សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប

មហាវិទ្យាល័យវិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា

មុខវិជ្ជា ៖ Object Programming Language

មេរៀនទី២ ៖ **ប្រភេទទិន្នន័យ អថេរ និង** Arrays

សាស្ត្រាចារ្យ ៖ ឈុំ ខឿន

ទូរស័ព្ទលេខ៖ ០៩២ ៧៩ ៦៥ ៧៩ / ០១៦ ៥២៧ ៥៦៧

https://elearning.aeu.cloud

សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប

មហាវិទ្យាល័យ វិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា

មុខវិជ្ជា៖

Object Programming Language

មេរៀនទី២៖ ប្រភេទទិន្នន័យ អថេរ និង Arrays

គណៈកម្មការអភិវឌ្ឍន៍កម្មវិធីសិក្សា

© 2020

មាតិកា

1. ប្រភេទទិន្នន័យធម្មតា (Simple Data Types)
2. ការប្រើពាក្យ (Lexical Issues)
3. អថេរ (Variables)
4. ការបំប្លែង និងការបញ្ជាក់ប្រភេទទិន្នន័យ (Type Conversion and Casting)
5. Arrays

ក្រោយពីបានសិក្សាជំពូកនេះចប់ សិស្សានុសិស្សមានលទ្ធភាព៖

- ស្គាល់ និងចេះប្រើប្រាស់ប្រភេទទិន្នន័យធម្មតា
- ស្គាល់ ចេះប្រកាស និងចេះប្រើប្រាស់អថេរ
- ស្គាល់ ចេះប្រកាស និងចេះប្រើប្រាស់អថេរជា Arrays
- ចេះសរសេរកម្មវិធី Java ដែលមានការប្រកាសអថេរដោយប្រើប្រាស់ប្រភេទទិន្នន័យធម្មតា
- ចេះសរសេរកម្មវិធី Java ដែលមានការប្រកាសអថេរជា Array មួយវិមាត្រ និងច្រើនវិមាត្រ

- ចាវ៉ា គឺជាភាសាដែលមានការកំណត់ប្រភេទទិន្នន័យច្បាស់លាស់ ដែលជាផ្នែកមួយ ធ្វើឱ្យមានភាពប្រាកដប្រជា និងមានសុវត្ថិភាព
- រាល់ការប្រកាសអថេរ និង Expression ត្រូវតែមានប្រភេទទិន្នន័យ
- រាល់ប្រភេទទិន្នន័យ សុទ្ធតែត្រូវបានកំណត់យ៉ាងច្បាស់លាស់
- គ្រប់ការកំណត់តម្លៃ (ទោះនៅផ្នែកខាងក្រៅ Method ឬតាមរយៈប៉ារ៉ាម៉ែត្ររបស់ Method ក៏ដោយ) ពេលយកមកប្រើប្រាស់ សុទ្ធតែត្រូវបានត្រួតពិនិត្យដើម្បីឱ្យមានភាពត្រូវគ្នានៃប្រភេទទិន្នន័យ
- ប្រភេទទិន្នន័យណាមួយដែលមិនត្រូវគ្នា នឹងមាន Error កើតឡើង ហើយត្រូវតែធ្វើការកែសម្រួលឱ្យបានរួចរាល់សិន មុននឹង Compiler បញ្ចប់ការបំប្លែងដោយជោគជ័យ។

1. ប្រភេទទិន្នន័យធម្មតា (Simple Data Types)

- Java មានប្រភេទទិន្នន័យធម្មតា ចំនួន ៨ គឺ៖ byte, short, int, long, char, float, double និង boolean ។
- ប្រភេទទិន្នន័យទាំងនេះ ចែកជាបួនក្រុម គឺ៖
  - **ក្រុមលេខគត់**(Integer)៖ ក្រុមនេះរួមមាន byte, short, int និង long ដែលជាប្រភេទទិន្នន័យតំណាងឱ្យលេខចំនួនគត់ ។
  - **ក្រុមលេខទសភាគ**(Floating-point number)៖ ក្រុមនេះរួមមាន float និង double ដែលជាប្រភេទទិន្នន័យតំណាងឱ្យលេខទសភាគ។
  - **ក្រុមតួអក្សរ** (Character)៖ ក្រុមនេះមានតែ char មួយគត់ ដែលតំណាងឱ្យនិមិត្តសញ្ញានៅក្នុងក្រុមនៃអក្សរ(តួអក្សរ និងលេខ)។
  - **ក្រុមតក្ក** (Boolean)៖ ក្រុមនេះមានតែ boolean មួយគត់ ដែលជាប្រភេទទិន្នន័យពិសេសសម្រាប់តំណាងឱ្យតម្លៃពិត និងមិនពិត។

1.1. ប្រភេទទិន្នន័យចំនួនគត់

- Java កំណត់ប្រភេទទិន្នន័យចំនួនគត់ ៤ប្រភេទ៖ byte, short, int និង long ។
- ទំហំ និងតម្លៃដែនកំណត់នៃប្រភេទទិន្នន័យចំនួនគត់ ដូចខាងក្រោម៖

ឈ្មោះ	ទំហំ	តម្លៃដែនកំណត់
long	64 bits (8 bytes)	ចាប់ពី -9,223,372,036,854,775,808 ដល់ 9,223,372,036,854,775,807
int	32 bits (4 bytes)	ចាប់ពី -2,147,483,648 ដល់ 2,147,483,647
short	16 bits (2 bytes)	ចាប់ពី -32,768 ដល់ 32,767
byte	8 bits (1 byte)	ចាប់ពី -128 ដល់ 127

- ឧទាហរណ៍ខាងក្រោមនេះ ជាកម្មវិធីមួយដែលគណនាចំនួន Miles ដែលពន្លឺនឹងធ្វើដំណើរក្នុងរយៈពេលនៃចំនួនថ្ងៃដែលបានបញ្ជាក់ៈ

```
public class Light {
    public static void main(String[] args) {
        int lightspeed;
        long days, seconds, 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;
        System.out.print("In " + days);
        System.out.print(" days, light will travel about ");
        System.out.println(distance + " miles.");
    }
}
```

