Java Break 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.

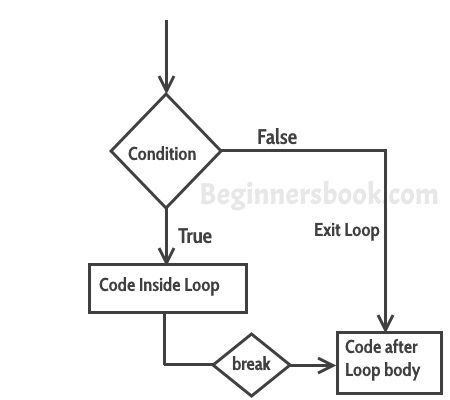
The Java *break* statement is used to break loop or [switch](https://www.javatpoint.com/java-switch) statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

We can use Java break statement in all types of loops such as [for loop](https://www.javatpoint.com/java-for-loop), [while loop](https://www.javatpoint.com/java-while-loop) and [do-while loop](https://www.javatpoint.com/java-do-while-loop).

Break statement is almost always used with decision-making statements ([Java if...else Statement](https://www.programiz.com/java-programming/if-else-statement)).

Here is the syntax of the break statement in Java:

break;

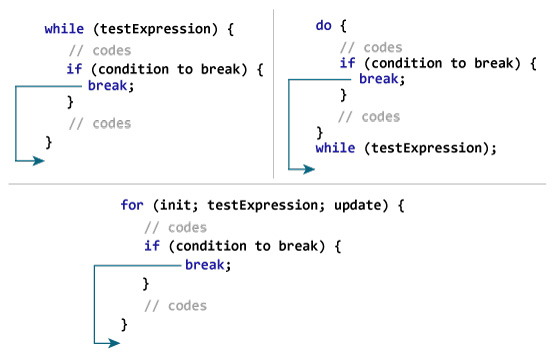


It is also used in [switch case](https://beginnersbook.com/2017/08/java-switch-case/) control. Generally all cases in switch case are followed by a break statement so that whenever the program control jumps to a case, it doesn’t execute subsequent cases . As soon as a break is encountered in switch-case block, the control comes out of the switch-case body.

In Java, a break statement is majorly used for:

* To exit a loop.
* Used as a “civilized” form of goto.( but goto is not used in java and C++)
* Terminate a sequence in a switch statement.

**How break statement works?**



## Example – Use of break in a for loop

The thing you can see here. As soon as the var value hits 99, the [for loop](https://beginnersbook.com/2015/03/for-loop-in-java-with-example/) gets terminated.

|  |  |  |
| --- | --- | --- |
| public class BreakExample2 {  public static void main(String args[]){ int var;  for (var =100; var>=10; var --)  {  System.out.println("var: "+var);  if (var==99)  {  break;  }  }  System.out.println("Out of for-loop");  }  } |  | **Output:**  var: 100  var: 99  Out of for-loop |

## Example – Use of break in a while loop

In the example below, we have a [while loop](https://beginnersbook.com/2015/03/while-loop-in-java-with-examples/) running from o to 100 but since we have a break statement that only occurs when the loop value reaches 2, the loop gets terminated and the control gets passed to the next statement in program after the loop body.

|  |  |  |
| --- | --- | --- |
| public class BreakExample1 {  public static void main(String args[]){  int num =0;  while(num<=100)  {  System.out.println("Value of variable is: "+num);  if (num==2)  {  break;  }  num++;  }  System.out.println("Out of while-loop");  }  } |  | **Output:**  Value of variable is: 0  Value of variable is: 1  Value of variable is: 2  Out of while-loop |

## Example – Use of break statement in switch-case

public class BreakExample3 {

public static void main(String args[]){

int num=2;

switch (num)

{

case 1:

System.out.println("Case 1 ");

break;

case 2:

System.out.println("Case 2 ");

break;

case 3:

System.out.println("Case 3 ");

break;

default:

System.out.println("Default ");

}

}

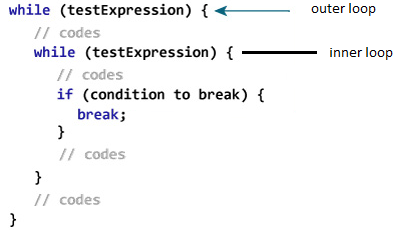
}

**Output:** Case 2

In this example, we have break statement after each Case block, this is because if we don’t have it then the subsequent case block would also execute. The output of the same program without break would be Case 2 Case 3 Default.

