



# **សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប**

## **មហាវិទ្យាល័យវិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា**

**មុខវិជ្ជា**            ៖ Object Programming Language

**មេរៀនទី៤**       ៖ **ប្លង់រៀបចំការងារកម្រិត** ( Control Statements )

**សាស្ត្រាចារ្យ** ៖ **ឈុំ ខឿន**

<https://elearning.aeu.cloud>

# សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប

មហាវិទ្យាល័យ វិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា



មុខវិជ្ជា៖

Object Programming Language

មេរៀនទី៤៖ បង្កើនប្លាត់កូដ  
( Control Statements )

គណៈកម្មការអភិវឌ្ឍន៍កម្មវិធីសិក្សា

© 2020

## មាតិកា

1. ប្លង់ជ្រើសរើស ( Selection Statement )
2. ប្លង់ធ្វើលំដាប់ ( Iteration Statement )
3. ប្លង់លោតរំលង ( Jump Statement )

ក្រោយពីបានសិក្សាជំពូកនេះ សិស្សានុសិស្សមានសមត្ថភាព៖

- ស្គាល់ និងចេះប្រើប្រាស់ឃ្លាជ្រើសរើស ( Selection Statement )
- ស្គាល់ និងចេះប្រើប្រាស់ឃ្លារង្វិលជុំ ( Iteration Statement )
- ស្គាល់ និងចេះប្រើប្រាស់ឃ្លាលោតរំលង ( Jump Statement )
- ចេះសរសេរកម្មវិធីចាត់ ដោយប្រើប្រាស់ឃ្លាជ្រើសរើស ឃ្លារង្វិលជុំ និងឃ្លាលោតរំលង
- ចេះវិភាគអំពីដំណើរការនៃការអនុវត្តរបស់ឃ្លាជ្រើសរើស ឃ្លារង្វិលជុំ និងឃ្លាលោតរំលង

- ឃ្លាផ្ទៀងផ្ទាត់លក្ខខណ្ឌ ត្រូវបានចែកចេញជាបីក្រុម គឺ៖
  1. ឃ្លាជ្រើសរើស ( Selection )
  2. ឃ្លារង្វិលជុំ ( Iteration )
  3. ឃ្លាលោតរំលង ( Jump )
- ឃ្លាជ្រើសរើស ( Selection ) អនុញ្ញាតឱ្យកម្មវិធីជ្រើសរើសផ្លូវសម្រាប់អនុវត្ត ដោយផ្អែកលើលទ្ធផលលក្ខខណ្ឌរបស់កន្សោម ( Expression ) ឬលក្ខណៈនៃអថេរណាមួយ។
- ឃ្លារង្វិលជុំ ( Iteration ) អនុញ្ញាតឱ្យកម្មវិធីអនុវត្តឃ្លា ( Statements ) មួយ ឬច្រើនសារចុះសារឡើង ម្តងហើយម្តងទៀតបាន ( មានន័យថា ឃ្លារង្វិលជុំបង្កើតទម្រង់ Loops ) ។
- ឃ្លាលោតរំលង ( Jump ) អនុញ្ញាតឱ្យកម្មវិធីអនុវត្តក្នុងទម្រង់មួយមិនទៅតាមលំដាប់លំដោយនោះទេ។

# 1. ប្រាជ្ញាសាស្ត្រ ( Selection Statements )

- ភាសាកម្មវិធី Java បានផ្តល់នូវប្រភេទប្រាជ្ញាសាស្ត្រ ( Selection Statements ) ពីរ គឺ if និង switch ។
- ប្រាជ្ញាសាស្ត្រនេះអនុញ្ញាតឱ្យមានការត្រួតពិនិត្យ សម្រាប់ដំណើរការអនុវត្តន៍ នៃកម្មវិធីរបស់យើង ដោយផ្អែកលើលក្ខខណ្ឌដែលបានកំណត់។

## 1.1. ប្រាជ្ញា if (if statement)

- ប្រាជ្ញា if ក្នុងកម្មវិធីចារឹក ជាប្រាជ្ញាដែលមានពីរផ្នែក និងមានកំណត់ លក្ខខណ្ឌ។
- គេប្រើវា ដើម្បីធ្វើឱ្យកម្មវិធីប្រាជ្ញាសាស្ត្រផ្លូវមួយក្នុងចំណោមផ្លូវពីរផ្សេង គ្នាសម្រាប់ការអនុវត្ត។

- ឃ្លាខាងក្រោម គឺទម្រង់របស់ឃ្លា `if` ៖

```

if ( condition ) statement1;    // when condition is true,
                                // statement1 is executed.
else statement2;                // when condition is false,
                                // statement2 is executed.

