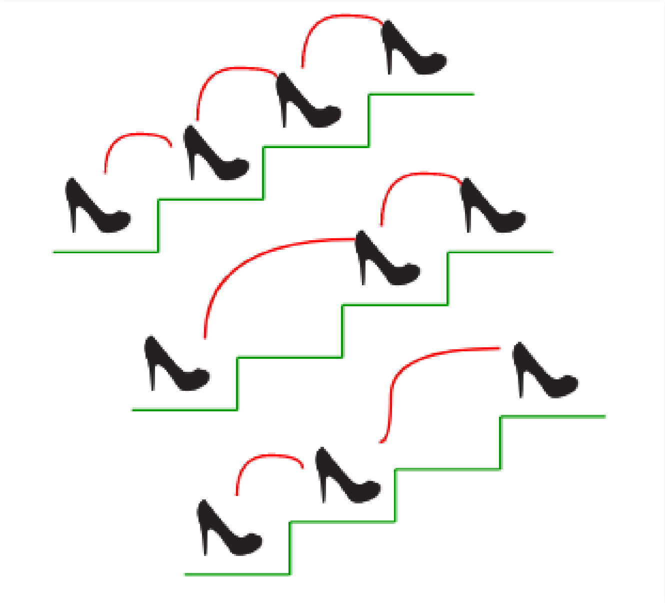
Usage of Break keyword in Java

Last Updated : 04 Dec, 2021

Break keyword is often used inside loops control structures and switch statements. It is used to terminate loops and switch statements in java. When the break keyword is encountered within a loop, the loop is immediately terminated and the program control goes to the next statement following the loop. When the break keyword is used in a nested loop, only the inner loop will get terminated. Even it is used with if statement to terminated when a certain condition is met.

The break keyword has special usage inside the switch statement. Every case in the switch statement is followed by a break keyword, such that whenever the program control jumps to the next case, it wouldn't execute subsequent cases.

Real-life Illustration:



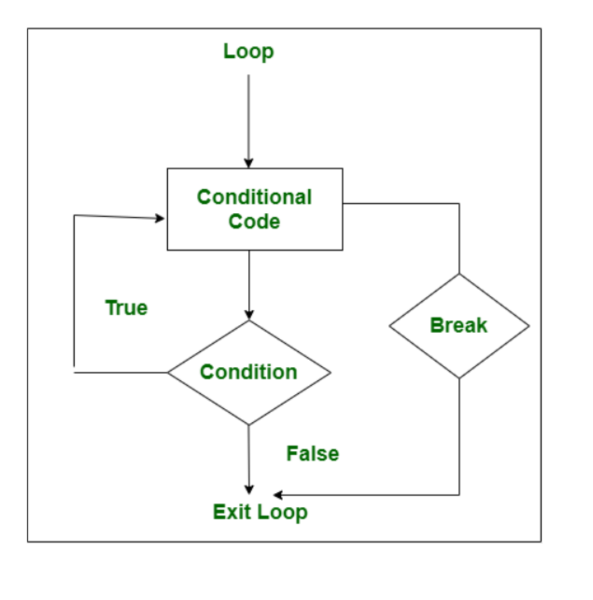
> *Consider a person climbing stairs. Now the red line here is depicting the stair on whi ch his/her shoe slips and he/she rolls backs to the base. Remember if he/she slips, he/she will always be landing on the base.*
>
> *Here in computers in programming language there is a keyword called 'break' which will abruptly stop the further execution of all the statements following it defined in th at scope.*

Syntax:

break keyword along with a semicolon

```
break;
```

Flow chart of Break statement



Use of Break Keyword is mostly found in loop concepts:

Types of loops	Usage of ‘break’ keyword
Simple loop	While the goal of insertion, delete, or updation is completed and there itself wants the program to terminate
Nested loop	When the user wants to take command from innermost loop to outer loop
Infinite loop	When the program enters into an infinite looping state

Case 1: Break keyword inside FOR loop

In order to show how to use the break keyword within For loop. Considering when the value of ‘i’ becomes 39, the break keyword plays its role and terminate the for a loop. All values of ‘i’ before 39, are displayed in the result.

```
// Java Program to show use of break statement
// inside the For loop
public class GFG {

    // Main driver code
    public static void main(String[] args)
    {
        // For loop for iteration
        for (int i = 3; i <= 50; i += 3) {
            if (i == 39) {
                // Using Break keyword to suspend
                // loop when i become 39
                break;
            }

            // Printing elements showcasing break usage
            System.out.print(i + " ");
        }
    }
}
```

Output:

```
3 6 9 12 15 18 21 24 27 30 33 36
```

Case 2: Break keyword inside WHILE loop

In order to show how to use the break keyword within the While loop. Considering when the value of ‘i’ becomes 15, the break keyword plays its role and terminate the for a loop. All values of ‘i’ greater than 15 till 35, are displayed in the result.

```
// Java Program to illustrate the use of break statement
// inside the While loop

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

```
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // While loop for iteration

        // Creating and initializing variable
        // outside loop
        int i = 35;

        // Condition check
        while (i >= 10) {
            if (i == 15) {
                // Using Break keyword to suspend
                // loop when i become 15

                // Decrementing variable initialized above
                i--;

                // Break statement
                break;
            }

            // Printing elements showcasing break
            System.out.print(i + " ");
            i--;
        }
    }
}
```

Output:

```
35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
```

Case 3: Break keyword inside DO-WHILE loop

In order to show how to use the break keyword within the Do-while loop. Considering when the value of ‘i’ becomes 80, the break keyword plays its role and terminate the for a loop. All values of i” before 80, are displayed in the result.

```
// Java Program to illustrate the use of break statement
// Inside the do-While loop
```

```java
// Importing all generic classes
import java.util.*;

// Importing specific class to take
// user input from the user
import java.util.Scanner;

public class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // Using do-While loop

        // Creating and initializing a variable
        // over which execution occurs
        int i = 0;

        // do loop consisting of executable statements
        do {
            if (i == 80) {

                // Incrementing variable by 5
                i += 5;

                // Using Break keyword to
                // suspend loop when i become 80
                break;
            }

            System.out.print(i + "  ");
            // Printing elements to show break usage

            i += 5;
        }
        // Condition check
        while (i <= 100);
    }
}
```

Output:

```
0  5  10  15  20  25  30  35  40  45  50  55  60  65  70  75
```

Case 4: Break keyword inside the Switch statement

In order to show how to use the break keyword within For loop. The break keyword is used at the bottom of every switch case to terminate each statement sequence and prevent the mixing of switch-case statements.

```java
// Java Program to illustrate the use of break statement
// inside the Switch Statement

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

public class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        // Declaring a variable for switch expression
        int numb = 20;

        // Switch expression
        switch (numb) {

        // Case statements
        case 10:
            System.out.println("TEN");

            // Break keyword to suspend the switch case
            // if given condition is fulfilled
            break;

        case 20:
            System.out.println("TWENTY");
            break;

        case 30:
            System.out.println("THIRTY");
            break;

            // Default case statement
        default:
            System.out.println("INFINITE");
        }
    }
}
```

Output:

```
TWENTY
```