1.2. ប្រភេទទិន្នន័យចំនួនទសភាគ

- ប្រភេទទិន្នន័យចំនួនទសភាគ (ឬហៅថាចំនួនពិត) ដែលគេប្រើនៅពេលវាយតម្លៃ Expression ដែលត្រូវការភាពជាក់លាក់នៃតម្លៃលេខ
- ឧទាហរណ៍៖ ការគណនា ឫសការេ ស៊ីនុស ឬ កូស៊ីនុស ដែលលទ្ធផលត្រូវការតម្លៃទសភាគជាក់លាក់។ ប្រភេទទិន្នន័យទសភាគមាន២ប្រភេទ គឺ៖ float និង double ។
- ទំហំ និងតម្លៃដែនកំណត់នៃប្រភេទទិន្នន័យលេខទសភាគ ដូចជា៖

ឈ្មោះ	ទំហំគិតជា Bits	តម្លៃដែនកំណត់
double	64 bits	±1.79769313486231570E+308
float	32 bits	±3.40292347E+38

- ឧទាហរណ៍ខាងក្រោមនេះ ជាកម្មវិធីតូចមួយ ដែលប្រើអថេរដែលមានប្រភេទ double ដើម្បីគណនាក្រឡាផ្ទៃនៃរង្វង់មួយ៖

```
// compute the area of circle
public class Area {
    public static void main(String[] args) {
        double pi, r, a;
        r = 10.8;  // radius of circle
        pi = 3.1416;  // pi, approximately
        a = pi * r * r; // compute area
        System.out.println("Area of circle is : " + a);
    }
}
```

1.3. ប្រភេទទិន្នន័យតួអក្សរ

- នៅក្នុងភាសា Java ប្រភេទទិន្នន័យដែលប្រើសម្រាប់ផ្ទុកតួអក្សរ គឺ char ។ Java មានប្រើ Unicode សម្រាប់តំណាងឱ្យតួអក្សរនីមួយៗ។
- Unicode កំណត់ឱ្យតួអក្សរអន្តរជាតិពេញលេញ ដែលតំណាងឱ្យគ្រប់តួអក្សរដែលមាននៅក្នុភាសាមនុស្ស។ លក្ខណៈនេះ វាត្រូវការទំហំចំនួន 16Bits ។
- ហេតុនេះ នៅក្នុងភាសា Java, char គឺជាប្រភេទទិន្នន័យដែលមានទំហំ 16Bits និងមានតម្លៃដែលកំណត់ពី 0 រហូតដល់ 65,536 ។ វាពុំមានតម្លៃអវិជ្ជមានទេ។

- ឧទាហរណ៍ខាងក្រោមនេះ ជាកម្មវិធីដែលប្រើអថេរប្រភេទ char ៖

```
// Demonstrate char data type
public class CharDemo {
    public static void main(String[] args) {
        char ch1, ch2;
        ch1 = 88;
        ch2 = 'Y';
        System.out.print("ch1 and ch2 : ");
        System.out.println(ch1 + " " + ch2);
    }
}
```

ចូរកត់សម្គាល់ថា ch1 ត្រូវបានកំណត់តម្លៃ 88 ដែលជាតម្លៃ Unicode ហើយត្រូវគ្នានឹងអក្សរ X ។ ទោះបីជា char មិនមែនជាប្រភេទទិន្នន័យលេខគត់ក៏ដោយ ក៏គេអាចធ្វើប្រមាណវិធីជាមួយនឹងវាដូចចំនួនគត់ដែរ។ លក្ខណៈនេះ អាចឱ្យយើងបូកពីរតួអក្សរបញ្ចូលគ្នាបាន ឬអាចបង្កើនតម្លៃនៃអថេរតួអក្សរបាន។

- សូមពិនិត្យមើលឧទាហរណ៍ខាងក្រោម៖

```
// char variables behave like integers
public class CharDemo2 {
    public static void main(String[] args) {
        char ch;
        ch = 'X';
        System.out.println("ch contains " + ch);
        ch++;   // increment ch
        System.out.println("ch is now " + ch);
    }
}
```

នៅក្នុងកម្មវិធីខាងលើ ch ឱ្យតម្លៃដំបូង គឺ X បន្ទាប់មក ch បង្កើនមួយឯកតា។ ការបង្កើនមួយឯកតា នាំឱ្យ ch មានតម្លៃជា Y ដែលជាតួអក្សរបន្ទាប់ពី X នៅក្នុងលំដាប់ Unicode។

1.4. ប្រភេទទិន្នន័យតក្ក

- Java មានប្រភេទទិន្នន័យមួយគឺ boolean ដែលសម្រាប់តំណាងឱ្យតម្លៃតក្ក ។
- ប្រភេទទិន្នន័យ boolean នេះ គឺតំណាងឱ្យតម្លៃពិត (true) និងតម្លៃមិនពិត (false) ។ វាជាប្រភេទទិន្នន័យដែលបានមកពីការផ្តល់តម្លៃ តាមរយៈសញ្ញាលេខនព្វន្តធៀប ដូចជា a < b ។
- boolean ក៏ត្រូវការប្រើក្នុងកន្សោមលក្ខខណ្ឌនៃឃ្លា if និង for ផងដែរ។
- ឧទាហរណ៍ខាងក្រោម បង្ហាញពីការប្រើប្រភេទទិន្នន័យ boolean ៖

```
public class BoolTest {
    public static void main(String[] args) {
        boolean b = false;
        System.out.println("b is " + b);
        b = true;
        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
        System.out.println("10 > 9 is " + (10 > 9));
    }
}
```

កម្មវិធីខាងលើនេះ មានបីបញ្ហាដែលគួរឱ្យចាប់អារម្មណ៍៖

1. កាលណាតម្លៃ boolean មួយត្រូវបានបញ្ចេញលទ្ធផលតាមរយៈ println( ) ពេលនោះតម្លៃ true ឬ false ត្រូវបានបង្ហាញ។
2. តម្លៃរបស់អថេរ boolean មួយ គឺគ្រប់គ្រាន់សម្រាប់វាត្រួតពិនិត្យនូវឃ្លា if ។ ហេតុនេះ មិនបាច់ចាំសរសេរ if(b==true) ដូច្នេះទេ។
3. លទ្ធផលនៃសញ្ញាប្រៀបធៀប ដូចជា < គឺជាតម្លៃ boolean។