```

ក្នុងនេះ `statement1` ឬ `statement2` ជាឃ្លាមួយ ឬច្រើនដែលសរសេរក្នុង `{}` ( ដែលហៅថា Block ) ។ `condition` គឺជាកន្សោមមួយដែលផ្តល់តម្លៃ `boolean` ។ ឃ្លា `else` ជាឃ្លាដែលជួនកាលប្រើជួនកាលមិនប្រើ ( Optional ) ។

- សូមពិនិត្យមើលឧទាហរណ៍ខាងក្រោម៖

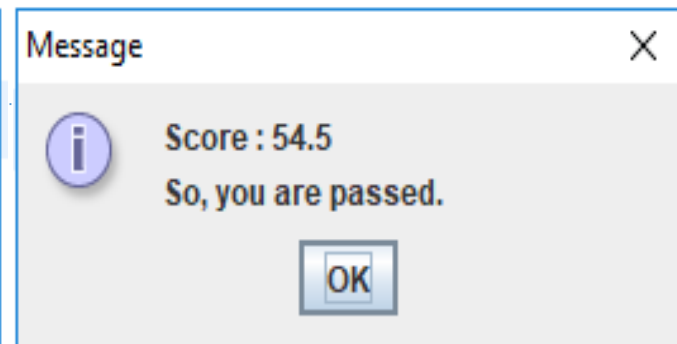
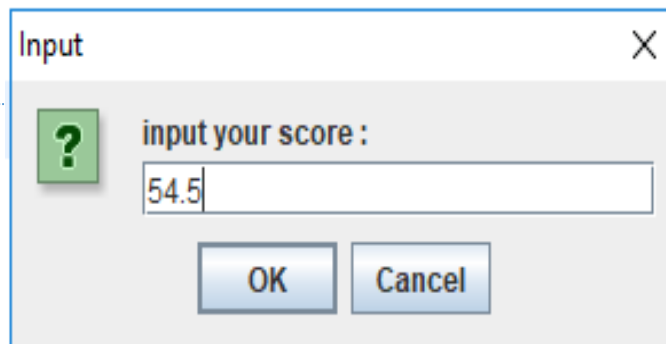
```

int a, b;
// ...
if( a < b )  a = 0;
else  b = 0;

```

- សូមពិនិត្យមើលឧទាហរណ៍ខាងក្រោម៖

```
import javax.swing.JOptionPane;
public class IETest {
    public static void main(String[] args) {
        float score;
        String Sscore = JOptionPane.showInputDialog(
            "input your score : ");
        score = Float.parseFloat(Sscore);
        if(score >= 50) {
            JOptionPane.showMessageDialog(null, "Score : " + score +
                "\nSo, you are passed.");
        }
        else {
            JOptionPane.showMessageDialog(null, "Score : " + score +
                "\nSo, you are failed.");
        }
    }
}
```





### 1.1.1. ឃ្លា if មួយ ស្ថិតក្នុងឃ្លា if មួយទៀត (Nested if)

- Nested if គឺជាឃ្លា if មួយ ដែលស្ថិតនៅក្នុងឃ្លា if ឬ else ណាមួយផ្សេងទៀត។
- កាលណាយើងប្រើ Nested if យើងត្រូវចាំថាឃ្លា else ដែលជារបស់ if គឺស្ថិតនៅក្នុង Block ជាមួយគ្នា។

ឧទាហរណ៍៖

```
if ( i == 10 ) {  
    if ( j < 20 ) a = b;  
    if ( k > 100 ) c = d;    // This if is  
    else a = c;             // associated with this else  
}  
else a = d;                 // This else refers to if ( i == 10 )
```

## 1.1.2. លំដាប់ថ្នាក់នៃ if-else-if (if-else-if ladder)

- ការប្រើ Nested ifs បន្តបន្ទាប់គ្នា ហៅថា លំដាប់ថ្នាក់នៃ if-else-if ។  
ទម្រង់របស់វា គឺ៖

```
if ( condition )  
    statement;  
else if ( condition )  
    statement;  
else if ( condition )  
    statement;  
.  
.  
else  
    statement;
```

