សរបន្ទាប់ពី X នៅក្នុងលំដាប់ ASCII (និង Unicode)។

១.៤ **<u>ប្រភេទទិន្នន័យបែបតក្ក</u>** (Boolean)

ប្រភេទទិន្នន័យ boolean ផ្តល់នូវទីតាំងផ្ទុកទំហំ 1 bit តាងអោយតំលៃពិត និង មិនពិត។ នេះជាប្រភេទទិន្នន័យដែលបានមកពីការអោយតំលៃនៃសញ្ញាណនព្វន្តលេខធៀប ដូចជា a<b ។ boolean ត្រូវការប្រើនៅក្នុងកន្សោមលក្ខ័ណ្ឌនៃឃ្លា if និង for ជាដើម។

កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការប្រើប្រាស់ boolean:

```
//demonstrate boolean values.
Class BooleanTest{
      public static void main(String Args[]){
            boolean b;

            b=false;
            System.out.println("b is " + b);
            b=true;
            System.out.println("b is " + b);
            //a boolean value can control the if statement.
            if (b) System.out.println("This is executed");

            b=false;
            if (b) System.out.println("This is not executed");

            //outcome of a relational operator is a boolean value
            System.out.println("10 > 9 is " + (10 > 9));
      }
}
```

២. ការប្រើពាក្យ(*Lexical Issues*):

២.១ តំណកឃ្លា(White space)

ភាសា Java គឺជាភាសាមានទំរង់សេរី។ នេះមានន័យថា គេមិចចាំបាច់ អនុលោមតាមគោលការណ៍នៃការដកឃ្លាចុះដើមបន្ទាត់ពិសេសណាមួយ។ តំណកឃ្លា គឺជាការដកឃ្លាមួយ space, មួយ tab, និង មួយបន្ទាត់។

២.២ Identifiers

Identifiers ត្រូវបានប្រើចំពោះឈ្មោះ Class, ឈ្មោះ Method និងឈ្មោះអញ្ញាត្ត។ Identifier មួយអាចជាលំដាប់អក្សរបែបពិពណ៌នាណាមួយ អក្សរតូចធំ លេខ ឬតួអក្សរសញ្ញា underscore និង តួអក្សរសញ្ញា $។ វាមិនត្រូវចាប់ផ្តើមដោយលេខ ព្រោះវាអាចច្រលំនឹងលេខ។ នេះជាឧទាហរណ៍ខ្លះនៃ Idetifiers ដែលត្រឹមត្រូវ:

AvgTemp	count	a4	$test	this_is_ok

ឈ្មោះអញ្ញាត្តមិនត្រឹមត្រូវមាន:

2count	high-temp	Not/ok

២.៣ ក្បួនសំណេរអោយបានត្រឹមត្រូវ(Literals)

តំលៃអថេរមួយនៅក្នុង Java ត្រូវបានបង្កើតឡើងដោយប្រើតំណាងពាក្យមួយតាមក្បួនសំនេរអោយតំលៃនោះ។ ឧទាហរណ៍ខាងក្រោមនេះជាការតាងពាក្យសំនេរមួយចំនួនអោយបានត្រឹមត្រូវ:

100	98.6	'X'	"This is a test"

២.៣.១ ក្បួនសំណេរប្រភេទចំនួនគត់អោយបានត្រឹមត្រូវ(Integer Literals)

ចំនួនគត់ ជាប្រភេទទិន្នន័យដែលប្រើញឹកញាប់ក្នុងកម្មវិធីធម្មតា ឧទាហរណ៍1, 2, 3, និង 42។ តំលៃលេខទាំងនេះជាតំលៃលេខប្រព័ន្ធគោលដប់។ នៅក្នុង Java ក៏មានតំលៃពីរប្រភេទផ្សេងទៀត ដែលស្ថិតក្នុងលក្ខណៈចំនួនគត់ដែរ គឺប្រភេទ Octal (ប្រព័ន្ធគោល ៨) និងប្រភេទ Hexadecimal (ប្រព័ន្ធគោល ១៦)។ នៅក្នុងភាសា Java

គេអាចសំគាល់តំលៃ octal បានដោយប្រើលេខសូន្យនៅពីមុខ។ តំលៃលេខធម្មតា ក្នុងប្រព័ន្ធគោលដប់មិនអាចដាក់លេខសូន្យនៅពីមុខបានទេ។ ហេតុនេះតំលៃ 09 ហាក់ដូចជាត្រឹមត្រូវ តែផ្ទុយទៅវិញវាធ្វើអោយ Error មួយកើតឡើងពី compiler ដោយលេខ 9 ស្ថិតនៅក្រៅ octal ដែលស្ថិតនៅចន្លោះ 1 ទៅ 7។ រីឯ Hexadecimal គេអាចសំគាល់បានដោយដាក់ 0x ឬ 0X នៅពីមុខ។ តំលៃលេខ Hexadecimal ស្ថិតនៅចន្លោះ ពី 0 ទៅ 15។