2. ការប្រើពាក្យ (Lexical Issues)

2.1. ការដកឃ្លា (Whitespace)

- ភាសា Java គឺជាភាសាដែលមានទម្រង់សេរី។ មានន័យថា គេមិនចាំបាច់ធ្វើតាមគោលការណ៍នៃការដកឃ្លា ឬការចុះដើមបន្ទាត់ពិសេសណាមួយទេ។
- ក្នុងកម្មវិធី យើងអាចសរសេរនៅលើបន្ទាត់មួយ ឬតាមរបៀបផ្សេងៗទៀតដែលយើងមានអារម្មណ៍ថាចង់វាយវា។
- នៅក្នុង Java ការដកឃ្លាសំដៅលើ Space, Tab ឬ Newline។

2.2. ការកំណត់អត្តសញ្ញាណ (Identifier)

- Identifiers គឺជាពាក្យដែលប្រើសម្រាប់ដាក់ឈ្មោះ Classes ឈ្មោះ Methods ឬឈ្មោះអថេរដាដើម។

- Identifier មួយ អាចជាលំដាប់នៃអក្សរតូច និងអក្សរធំ ជាលេខ ឬជាតួអក្សរ, សញ្ញា Underscore (_) ឬសញ្ញាដុល្លារ ($) ។ វាមិនត្រូវចាប់ផ្តើមដោយលេខទេ ព្រោះវាអាចច្រឡំនឹងតម្លៃលេខ។
- ឧទាហរណ៍៖
  - Identifiers ត្រឹមត្រូវ៖

    AvgTemp count a4 $test this_is_ok

  - Identifiers មិនត្រឹមត្រូវ៖

    2count high-temp Not/ok

2.3. ក្បួនសំណេរឱ្យបានត្រឹមត្រូវ (Literals)

- តម្លៃថេរមួយនៅក្នុងភាសា Java ត្រូវបានបង្កើតឡើងដោយប្រើប្រាស់ពាក្យមួយតាមក្បួនសំណេរសម្រាប់តំណាងឱ្យតម្លៃនោះ ។
- ឧទាហរណ៍ខាងក្រោម តាងពាក្យសំណេរមួយចំនួនឱ្យបានត្រឹមត្រូវ៖

100 98.6 'X' "This is a test"

ពីឆ្វេងទៅស្តាំ ពាក្យសំណេរទី១គឺជាចំនួនគត់ ទី២គឺជាចំនួនទសភាគ ទី៣គឺជាតួអក្សរ និងទី៤គឺជាសំណុំអក្សរ (String) ។

- ក្បួនសំណេរមួយឱ្យបានត្រឹមត្រូវ អាចប្រើនៅកន្លែងណាក៏បាន នូវតម្លៃនៃប្រភេទទិន្នន័យដែលវាអាចប្រើបាន។

2.3.1. ក្បួនសំណេរតម្លៃលេខគត់ឱ្យបានត្រឹមត្រូវ

- តម្លៃលេខគត់ ជាប្រភេទទិន្នន័យដែលប្រើញឹកញាប់នៅក្នុងកម្មវិធី ។ ឧទាហរណ៍: 1, 2, 3 និង 42 ។ តម្លៃលេខទាំងនេះ ជាតម្លៃលេខប្រព័ន្ធគោល១០ (decimal values)។
- នៅក្នុង Java ក៏មានតម្លៃលេខពីរប្រភេទផ្សេងទៀតដែរ ដែលស្ថិតក្នុងលក្ខណៈតម្លៃលេខគត់នោះដែរ គឺតម្លៃលេខប្រព័ន្ធគោល៨ (octal values) និងតម្លៃលេខប្រព័ន្ធគោល១៦ (hexadecimal values)។
- នៅក្នុង Java គេអាចសម្គាល់តម្លៃលេខប្រព័ន្ធគោល៨ ដោយប្រើលេខ 0 ដាក់ពីមុខ។
- តម្លៃលេខក្នុងប្រព័ន្ធគោល១០ មិនអាចដាក់លេខ 0 នៅពីមុខទេ។

- ហេតុនេះ តម្លៃ 09 ហាក់ដូចជាត្រឹមត្រូវ តែផ្ទុយទៅវិញ វាធ្វើឱ្យមាន Error កើតឡើងពី Compiler ដោយលេខ 9 ស្ថិតនៅខាងក្រៅព្រំដែនតម្លៃរបស់ប្រព័ន្ធគោល៨ (ប្រព័ន្ធគោល៨ មានតម្លៃចន្លោះពី 0 ទៅ 7) ។
- រីឯតម្លៃលេខប្រព័ន្ធគោល១៦ គេអាចសំគាល់បាន ដោយដាក់ 0x រឺ 0X នៅពីមុខ (ប្រព័ន្ធគោល១៦ មានតម្លៃចន្លោះពី 0 ទៅ 15 ដោយគេប្រើ A ដល់ F ឬ a ដល់ f ជំនួសឱ្យពីលេខ 10 ដល់ 15) ។
- ចំពោះតម្លៃលេខគត់ប្រភេទប្រភេទ long បើចង់ឱ្យអថេរមួយផ្ទុកតម្លៃធំ នោះគេត្រូវបញ្ចាក់នៅខាងក្រោយលេខនូវអក្សរ L (l ឬ L)។

ឧទាហរណ៍៖ 0x7fffffffffffffffL ឬ 9223372036854775807L

- ស្វ័យគុណ តាងដោយអក្សរ E ឬ e ដោយមានលេខប្រព័ន្ធគោលដប់នៅដាប់ពីក្រោយ (ជាលេខវិជ្ជមាន ឬ អវិជ្ជមាន)។

  ឧទាហរណ៍៖ 6.022E23, 314159E-05 និង 2e+100 ជាដើម។

- ជាធម្មតា នៅក្នុងភាសា Java តម្លៃលេខទសភាគដែលបញ្ជាក់ឱ្យប្រភេទទិន្នន័យ double គេអាចដាក់ ឬមិនបាច់ដាក់អក្សរ D ឬ d នៅពីក្រោយលេខ។ ផ្ទុយទៅវិញ គេត្រូវតែដាក់អក្សរ F ឬ f នៅពីក្រោយតម្លៃលេខទសភាគ ដើម្បីបញ្ជាក់នូវប្រភេទទិន្នន័យ float ។

