BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Looping statements in Java.

Loops in Java:-

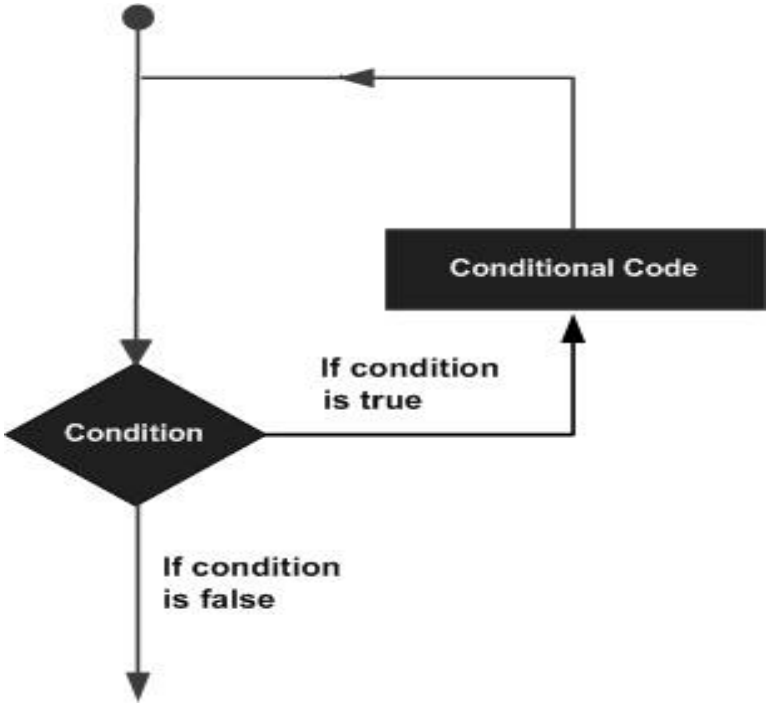
Loops can execute a block of code as long as a specified condition is reached.

Loops are handy because they save time, reduce errors, and they make code more readable.

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −



Types of Loop:-

Java programming language provides the following types of loop to handle looping requirements.

- ➤ For Loop
- ➤ While Loop
- ➤ Do While Loop

For Loop:-

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to be executed a specific number of times.

A for loop is useful when you know how many times a task is to be repeated.

Syntax:- The syntax of a for loop is –

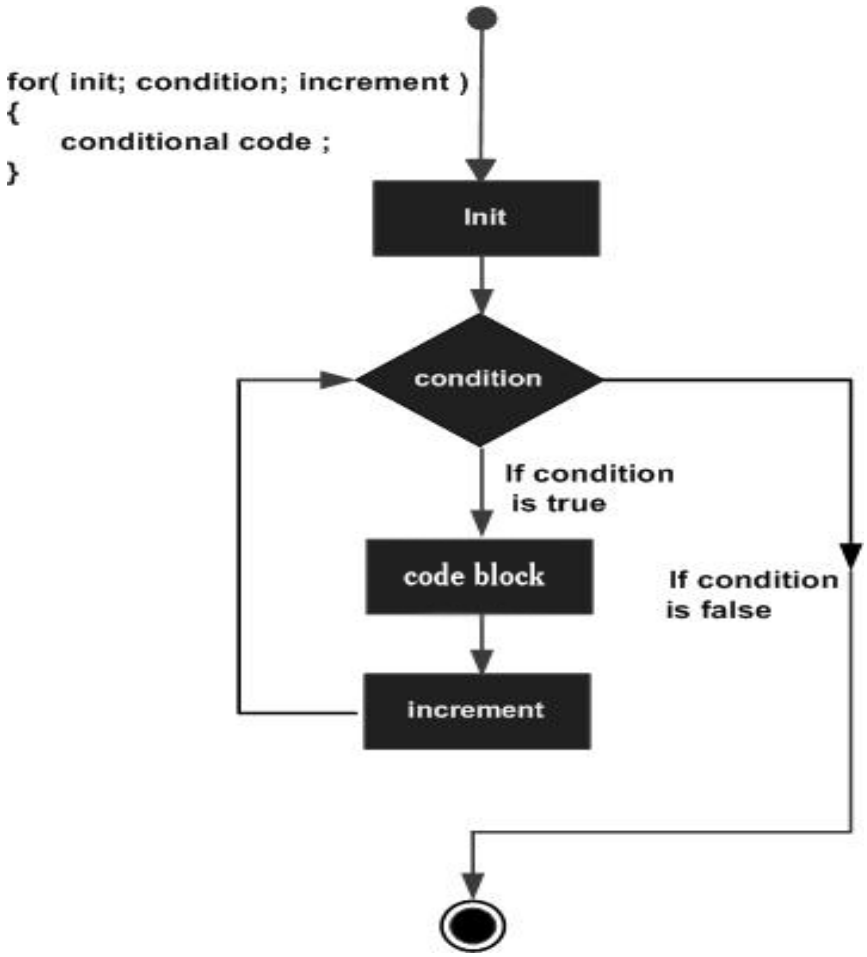
```
for(initialization, condition, increment/decrement)
{
//block of statements
}
```

