មជ្ឈមណ្ឌលកូរ៉ែ សហ្វវែរ អេច អ ឌី

Korea Software HRD Center

ការគ្រប់គ្រងលំហូរបស់ Statement

Control Flow Statement

ណែនាំដោយ : Dr. Kim Tae Kyung

Webcash

KOICA
Korea International Cooperation Agency

๖๓๓

๑. If-then / if-then-else Statement

๒. Switch-case Statement

๓. Loop Statement

๔. Break & continue Keyword

๕. Arrays: One-Dimensional Array

១. If-then / if-then-else Statement

- ការប្រើប្រាស់ If Statements
- ការប្រើប្រាស់ Switch Statements
- ការប្រៀបធៀប If Statements ជាមួយនឹង Switch Statements

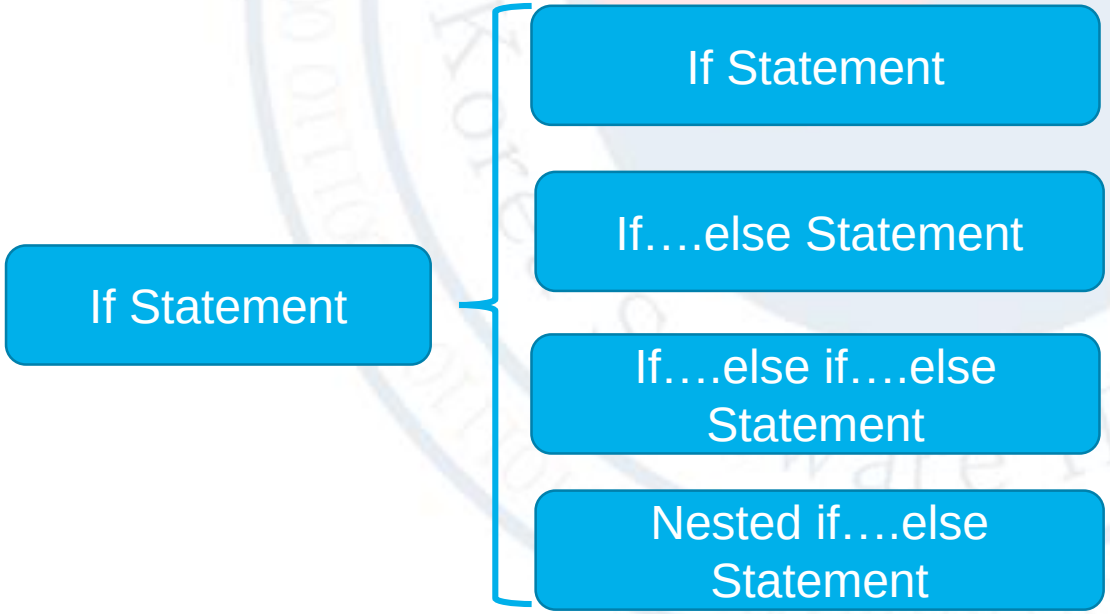
១. If-then / if-then-else Statement (ត)

- នៅក្នុងភាសា Java យើងអាចធ្វើការកំណត់លក្ខខ័ណ្ឌក្នុងប្រតិបត្តិការណ៍ការងាររបស់យើងបាន តាមរយៈ ២ ប្រភេទខុសៗគ្នា៖
  - ការប្រើប្រាស់នូវ If Statement
  - ការប្រើប្រាស់នូវ Switch Statement

១. If Statement (តរ)

- If Statement ត្រូវបានគេប្រើប្រាស់សំរាប់ត្រួតពិនិត្យលក្ខខ័ណ្ឌ ប្រសិនបើលក្ខខ័ណ្ឌនោះពិត នាំអោយ code នៅក្នុង block ធ្វើការ។

- If Statement ត្រូវបានបែងចែកចេញជា 4 ប្រភេទគឺ៖



១. If Statement (ក)

- If Statement ត្រូវបានគេប្រើប្រាស់សំរាប់ត្រួតពិនិត្យលក្ខខ័ណ្ឌ ប្រសិនបើលក្ខខ័ណ្ឌនោះពិត នាំអោយ code នៅក្នុង block ធ្វើការ។