  ឧទាហរណ៍៖

  double d = 23.25; ឬ double d = 23.25D;

  float f = 43.23f;

2.3.2. ការដាក់សំណេរតម្លៃលេខទសភាគឱ្យបានត្រឹមត្រូវ

- តម្លៃទសភាគ គឺជាតម្លៃលេខប្រព័ន្ធគោលដប់ ដែលមានតម្លៃនៅខាងក្រោយក្បៀស។
- គេអាចសរសេរតម្លៃលេខទសភាគ តាមលក្ខណៈបែបធម្មតា (standard notation) ឬតាមលក្ខណៈវិទ្យាសាស្ត្រ (scientific notation) ។
- តម្លៃលេខទសភាគ តាមលក្ខណៈធម្មតា គឺបង្កើឡើងដោយលេខប្រព័ន្ធគោល១០ ជាមួយសញ្ញាចំនុច (.) ហើយមានតម្លៃទសភាគនៅជាប់ខាងក្រោយ ដូចជា 2.0, 3.14159 និង 0.6667 ។
- ចំណែកឯតម្លៃលេខទសភាគ តាមលក្ខណៈបែបវិទ្យាសាស្ត្រវិញ ប្រើដូចលក្ខណៈបែបធម្មតាដែរ ហើយបន្ថែមស្វ័យគុណនៃ១០ ជាមួយនឹងលេខដែលត្រូវគុណ ។

2.3.3. ក្បួនសំណេរតម្លៃតក្កឱ្យបានត្រឹមត្រូវ

- តម្លៃតក្ក មានតែពីរប៉ុណ្ណោះ គឺ true និង false។
- គេមិនអាចបំលែងតម្លៃ true និង false ឱ្យទៅជាតម្លៃលេខបានទេ ហើយគេក៏មិនអាចយកតម្លៃលេខ 1 ជំនួសឱ្យ true និងតម្លៃលេខ 0 ជំនួសឱ្យ false បានទៀយ។

2.3.4. ក្បួនសំណេរតម្លៃតួអក្សរឱ្យបានត្រឹមត្រូវ

- នៅក្នុង Java តួអក្សរត្រូវបានរៀបតាមលេខរៀងអក្សរទៅជាតួអក្សរ Unicode ។ វាមានទំហំ 16 bits ដែលអាចបំលែងទៅជាចំនួនគត់ និង ប្រើជាមួយសញ្ញាប្រមាណវិធី ដូចជា បូក ឬដក ជាដើមបាន។ ក្បួនសំណេរតម្លៃតួអក្សរ គឺត្រូវសរសេរដាក់ក្នុងចន្លោះសញ្ញា ‘ ’ ។

  ឧទាហរណ៍៖ ‘a’, ‘Z’ ឬ ‘@’ ជាដើម។

- ចំពោះតួអក្សរណាដែលមិនអាចសរសេរតែខ្លួនវាផ្ទាល់នោះ មានដូចជា Escape sequences មួយចំនួន តម្រូវឱ្យយើងបញ្ចូលនូវសញ្ញា '\' ដាក់ជាមួយ ដូចជា '\n' សម្រាប់តួអក្សរចុះដើមបន្ទាត់។
- ក្រៅពីនេះ គេក៏អាចប្រើសញ្ញា \ ជាមួយតម្លៃលេខប្រព័ន្ធគោល៨ និងប្រព័ន្ធគោល១៦ បានដែរ ពេលនោះវានឹងផ្តល់តម្លៃជាតួអក្សរដែលត្រូវគ្នា។
- ចំពោះតម្លៃលេខប្រព័ន្ធគោល៨ គេអាចដាក់លេខចំនួន៣ខ្ទង់ នៅខាងក្រោយសញ្ញា \ ឧទាហរណ៍ '\141' ដែលត្រូវនឹងតួអក្សរ 'a' ។
- ដោយឡែកចំពោះតម្លៃលេខប្រព័ន្ធគោល១៦ គេត្រូវប្រើតួអក្សរ '\u' ហើយដាក់លេខចំនួន៤ខ្ទង់ នៅពីក្រោយ ឧទាហរណ៍ '\u0061' ដែលត្រូវនឹងតួអក្សរ 'a' ។

- តារាងខាងក្រោមនេះ បង្ហាញពីទម្រង់អក្សរដែលប្រើជាមួយសញ្ញា \ ៖

Escape Sequence	Description
\ddd	Octal character (ddd)
\uxxxx	Hexadecimal UNICODE Character (xxxx)
\′	Single quote
\′′	Double quote
\\	Backslash
\n	New line (also known as line feed)
\t	Tab
\b	Backspace

2.3.5. ក្បួនសំណេរតម្លៃ String ឱ្យបានត្រឹមត្រូវ

- ក្បួនសំណេរតម្លៃ String នៅក្នុងភាសា Java បានកំណត់ដូចភាសាដទៃទៀតដែរ គឺសំណុំអក្សរត្រូវសរសេរដាក់នៅក្នុងចន្លោះ " " ។ ឧទាហរណ៍៖

  “Hello, World”

  “two\nlines”

  “\“This is in quotes\””

- គួរចងចាំថា String មួយរបស់ Java ត្រូវតែចាប់ផ្តើម និងបញ្ចប់ទៅវិញនៅលើបន្ទាត់តែមួយ ព្រោះវាពុំមាននិមិត្តសញ្ញា ឬអក្សរណាមួយសម្រាប់ភ្ជាប់ String នៅលើបន្ទាត់ពីរផ្សេងគ្នា ដូចភាសាផ្សេងទៀយ។

2.4. ការអធិប្បាយ (Comments)