ចំពោះចំនួនគត់ប្រភេទ long នោះ បើចង់អោយអញ្ញាត្តមួយផ្ទុកតំលៃធំនោះ គេត្រូវបញ្ជាក់នៅខាងក្រោយលេខនូវអក្សរ L (l ឬ L)។ ឧទាហរណ៍ 0x7fffffffffffffL ឬ 9223372036854775807L។

២.៣.២ ក្បួនសំណេរប្រភេទទសភាគអោយបានត្រឹមត្រូវ(Floating-point Literals)

ចំនួនទសភាគ គឺជាតំលៃលេខប្រព័ន្ធគោលដប់ ដែលមានតំលៃនៅក្រោយក្បៀស។ គេអាចសរសេរចំនួនទសភាគបានតាមលក្ខណៈធម្មតា (standard notation) ឬលក្ខណៈបែបវិទ្យាសាស្ត្រ (scientific notation)។ ចំនួនទសសភាគតាមលក្ខណៈធម្មតា គឺបង្កឡើងដោយលេខប្រព័ន្ធគោលដប់ជាមួយសញ្ញាចំនុច ហើយតំលៃទសភាកនៅជាប់ខាងក្រោយ ដូចជា 2.0, 3.14159 និង 0.6667។ ចំណែកឯចំនួនទសភាគជាលក្ខណៈវិទ្យាសាស្ត្រ ប្រើដូចលក្ខណៈទូទៅដែរ ហើយបន្ថែមស្វ័យគុណនៃ 10 ជាមួយនិងលេខដែលត្រូវគុណ។ ស្វ័យគុណតាងដោយអក្សរ E ឬ e ដោយមានលេខប្រព័ន្ធគោលដប់នៅជាប់ពីក្រោយ (អាចជាលេខវិជ្ជមាន ឬ អវិជ្ជមាន)។ ឧទាហរណ៍ 6.022E23, 314259E-05 និង 2e+100។

នៅក្នុងភាសា Java ជាធម្មតាចំនួនទសភាគបញ្ជាកអោយប្រភេទ double គេអាចដាក់ ឬមិនដាក់អក្សរ D ឬ d នៅពីក្រោយលេខ។ ផ្ទុយទៅវិញគេត្រូវតែដាក់អក្សរ F ឬ f នៅពីក្រោយលេខទសភាគ ដើម្បីបញ្ជាក់ប្រភេទ float។

២.៣.៣ ក្បួនសំណេរប្រភេទតក្កអោយបានត្រឹមត្រូវ(Boolean Literals)

តំលៃតក្ក មានតែពីរប៉ុណ្ណោះ គឺ True និង False។ គេមិនអាចបំលែងតំលៃ True និង False អោយទៅជាតំលៃលេខបាននោះទេ ហើយគេក៏មិនអាចយកតំលៃលេខ 1 ទៅជំនួសអោយ True និង លេខ 0 ជំនួសអោយ False ដែរ។

២.៣.៤ ក្បួនសំណេរប្រភេទអក្សរអោយបានត្រឹមត្រូវ(Character Literals)

នៅក្នុង Java តួអក្សរត្រូវរៀបទៅតាម Unicode។ វាមានទំហំ 16 bits ដែលអាចបំលែងទៅជាចំនួនគត់ និងប្រើជាមួយសញ្ញាណនព្វន្តលេខ ដូចជា បូក និងដក ជាដើម។ អក្សរត្រូវបានសរសេរនៅក្នុង '.....'។ គ្រប់តួអក្សរ ASCII ទាំងអស់ អាចសរសេរដាក់ក្នុងចន្លោះសញ្ញាអព្ភន្តរៈបានដោយផ្ទាល់ ដូចជា 'a', 'z', និង '@'។ ចំពោះតួអក្សរដែលមិនសរសេរដោយផ្ទាល់នោះ តំរូវអោយយើងបញ្ចូសញ្ញា "\" ដាក់ជាមួយដូចជា '\n' សំរាប់តួអក្សរចុះដើមបន្ទាត់។ ក្រៅពីនេះគេអាចប្រើសញ្ញា \ ជាមួយចំនួន octal និង hexadecimal ផងដែរ ពេលនោះវានឹងអោយតំលៃជាតួអក្សរដែលត្រូវគ្នា។ ចំពោះ octal គេអាចដាក់លេខចំនួន 3 ខ្ទង់នៅខាងក្រោយសញ្ញានោះ ឧទាហរណ៍ '\141' ត្រូវគ្នានឹងអក្សរ 'a'។ ដោយឡែកចំពោះលេខ hexadecimal គេត្រូវប្រើសញ្ញា '\u' ហើយដាក់លេខចំនួន 4 ខ្ទង់នៅពី

