Loop 4.3

Loops in Java
In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in java.

1. for loop
2. while loop
3. do-while loop

Java For Loop
The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

There are three types of for loops in java.

1. Simple For Loop
2. For-each or Enhanced For Loop
3. Labeled For Loop

Java Simple For Loop
A simple for loop is the same as C/C++. We can initialize the variable, check condition and increment/decrement value. It consists of four parts:

1. Initialization: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.

2. Condition: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.

3. Statement: The statement of the loop is executed each time until the second condition is false.
Increment/Decrement: It increments or decrements the variable value. It is an optional condition.

Syntax:

```
for(initialization;condition;incr/decr){
//statement or code to be executed
}
```

Example :

```
//Java Program to demonstrate the example of for loop
//which prints table of 1
public class ForExample {
public static void main(String[] args) {
   //Code of Java for loop
   for(int i=1;i<=10;i++){
      System.out.println(i);
```

```
    }
}
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Java for-each Loop

The for-each loop is used to traverse array or collection in java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.

It works on elements basis not index. It returns element one by one in the defined variable.

Syntax:

```
for(Type var:array){
```

Example :

```
//Java For-each loop example which prints the
//elements of the array
public class ForEachExample {
public static void main(String[] args) {
    //Declaring an array
    int arr[]={12,23,44,56,78};
    //Printing array using for-each loop
    for(int i:arr){
        System.out.println(i);
    }
}
}
```

Output:

```
12
23
44
56
78
```

Java Labeled For Loop

We can have a name of each Java for loop. To do so, we use label before the for loop. It is

useful if we have nested for loop so that we can break/continue specific for loop.

Usually, break and continue keywords breaks/continues the innermost for loop only.

Syntax:

```
labelname:
for(initialization;condition;incr/decr){
//code to be executed
}
```

Example :

//A Java program to demonstrate the use of labeled for loop

Output:

```
1 1
1 2
1 3
2 1
```

If you use break bb;, it will break inner loop only which is the default behavior of any loop.

```
public class LabeledForExample2 {
public static void main(String[] args) {
    aa:
        for(int i=1;i<=3;i++){
            bb:
                for(int j=1;j<=3;j++){
                    if(i==2&&j==2){
                        break bb;
                    }
                    System.out.println(i+" "+j);
                }
        }
}
}
```

Output:

```
1 1
1 2
1 3
2 1
3 1
3 2
3 3
```

Java Infinitive For Loop

If you use two semicolons ;; in the for loop, it will be infinitive for loop.

Syntax:

```
for(;;){
//code to be executed
}
```

```
//Java program to demonstrate the use of infinite for loop
//which prints an statement
public class ForExample {
public static void main(String[] args) {
    //Using no condition in for loop
    for(;;){
        System.out.println("infinitive loop");
    }
}
}
```

Output:

```
infinitive loop
infinitive loop
infinitive loop
infinitive loop
infinitive loop
ctrl+c
```

Now, you need to press ctrl+c to exit from the program.