BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Loop Control statements and Labelled Loop in Java.

Loop Control Statement:-

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Java supports the following control statements.

Brake Statement:-

Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.

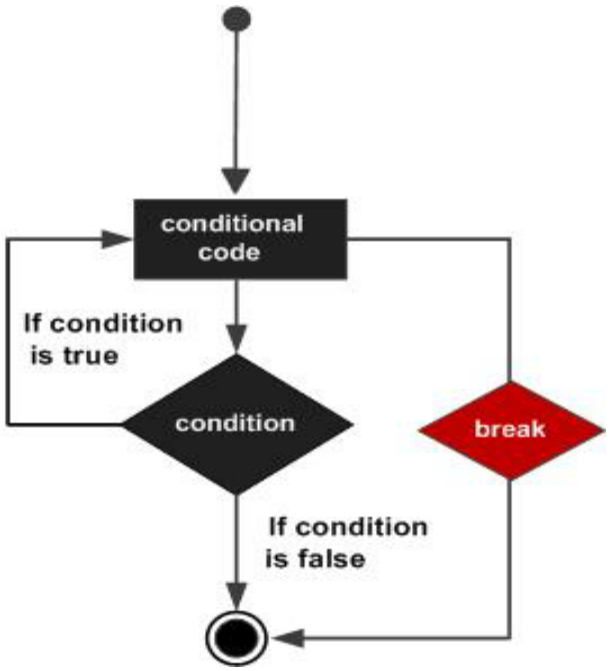
The break statement in Java programming language has the following two usages −

- When the break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.
- It can be used to terminate a case in the switch statement (covered in the next chapter).

Syntax:-

The syntax of a break is single statement inside any loop −
break;

Flow Diagram:-



Example:-

```
public class Test
{
  public static void main(String args[])
  {
    int [] numbers = {10, 20, 30, 40, 50};
    for(int x : numbers )
    {
      if( x == 30 )
      {
        break;
```

```
        }
        System.out.print( x );
        System.out.print("\n");
     }
  }
}
```

Output:- This will produce the following result –

10

20

Continue Statement:-

Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

The continue keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop.

- In a for loop, the continue keyword causes control to immediately jump to the update statement.
- In a while loop or do/while loop, control immediately jumps to the Boolean expression.

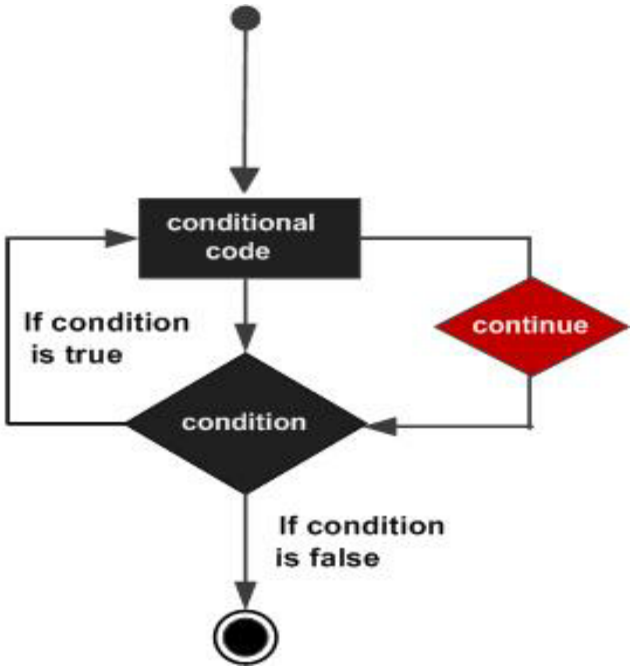
Syntax:-

The syntax of a continue is a single statement inside any

loop –

continue;

Flow Diagram:-



Example:-

```
public class Test
{
   public static void main(String args[])
   {
      int [] numbers = {10, 20, 30, 40, 50};
      for(int x : numbers )
      {
         if( x == 30 )
```

```
        {
            continue;
         }
         System.out.print( x );
         System.out.print("\n");
      }
   }
}
```

Output:- This will produce the following result −

```
10
20
40
50
```

Labeled Loop in Java:-

A label is a valid variable name that denotes the name of the loop to where the control of execution should jump. To label a loop, place the label before the loop with a colon at the end. Therefore, a loop with the label is called a labeled loop.

In layman terms, we can say that label is nothing but to provide a name to a loop. It is a good habit to label a loop

when using a nested loop. We can also use labels with continue and break statements.

Labeled for Loop:-

Labeling a for loop is useful when we want to break or continue a specific for loop according to requirement. If we put a break statement inside an inner for loop, the compiler will jump out from the inner loop and continue with the outer loop again.

What if we need to jump out from the outer loop using the break statement given inside the inner loop? The answer is, we should define the label along with the colon(:) sign before the loop.

Syntax:

```
labelname:
for(initialization; condition; incr/decr)
{
//functionality of the loop
}
```

Example:-

```
public class LabeledForLoop
{
public static void main(String args[])
{
```

```
int i, j;
//outer loop
outer:     //label
for(i=1;i<=5;i++)
{
System.out.println();
//inner loop
inner:      //label
for(j=1;j<=10;j++)
{
System.out.print(j + " ");
if(j==9)
break inner;
}
}
}
}
```

Output:

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

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

Labeled while Loop:-

Syntax:

```
labelName:
while ( ... )
{
//statements to execute
}
```

Example:-

```
public class LabledWhileLoop
{
public static void main(String args[])
{
int i = 0;
whilelabel:
while (i < 5)
{
```

```
System.out.println("outer value of i= " + i);
i++;
forlabel:
for (int j = 0; j < 5; j++)
{
if (j > 0)
{
//execution transfer to the for loop
continue forlabel;
}  //end of if
if (i > 1)
{
//execution transfer to the while loop
continue whilelabel;
}  //end of if
System.out.println("inner value of i= " + i + ", j= " + j);
}  //end of for
}  //end of while
}  //end of main
}
```

Output:-

outer value of i= 0

inner value of i= 1, j= 0

outer value of i= 1

outer value of i= 2

outer value of i= 3

outer value of i= 4