ក្រោយ ឧទាហរណ៍ '\u0061' ត្រូវគ្នានឹងតួអក្សរ 'a'។ តារាងខាងក្រោមនេះបង្ហាញពីតារាងតួអក្សរដែលប្រើជាមួយសញ្ញា '\'។

\ddd	Octal character (*ddd*)
\uxxxx (*xxx*)	Hexadecimal UNICODE character
\'	Single quote
\"	Double quote
\\	Backslash
\r	Carriage return
\n	New line (also known as line feed)
\f	Form feed
\t	Tab

២.៣.៥ ក្បួនសំណេរ String អោយបានត្រឹមត្រូវ(String Literals)

ក្បួនសំនេរ String នៅក្នុងភាសា Java បានកំនត់ដូចភាសាដ៏ទៃទៀតដែរ គឺអក្សរត្រូវសរសេរដាក់នៅក្នុងចន្លោះសញ្ញា "...."។ ឧទាហរណ៍

"Hello, world"
"two\nlines"
"\"this is in quotes\""

គួរចងចាំថា string របស់ Java ត្រូវតែចាប់ផ្តើម និង បញ្ចប់នៅលើបន្ទាត់តែមួយ ព្រោះវាពុំមានអក្សរ ឬសញ្ញា សំរាប់ភ្ជាប់ string នៅលើបន្ទាត់ពីរផ្សេងគ្នា ដូចភាសាផ្សេងឡើយ។

២.៤ ការអធិប្បាយ(Comments)

នៅក្នុង Java comment មានបីប្រភេទ: សំរាប់ comment មួយបន្ទាត់គឺ //............... និង comment ច្រើនបន្ទាត់គឺ /*.......................*/។ ចំណែកឯប្រភេទទី 3 គឺប្រើសំរាប់កំណត់ត្រាឯកសារ ជាពិសេសគេប្រើសំរាប់បង្កើតឯកសារ HTML មួយ ដែលកត់ត្រាឯកសារកម្មវិធីថ្មី។ គេអាចសំគាល់បានដោយវាចាប់ផ្តើមដោយ /** ហើយបញ្ចប់វិញដោយ */។

២.៥ ពាក្យគន្លឹះរបស់ Java(The java reserved keywords)

វាមានពាក្យគន្លឹះសំខាន់ៗ ជាច្រើននៅក្នុងភាសា Java។ ពាក្យគន្លឹះបន្សំជាមួយក្បួននៃសញ្ញាណនព្វន្តលេខ និង សញ្ញាខ័ណ្ឌបំបែក បង្កើតបានជានិយមន័យនៃភាសា Java ។ពាក្យគន្លឹះទាំងនេះមិនអាចប្រើជា ឈ្មោះអញ្ញាត្ត ឈ្មោះ Class ឬ ឈ្មោះ Method ឡើយ។

abstract	continue	finally	interface	public	this	boolean
default	float	long	return	throw	break	do
for	native	short	throws	byte	double	if
new	static	transient	case	else	implements	null
strictfp	true	catch	extends	import	package	super
try	char	false	instanceof	private	switch	void
class	final	int	protected	synchronized	volatile	while

២.៦ សញ្ញាខ័ណ្ឌបំបែក(Separators)

នៅក្នុងភាសា Java មានតួអក្សរមួយចំនួនតូចដែលប្រើជាសញ្ញាខ័ណ្ឌបំបែក។ សញ្ញាដែលគេប្រើជាញឹកញាប់ជាងគេនោះគឺ (;) ដែលប្រើសំរាប់បញ្ចប់ឃ្លាព្រយោគ។ ខាងក្រោមជាសញ្ញាទាំងនោះ និងអត្ថន័យរបស់វា:

និមិត្តសញ្ញា	គោលបំណង
()	ប្រើសំរាប់រាយ parameters នៅក្នុង methods។ វាអាចប្រើសំរាប់កំនត់អទិភាព នៅក្នុងកន្សោមលេខ ប្រើក្នុងលក្ខ័ណនៃតំណើរប្រតិបត្តិដដែលៗ ហើយនិងការដាក់អមបញ្ជាក់ប្រភេទទិន្នន័យ។
{}	ប្រើសំរាប់កំនត់តំលៃតំបូងទៅអោយ Array។ ហើយប្រើសំរាប់កំនត់ទំហំ Block នៃ Code អោយ Class, Method និង អញ្ញាត្ត។
[]	ប្រើនៅពេលប្រកាស Array និង ការបញ្ជាក់អោយធាតុនៃ Array។
;	ប្រើសំរាប់បញ្ចប់ឃ្លាព្រយោគ។
,	ប្រើសំរាប់ខ័ណ្ឌidentifiers នៅបន្តបន្ទាប់គ្នា ពេលប្រកាសអញ្ញត្តិ និង នៅក្នុងឃ្លា for។
.	ប្រើសំរាប់ប័ណ្ណឈ្មោះ packages ពី subpackage ហើយនិង class។ ប្រើសំរាប់បញ្ជាក់អញ្ញាត្តិ និង methods របស់ objects ផងដែរ។