- ខាងក្រោមនេះ គឺជាកម្មវិធីមួយដែលប្រើលំដាប់ថ្នាក់នៃ if-else-if ដើម្បីកំណត់ខែនៅក្នុងរដូវប្រចាំឆ្នាំ៖

```
public class IfEsle {
    public static void main(String[] args) {
        int month = 4; // April
        String season;
        if(month==12 || month==1 || month==2)
            season = "Winter";
        else if(month==3 || month==4 || month==5)
            season = "Spring";
        else if(month==6 || month==7 || month==8)
            season = "Summer";
        else if(month==9 || month==10 || month==11)
            season = "Spring";
        else
            season = "Bogus Month";
        System.out.println("April is in the " + season + ".");
    }
}
```

## 1.2. ឃ្លា switch (switch statement)

- ឃ្លា switch នៅក្នុង Java គឺជាឃ្លាដែលមានច្រើនផ្នែក អនុវត្តបានច្រើនផ្លូវ។
- switch ផ្តល់នូវវិធីងាយស្រួល ដែលធ្វើឱ្យការអនុវត្តរបស់កម្មវិធីជ្រើសរើសយកផ្នែកផ្សេងៗនៃកូដ ដោយផ្អែកលើតម្លៃនៃកន្សោម ( Expression ) ។ ដូច្នេះ វាផ្តល់នូវលក្ខណៈមួយ ដែលប្រសើរជាងឃ្លា if-else-if ។

ខាងក្រោមនេះ គឺជាទម្រង់របស់ឃ្លា switch ៖

```
switch ( expression ) {  
    case value 1:  
        // statement sequence  
        break;
```

```
case value 2:  
    // statement sequence  
    break;
```

·  
·  
·

```
case valueN:  
    // statement sequence  
    break;
```

```
default:  
    // default statement sequence
```

```
}
```

ក្នុងនេះ *expression* ត្រូវតែមានប្រភេទទិន្នន័យជា byte, short, int ឬ char ។ តម្លៃនីមួយៗ ដែលប្រើនៅក្នុងឃ្លា case ត្រូវតែមានប្រភេទទិន្នន័យត្រូវគ្នាជាមួយនឹង *expression* ។

- តម្លៃនៃ case នីមួយៗ ត្រូវតែជាតម្លៃដែលមានតែមួយ (Unique literal) ។ មានន័យថា តម្លៃនៃ case ត្រូវតែជាចំនួនថេរ (constant) មិនមែនជាអថេរ (variable) ទេ។ តម្លៃ case មិនអនុញ្ញាតឱ្យមានលេខដូចគ្នាឡើយ។
- ឃ្លា break ត្រូវបានប្រើនៅក្នុង switch ដើម្បីបញ្ចប់លំដាប់ឃ្លា (statement sequence) ។ នេះគឺ ជាឥទ្ធិពលនៃដំណើរការលោតចេញក្រៅរបស់ switch ។

កម្មវិធីខាងក្រោម បង្ហាញពីការប្រើឃ្លា switch ៖

```
public class SampleSwitch {  
    public static void main(String[] args) {  
        for(int i = 0; i < 6; i++) {  
            switch(i) {  
                case 0: System.out.println("i is zero"); break;  
            }  
        }  
    }  
}
```

```
case 1: System.out.println("i is one"); break;
case 2: System.out.println("i is two"); break;
case 3: System.out.println("i is three"); break;
default:
    System.out.println("i is greater than 3");
}
```

- ឃ្លា break ជូនការប្រើ ជូនកាលមិនប្រើ ( optional ) ។
- ប្រសិនបើយើងលុប break ចោល នោះការអនុវត្តនឹងបន្តចូលទៅក្នុង case បន្ទាប់។
- ជូនកាលមាន case ច្រើន ដោយគ្មានឃ្លា break នៅចន្លោះ case ទាំងនោះ។

- សូមពិនិត្យមើលកម្មវិធីខាងក្រោមនេះ៖

```

public class Switch {
    public static void main(String[] args) {
        String month = "Apr";
        String season;
        switch(month) {
            case "Dec":
            case "Jan":
            case "Feb": season = "Winter"; break;
            case "Mar":
            case "Apr":
            case "May": season = "Spring"; break;
            case "Jun":
            case "July":
            case "Aug": season = "Summer"; break;
            case "Sep":
            case "Oct":
            case "Nov": season = "Autumn"; break;
            default: season = "Bogus month.";
        }

        System.out.println("April is in the " + season);
    }
}