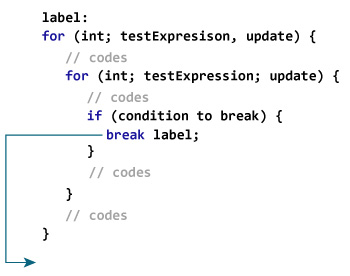
**Java break and Nested Loop**

In the case of [nested loops](https://www.programiz.com/java-programming/nested-loop), the break statement terminates the innermost loop.



Here, the break statement terminates the innermost while loop, and control jumps to the outer loop.

**Labeled break Statement**



Here, the break statement is terminating the labeled statement (i.e. outer loop). Then, the control of the program jumps to the statement after the labeled statement.

Java Continue Statement:

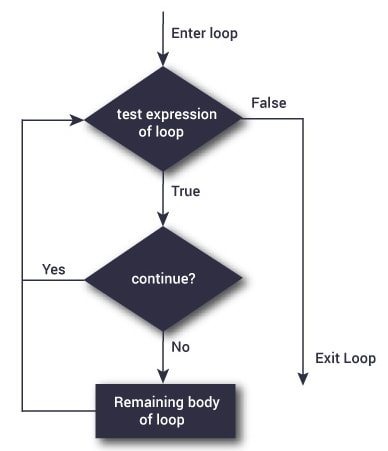
The **continue** keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop.

* 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.

Here's the syntax of the continue statement.

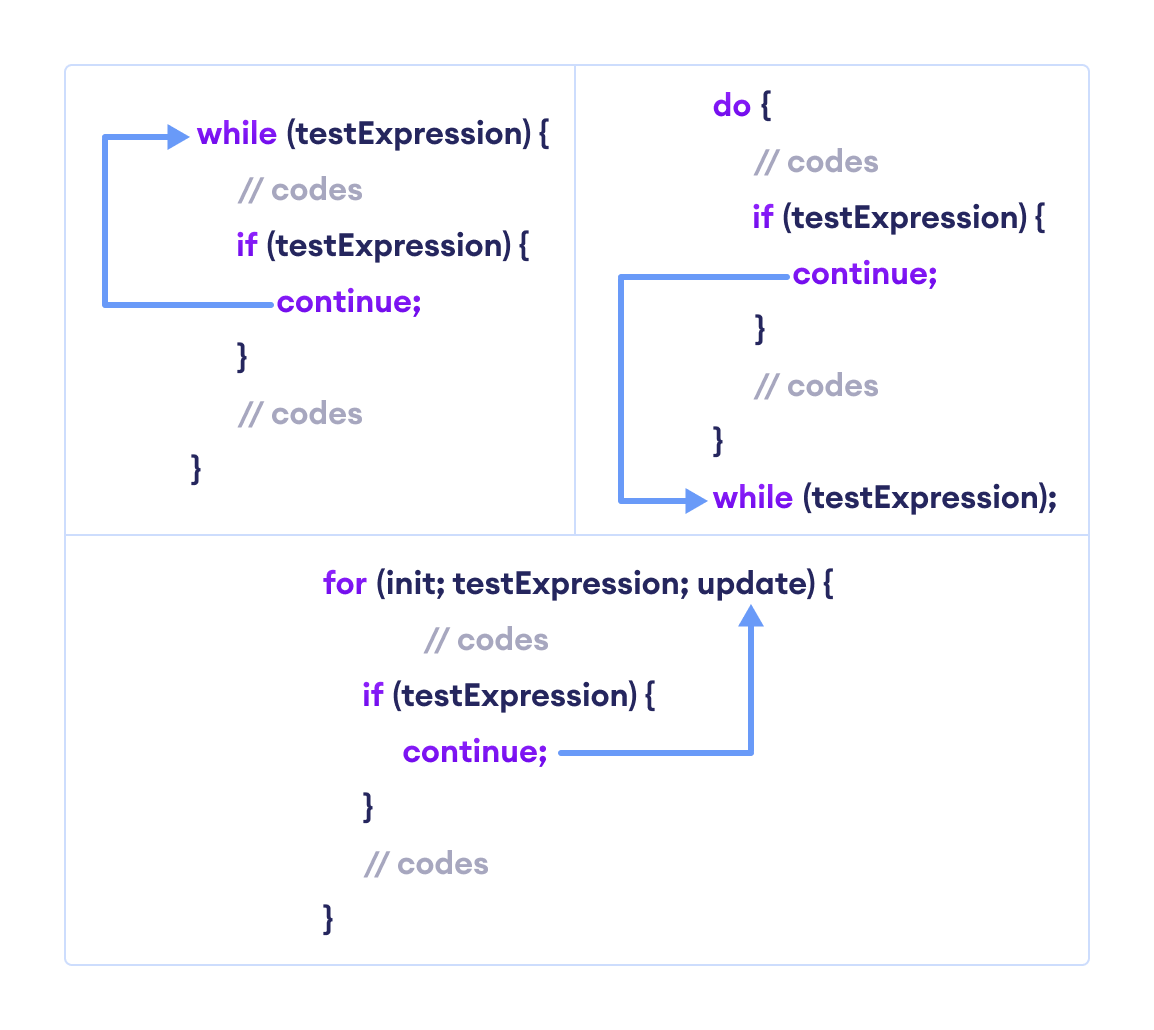
continue;

The continue statement is almost always used in decision-making statements ([if...else Statement](https://www.programiz.com/java-programming/if-else-statement)).