- ការអធិប្បាយដែលបានប្រើនៅក្នុងភាសា Java មានបីប្រភេទ។
- យើងបានជួបប្រទះការអធិប្បាយ នៅក្នុងកម្មវិធីកន្លងមក ចំនួនពីរួចមកហើយ គឺការអធិប្បាយលើបន្ទាត់តែមួយដែលប្រើនូវសញ្ញា //..... និងការអធិប្បាយមានច្រើនបន្ទាត់ដែលសម្គាល់បានដោយប្រើសញ្ញា /*.....*/ ។
- រីឯការអធិប្បាយប្រភេទទី៣ ហៅថាការអធិប្បាយកំណត់ត្រាឯកសារ (documentation comment) ។ ការអធិប្បាយប្រភេទនេះ ប្រើសម្រាប់បង្កើតឯកសារ HTML មួយដែលកត់ត្រាឯកសារកម្មវិធី ។ គេអាចសម្គាល់ការអធិប្បាយបែបនេះបាន ដែលវាចាប់ផ្តើមដោយ /** ហើយបញ្ចប់ដោយ */ ។

2.5. ពាក្យគន្លឹះនៅក្នុងភាសា Java (Keywords in Java)

- នៅក្នុងភាសា Java មានពាក្យគន្លឹះជាច្រើន។ ពាក្យគន្លឹះទាំងនោះមិនអនុញ្ញាតឱ្យប្រើឡើងវិញក្នុងការដាក់ឈ្មោះអថេរ ឈ្មោះ Class ឬឈ្មោះ Method បានឡើយ។ ពាក្យគន្លឹះទាំងនោះ មានដូចជា៖

  abstract continue finally interface public this

  class boolean default float long return throw
  break do for final short throws byte double
  if new static protected case else implements

  null while true catch extends import package

  super try char false int private switch void

- ពាក្យគន្លឹះដែលបានរក្សាទុកប៉ុន្តែមិនទាន់ប្រើក្នុងភាសា Java ។ Java បានរក្សាទុកនូវពាក្យគន្លឹះមួយចំនួនសម្រាប់ប្រើនៅពេលអនាគត គឺ៖

  byvalue cast const future generic goto

  inner operator outer rest var

2.6. **សញ្ញាខណ្ឌបំបែក** (Separators)

- នៅក្នុងភាសា Java មាននិមិត្តសញ្ញាមួយចំនួន ដែលប្រើជាសញ្ញាខណ្ឌបំបែក ។ សញ្ញាខណ្ឌបំបែក ដែលត្រូវបានប្រើញឹកញាប់ជាងគេនៅក្នុងភាសា Java នោះ គឺសញ្ញា ; ។ សញ្ញា ; ប្រើសម្រាប់បញ្ចប់ឃ្លានៅក្នុងកូដ។ ខាងក្រោមនេះ បង្ហាញនូវសញ្ញាខណ្ឌបំបែក និងអត្ថន័យរបស់វា៖

និមិត្តសញ្ញា	ឈ្មោះ	គោលបំណង
( )	Parentheses	ប្រើសម្រាប់ផ្ទុកប៉ារ៉ាម៉ែត្ររបស់ Method។ វាអាចប្រើដើម្បីកំណត់អាទិភាពនៅក្នុងកន្សោមឃ្លា។ ប្រើក្នុងឃ្លាលក្ខខណ្ឌ និងការដាក់អមបញ្ជាក់នូវប្រភេទទិន្នន័យ (cast) ដើម្បីបំលែងប្រភេទទិន្នន័យ។

និមិត្តសញ្ញា	ឈ្មោះ	គោលបំណង
{}	Braces	ប្រើសម្រាប់ផ្ទុកតម្លៃ Array ។ ហើយប្រើសម្រាប់កំណត់ Block នៃកូដឱ្យ Class និង Method ជាដើម។
[]	Brackets	ប្រើនៅពេលប្រកាស Array និងការបញ្ជាក់ឱ្យតម្លៃនៃធាតុរបស់ Array។
;	Semicolon	ប្រើសម្រាប់បញ្ចប់ឃ្លានៅក្នុងកូដ។
,	Comma	ប្រើសម្រាប់ខណ្ឌបំបែក Identifiers នៅពេលប្រកាសអថេរនៅបន្តបន្ទាប់គ្នា។
.	Period	សម្រាប់ខណ្ឌឈ្មោះ Package ពី Sub-package ជាមួយ Class។ គេក៏ប្រើសម្រាប់ខណ្ឌបំបែករវាងអថេរ និង Method ពី Object ផងដែរ។

3. អថេរ (Variables)

3.1. ការប្រកាសអថេរ (Declaring variables)

- នៅក្នុងភាសា Java គ្រប់អថេរទាំងអស់ត្រូវតែប្រកាសជាមុន មុននឹងយកវាទៅប្រើប្រាស់។
- ទម្រង់នៃការប្រកាសអថេរមានដូចខាងក្រោម៖

```
type identifier1 [= value], identifier1 [= value], …;
```

ក្នុងទម្រង់នៃការប្រកាសនេះ type គឺជាប្រភេទទិន្នន័យធម្មតាណាមួយរបស់ Java ឬជាឈ្មោះរបស់ Class ឬ Interface ណាមួយដែលបានបង្កើត។

- យើងកំណត់តម្លៃឱ្យអថេរ ដោយប្រើសញ្ញាស្មើ និងតម្លៃណាមួយ។
- ត្រូវចាំថា កន្សោមកំណត់តម្លៃ ត្រូវផ្តល់ជាតម្លៃនៃប្រភេទទិន្នន័យដូចគ្នានឹងប្រភេទទិន្នន័យពេលប្រកាសអថេរនោះ។
- សញ្ញា Comma (,) ដើម្បីការប្រកាសអថេរច្រើន ដែលមានប្រភេទទិន្នន័យដូចគ្នា យើងប្រើបំបែកពីអថេរមួយ ទៅអថេរមួយទៀត។
- ឧទាហរណ៍៖

```
int a, b, c;          // declares 3 ints a, b, and c.
int d = 3, e, f = 5; // declares 3 ints, initializing d and f.
byte z = 22;      // initializes z.
double pi = 3.14159; // declares & initialize pi as double.
char x  = 'x';     // the variable x has the value 'x'.
```

3.2. ការកំណត់តម្លៃមានភាពប្រែប្រួល (Dynamic Initialization)