```



- switch មួយអាចប្រើនៅក្នុង switch មួយទៀត ដូចខាងក្រោមនេះ៖

```
public class NestedSwitch {  
    public static void main(String[] args) {  
        int count = 1;  
        int target = 0;  
        switch(count) {  
            case 1:  
                switch(target) { // nested switch  
                    case 0:  
                        System.out.println("target is zero");  
                        break;  
                    case 1: // not conflicts with outer switch  
                        System.out.println("target is one");  
                        break;  
                }  
                break;  
            case 2: // ...  
        }  
    }  
}
```

## 2. ប្ប្រែងលំដាប់ ( Iteration Statements )

- ប្ប្រែងលំដាប់ ( Iteration ) របស់ Java មាន៖
  1. while
  2. do-while
  3. for
- ប្ប្រែងទាំងនេះបង្កើតនូវទម្រង់មួយ ដែលហៅថា Loop ។

### 2.1. The while Loop

- while loop គឺជាប្រភេទប្ប្រែងលំដាប់ ( Looping statement ) ដ៏សំខាន់មួយ។
- វាអនុវត្តចំពោះប្ប្រែង ឬ block ណាមួយ សារចុះសារឡើង នៅពេលដែល expression ដែលត្រូវត្រួតពិនិត្យមានតម្លៃពិត។

- ទម្រង់ទូទៅរបស់វា គឺ៖

```
while( condition ) {  
    // body of loop  
}
```

ក្នុងនេះ condition ជាកន្សោមប្រភេទ boolean ណាមួយ។  
body of loop នឹងអនុវត្ត ដ៏រាបណាកន្សោមលក្ខខណ្ឌ (conditional expression) មានតម្លៃពិត។ កាលណា condition មានតម្លៃមិនពិត វានឹងអនុវត្តចំពោះកូដ ដែលស្ថិតនៅបន្ទាប់ពី Loop ។

- កម្មវិធីខាងក្រោម ប្រើ while loop ដែលតម្លៃថយចុះ ពី 10 មកក្រោម ។  
លទ្ធផលនៃកម្មវិធីនេះផ្តល់ពាក្យ tick ចំនួន 10 បន្ទាត់។

```
public class While {  
    public static void main(String[] args) {  
        int n = 10;
```

```
System.out.println("The result: ");
while(n > 0) {
    System.out.println("tick " + n);
    n--;
}
}
```

```
The result:
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

- ដោយ while loop ត្រូវពិនិត្យកន្សោមលក្ខខណ្ឌ នៅផ្នែកខាងលើ របស់ loop ជាហេតុធ្វើឱ្យតួរបស់ loop មិនបានអនុវត្តឡើយ សូម្បីតែ ម្តងក៏ដោយ ប្រសិនបើនៅពេលចាប់ផ្តើមលក្ខខណ្ឌមានតម្លៃមិនពិត។

- ឧទាហរណ៍ខាងក្រោម ការប្រើ `println( )` មិនត្រូវបានអនុវត្តទេ៖

```
int a = 10, b = 20;
while ( a > b )
```

```
System.out.println( "This will not be displayed" );
```

- `body` របស់ `while` (ឬ `loop` ណាមួយផ្សេងទៀតរបស់ Java) អាចនៅទទេ។ នេះគឺដោយសារតែឃ្លាទទេ (ឃ្លាដែលមានតែ `semicolon`) អាចប្រើបាននៅក្នុងកម្មវិធី Java ។ សូមពិនិត្យមើលឧទាហរណ៍៖

```
public class NoBody {
    public static void main(String[] args) {
        int i, j;
        i = 100;
        j = 200;
        // Find midpoint between i and j
        while(++i < --j); // No body in this loop
        System.out.println("Midpoint is " + i);
    }
}
```

## 2.2. The **do-while** Loop

- do-while loop នឹងអនុវត្ត body របស់វា យ៉ាងហោចណាស់ក៏ម្តងដែរ ដោយសារកន្សោមលក្ខខណ្ឌ របស់វាស្ថិតនៅខាងក្រោមនៃតួ loop ។

- ទម្រង់ទូទៅរបស់វា គឺ៖