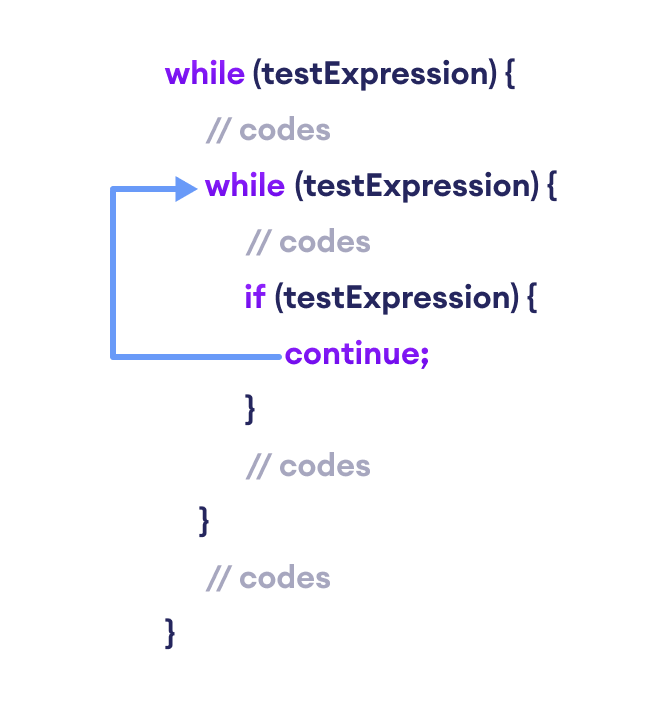
The continue statement is almost always used in decision-making statements ([if...else Statement](https://www.programiz.com/java-programming/if-else-statement)).

**Working of Java continue statement**



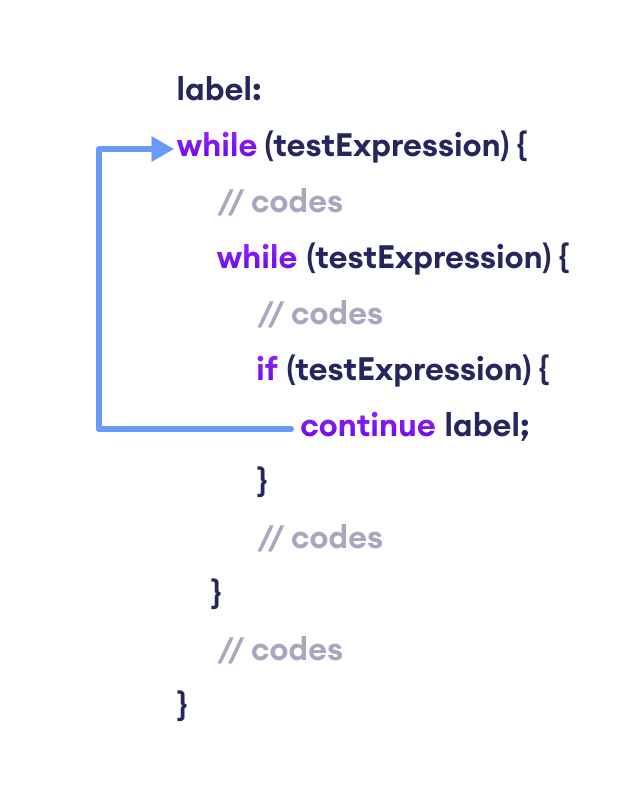
**Java continue with Nested Loop**

In the case of [nested loops in Java](https://www.programiz.com/java-programming/nested-loop), the continue statement skips the current iteration of the innermost loop.



**Labeled continue Statement**

Here, the continue statement skips the current iteration of the loop specified by label.



We can see that the label identifier specifies the outer loop. Notice the use of the continue inside the inner loop.

Here, the continue statement is skipping the current iteration of the labeled statement (i.e. outer loop). Then, the program control goes to the next iteration of the labeled statement.

Difference Between Break and Continue