- Syntax

```
if(លក្ខខ័ណ្ឌ) {

        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌនោះពិត
}
```

១. If Statement (តរ)

- សូមក្រឡេកមើលឧទាហរណ៍ខាងក្រោម៖

```
public class Test {

        public static void main(String args[]){

                int x = 10;

                if( x < 20 ){

                        System.out.print("This is if statement");
                }
        }
}
```

១. If……else Statement (ត)

- If…..else Statement ត្រូវបានគេប្រើប្រាស់សំរាប់ដាក់លក្ខខ័ណ្ឌមួយ ប្រសិនបើលក្ខខ័ណ្ឌនោះពិត នោះ code ក្នុង if blockនឹងប្រតិបត្តិ ហើយបើលក្ខខ័ណ្ឌនោះមិនពិត នោះ code ក្នុង else block នឹងអនុវត្ត។
- Syntax

```
if(លក្ខខ័ណ្ឌ) {

        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌនោះពិត
} else{
        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌនោះមិនពិត
}
```

១. If……else Statement (ត)

- សូមក្រឡេកមើលឧទាហរណ៍ខាងក្រោម៖

```
public class Test {
        public static void main(String args[]){
                int x = 30;
                if( x < 20 ){
                        System.out.print("This is if statement");
                }else{
                        System.out.print("This is else statement");
                 }
        }
}
```

១. If……else if……else Statement (ត)

- If….else if Statement ត្រូវបានគេប្រើប្រាស់សំរាប់ដាក់លក្ខខ័ណ្ឌច្រើន ប្រសិនបើលក្ខខ័ណ្ឌនោះពិត នោះ code ក្នុង blockនឹងប្រតិបត្តិ ហើយបើលក្ខខ័ណ្ឌនោះមិនពិត នោះ code ក្នុង else if block ខាងក្រោមនឹងប្រតិបត្តិ ផ្ទុយមកវិញវានឹងរំលងលក្ខខ័ណ្ឌដទៃទៀតដោយមិនប្រតិបត្តិ។

- Syntax

```
If (លក្ខខ័ណ្ឌទី១) {
        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌទី១ពិត
} else if (លក្ខខ័ណ្ឌទី២) {
        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌទី២ពិត
} else if (លក្ខខ័ណ្ឌទី៣) {
        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌទី៣ពិត
} else {
        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌទាំងនោះមិនពិត
}
```

១. If……else if……else Statement (ត)

- សូមក្រឡេកមើលឧទាហរណ៍ខាងក្រោម៖

```
public class Test {
        public static void main(String args[]){
                int x = 30;
                if( x == 10 ){
                        System.out.print("Value of X is 10");
                }else if( x == 20 ){
                        System.out.print("Value of X is 20");
                }else if( x == 30 ){
                        System.out.print("Value of X is 30");
                }else{
                        System.out.print("This is else statement");
                }
        }
}
```

១. Nested if……else Statement (ក)

- Nested if…else Statement ត្រូវបានគេប្រើប្រាស់សំរាប់ដាក់លក្ខខ័ណ្ឌនៅក្នុងលក្ខខ័ណ្ឌ។
- Syntax

```
If (លក្ខខ័ណ្ឌទី១) {
    //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌទី១ពិត
    if (លក្ខខ័ណ្ឌទី២) {
        //Statements នឹងប្រតិបត្តិនៅពេលដែលលក្ខខ័ណ្ឌទី ១ ពិត
        // ហើយនិងលក្ខខ័ណ្ឌទី ២ ពិត
    }
}
```

១. Nested if……else Statement (ត)

- សូមក្រឡេកមើលឧទាហរណ៍ខាងក្រោម៖

```
public class Test {
        public static void main(String args[]){
                int x = 30;
                int y = 10;
                if( x == 30 ){
                        if( y == 10 ){
                                System.out.print("X = 30 and Y = 10");
                        }
                }
        }
}
```

២. Switch-case Statement

- តើអ្វីទៅជា Switch Statement ?