- Java អនុញ្ញាតឱ្យយើងកំណត់តម្លៃឱ្យអថេរមួយ ដែលមានលក្ខណៈប្រែប្រួល ដោយប្រើកន្សោមលេខត្រឹមត្រូវណាមួយ នៅពេលប្រកាស។
- ឧទាហរណ៍៖

```
int x = 11, y = 22;
int z = x + y;          // z = 33
```

```
1 public class DynInit {
2     public static void main(String[] args) {
3         double a = 3.00, b = 4.0;
4         // c is dynamically initialized
5         double c = Math.sqrt(a*a + b*b);
6         System.out.println("Hypotenuse : " + c);
7     }
8 }
```

3.3. ដំហំនៃតាំង និងការរស់នៅនៃអថេរ (The Scope and Lifetime of Variables)

- Java អាចឱ្យប្រកាសអញ្ញាតនៅក្នុង Block ណាមួយ។ Block ចាប់ផ្តើមដោយសញ្ញា { និងបញ្ចប់វិញដោយសញ្ញា } ។
- Block កំណត់ទំហំមួយ ហៅថា Scope ។
- ភាសាសំណេរកម្មវិធីភាគច្រើន បានកំណត់ទំហំ Scope ពីរប្រភេទ គឺសាកល (Global) និងផ្ទៃក្នុង (Local)។
- នៅក្នុងភាសា Java ទំហំ Scope សំខាន់ៗមានពីរ គឺកំណត់ដោយ Class និងកំណត់ដោយ Method។ Scope មួយអាចស្ថិតនៅក្នុង Scope មួយទៀត។

- សូមសង្កេតមើលឧទាហរណ៍ខាងក្រោម៖

```
// Demonstrate block scope
public class Scope {
    public static void main(String[] args) {
        int x = 10; // known to all code within main
        if(x==10) { // start new scope
            int y = 20; // known only to 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 know here
        System.out.println("Now, x is : " + x);
    }
}
```

អថេរត្រូវបានប្រកាសនៅពេល Scope របស់វាចាប់ផ្តើម ហើយត្រូវបំបាត់ទៅវិញនៅពេល Scope របស់វាបញ្ចប់។ ពេលវេលាដែលអថេរមួយប្រើបាន ត្រូវបានកំណត់ទៅនឹងព្រំដែននៃ Scope របស់វា។

- ប្រសិនបើការប្រកាសអថេរមួយមានការកំណត់តម្លៃដំបូង អថេរនោះអាចនឹងត្រូវបានកំណត់តម្លៃជាថ្មីរៀងរាល់ Block ដែលបានប្រកាសវា។ សូមសង្កេតមើលឧទាហរណ៍ខាងក្រោម៖

```
// Demonstrate lifetime of a variable
public 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 always prints -1
            y = 100;
            System.out.println("y is now : " + y);
        }
    }
}
```

4. ការបំប្លែង និងការបញ្ជាក់ប្រភេទទិន្នន័យ (Type Conversion and Casting)

- ការកំណត់តម្លៃអថេរនៃប្រភេទទិន្នន័យមួយ ទៅឱ្យអថេរដែលមានប្រភេទទិន្នន័យផ្សេង បើសិនជាប្រភេទទិន្នន័យទាំងពីរត្រូវគ្នា នោះ Java នឹងធ្វើការបំប្លែងឱ្យដោយស្វ័យប្រវត្តិ។

- ប៉ុន្តែ មិនមែនគ្រប់ប្រភេទទិន្នន័យទាំងអស់ សុទ្ធតែត្រូវគ្នានោះទេ ហេតុនេះរាល់ប្រភេទទិន្នន័យទាំងអស់ ក៏មិនអនុញ្ញាតឱ្យធ្វើការបំប្លែងដោយស្វ័យប្រវត្តិដែរ។ ចំពោះករណីនោះ គេក៏នៅតែអាចធ្វើការបំប្លែងប្រភេទទិន្នន័យដែលមិនត្រូវគ្នាបានដែរ។

- ដើម្បីធ្វើការបំប្លែងប្រភេទទិន្នន័យដែលមិនត្រូវគ្នាបែបនោះ គេត្រូវតែបញ្ជាក់ឱ្យច្បាស់លាស់រវាងប្រភេទទិន្នន័យដែលមិនត្រូវគ្នានោះ ។

4.1. ការបំប្លែងប្រភេទទិន្នន័យដោយស្វ័យប្រវត្តិ (Java's Automatic Conversion)

- កាលណាទិន្នន័យមួយប្រភេទ ត្រូវបានកំណត់ឱ្យអថេរនៃប្រភេទផ្សេង នោះការបំប្លែងប្រភេទទិន្នន័យដោយស្វ័យប្រវត្តិនឹងកើតឡើង ប្រសិនបើត្រូវនឹងលក្ខខណ្ឌដូចខាងក្រោម៖

  1. ប្រភេទទិន្នន័យទាំងពីរត្រូវគ្នា

  2. ប្រភេទទិន្នន័យគោលដៅ(អង្គខាងឆ្វេង) ធំជាងប្រភេទទិន្នន័យដើម(នៅអង្គខាងស្តាំ)

- ទោះជាយ៉ាងណាក៏ដោយ ប្រភេទទិន្នន័យតម្លៃលេខទសភាគ មិនត្រូវគ្នានឹង char ឬ boolean ទេ។ ហើយ char និង boolean ក៏មិនត្រូវគ្នាដែរ។

4.2. ការបំប្លែង និងបញ្ជាក់ប្រភេទទិន្នន័យដែលមិនត្រូវគ្នា (Casting Incompatible Types)

- ដើម្បីបំប្លែងប្រភេទទិន្នន័យពីរមិនត្រូវគ្នា នោះយើងត្រូវប្រើលក្ខណៈបញ្ជាក់បន្ថែម (cast) ដែលមានទម្រង់ទូទៅ៖

  (target-type) value;

  ក្នុងទម្រង់ខាងលើនេះ target-type ជាឈ្មោះប្រភេទទិន្នន័យដែលយើងត្រូវការ ដើម្បីបំប្លែង value ឱ្យទៅជាប្រភេទទិន្នន័យនោះ ។