```
do {
    // Body of loop
} while ( condition );
```

ជាដំបូង do-while loop វាអនុវត្ត Body of loop ជាមុន បន្ទាប់មក វាពិនិត្យមើលកន្សោមលក្ខខណ្ឌជាក្រោយ។ ប្រសិន កន្សោមនេះ មានតម្លៃពិត នោះ loop នឹងអនុវត្តសារប៉ុះសារ ឡើង។ បើពុំដូច្នោះទេ loop នឹងត្រូវបានបញ្ចប់។ គ្រប់ loop, condition ជាកន្សោម boolean ។

- ខាងក្រោមនេះ គឺជាកម្មវិធីមួយដែលកែប្រែទៅលើកម្មវិធី “tick” ដែលនឹងត្រូវប្រើ do-while loop ។ វាក៏ផ្តល់លទ្ធផលដូចកម្មវិធីដែលប្រើ while ដែរ។

```
public class Dowhile {  
    public static void main(String[] args) {  
        int n = 10;  
        System.out.println("The result is" +  
                             "\n-----");  
        do {  
            System.out.println("tick " + n);  
            n--;  
        } while(n > 0);  
    }  
}
```

## 2.3. The **for** loop

- for loop គឺជាឃ្លា ដែលអាចប្រើបានច្រើនយ៉ាង និងមានលក្ខណៈល្អប្រសើរ។ ខាងក្រោមនេះគឺជា ទម្រង់ទូទៅរបស់ឃ្លា for ៖

```
for ( initialization; condition; iteration ) {  
    // body of loop  
}
```

- *initialization* គឺជាកន្លែងដែលអថេរត្រូវកំណត់តម្លៃដំបូង។
- *condition* ជាកន្លែងប្រៀបធៀបអថេរទៅនឹងតម្លៃណាមួយ។
- *iteration* ជាកន្លែងអថេរត្រូវបង្កើន ឬបន្ថយតម្លៃម្តងមួយឯកតា។
- *initialization* ត្រូវបានអនុវត្តតែម្តងគត់ នៅពេល for loop ត្រូវបានចាប់ផ្តើម បន្ទាប់មក *condition* ត្រូវបានពិនិត្យ ប្រសិនវាមានតម្លៃពិត នោះគួរបស់ loop ត្រូវបានអនុវត្ត។ បន្ទាប់មកទៀត អថេរត្រូវបានបង្កើន ឬបន្ថយមួយឯកតានៅក្នុងផ្នែក *iteration* ។



- ដំណើររបស់ for loop គឺ៖ condition => body of loop => iteration ចេះតែបន្តរហូតដល់ condition មានតម្លៃមិនពិត ពេលនោះ loop ក៏ត្រូវបានបញ្ចប់។
- នេះគឺជាកម្មវិធីមួយ ដែលសរសេរឡើងវិញនៃកម្មវិធី "tick" ដោយប្រើ for loop ៖

```
public class ForTick {
    public static void main(String[] args) {
        int n;
        System.out.println("The result is" +
                           "\n-----");
        for(n = 10; n > 0; n--) {
            System.out.println("tick " + n);
        }
    }
}
```

## 2.4. ការប្រកាសអថេរនៅក្នុង for loop

- អថេរដែលបានប្រកាសនៅក្នុង for loop មួយ ត្រូវបានប្រើសម្រាប់ តែ loop នោះតែប៉ុណ្ណោះ មិនត្រូវបានប្រើនៅកន្លែងផ្សេងទៀត។ ករណីនេះ យើងអាចប្រកាសអថេរក្នុងផ្នែក initialization នៃ for បាន។
- សូមពិនិត្យមើលកម្មវិធីមួយ ដែលសរសេរឡើងវិញ ដោយអថេរ n ត្រូវបានប្រកាសជា int នៅក្នុង for ៖

