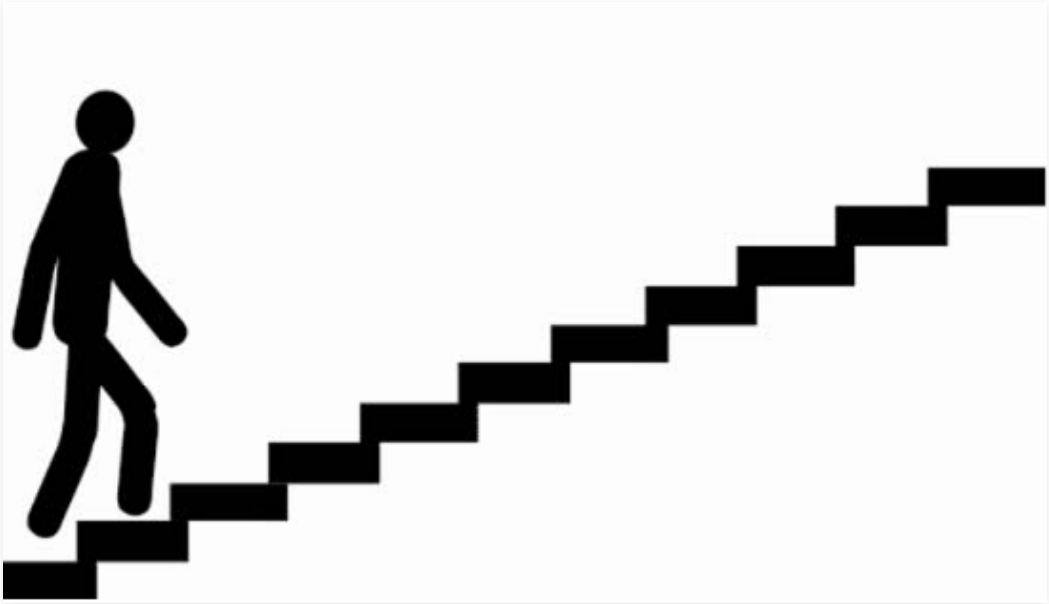
Continue Statement in Java

Difficulty Level : Basic Last Updated : 04 Feb, 2022

Suppose a person wants code to execute for the values as per the code is designed to be executed but forcefully the same user wants to skip out the execution for which code should have been executed as designed above but will not as per the demand of the user. In simpler words, it is a decision-making problem as per the demand of the user.

Real-Life Example:



Consider a man is climbing up to go to his house in between there are 11 stairs. Bei ng in hurry to climb up he directly stepped onto 3 staircases and then 4, 5, 6, 7, 8, 9 and jumps to last one. During this he missed out staircase 1st, 2nd and 10th and he completed the goal to reach his house. He continued his journey skipping staircase of. his choices.

In computers, it interprets staircases which is/are supposed to be skipped as 'contin ue'. The action to miss out execution which are supposed to be executed, is interpre ted as continue statement be it any programming language.

Continue statement is often used inside in programming languages inside loops control structures. Inside the loop, **when a continue statement is encountered the control directly jumps to the beginning of the loop for the next iteration instead of**

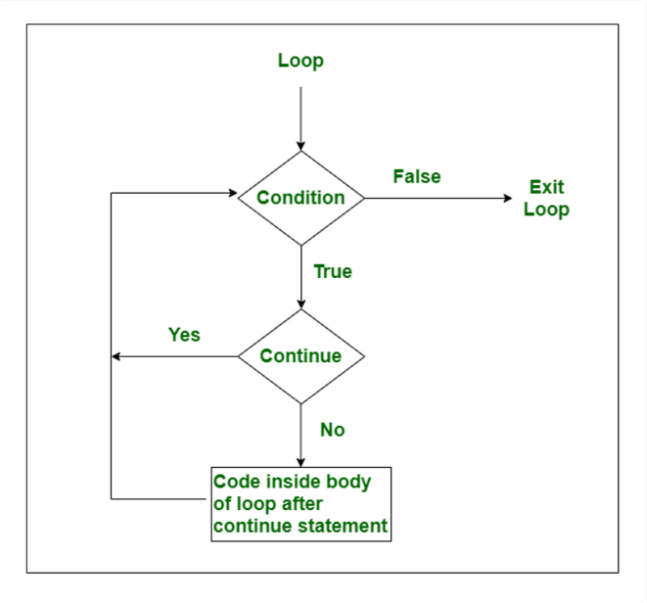
executing the statements of the current iteration. The continue statement is used when we want to skip a particular condition and continue the rest execution. Java continue statement is used for all type of loops but it is generally used in for, while, and do-while loops.