- Switch Statement ត្រូវបានគេប្រើប្រាស់សំរាប់ធ្វៀងផ្ទាត់នូវ expression មួយជាមួយនឹងតម្លៃណាមួយដែលយើងចង់ធ្វៀងផ្ទាត់ ហើយវានឹងធ្វើការត្រួតពិនិត្យជាមួយនឹង case ណាមួយក្នុងលក្ខខ័ណ្ឌដូចទៅនឹង expression នោះ code នឹងប្រតិបត្តិ។

  Expression មានដូចជា៖ int, short, byte, char, and string ។

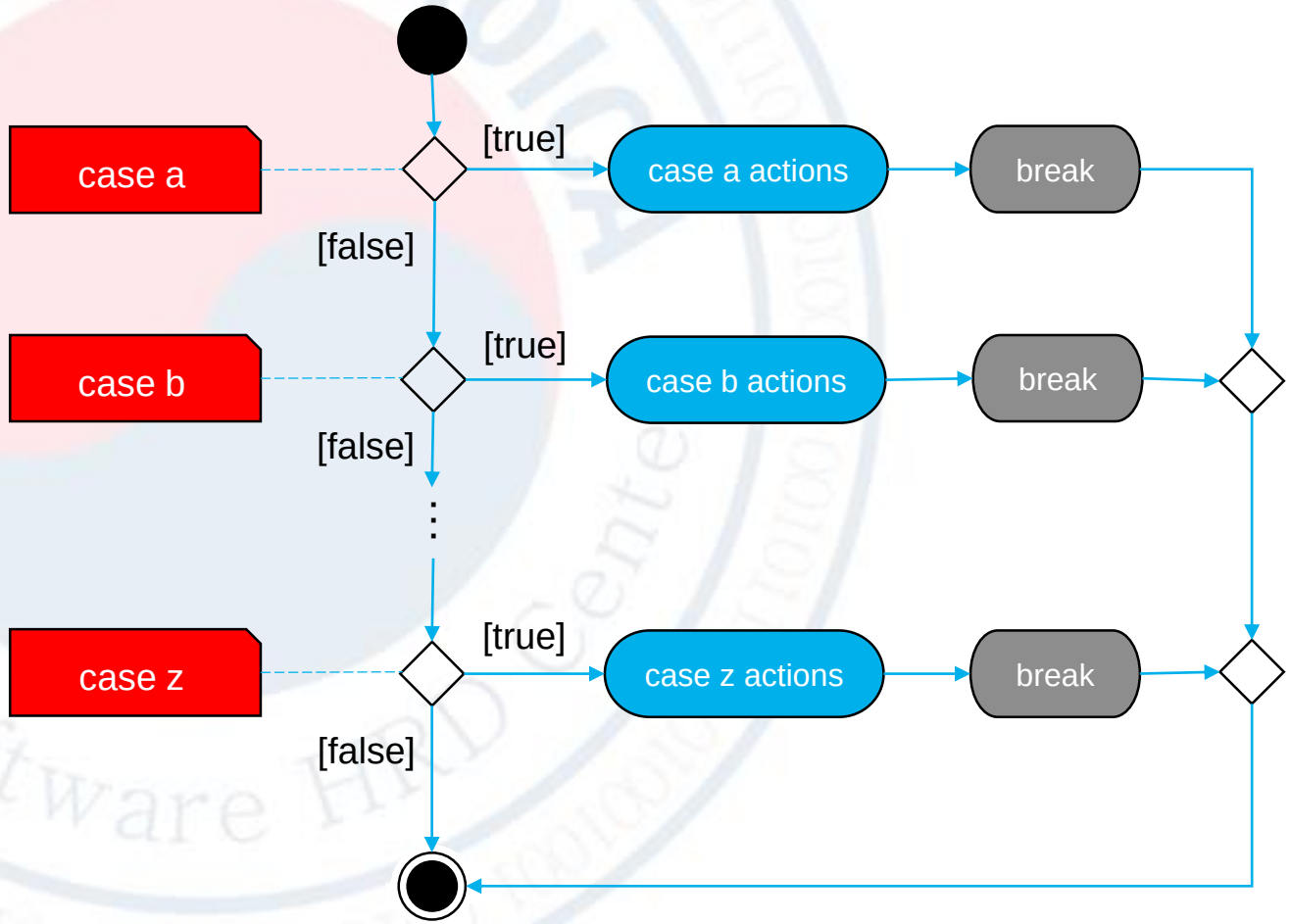
5. Switch-case Statement

- Syntax and diagram

```
switch (expression) {
case value:
        // Statements
        break; // optional
case value:
        // Statements
        break; // optional


// You can have any number of
// case statements.


default: // Optional
        // Statements
}
```



២. Switch-case Statement (ត)

- សូមក្រឡេកមើលឧទាហរណ៍ខាងក្រោម៖

```
public class Test {
        public static void main(String args[]){
        int num = 2;
        switch(num) {
                case 1 :
                        System.out.println(“Monday");
                        break;
                case 2:
                        System.out.println(“Tuesday");
                        break;
                default :
                        System.out.println(“Sunday");
        }
}
```

២. ការប្រៀបធៀប If Statements ជាមួយនិង Switch Statements

If Statements	Switch Statements
ត្រួតពិនិត្យមើលលក្ខខ័ណ្ឌ else if statement រហូតដល់លក្ខខ័ណ្ឌនោះពិត	ពិនិត្យលក្ខខ័ណ្ឌជាមួយនឹង expression ណាមួយដែលយើងចង់ផ្ទៀងផ្ទាត់។
មានភាព flexible ដែលអាចអោយយើងពិនិត្យលក្ខខ័ណ្ឌបានច្រើន។	មិនសូវ flexible ពីព្រោះវាអនុញ្ញាតអោយយើងធ្វើការ test តែទៅលើ expression តែមួយប៉ុណ្ណោះ។