៣. អញ្ញាត្តិ(*Variables*):

៣.១ ការប្រកាសអញ្ញាត្តិ(Declaring a variable)

គ្រប់អញ្ញត្តិទាំអស់ ត្រូវតែប្រកាសមុននឹងយកមកប្រើ នៅក្នុងភាសា Java។ ទំរង់នៃការប្រកាសអញ្ញាត្តិមានដូចខាងក្រោមនេះ:

```
type  indentifier[=value][, indetifier[=value]...];
```

ដែលក្នុងនោះ type គឺជាប្រភេទទិន្នន័យណាមួយរបស់ Java ឬជាឈ្មោះរបស់ Class ឬ Interface។ identifier គឺជាឈ្មោះអញ្ញាត្តិ។ យើងអាចកំនត់តំលៃតំបូងទៅអោយអញ្ញាត្តិ ដោយបញ្ជាក់សញ្ញាស្មើ និង តំលៃមួយ។ គួរចងចាំថាកន្សោមកំនត់តំលៃ ត្រូវអោយលទ្ធផលជាតំលៃនៃប្រភេទទិន្នន័យដូចគ្នា នឹងប្រភេទទិន្នន័យដែលកំនត់អោយអញ្ញាត្តិ។ ការប្រកាសអញ្ញាត្តិច្រើនដែលមានប្រភេទទិន្នន័យដូចគ្នា យើងអាចប្រើសញ្ញាក្បៀស (,) ដើម្បីញែកពីអញ្ញាត្តិមួយ ទៅអញ្ញាត្តិមួយទៀត។ ឧទាហរណ៍:

```
int a, b, c;        //declares three ints a, b, and c.
int d=3,e ,f=5;     //declares three more ints, initializing d and f.
byte z=22;          //initializes z.
double pi=3.14159; //declares an  approximation of pi.
char x='X';         //the variable x has the value 'X'.
```

៣.២ ការកំនត់តំលៃដែលមានលក្ខណៈប្រែប្រួល(Dynamic initialization)

Java អាចអោយអញ្ញាត្តិកំនត់តំលៃតំបូងដែលមានលក្ខណៈប្រែប្រួល ដោយប្រើកន្សោមលេខត្រឹមត្រូវណាមួយ កំនត់ទៅអោយអញ្ញាត្តិពេលប្រកាស។

កម្មវិធីខាងក្រោមនេះជាកម្មវិធីតូចមួយ គណនាប្រវែងអ៊ីប៉ូតេនុសនៃត្រីកោណកែង កាលណាស្គាល់ប្រវែងជ្រុងពីរទៀត។

```
//demostrate dynamic initialization.
class DynInit{
      public static void main(String Args[]){
            double a = 3.0, b = 4.0;
            //c is dynamically initialized
            double c = Math.sqrt(a * a + b * b);
            System.out.println("Hypotenuse is " + c);
      }
}
```

៣.៣ ទំហំទីតាំង និង ការិតស្ថិតនៅនៃអញ្ញាត្តិ(The scope and lifetime of variables)

Java អាចអោយអញ្ញាត្តិប្រកាសនៅក្នុង Block ណាមួយ។ Block ចាប់ផ្តើដោយសញ្ញា { និង បញ្ចប់ដោយសញ្ញា }។ Block កំណត់ទំហំមួយហៅថា Scope។

ភាសាសំនេរភាគច្រើន បានកំនត់ទំហំ Scope ពីរប្រភេទ គឺ global និង local។ នៅក្នុងភាសា Java ទំហំ Scope សំខាន់ពីរគឺ Class និង Method។ Scope មួយស្ថិតនៅក្នុង Scope មួយទៀត ដើម្បីលក្ខណះនេះ ចូរសង្កេត កម្មវិធីខាងក្រោម:

```
//Demonstrate block scope
class Scope{
      public static void main(String Args[]){
            int x;            //known to all code within main

            x=10;
            if(x==10){        //start new scope
                  int y=20;   //known only in this block
                  //x and y both known here
                  System.out.println("x and y: " + x + " " + y);
                  x=y * 2;
            }
            //y=100;          //Error! y not known here
            //x is still known here
            System.out.println("x is " + x);
      }
```

អញ្ញាត្តិត្រូវបង្កើតឡើងនៅពេល scope របស់វាឈានចូល ហើយត្រូវបំបាត់នៅពេល scope របស់វាចាកចេញផុត។ ការិតស្ថិតនៅរបស់អញ្ញាត្តិមួយ គឺត្រូវកំនត់ដោយ scope របស់វា។