- នៅពេលបំប្លែងប្រភេទទិន្នន័យលេខទសភាគ ដែលត្រូវបានកំណត់ឱ្យទៅជាប្រភេទទិន្នន័យលេខគត់ គឺមានការកាត់ចោលផ្នែកទសភាគចោល ដោយសារប្រភេទទិន្នន័យលេខគត់ មិនមានផ្នែកទសភាគទេ។

- ឧទាហរណ៍៖ បើតម្លៃ 1.23 ត្រូវកំណត់ទៅជាប្រភេទលេខគត់ នោះតម្លៃលទ្ធផលទទួលបាន គឺ 1 ចំណែកឯ 0.23 ត្រូវបានកាត់ចោល ។

- កម្មវិធីខាងក្រោមនេះ បង្ហាញពីការបំលែងប្រភេទទិន្នន័យមួយចំនួនដែលត្រូវបញ្ជាក់បន្ថែម៖

```
// Demonstrate cast
public class Conversion {
    public static void main(String[] args) {
        byte b;
        int i = 257;
        double d = 323.142;
        System.out.println("Conversion of int to byte.");
        b = (byte) i;
        System.out.println("i and b : " + i + " " + b);
        System.out.println("\nConversion of double to int.");
        i = (int) d;
        System.out.println("d and i : " + d + " " + i);
        System.out.println("\nConversion of double to byte.");
        b = (byte) d;
        System.out.println("d and b : " + d + " " + b);
    }
}
```

- តាមរយៈកម្មវិធីខាងលើ យើងសង្កេតឃើញថាមានការបំប្លែងប្រភេទទិន្នន័យពីមួយទៅមួយផ្សេងទៀត។
- នៅពេលតម្លៃ 257 ដែលមានប្រភេទទិន្នន័យ int ត្រូវបានបំប្លែងឱ្យជាប្រភេទទិន្នន័យ byte នោះលទ្ធផលគឺជាសំណល់នៃ 257 ចែកនឹង 256(តម្លៃដែនកំណត់នៃ byte) គឺស្មើនឹង 1 ។
- នៅពេល d បំលែងទៅជា int នោះផ្នែកទសភាគត្រូវបាត់បង់។
- នៅពេល d បំលែងទៅជា byte នោះផ្នែកទសភាគត្រូវបាត់បង់ ហើយតម្លៃត្រូវបែងចែក ដែលជាសំណល់នៃផលចែកជាមួយ 256 ហើយលទ្ធផល គឺ 67 ។

4.3. គោលការណ៍បង្កើនប្រភេទទិន្នន័យ (The Type Promotion Rules)

- ការលើកប្រភេទទិន្នន័យ byte និង short ទៅជា int នោះ Java មានគោលការណ៍បង្កើនប្រភេទទិន្នន័យ ដែលត្រូវប្រើប្រាស់។

```
public class Promote {
    public static void main(String[] args) {
        byte b = 42;
        char ch = 'a';
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = .1234;
        double resutl = (f*b) + (i/ch) - (d*s);
        System.out.println((f*b) + " + " + (i/ch) +
                           " - " + (d*s));
        System.out.println("Result = " + resutl);
    }
}
```

5. អឺរេ (Arrays)

- Array គឺជាសំណុំអថេរដែលមានប្រភេទទិន្នន័យដូចគ្នា ក្រោមឈ្មោះមួយ តែខុសគ្នាដោយលេខ index ។
- Array ត្រូវបានបង្កើតឡើង ដោយប្រើប្រភេទទិន្នន័យណាមួយ និងអាចមានទំហំមួយវិមាត្រ ឬច្រើនវិមាត្រ។

5.1. Arrays មួយវិមាត្រ (One-Dimensional Arrays)

- Array មួយវិមាត្រជាសំណុំអថេរដែលមានប្រភេទទិន្នន័យដូចគ្នា។
- ដើម្បីបង្កើត Array មួយ ដំបូងយើងត្រូវប្រកាសអថេរនៃ Array នោះដោយមានប្រភេទទិន្នន័យណាមួយ។
- ទម្រង់ទូទៅនៃការប្រកាស Array មួយវិមាត្រ គឺ៖

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

- ឧទាហរណ៍ខាងក្រោម ជាការប្រកាស Array មួយឈ្មោះ month-days ដែលបានបញ្ជាក់ប្រភេទទិន្នន័យ int ៖

```
int month_days [];
```

ទោះបីការប្រកាសនេះបានធ្វើឡើងដោយមាន month-days ជាអថេររបស់ Array ក៏ដោយ តែតាមពិតគ្មាន Array កើតឡើងនៅឡើយទេ។ តម្លៃនៃ month-days ត្រូវបានកំណត់ជា null ដែលតាងឱ្យ Array មួយគ្មានតម្លៃ។

- ដើម្បីភ្ជាប់ month-days ជាមួយនឹងរូបរាង Array ពិតប្រាកដនៃប្រភេទតម្លៃលេខគត់ យើងត្រូវបង្កើតទីតាំងមួយ ដោយប្រើ new និងកំណត់តម្លៃទៅឱ្យ month-days ។ new គឺជាសញ្ញាណលេខនព្វន្តពិសេសមួយ សម្រាប់បង្កើតទីតាំងក្នុង memory ។

- Array គឺជាសំណុំអថេរដែលមានប្រភេទទិន្នន័យដូចគ្នា ក្រោមឈ្មោះមួយ តែខុសគ្នាដោយលេខ index ។
- Array ត្រូវបានបង្កើតឡើង ដោយប្រើប្រភេទទិន្នន័យណាមួយ និងអាចមានទំហំមួយវិមាត្រ ឬច្រើនវិមាត្រ។
- ទម្រង់ទូទៅរបស់ new ដែលប្រើជាមួយ Array មួយវិមាត្រ គឺ៖

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

ដែលក្នុងនេះ type បញ្ជាក់ពីប្រភេទទិន្នន័យដែលត្រូវបង្កើតទីតាំង។ size កំណត់ចំនួនធាតុ Array និង array-var ជាអថេររបស់ Array ។