```
public class ForTickAgain {
    public static void main(String[] args) {
        // Here, n is declared inside for loop
        System.out.println("The result is" +
                           "\n-----");
        for(int n = 10; n > 0; n--) {
            System.out.println("tick " + n);
        }
    }
}
```

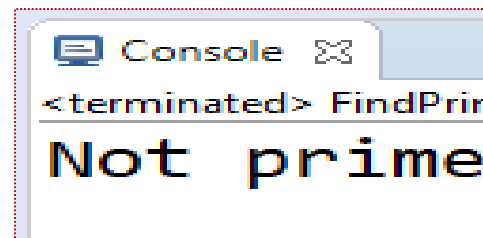
- នៅពេលដែលអ្នកប្រកាសអថេរនៅក្នុង for loop អ្នកត្រូវតែចាំនូវ ចំណុចសំខាន់មួយ គឺ scope ត្រូវបិទ កាលណាឃ្លា for ត្រូវបញ្ចប់។ មានន័យថា scope របស់អថេរកំណត់ព្រំដែនដោយ for loop ។
- កាលណាអថេរផ្ទៀងផ្ទាត់ loop មិនត្រូវយកទៅប្រើនៅកន្លែងផ្សេង ទៀតទេ ពេលនោះគេត្រូវប្រកាសនៅខាងក្នុង for ។
- ខាងក្រោមនេះ គឺកម្មវិធីមួយដែលគណនានូវចំនួនបឋម។ ចូរកត់ សម្គាល់ថា អថេរ i របស់ for ត្រូវបានប្រកាសនៅក្នុង for ពីព្រោះ វាមិនចាំបាច់សម្រាប់ប្រើប្រាស់នៅកន្លែងផ្សេងឡើយ។

```

public class FindPrime {
    public static void main(String[] args) {
        int num;
        boolean isPrime = true;
        num = 14;
        for(int i = 2; i < num/2; i++) {
            if(num%i==0) {
                isPrime = false;
                break;
            }
        }
        if(isPrime) System.out.println("Prime");
        else System.out.println("Not prime");
    }
}

```

លទ្ធផលនៃកម្មវិធីខាងលើ គឺ៖



## 2.5. ការប្រើសញ្ញា Comma (Using the Comma)

- Java អនុញ្ញាតឱ្យមានឃ្លាច្រើននៅក្នុងផ្នែក initialization និង iteration របស់ for ដើម្បីអាចឱ្យអថេរមួយ ឬច្រើន ត្រួតពិនិត្យ for loop ។ ឃ្លានីមួយៗញែកផ្ដាច់ពីគ្នាដោយសញ្ញា Comma (,) ។
- កម្មវិធីខាងក្រោម បង្ហាញពីការប្រើសញ្ញា Comma នៅក្នុង for loop ៖

```
public class Comma {
    public static void main(String[] args) {
        int a, b;
        for(a = 1, b = 4; a < b; a++, b--) {
            System.out.println("a = " + a);
            System.out.println("b = " + b);
        }
    }
}
```

## 2.6. Loop មួយ នៅក្នុង Loop មួយទៀត (Nested Loops)

- Java អនុញ្ញាតឱ្យប្រើ loop មួយស្ថិតនៅក្នុង loop មួយទៀតបាន ដូចជា ភាសាសរសេរកម្មវិធីដទៃទៀតដែរ ។
- សូមពិនិត្យមើលកម្មវិធីមួយ ស្តីពីការប្រើ for loop មួយ ស្ថិតនៅក្នុង for loop មួយទៀត៖

```
public class NestedLoop {
    public static void main(String[] args) {
        int i, j;
        for(i = 0; i < 10; i++) {
            for(j = 0; j < 10; j++) {
                System.out.print("$ ");
            }
            System.out.println();
        }
    }
}
```

## ៣. ឃ្លាលោតរំលង ( Jump Statements )

- Java មានឃ្លាលោតរំលង ចំនួន៣ គឺ break, continue និង return ។

### 3.1. ការប្រើ break

- ឃ្លា break អាចប្រើបាន៣យ៉ាង៖
  1. ទី១៖ ប្រើសម្រាប់បញ្ចប់លំដាប់ឃ្លានៅក្នុង switch
  2. ទី២៖ ប្រើសម្រាប់ចាកចេញពី loop
  3. ទី៣៖ ប្រើដូចជាទម្រង់របស់ goto ។
- ដោយសារចំណុចទី១ យើងបានឃើញពីប្រើប្រាស់វារួចហើយ ដូច្នេះពេលនេះយើងលើកយកតែចំណុចទី២ និងទី៣ មកសិក្សា។

### 3.1.1. ការប្រើ break ដើម្បីចាកចេញពី Loop

- break យើងប្រើសម្រាប់ធ្វើឱ្យ loop មួយបញ្ចប់ភ្លាមៗបាន ដោយមិនអនុវត្តកន្សោមលក្ខខណ្ឌ និង code ដែលនៅសេសសល់នៅក្នុង body របស់ loop ទៀតទេ
- កាលណា break ត្រូវបានប្រើក្នុង loop វាធ្វើឱ្យ loop ត្រូវបានបញ្ចប់ ហើយបន្តអនុវត្តទៅកាន់ឃ្លាបន្ទាប់ពី loop ។ សូមពិនិត្យឧទាហរណ៍៖