បើសិនជាការប្រកាសអញ្ញាត្តិមួយមានកំនត់តំលៃដំបូង ពេលនោះវានឹងត្រូវកំនត់តំលៃដំបូងសារជាថ្មី រៀងរាល់

Block ដែលវាប្រកាសឈានចូលមកដល់។ ឧទាហរណ៍ ចូរសង្កេតកម្មវិធីខាងក្រោម:

```
//demonstrate lifetime of a variable
class LifeTime{
      public static void main(String Args[]){
            int x;

            for(x=0;x<3;x++){
                  int y=-1;
                  //y is initialized each time block is entered

                  System.out.println(" y is: " + y);
                  //this is always print -1

                  y=100;
                  System.out.println("y is now: " + y);
            }
      }
}
```

៤. ការបំលែងប្រភេទទិន្នន័យ និង ការបញ្ជាក់ប្រភេទទិន្នន័យ(*Type conversion and casting*):

ការកំនត់តំលៃនៃប្រភេទទិន្នន័យមួយទៅអោយអញ្ញាត្តិដែលមានប្រភេទទិន្នន័យផ្សេង ប្រសិនបើប្រភេទទិន្នន័យទាំងពីរត្រូវគ្នានោះ Java នឹងធ្វើការបំលែងដោយស្វ័យប្រវត្តិ។ ទោះជាយ៉ាងណាក៏ដោយ មិនមែនគ្រប់ប្រភេទទិន្នន័យទាំងអស់សុទ្ធតែត្រូវគ្នានោះទេ ហេតុនេះ រាល់ប្រភេទទិន្នន័យទាំងអស់ មិនអនុញ្ញាត្តិអោយធ្វើការបំលែងដោយស្វ័យប្រវត្តដែរ។ ក្នុងខណៈនោះគេនៅតែអាច ធ្វើការបំលែងប្រភេទទិន្នន័យ ដែលមិនត្រូវគ្នាបានដែរ។ ដើម្បីធ្វើការបំលែងលក្ខណៈបែបនោះ គេត្រូវប្រើការបញ្ជាក់ច្បាស់លាស់ រវាងប្រភេទទិន្នន័យមិនត្រូវគ្នានោះ។

៤.១ ការបំលែងប្រភេទទិន្នន័យដោយស្វ័យប្រវត្ត(Java’s automatic conversions)

កាលណាទិន្នន័យមួយប្រភេទ ត្រូវបានកំនត់អោយអញ្ញត្តិនៃប្រភេទទិន្នន័យផ្សេងនោះ ការបំលែងប្រភេទទិន្នន័យដោយស្វ័យប្រវត្តនឹងចាប់ប្រតិបត្តិការ បើសិនជួបលក្ខ័ណ្ឌដូចខាងក្រោម:

-ប្រភេទទិន្នន័យទាំងពីរត្រូវគ្នា

-ប្រភេទទិន្នន័យគោលដៅ (នៅអង្គខាងឆ្វេង) ធំជាងប្រភេទទិន្នន័យដើម (នៅអង្គខាងស្តាំ)។ ទោះជាយ៉ាងណាក៏ដោយ ប្រភេទទិន្នន័យជាលេខ មិនត្រូវគ្នានឹង char ឬ boolean នោះទេ។ ហើយ char និង boolean ក៏មិនត្រូវគ្នាដែរ។

៤.២ ការបំលែងប្រភេទទិន្នន័យមិនត្រូវគ្នា ដោយបញ្ជាក់បន្ថែមប្រភេទទិន្នន័យ(Casting incompatible types)

ដើម្បីធ្វើការបំលែងប្រភេទទិន្នន័យពីរដែលមិនត្រូវគ្នា នោះយើងត្រូវប្រើលក្ខណៈបញ្ជាក់បន្ថែម (Cast)។ វាមានទំរង់ទូទៅគឺ:

```
(target-type) value
```

target-type កំនត់ប្រភេទទិន្នន័យដែលយើងត្រូវការ ដើម្បីបំលែងតមលៃអោយទៅប្រភេទទិន្នន័យ

នោះ។

វាមានការបំលែងប្រភេទទិន្នន័យផ្សេងកើតឡើង នៅពេលតំលៃទសភាគមួយត្រូវបានកំនត់ទៅជាប្រភេទទិន្នន័យចំនួនគត់: ការតំរឹម។ ដូចយើងបានដឹងហើយថា ចំនួនគត់មិនមានផ្នែកទសភាគនោះទេ។ ហេតុនេះនៅពេលដែលចំនួនទសភាគមួយ ត្រូវកំនត់អោយទៅជាចំនួនគត់នោះ ផ្នែកទសភាគត្រូវបាត់បង់។ ឧទាហរណ៍ប្រសិនបើតំលៃ 1.23 ត្រូវកំនត់អោយទៅជាចំនួនគត់នោះ លទ្ធផលដែលទទួលបានគឺ 1។ 0.23 ត្រូវបានតំរឹម។

កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការបំលែងប្រភេទទិន្នន័យខ្លះ ដែលត្រូវការបញ្ជាក់បន្ថែម:

```
//demonstrate casts
class Conversion{
      public static void main(String Args[]){
            byte b;
            int i=257;
            double d=323.142;

            system.out.println("\nConversion of integer to byte");
            b=(byte) i;
            System.out.println("b and i " + b + "" + i);

            system.out.println("\nConversion of double to integer");
            i=(int) d;
            System.out.println("i and d " + i + "" + d);

            system.out.println("\nConversion of double to byte");
            b=(byte) d;
            System.out.println("b and d " + b + "" + d);

      }
}
```

ចូរសង្កេតមើលការបំលែងប្រភេទទិន្នន័យនិមួយៗ។ នៅពេលតំលៃ 257 ត្រូវបញ្ជាក់បន្ថែមទៅអោយ អញ្ញាត្តិប្រភេទ byte នោះលទ្ធផលគឺជាសំណល់នៃផលចែករវាង 257 ជាមួយ 256 (ដែលជាតំលៃដែនកំនត់នៃ byte) គឺស្មើនឹង 1 នៅក្នុងករណីនេះ។ នៅពេល d ត្រូវបំលែងជា int នោះផ្នែកទសភាគត្រូវបានបាត់បង់។ នៅពេល d ត្រូវបំលែងជា byte នោះផ្នែកទសភាគត្រូបាត់បង់ រួចហើយតំលៃត្រូវថយចុះដោយរកតំលៃនៃផលចែកជាមួយ 256 ហើយលទ្ធផលទទួលបានគឺ 67។

៤.៣ <u>គោលការណ៍បង្កើតប្រភេទទិន្នន័យ</u>(The type promotion rules)

លើសពីនេះ ការលើកប្រភេទទិន្នន័យ byte និង short ទៅជា int នោះ Java មានគោលការណ៍បង្កើនប្រភេទទិន្នន័យជាច្រើន ដែលប្រើនៅក្នុងកន្សោម។

កម្មវិធីខាងក្រោមនេះ បង្ហាញពីរបៀបដែលតំលៃនិមួយៗ ទទួលការបង្កើនប្រភេទទិន្នន័យដើម្បីធ្វើអោយត្រូវគ្នានឹងតំលៃទីពីរចំពោះសញ្ញាណព្វន្តលេខនិមួយៗ

```
class Promote{
      public static void main(String Args[]){
            byte b=42;
            char c='a';
            short s=1024;
            int i=50000;
            float f=5.67f;
            double d=0.1234;
            double result=(f*b) + (i/c) –(d*s);
            System.out.println((f*b) + "+" + (i/c) + "-" + (d*s));
            System.out.println("restult= " + result);
      }
}
```

៥. Arrays:

Array គឺជាសំនុំអញ្ញាត្តិដែលមានប្រភេទទិន្នន័យដូចគ្នា ក្រោមឈ្មោះរួមមួយ គ្រាន់តែខុសគ្នាដោយលេខ Index។ Array នៃប្រភេទទិន្នន័យណាមួយក៏ដោយ អាចបង្កើតឡើង និងមានទំហំ មួយ ឬច្រើនវិមាត្រ។

៥.១ Array មួយវិមាត្រ(One-Dimensional Arrays)

Array មួយវិមាត្រ ជាសំណុំអញ្ញាត្តិដែលមានប្រភេទទិន្នន័យដូចគ្នា។ ដើម្បីបង្កើត Array មួយយើងត្រូវបង្កើតអញ្ញាត្តិ Array មួយនៃប្រភេទទិន្នន័យដែលត្រូវការ។ ទំរង់ទូទៅនៃការប្រកាស Array មួយវិមាត្រគឺ:

```
type var-name[];
```

ក្នុងនោះ type បញ្ជាក់ប្រភេទទិន្នន័យអោយ Array។ ឧទាហរណ៍ខាងក្រោមនេះ Array មួយឈ្មោះ month_days ដែលមានប្រភេទទិន្នន័យជា int។

```
int month_days[];
```

ទោះបីជាការប្រកាសនេះបានរៀបចំឡើងនូវ month_days ជាអញ្ញាត្តិ Array ក៏ដោយ តែតាមពិតគ្មាន Array កើតឡើងនោះទេ។ តំលៃ month_days ត្រូវកំនត់ជា null ដែលតាងអោយ Array មួយគ្មានតំលៃ។ ដើម្បីភ្ជាប់ month_days ជាមួយនឹងរូបរាង array ពិតប្រាកដនៃប្រភេទចំនួនគត់ យើងត្រូវបង្កើតទីតាំងមួយ ដោយប្រើ new និងកំនត់តំលៃទៅនឹង month_days។ new ជាសញ្ញាណនព្វន្តលេខពិសេសមួយ អាចបង្កើតទីតាំងក្នុង memory។