បោះតម្លៃជា true false(zero/non-zero)	ធ្វើការជាមួយ equality operator
ទទួលយក all data types	ទទួលយកតែ primitive type តែប៉ុណ្ណោះដូចជា key ហើយនឹង constants for cases
Performance of if statement will be slow for multiple condition	Performance is faster because switch uses index mapping

៣. Loop / Repetition Statement

- អ្វីទៅគឺជា Loop?
  - Loop គឺជាការដំណើរការបណ្តុំនៃកូដ ជាច្រើនដង។
- Loop Statements របស់ Java ចែកចេញជា ២ ប្រភេទដូចខាងក្រោម៖
  - **Pre-test Loop**
    - ✓ For Loop
    - ✓ For Each Loop
    - ✓ While Loop
  - **Post-test Loop**
    - ✓ Do-while loop

៣. Loop / Repetition Statement (៤)

- Pre-test Loop គឺជា Control Structure មួយដែលធ្វើងផ្ទាត់លក្ខខ័ណ្ឌមុន ទើបដំណើរការ Loop តាមក្រោយ។
- ប្រសិនបើ លក្ខខ័ណ្ឌពិត វានឹងអនុវត្តន៍ Statement នៅក្នុង Loop រហូតដល់ Loop បញ្ឈប់ ។
- ប្រសិនបើ លក្ខខ័ណ្ឌមិនពិត វានឹងមិន អនុវត្តន៍នូវ Statement នៅក្នុង Loop ណាមួយឡើយ ហើយ វានឹងចាក់ចេញ ពី Loop ។

๓. Loop / Repetition Statement (๕)

- Syntax: **For Loop**

```
for (initialaztion; Boolean expression; update control) {
        Statements; // Loop body
    }

Or for (;;){// infinite for loop}
```

- Example:

```
for (int i = 0; i <= 20; i++) {
        System.out.println("value of i : " + i);
}
```

๓. Loop / Repetition Statement (๕)

- Syntax: **For Each**

```
for (Datatype variableName: array or collection) {
        //Statements; // Loop body
    }
```