```
public class BreakLoop {
    public static void main(String[] args) {
        for(int i = 0; i < 100; i++) {
            if(i==10) break; // terminate at i = 10
            System.out.println("i : " + i);
        }
        System.out.println("Loop complete.");
    }
}
```



- break អាចប្រើជាមួយគ្រប់ប្រភេទនៃ loop របស់ Java
- ខាងក្រោមនេះ គឺជាកម្មវិធីមួយ ដែលយកកម្មវិធីខាងលើ ប្រើជាមួយ while loop វិញម្តង ដែលលទ្ធផលទទួលបាន គឺដូចនឹងកម្មវិធីខាងលើដែរ៖

```
public class BreakWhile {  
    public static void main(String[] args) {  
        int i = 0;  
        while(i < 100) {  
            if (i == 10)  
                break; // terminate at i = 10  
            System.out.println("i : " + i);  
            i++;  
        }  
        System.out.println("Loop complete.");  
    }  
}
```

- កាលណាប្រើនៅក្នុង Nested loop, ឃ្លា break នឹងបញ្ចប់តែ loop ដែលស្ថិតនៅខាងក្នុងប៉ុណ្ណោះ។
- ឧទាហរណ៍៖

```
public class BreakNestedLoop {  
    public static void main(String[] args) {  
        for(int i = 0; i < 3; i++) {  
            System.out.println("Pass " + i + ": ");  
            for(int j = 0; j < 100; j++) {  
                if(j == 10) break; // terminate at j = 10  
                System.out.println(j + " ");  
            }  
            System.out.println();  
        }  
        System.out.println("Loops complete");  
    }  
}
```

### 3.1.2. ការប្រើ break ដូចជាទម្រង់របស់ goto

- ឃ្លា break ក៏អាចប្រើ ដើម្បីបង្កើតទម្រង់ឃ្លា goto បានដែរ ដែលត្រូវបានហៅថាឃ្លា Labeled break ។
- ទម្រង់ទូទៅរបស់វា គឺ៖

`break label;`

ក្នុងនេះ `label` គឺជាឈ្មោះរបស់ Label ដែលសម្រាប់សំគាល់ Block របស់ code ។

- . នៅពេលទម្រង់របស់ break ខាងលើ ត្រូវបានអនុវត្ត នោះវាសំដៅទៅរកឈ្មោះ block នៃ code នោះ
- . ដើម្បីដាក់ឈ្មោះឱ្យ block មួយ យើងដាក់ label នៅខាងដើម block ដោយមានសញ្ញា Colon (:) នៅពីក្រោយវា។

- កម្មវិធីខាងក្រោម បង្ហាញពី Nested blocks ចំនួន៣ជាន់ ដែល block នីមួយៗមាន label ផ្ទាល់របស់វា៖

```
public class BreakLabel {
    public static void main(String[] args) {
        boolean t = true;
        first: {
            second: {
                third: {
                    System.out.println("Before the break");
                    if(t) break second; // break the second.
                    System.out.println("This won't execute.");
                }
                System.out.println("This won't execut.");
            }
            System.out.println("This is after second block.");
        }
    }
}
```

- ជាទូទៅ label break ត្រូវបានប្រើដើម្បីចាកចេញពី Nested block ។
- នៅក្នុងកម្មវិធីខាងក្រោមនេះ loop ខាងក្រៅ (outer loop) អនុវត្តបានតែម្តង៖

```
public class BreakNestLoop {
    public static void main(String[] args) {
        outer: for(int i = 0; i < 3; i++) {
            System.out.println("Pass " + i + ": ");
            for(int j = 0; j < 100; j++) {
                if(j == 10) break outer; // exit both loop
                System.out.println("\t" + j);
            }
            System.out.println("This will not print.");
        }
        System.out.println("Loops complete.");
    }
}
```

- នៅពេល loop ខាងក្នុង(inner loop) បញ្ចប់(break) ដំណើរការ loop ខាងក្រៅ(outer loop) ពេលនោះ loops ទាំងពីរត្រូវបានបញ្ចប់។
- ប៉ុន្តែ យើងមិនអាច break label ណាមួយដែលស្ថិតនៅក្នុង block ផ្សេងគ្នាបានឡើយ។
- ក៏ម្ចាស់ក៏ខាងក្រោមនេះ loop ខាងក្រៅ អនុវត្តបានតែម្តងគត់៖