Here is the flow of control in a for loop –

- The **initialization** step is executed first, and only once. This step allows you to declare and initialize any loop control variables and this step ends with a semi colon (;).
- Next, the **Boolean expression/ condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop will not be executed and control jumps to the next statement past the for loop.
- After the **body** of the for loop gets executed, the control jumps back up to the update (increment/decrement) statement. This statement allows you to update any loop control variables. This statement can be left blank with a semicolon at the end.
- The **Boolean expression/Condition** is now evaluated again. If it is true, the loop executes and the process repeats (body of loop, then update (increment/decrement) step, then Boolean expression).

After the Boolean expression is false, the for loop terminates.

Flow diagram of for loop:-



Example:- Following is an example code of the for loop in Java.

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

```
      for(int x = 10; x < 20; x = x + 1)
      {
         System.out.print("value of x : " + x  +"\n");
      }
   }
}
```

Output:- This will produce the following result −

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```

Java for-each loop/enhanced for loop:-

As of Java 5, the enhanced for loop was introduced. This is mainly used to traverse collection of elements including arrays.

Java provides an enhanced for loop to traverse the data structures like array or collection. In the for-each loop, we don't need to update the loop variable.

Syntax:-

```
for(data_type var : array_name/collection_name)
{
//statements
}
```

We can also write this syntax in following way.

```
for(declaration : expression)
{
  // Statements
}
```

Declaration – The newly declared block variable, is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.

Expression – This evaluates to the array you need to loop through. The expression can be an array variable or method call that returns an array.

Example:-

```
public class Calculation
{
public static void main(String[] args)
{
String[] names = {"C","C++","Java","Python","JavaScript"};
System.out.println("Printing the content of the array names:\n");
for(String name:names)
 {
System.out.println(name);
}
}
}
```

Output:

```
Printing the content of the array names:
Java
```

C

C++

Python

JavaScript

Other Example:-

```
public class Test
{
  public static void main(String args[])
  {
    int [] numbers = {10, 20, 30, 40, 50};
    for(int x : numbers )
    {
      System.out.print( x );
      System.out.print(",");
    }
    System.out.print("\n");
    String [] names = {"Ram", "Shyam", "Mohan", "Sohan"};
    for( String name : names )
    {
```

```
        System.out.print( name );
        System.out.print(",");
      }
   }
}
```

Output:- This will produce the following result −

10, 20, 30, 40, 50,

Ram, Shyam, Mohan, Sohan,

While Loop:-

It repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

A while loop statement in Java programming language repeatedly executes a target statement as long as a given condition is true.

The while loop is also used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop. Unlike for loop, the initialization and increment/decrement doesn't take place inside the loop statement in while loop.

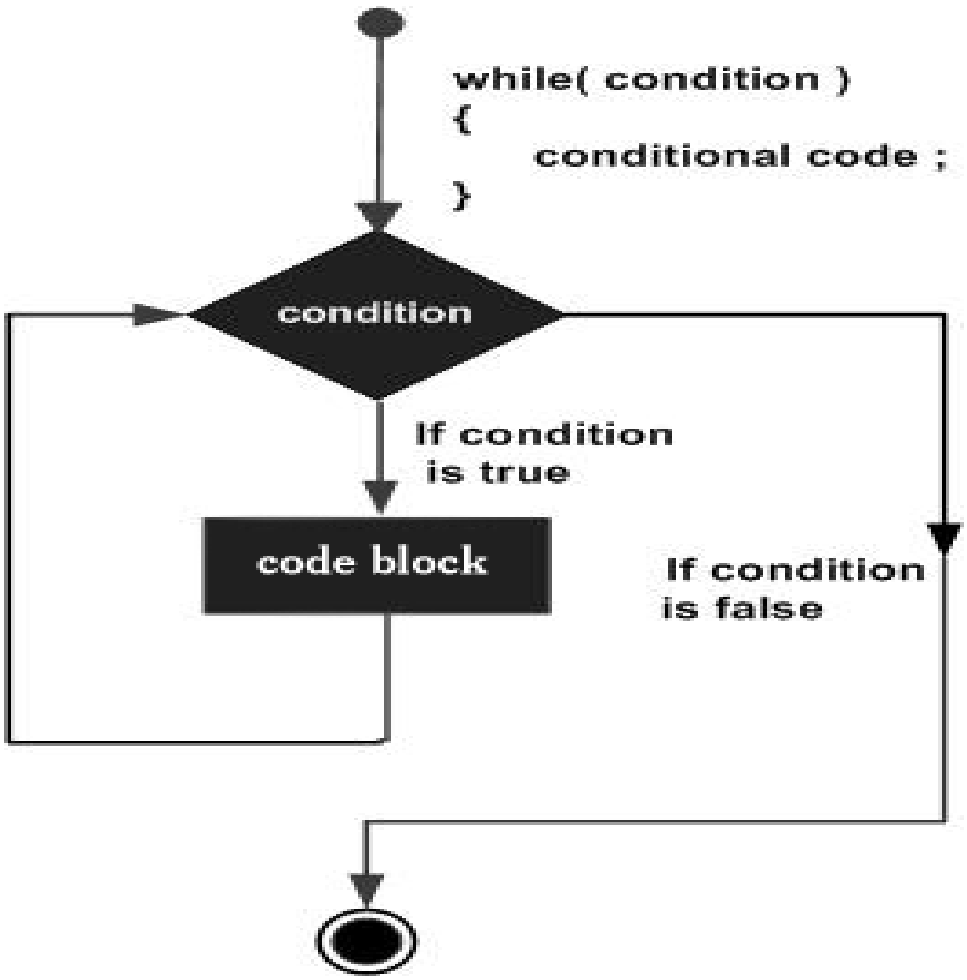
It is also known as the entry-controlled loop since the

condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

Syntax:-

```
while(condition)
{
//looping statements
}
```

Flow Diagram:-



Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any non zero value.

When executing, if the boolean_expression result is true, then the actions inside the loop will be executed. This will continue as long as the expression result is true.

When the condition becomes false, program control passes to the line immediately following the loop.

Here, key point of the while loop is that the loop might not ever run. When the expression is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Example:-

```
public class Test
{
  public static void main(String args[])
 {
    int x = 10;
    while( x < 20 )
    {
      System.out.print("value of x : " + x );
      x++;
```

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

Output:- This will produce the following result–

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```

Do... While Loop:-

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

The do-while loop checks the condition at the end of the loop after executing the loop statements. When the number

of iteration is not known and we have to execute the loop at least once, we can use do-while loop.

It is also known as the exit-controlled loop since the condition is not checked in advance.

Syntax:-

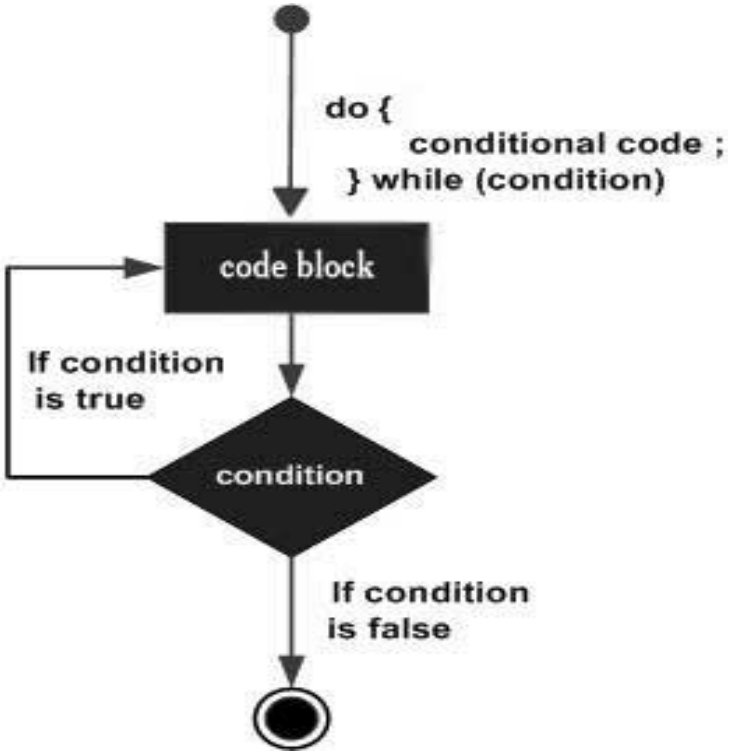
Following is the syntax of a do...while loop −

```
do
{
   // Statements
}
while(Boolean_expression);
```

Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute once before the Boolean is tested.

If the Boolean expression is true, the control jumps back up to do statement, and the statements in the loop execute again. This process repeats until the Boolean expression is false.

Flow Diagram:-



Example:-

```
public class Test
{
  public static void main(String args[])
  {
    int x = 10;
    do
   {
      System.out.print("value of x : " + x );
      x++;
      System.out.print("\n");
```

```
      }
      while( x < 20 );
   }
}
```

Output:- This will produce the following result −

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```