- Example:

```
for (String string: arr) {
    System.out.println(string);
}
```

৩. Loop / Repetition Statement (৪)

- Syntax:

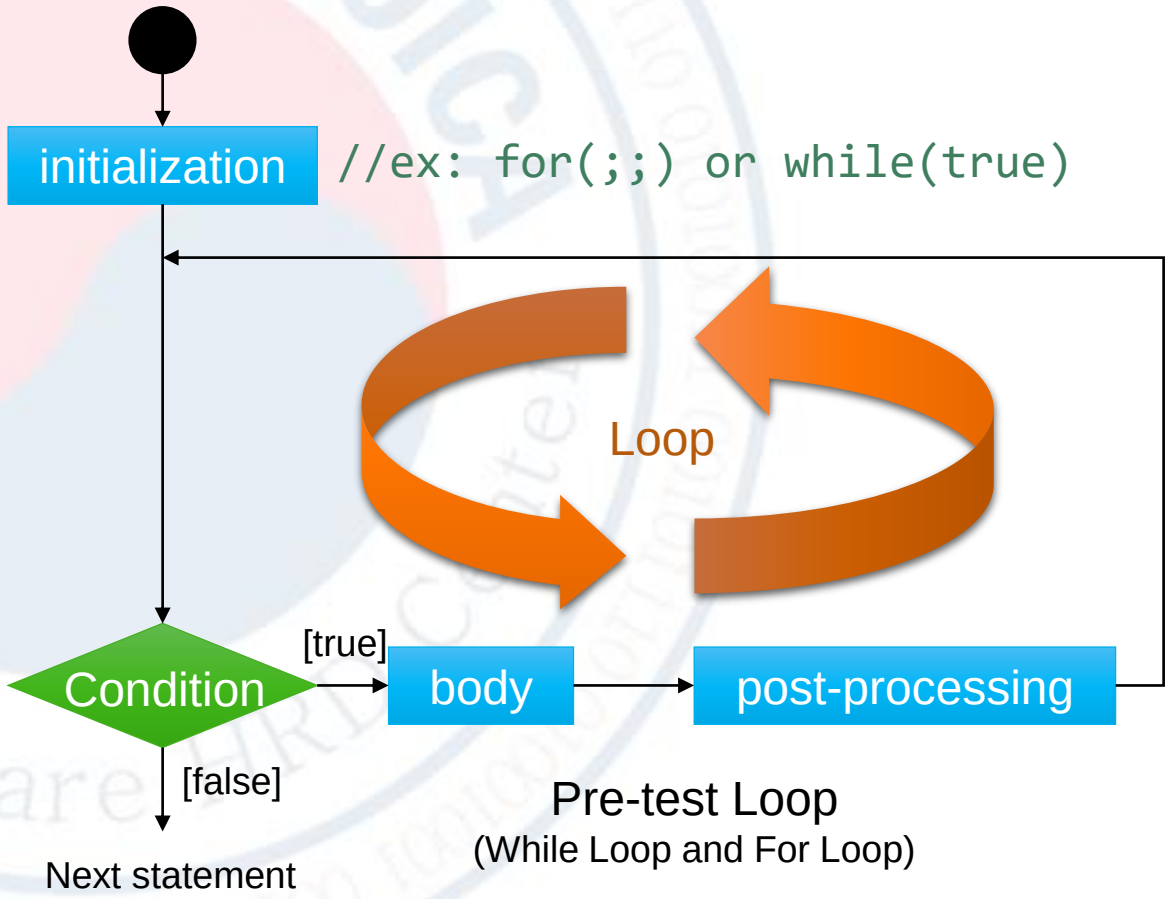
```
while (Boolean_expression) // condition to be checked
        {
            Statement(s); // loop body
        }
```

- Example:

```
int x = 10; // initial value
while (x < 20) {
        System.out.println("value of x : " + x);
        x++; // increment by 1
}
```