ទំរង់ទូទៅរបស់ new ដែលគេប្រើជាមួយនឹង array មួយវិមាត្រ មានដូចខាងក្រោម:

```
array-var=new type[size];
```

ដែលក្នុងនោះ type បញ្ជាក់ពីប្រភេទទិន្នន័យដែលត្រូវបង្កើតទីតាំង។ រីឯ size កំនត់ចំនួនធាតុក្នុង array និង array-var ជាអញ្ញាត្តិ array ដែលត្រូវភ្ជាប់ទៅនឹង array។ បានន័យថា ប្រើ new សំរាប់បង្កើតទីតាំងអោយ array មួយនោះ យើងត្រូវកំនត់ប្រភេទទិន្នន័យ និងចំនួនធាតុដើម្បីបង្កើតទីតាំង។ ធាតុទាំងអស់របស់ array ដែលបង្កើតទីតាំងដោយប្រើ new នោះ នឹងត្រូវកំនត់តំលៃដំបូងស្មើនឹងសូន្យ ជាស្វ័យប្រវត្តិ។ នៅក្នុងឧទាហរណ៍នេះ បង្កើតទីតាំងរបស់ array មួយដែលមាន 12 ធាតុនៃប្រភេទចំនួនគត់ ហើយភ្ជាប់វាទៅនឹង month_days។

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

នេះជាកម្មវិធីមួយបង្កើត array មួយនៃចំនួនថ្ងៃ នៅក្នុងខែនិមួយៗ:

```
//demonstrate a one-dimensional array
class Array{
      public static void main(String Args[]){
            int month_days[]=new int[12];
            month_days[0]=31;
            month_days[1]=28;
            month_days[2]=31;
            month_days[3]=30;
            month_days[4]=31;
            month_days[5]=30;
            month_days[6]=31;
            month_days[7]=31;
            month_days[8]=30;
            month_days[9]=31;
            month_days[10]=30;
            month_days[11]=31;

            System.out.println("April has " + month_days[3] + " days.");

      }
}
```

Array អាចកំនត់តំលៃដំបូងអោយវានៅពេលប្រកាសវា។ ការកំនត់តំលៃដំបូងនៃ array មួយគឺជាការរាយតំលៃនៃធាតុនិមួយៗ ដែលឃ្លាតពីគ្នាដោយសញ្ញាក្បៀស និង អមដោយសញ្ញា {}។ Array នឹងបង្កើតឡើងមានទំហំល្មមគ្រប់គ្រាន់សំរាប់ដាក់ចំនួនធាតុ ដែលយើងកំនត់អោយក្នុង array ដោយស្វ័យប្រវត្តិ។ វាមិនត្រូវការប្រើ new ទេ។ ចូរសង្កេតមើលឧទាហរណ៍ខាងក្រោម:

```
//An improved version of the previous program.
class AutoArray{
      public static void main(String Args[]){
            int month_days[]={31,28,31,30,31,30,31,31,30,31,30,31};
            System.out.println("April has " + month_days[3] + " days.");
      }
}
```

```
//Average an array of value.
class Average{
      public static void main(String Args[]){
            double nums[]={10.1,11.2,12.3,13.4,14.5};
            double result=0;
            int i;

            for(i=0;i<5;i++){
                  result=result + nums[i];
            }
            System.out.println("Average is " + result/5);
      }
}
```

៩.២ Array ច្រើនវិមាត្រ(Multidimensional Arrays)

នៅក្នុង Java , Array ច្រើនវិមាត្រគឺជា Array នៃ Array។ ដើម្បីប្រកាស Array ច្រើនវិមាត្រមួយ គេត្រូវកំនត់ Index បន្ថែមមួយទៀតដោយប្រើសញ្ញា []។ ឧទាហរណ៍ខាងក្រោមធ្វើការប្រកាស Array ពីរវិមាត្រឈ្មោះ towD។

```
int towD[][]=new int[4][5];
```

លក្ខណៈនេះកំនត់នូវ Array ទំហំ 4 គុណនិង 5 ហើយកំនត់ទៅអោយ towD។ ម៉ាទ្រីសនេះត្រូវបានប្រើជា Array នៃ Array ប្រភេទ int។

កម្មវិធីខាងក្រោមនេះកនត់តំលៃលេខអោយធាតុនិមួយៗ ក្នុង Array ពីឆ្វេងមកស្តាំ ពីលើមកក្រោម រួចហើយបង្ហាញតំលៃទាំងនេះ:

```
//Demonstrate a tow-dimensional array.
class TwoDArray{
      public static void main(String Args[]){
            int towD[][]=new int[4][5];
            int i,j,k=0;

            for(i=0;i<4;i++){
                  for(j=0;j<5;j++){
                        twoD[i][j]=k;
                        k++;
                  }
            }

            for(i=0;i<4;i++){
                  for(j=0;j<5;j++){
                        System.out.println(towD[i][j] + “ );
                        System.out.println();
                  }
            }
      }
}
```