- កាលណាយើងប្រើ new សម្រាប់បង្កើតទីតាំងឱ្យ Array មួយ យើងត្រូវតែកំណត់ប្រភេទទិន្នន័យ និងចំនួនធាតុរបស់វា។
- ធាតុទាំងអស់ក្នុង Array ដែលបង្កើតទីតាំង ដោយប្រើ new នោះនឹងត្រូវកំណត់តម្លៃដំបូងស្មើនឹងសូន្យដោយស្វ័យប្រវត្តិ។

- ឧទាហរណ៍ខាងក្រោម បានបង្កើតទីតាំងសម្រាប់ Array មួយ ដែលមាន ១២ធាតុ ប្រភេទទិន្នន័យចំនួនគត់។

month_days = new int [12] ;

```
public 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.");
    }
}
```

- យើងអាចកំណត់តម្លៃឱ្យ Array នៅពេលយើងប្រកាសវា។ ការកំណត់បែបនោះ គឺជាការរាយតម្លៃរបស់ធាតុនីមួយៗ ដែលផ្តាច់ពីគ្នាដោយ Comma និងអមដោយសញ្ញា { } ។ Array នឹងកើតឡើងដោយមានទំហំល្មមគ្រប់គ្រាន់សម្រាប់ដាក់ចំនួនធាតុ ដែលយើងបានកំណត់ឱ្យ Array ដោយស្វ័យប្រវត្តិ។ វាមិនត្រូវការប្រើ new ទេ។
- កូដនៅខាងក្រោមនេះ បង្កើត Array ដែលមានកំណត់តម្លៃដំបូងជាចំនួនគត់៖

```
public 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.");
    }
}
```

- ប្រព័ន្ធដំណើរការរបស់ Java នឹងពិនិត្យមើល ថាតើគ្រប់ Index ទាំងអស់របស់ Array នៅក្នុងជួរបានត្រឹមត្រូវដែរឬទេ។
- ឧទាហរណ៍៖ ប្រព័ន្ធដំណើរការពិនិត្យរបស់ Index នីមួយៗពី moth-days ដើម្បីឱ្យដឹងច្បាស់ថា វាស្ថិតនៅក្នុងតម្លៃចន្លោះពី 0 ដល់ 11 ដែរឬទេ។
- បើសិនជាយើងព្យាយាមប្រើធាតុដែលមានតម្លៃលេខ Index ក្រៅតម្លៃលេខ Index របស់ Array (ចំនួនអវិជ្ជមាន ឬចំនួនដែលមានលេខ Index ធំជាងក្នុងប្រវែង Array) នោះវានឹងបណ្តាលឱ្យមាន Error កើតឡើង។

- ខាងក្រោមនេះ ជាឧទាហរណ៍មួយទៀត ដែលប្រើ Array មួយវិមាត្រ ដើម្បីរកមធ្យមភាគនៃចំនួនដែលបានកំណត់ៈ

```
public 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);
    }
}
```

5.2. អារេច្រើនវិមាត្រ (Multi-Dimensional Arrays)

- Array ច្រើនវិមាត្រ គឺជា Array នៃ Array ។
- ដើម្បីប្រកាសអថេរ Array ច្រើនវិមាត្រ គេត្រូវកំណត់ Index បន្ថែមមួយទៀត ដោយប្រើសញ្ញា [] ។
- ឧទាហរណ៍ខាងក្រោម ជាការប្រកាសអថេរ Array ពីរវិមាត្រ ដែលមានឈ្មោះ towD ៖

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

លក្ខណៈនេះ កំណត់នូវ Array ទំហំ 4 គុណនឹង 5 ហើយកំណត់ទៅឱ្យ twoD ។ ម៉ាទ្រីសនេះ ត្រូវបានប្រើជា Array នៃ Array ប្រភេទ int។

- កម្មវិធីខាងក្រោមនេះ កំណត់តម្លៃលេខទៅឱ្យធាតុនីមួយៗក្នុង Array ពីឆ្វេងទៅស្តាំ ពីលើចុះក្រោម រួចហើយបង្ហាញតម្លៃទាំងនោះ៖

```
// Demonstrate a two-dimensional array
public class TwoDArray {
    public static void main(String[] args) {
        int twoD[][] = 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.print(twoD[i][j] + "\t");
                System.out.println();
        }
    }
}
```

- គេក៏អាចធ្វើការកំណត់តម្លៃដំបូងទៅឱ្យ Array ច្រើនវិមាត្របានដែរ។ ដើម្បីធ្វើការកំណត់តម្លៃដំបូង គេគ្រាន់តែប្រើសញ្ញា {} ដាក់អមវិមាត្រនីមួយៗនៃការកំណត់តម្លៃ។
- កម្មវិធីខាងក្រោម បង្កើតនូវម៉ាទ្រីសមួយ ដែលធាតុនីមួយៗផ្គុំផលគុណនៃ Index តាមជួរដេក និងជួរឈរ៖

```
// Initialize a two-dimensional array
public 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.print(m[i][j] + " ");
                System.out.println();
        }
    }
}
```

- កម្មវិធីខាងក្រោម បង្កើត Array បីវិមាត្រ ដែលមានទំហំ 3 x 4 x 5 ។ បន្ទាប់មក វាបញ្ចូលតម្លៃឱ្យធាតុនីមួយៗ នូវផលគុណនៃ Index របស់វា។ នៅទីបញ្ចប់វាបង្ហាញតម្លៃផលគុណនោះ។

```
//Demonstrate a three-dimensional array
public 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.print(threeD[i][j][k] + " ");
                                System.out.println();
                        }
                        System.out.println();
                    }
    }
}
```

5.3. ទម្រង់ប្រកាស Array ផ្សេងៗទៀត (Alternative Arrays Declaration Syntax)

- យើងអាចប្រើទម្រង់ផ្សេងៗទៀត សម្រាប់ការប្រកាស Array៖

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

- ឧទាហរណ៍ការប្រកាសអថេរទាំងពីរនៅខាងក្រោមនេះ គឺសមមូលគ្នា៖

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

- ការប្រកាសអថេរទាំងពីរខាងក្រោមនេះ ក៏សមមូលគ្នាដែរ៖

```
char twoD1[][] = new char[3][4];
char [][] twoD2 = new char[3][4];
```

សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប
ASIA EURO UNIVERSITY

អរគុណ
សម្រាប់ការយកចិត្តទុកដាក់

https://elearning.aeu.cloud