៣. Loop / Repetition Statement (ត)

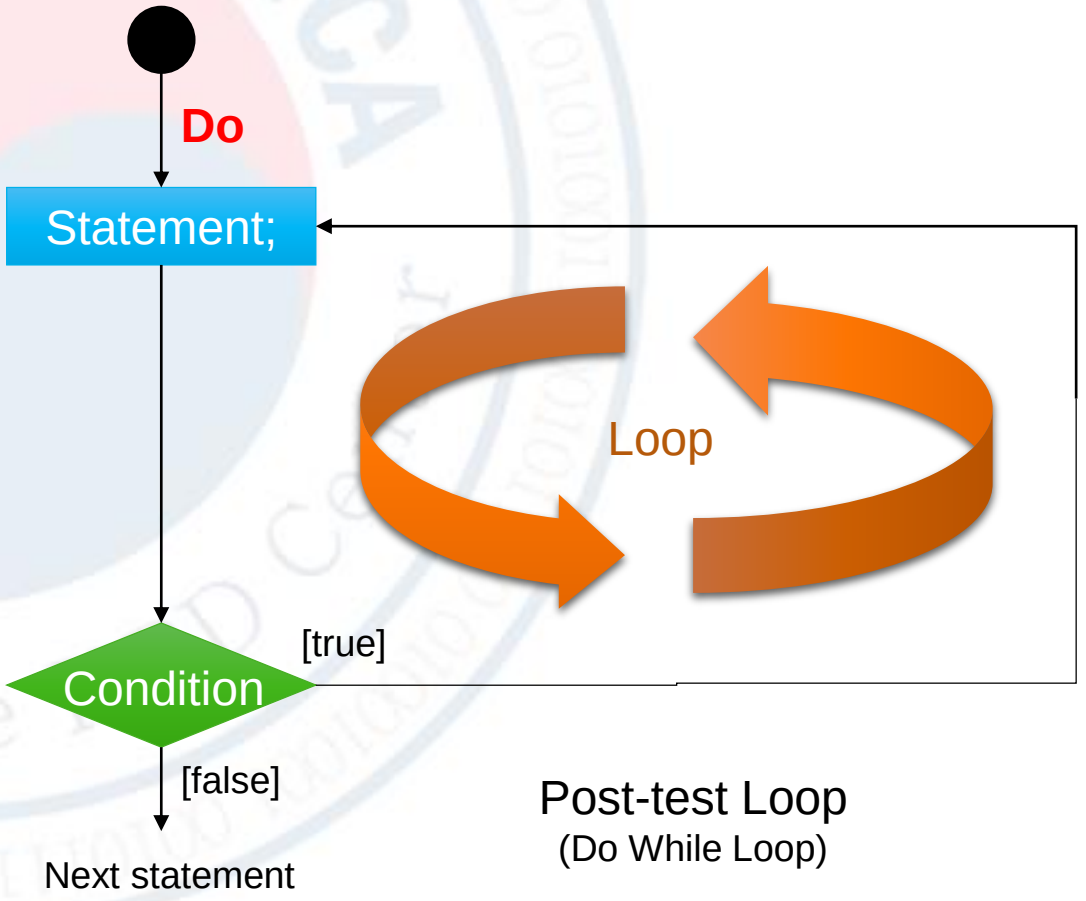
- ភាពខុសគ្នារវាង For Loop និង While Loop
  - For Loop: មានប្រយោជន៍ នៅពេលដែលយើងដឹងថា តើត្រូវធ្វើកិច្ចការមួយប៉ុន្មានដង ដែលជួយឲ្យយើងដំណើរ Loop តាមចំនួនដងជាក់លាក់។
  - While Loop: ប្រើនៅពេលដែលយើងមិនដឹងច្បាសពីលំហូរកិច្ចការថា តើត្រូវដំណើរការប៉ុន្មានដង វាធ្វើការរហូតដល់លក្ខខ័ណ្ឌពិតបានឈប់។