កាលណាយើងកំនត់ memory សំរាប់ array ច្រើនវិមាត្រ យើងត្រូវបញ្ជាក់ memory សំរាប់វិមាត្រទីមួយតែប៉ុណ្ណោះ(នៅខាងឆ្វេងគេបង្អស់)។ យើងអាចកំនត់វិមាត្រដែលនៅសល់ ដោយឡែកពីគ្នាបាន។ ឧទាហរណ៍ នៅពេលយើងកំនត់វិមាត្រតាមធម្មតានោះ យើងមិនចាំបាច់កំនត់ចំនួនធាតុដូចគ្នា ចំពោះវិមាត្រនិមួយៗឡើយ។ កម្មវិធីខាងក្រោមនេះ បង្កើតនូវ array ពីរវិមាត្រ ដែលមានទំហំនៃវិមាត្រទីពីរមិនស្មើគ្នា:

កម្មវិធីខាងក្រោមនេះ នឹងផ្តល់នូវលទ្ធផលដូចខាងក្រោម:

0
1 2
3 4 5
6 7 8 9

```
//Manually allocate differing size second dimension.
class TowDAgain{
      public static void main(String Args[]){
            int towD[][]=new int[4][];

            twoD[0]=new int[1];
            twoD[1]=new int[2];
            twoD[2]=new int[3];
            twoD[3]=new int[4];

            int i,j,k=0;
            for(i=0;i<4;i++)
                  for(j=0;j<i+1;j++){
                        towD[i][j]=k;
                        k++;
                  }

            for(i=0;i<4;i++)
                  for(j=0;j<i+1;j++){
                        System.out.println(twoD[i][j] + " ");
                        System.out.println();
                  }

      }
}
```

គេក៏អាចកំនត់តំលៃដំបូងអោយ Array ច្រើនវិមាត្របានដែរ។ ដើម្បីធ្វើដូច្នេះ គេគ្រាន់តែប្រើសញ្ញា {} ដាក់អមវិមាត្រនិមួយៗនៃការកំនត់តំលៃ។ កម្មវិធីខាងក្រោមនេះ បង្កើតនូវម៉ាទ្រីសមួយដែលធាតុនិមួយៗ ផ្ទុកផលគុណនៃ Index តាមជួរដេក និង ជួរឈរ។

```
//initialize a two-dimensional array
class Matrix{
      public static void main(String Args[]){
            double m[][]={
                              {0*0,1*0,2*0,3*0},
                              {0*1,1*1,2*1,3*1},
                              {0*2,1*2,2*2,3*2},
                              {0*3,1*3,2*3,3*3}
                         };
            int i,j;

            for(i=0;i<4;i++){
                  for(j=0;j<4;j++){
                        System.out.println(m[i][j] + " ");
                        System.out.println();
                  }
            }
      }
}
```

កម្មវិធីខាងក្រោមនេះ បង្កើត array បីវិមាត្រដែលមានទំហំ 3x4x5។ បន្ទាប់មកវាបញ្ចូលតំលៃអោយធាតុនិមួយៗ នូវផលគុណនៃ Index របស់វា។ នៅទីបញ្ចប់ វាបង្ហាញតំលៃនៃផលគុណនេះ។

```
//demonstrate a tree-dimensional array.
class ThreeDMatrix{
      public static void main(String Args[]){
            int threeD[][][]=new int[3][4][5];
            int i,j,k;

            for(i=0;i<3;i++)
                  for(j=0;j<4;j++)
                        for(k=0;k<5;k++)
                              threeD[i][j][k]=i * j * k;

            for(i=0;i<3;i++){
                  for(j=0;j<4;j++){
                        for(k=0;k<5;k++){
                              System.out.println(threeD[i][j][k] + " ");
                              System.out.println();
                        }
                  }
                  System.out.println();
            }

      }
}
```

កម្មវិធីនេះអោយលទ្ធផលដូចខាងក្រោម:

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12

0 0 0 0 0
0 2 4 6 8
0 4 8 12 16
0 6 12 18 24

៩.៣ **ទំរង់ប្រកាស Array មួយបែបទៀត**(Alternative array declaration syntax)

វាមានទំរង់មួយទៀត ដែលត្រូវប្រើសំរាប់ប្រកាស Array មួយ:

`type[] var-name;`

ឧទាហរណ៍ ការប្រកាសអញ្ញាត្តិពីរនៅខាងក្រោមនេះ គឺសមមូលគ្នា:

`int a1[]=new int[3];` និង int[] a2=new int[3];