- In the case of for loop, the continue keyword force control to jump immediately to the update statement.
- Whereas in the case of a while loop or do-while loop, control immediately jumps to the Boolean expression.

Syntax: continue keyword along with a semicolon

```
continue;
```

Flow Chart of Continue Statement



The above flowchart is most important for the understanding of this keyword. Always remember the condition is always placed inside diamond boxes and statements in rectangular boxes. Now jumping onto the implementation part

Case 1: Continue statement inside for loop

In this program, illustration for how to use the continue statement within For loop. When the value of ‘i’ becomes 10 or 12, the continue statement plays its role and skip their execution but for other values of’ ‘i’ the loop will run smoothly.

```
// Java Program to illustrate the use of continue statement

// Importing Classes/Files
import java.util.*;
public class GFG {

    // Main driver method
    public static void main(String args[])
    {
        // For loop for iteration
        for (int i = 0; i <= 15; i++) {

            // Check condition for continue
            if (i == 10 || i == 12) {

                // Using continue statement to skip the
                // execution of loop when i==10 or i==12
                continue;
            }
            // Printing elements to show continue statement
            System.out.print(i + " ");
        }
    }
}
```

Output :

```
0 1 2 3 4 5 6 7 8 9 11 13 14 15
```

Case 2: Continue statement inside while loop

In the above program, we give example, how to use the continue statement within the While loop. When the value of count becomes 7 or 15, the continue statement plays its role and skip their execution but for other values of the count, the loop will run smoothly.

```
// Java Program to illustrate the use of continue statement
// inside the While loop
public class GFG {

    // Main driver method
    public static void main(String args[])
    {
```

```
        // Initializing a variable say it count  to a value
        // greater than the value greater among the loop
        // values
        int count = 20;

        // While loop for iteration
        while (count >= 0) {
            if (count == 7 || count == 15) {
                count--;
                // Decrementing variable initialized above

                // Showing continue execution inside loop
                // skipping when count==7 or count==15
                continue;
            }

            // Printing values after continue statement
            System.out.print(count + " ");

            // Decrementing the count variable
            count--;
        }
    }
}
```

Output:

```
20 19 18 17 16 14 13 12 11 10 9 8 6 5 4 3 2 1 0
```

Case 3: Continue statement inside do while loop

In the above program, we give example, how to use the continue statement within the do-While loop. When the value of i becomes 4 or 18, the continue statement plays its role and skip their execution but for other values of i, the loop will run smoothly.

```
// Java Program to illustrate the use of continue statement
// inside the Do-While loop

// Importing generic Classes/Files
import java.util.*;

public class GFG {
```

```
    // Main driver method
    public static void main(String[] args)
    {
        // Creating and Initializing a variable
        int i = 0;

        // Do-While loop for iteration
        do {
            if (i == 4 || i == 18) {

                // Incrementing loop variable by 2
                i += 2;

                // Illustrating continue statement skipping
                // the execution  when i==7 or i==15
                continue;
            }

            // Printing to showcase continue affect
            System.out.println(i);

            // Incrementing variable by 2
            i += 2;

            // Condition check
        } while (i <= 35);
    }
}
```

Output:

```
0
2
6
8
10
12
14
16
20
22
24
26
28
30
```

32

34

Case 4: Continue statement inside Inner loop(Nested Loop)

In the above program, we give example, how to use the continue statement within Nested loops. When the value of i becomes 3 and j become 2, the continue statement plays its role and skip their execution but for other values of i and j, the loop will run smoothly.

```
// Java Program to illustrate the use of continue statement
// inside an inner loop or simply nested loops

// Importing generic Classes/Files
import java.util.*;

public class GFG {

    // Main drive method
    public static void main(String[] args)
    {
        // Outer loop for iteration
        for (int i = 1; i <= 4; i++) {

            // Inner loop for iteration
            for (int j = 1; j <= 3; j++) {
                if (i == 3 && j == 2) {

                    // Continue statement in inner loop to
                    // skip the execution when i==3 and j==2

                    continue;
                }

                // Print elements to showcase keyword affect
                System.out.println(i + " * " + j);
            }
        }
    }
}
```

Output:

```
1 * 1
1 * 2
1 * 3
2 * 1
2 * 2
2 * 3
3 * 1
3 * 3
4 * 1
4 * 2
4 * 3
```