៣. Loop / Repetition Statement (ត)

Post-test Loop គឺជា Control Structure មួយដែលដំណើរការ Loop ម្តងហើយ ទើបធ្វើងផ្ទាត់លក្ខខ័ណ្ឌតាមក្រោយ។ ហើយវាគឺជា Do While Loop។

- Do Loop Statement យ៉ាងហោចណាស់ ដំណើរការ code បានម្តង។



Post-test Loop
(Do While Loop)

๓. Loop / Repetition Statement (๕)

- Syntax:

```
do {
        Statement(s); // loop body
} while (Boolean_expression); // condition to be checked
```

- Example:

```
int x = 10;
do {
        System.out.println("value of x : " + x);
        x++;
} while (x < 20);
```

៤. Break & continue Keyword

Break and Continue Statements

- ក្រៅពី Control Structure ប្រភេទ Selection និង repetition Java ផ្តល់នូវ Statement Break និង Continue ដើម្បីកែសំរួល Flow of Control ។
- Break ប្រើសំរាប់ terminate block នៃកូដណាមួយ និងប្រើសម្រាប់បញ្ឈប់ Loop ឬ Switch ផងដែរ។
- ការប្រើប្រាស់ Break ចែកចេញជា ២ ប្រភេទគឺ៖
  - ប្រើ label
  - មិនប្រើ label

៤. Break & continue Keyword (ត)

- ការប្រើប្រាស់ break ជាមួយនិង label
  - ជាទូទៅគេប្រើប្រាស់ break statement ដោយប្រើ label ជាមួយនឹង nested loop ដើម្បីចាកចេញពី inner loop ទៅកាន់ outer loop ហើយដំណើរការនៃការ execute ត្រូវបានបញ្ចប់។

```
String s="Hello World";
outer: for (int i = 0; i < 10; i++) {
        for (int j = 0; j < s.length(); j++) {
            char c = s.charAt(j);
            if(c=='W'){
                break outer;
            }else if(c=='l'){
                continue;
            }
            System.out.print(c);
        }
}
```

Output: Heo

៤. Break & continue Keyword (ត)

- ការប្រើប្រាស់ break ដោយមិនប្រើ label
  - ជាទូទៅគេប្រើប្រាស់ break statement ដោយមិនប្រើ label ជាមួយនឹង loop ធម្មតាដើម្បីបញ្ចប់ដំណើរការរបស់ block នៃកូដមួយនោះ និងបញ្ឈប់ Loop ដែលនៅក្នុងគេបំផុត ហើយចាប់ផ្តើមដំណើរការ code ដែលនៅបន្ទាត់បន្ទាប់ ។

```
int[] numbers = { 10, 20, 30, 40, 50 };

for (int x : numbers) {
        if (x == 30) {
            break;
        }
        System.out.print(x + " ");
}
```

Output: 10 20

៤. Break & continue Keyword (ត)

- Continue statement ប្រើដើម្បី រំលង តំលៃណាមួយនៃ Loop Statement ហើយដំណើរការតំលៃបន្ទាប់ទីពីរតាមលំដាប់តទៅទៀត។
- ការប្រើប្រាស់ continue ចែកចេញជា ២ ប្រភេទគឺ៖
  - ប្រើ label
  - មិនប្រើ label

៤. Break & continue Keyword (ត)

- ការប្រើប្រាស់ conntinue ជាមួយនិង label
  - ជាទូទៅគេប្រើប្រាស់ continue statement ដោយប្រើ label ជាមួយនឹង nested loop ដើម្បីចាកចេញពី inner loop ទៅកាន់ outer loop ប៉ុន្តែការដំណើរការ execute នៅតែបន្ត។

```
String s = "Hello World";
outer: for (int i = 0; i < 10; i++) {
        for (int j = 0; j < s.length(); j++) {
            char c = s.charAt(j);
            if (c == 'l') {
                continue outer;
            }
            System.out.print(c);
        }
}
```

