17. Control flow Statements or Jump Statements

Jump statements are used to transfer the control of the program to the specific statements. In other words, jump statements transfer the execution control to the other part of the program. There are two types of jump statements in Java, i.e., break and continue.

As the name suggests, the break statement is used to break the current flow of the program and transfer the control to the next statement outside a loop or switch statement. However, it breaks only the inner loop in the case of the nested loop. The break statement cannot be used independently in the Java program, i.e., it can only be written inside the loop or switch statement. When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop. The Java break statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop. We can use Java break statement in all types of loops such as for loop , while loop and do-while loop.

17.1 break Statement

Syntax:

break;

break Example Program1:

```
//Java Program to demonstrate the use of break statement
//inside the for loop.
public class BreakExample1
{
      public static void main(String[] args)
      {
            //using for loop
            for(int i=1;i<=10;i++)
            {
                  if(i==5)
                  {
                  //breaking the loop
                  break;
                  }
            System.out.println(i);
            }
      }
}
```

Output:

```
1
2
3
4
```

break Example Program2:

```
//Java Program to illustrate the use of break statement
//inside an inner loop
public class BreakExample2
{
      public static void main(String[] args)
      {
      //outer loop
      for(int i=1;i<=3;i++)
      {
            //inner loop
            for(int j=1;j<=3;j++)
            {
            if(i==2&&j==2)
                  {
                        //using break statement inside the inner loop
                        break;
                  }
            System.out.println(i+" "+j);
            }
      }
      }
}
```

Output:

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

17.2 continue Statement

Unlike break statement, the continue statement doesn't break the loop, whereas, it skips the specific part of the loop and jumps to the next iteration of the loop immediately. The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop. The Java continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only. We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

continue;

continue Example Program1:

```
//Java Program to demonstrate the use of continue statement
//inside the for loop.
public class ContinueExample1
{
      public static void main(String[] args)
      {
            //for loop
            for(int i=1;i<=10;i++)
            {
                  if(i==5)
                  {
                        //using continue statement
                        continue;//it will skip the rest statement
                  }
            System.out.println(i);
            }
      }
}
```

Output:

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

continue Example Program2:

```
//Java Program to illustrate the use of continue statement
//inside an inner loop
public class ContinueExample2
{
      public static void main(String[] args)
      {
     //outer loop
     for(int i=1;i<=3;i++)
     {
            //inner loop
            for(int j=1;j<=3;j++)
            {
                  if(i==2&&j==2)
                  {
                        //using continue statement inside inner loop
                        continue;
```

```
                }
            System.out.println(i+" "+j);
            }
        }
        }
}
```

Output:

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