```
public class BreakError {
    public static void main(String[] args) {
        one: for(int i = 0; i < 3; i++) {
            System.out.println("Pass " + i + ": ");
        }
        for(int j = 0; j < 100; j++) {
            //if(j==10) break one; // Wrong
            System.out.println(j + " ");
        }
    }
}
```

## 3.2. ការប្រើ continue

- continue បង្ខំឱ្យ loop ធ្វើសកម្មភាពដដែលៗ ប៉ុន្តែបញ្ឈប់ការអនុវត្ត code ដែលនៅសេសសល់ក្នុង body របស់វា
- ការអនុវត្តបែបនេះ ជាលក្ខណៈរបស់ continue ដែលរំលងចោលឃ្លាខ្លះៗក្នុងតួនៃ loop និងឆ្ពោះទៅកាន់ផ្នែកខាងចុងនៃ loop នោះ
- កម្មវិធីខាងក្រោម ប្រើប្រាស់ continue ៖

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

- ស្រដៀងនឹងឃ្លា break ដែរ ឃ្លា continue អាចប្រើឈ្មោះ label ដើម្បីអនុញ្ញាតឱ្យ Loop ដែលយើងប្រើនោះធ្វើសកម្មភាពបន្ត
- កម្មវិធីខាងក្រោម ប្រើ continue ដើម្បីផ្តល់លទ្ធផលនូវតម្លៃជារាងតារាងត្រីកោណមួយ៖

```
public class ContinueLabel {
    public static void main(String[] args) {
        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*j));
            }
            System.out.println();
        }
    }
}
```

```
0
0 1
0 2 4
0 3 6 9
0 4 8 12 16
0 5 10 15 20 25
0 6 12 18 24 30 36
0 7 14 21 28 35 42 49
0 8 16 24 32 40 48 56 64
0 9 18 27 36 45 54 63 72 81
```



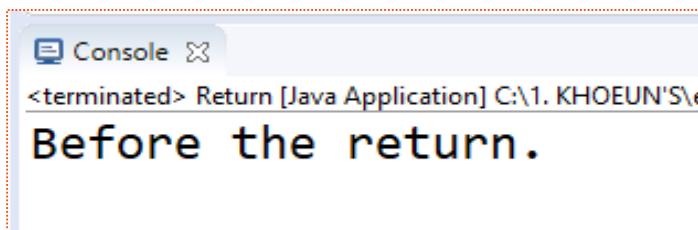
### 3.3. ការប្រើ return

- ឃ្លា return ប្រើសម្រាប់ផ្តល់តម្លៃពី method មួយច្បាស់លាស់
- វាផ្ទៀងផ្ទាត់កម្មវិធី ដើម្បីបញ្ជូនតម្លៃត្រឡប់ទៅឱ្យ Caller របស់ Method វិញ
- ហេតុនេះហើយ ទើបគេចាត់ទុកវាជាប្រភេទឃ្លា Jump ( Jumping Statement )
- កម្មវិធីខាងក្រោម បង្ហាញពីការប្រើ return ។
- នៅក្នុងកម្មវិធីនេះ return ធ្វើឱ្យការប្រតិបត្តិវិលទៅរកប្រព័ន្ធដំណើរការរបស់ Java ពីព្រោះប្រព័ន្ធដំណើរការរបស់ Java ជាអ្នកហៅក្នុង main( ) ។

- សូមពិនិត្យមើលកម្មវិធីខាងក្រោម៖

```
public class Return {
    public static void main(String[] args) {
        boolean t = true;
        System.out.println("Before the return.");
        if(t) return;    // return to caller
        System.out.println("This won't execute.");
    }
}
```

- លទ្ធផលនៃកម្មវិធីនេះ គឺដូចខាងក្រោម៖



Console [X]

<terminated> Return [Java Application] C:\1. KHOEUN'S\k  
Before the return.



សាកលវិទ្យាល័យ អាស៊ី អឺរ៉ុប

ASIA EURO UNIVERSITY

និរន្តរ៍ក្នុងការសិក្សា និងបង្រៀន

# អនុណា សម្រាប់ការយកចិត្តទុកដាក់

<https://elearning.aeu.cloud>