Output: HeHeHeHeHeHeHeHeHeHe

៤. Break & continue Keyword (ត)

- ការប្រើប្រាស់ continue ដោយមិនប្រើ label
  - ជាទូទៅគេប្រើប្រាស់ continue statement ដោយមិនប្រើ label ជាមួយនឹង loop ធម្មតា ដើម្បី skip នូវការ execute កូដណាមួយ។

```
int[] numbers = { 10, 20, 30, 40, 50 };

for (int x : numbers) {
        if (x == 30) {
                continue;
        }
        System.out.print(x + " ");
}
```

Output: 10 20 40 50

៥. Arrays

- យើងបានដឹងហើយថា Variable មានលទ្ធភាពផ្ទុកតម្លៃបានតែមួយប៉ុណ្ណោះក្នុងពេលតែមួយ។ ចុះឧទាហរណ៍ បើអ្នកចង់ផ្ទុកតម្លៃឈ្មោះ ចំនួន 100 នាក់តើអ្នកត្រូវបង្កើត variables ចំនួន 100 អ្នកឬ?
- ដូច្នេះដើម្បីដោះស្រាយបញ្ហានេះអ្នកអាចប្រើប្រាស់ Array។

៥. Arrays (ក)

- **អ្វីទៅជា Array?**
  - **Array** ជាសំណុំនៃ Variables សម្រាប់តំណាងឲ្យទិន្នន័យណាមួយ ដែល Variables ទាំងនោះមាន DataType ដូចៗគ្នា និងអាចផ្ទុកតម្លៃរៀងៗខ្លួនតាមរយៈ index របស់វា។
  - **Array** អាចជា Array មួយវិមាត្រ ឬ ច្រើនវិមាត្រ

0	1	2	3	4

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3

៥.១. Single Dimensional Array

- **ការបង្កើត Array មួយវិមាត្រ**
  - ទំរង់ទូទៅនៃការបង្កើត Array មួយវិមាត្រ

DataType[ ] ArrayName = new DataType[n]

Or

DataType ArrayName[ ] = new DataType[n]

n : ជាចំនួនធាតុនៃ Array

៥.១. Single Dimensional Array (ត)

- ឧទាហរណ៍

```
int[ ] score = new int [3];

int score[ ] = new int [3];
```

គ្រប់ធាតុ ជា integer Variable

៥.១. Single Dimensional Array (ត)

- ការផ្តល់តំលៃទៅអោយ Array នៅពេលបង្កើត

String[ ] titles = { "Mr." , "Mrs." , "Ms." };

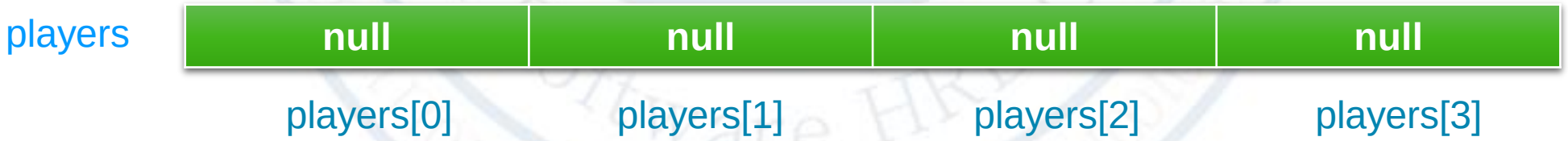
or String[ ] titles = new String[ ]{ "Mr." , "Mrs." , "Ms." };

- ការផ្តល់តំលៃទៅអោយ Array តាមរយៈ Index នៃធាតុនីមួយៗ

ArrayName [index] = Value;

Ex :

String[ ] players = new String[4];



player[0] = "Ronaldo";
player[1] = "Beckham";

៥.១. Single Dimensional Array (ត)

- ការទាញតំលៃពី Array

```
String name = player [0];
```

- ការបង្កើត Array នៃ Object

```
Student[ ] student = new Student [3];

student [0] = new Student (“Mike”);

student [1] = new Student (“John”);

student [2] = new Student (“Den”);
```

៥.១. Single Dimensional Array (ត)

- តំលៃ Default របស់ Array

Type	Default Value
Numeric Array	0
Character Array	‘\0’
Boolean	False
Object	null