JUMP STATEMENT IN JAVA

Jumping statements are control statements that transfer execution control from one point to another point in the program. There are **three Jump statements** that are provided in the Java programming language:

1. **Break statement.**
2. **Continue statement.**
3. **Return Statement**

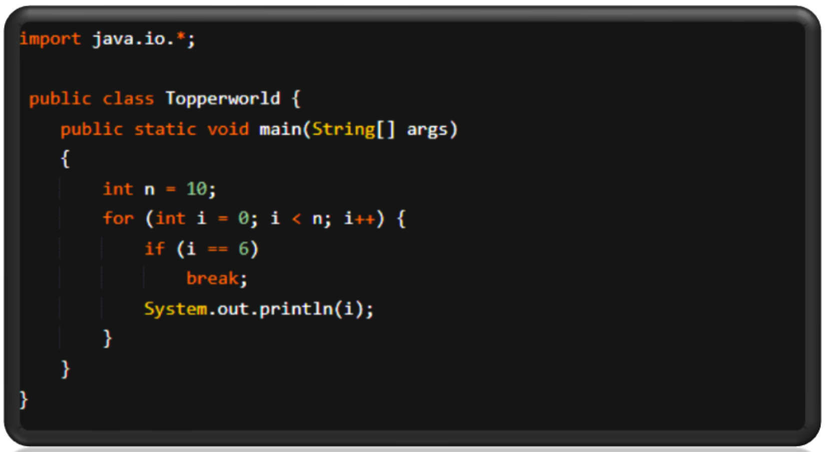
❖ Break statement

1. Using Break Statement to exit a loop:

In java, the break statement is used to terminate the execution of the nearest looping statement or switch statement. The break statement is widely used with the switch statement, for loop, while loop, do-while loop.

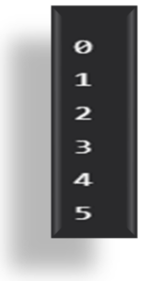
Syntax: break;

When a break statement is found inside a loop, the loop is terminated, and the control reaches the statement that follows the loop. **Here is an example:**



```
import java.io.*;

public class Topperworld {
    public static void main(String[] args)
    {
        int n = 10;
        for (int i = 0; i < n; i++) {
            if (i == 6)
                break;
            System.out.println(i);
        }
    }
}
```

Output:



```
0
1
2
3
4
5
```

As you see, the code is meant to print 1 to 10 numbers using for loop, but it prints only 1 to 5 . as soon as i is equal to 6, the control terminates the loop.

In a switch statement, if the break statement is missing, every case label is executed till the end of the switch.

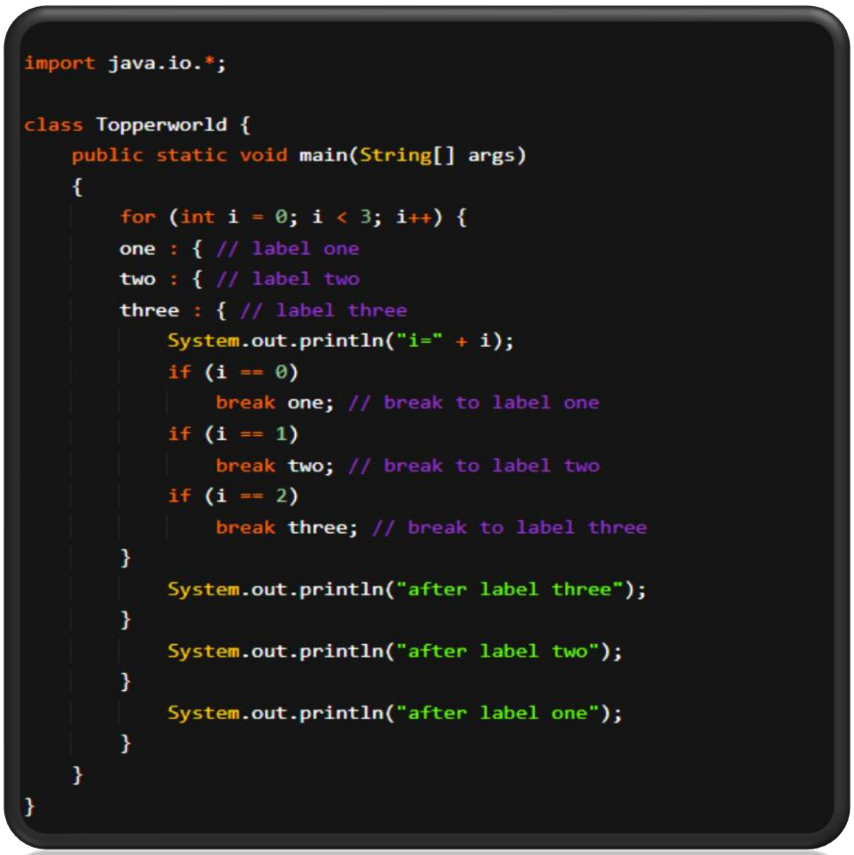
2. Use Break as a form of goto:

Java does not have a goto statement because it produces an unstructured way to alter the flow of program execution. Java illustrates an extended form of the break statement. This form of break works with the label. The label is the name of a label that identifies a statement or a block of code.

Syntax: `break label;`

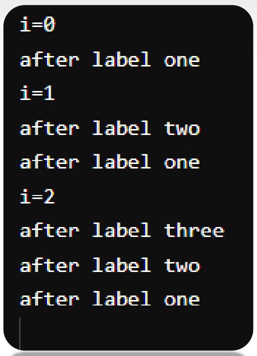
When this form of break executes, control jumps out of the labeled statement or block.

Here is an example:



```java
import java.io.*;

class Topperworld {
    public static void main(String[] args)
    {
        for (int i = 0; i < 3; i++) {
        one : { // label one
        two : { // label two
        three : { // label three
            System.out.println("i=" + i);
            if (i == 0)
                break one; // break to label one
            if (i == 1)
                break two; // break to label two
            if (i == 2)
                break three; // break to label three
        }
            System.out.println("after label three");
        }
            System.out.println("after label two");
        }
            System.out.println("after label one");
        }
    }
}
```

Output:



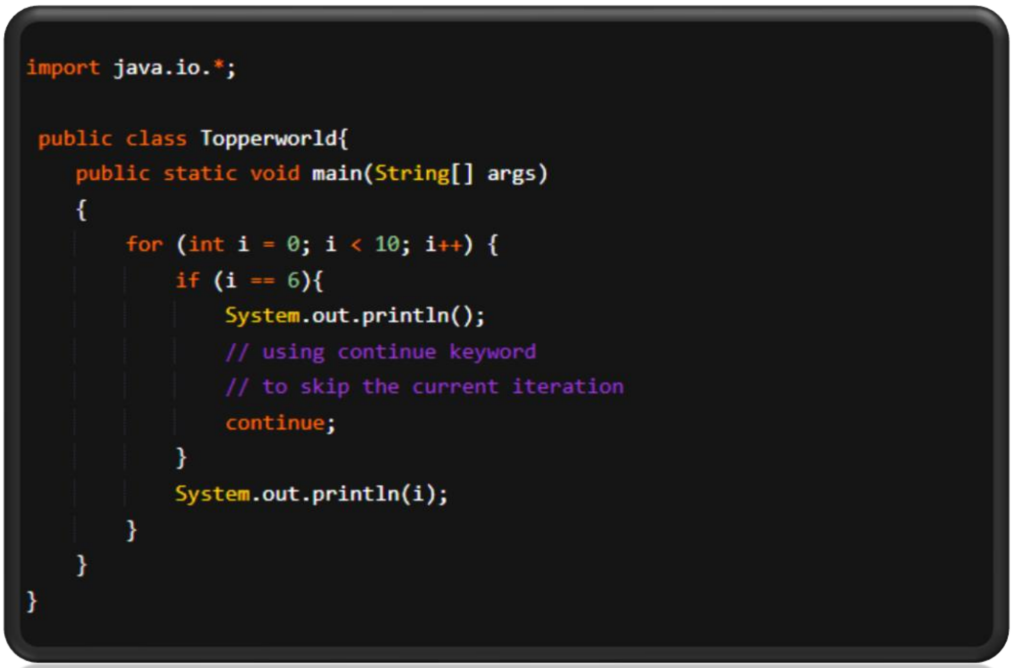
```
i=0
after label one
i=1
after label two
after label one
i=2
after label three
after label two
after label one
```

In the above program, when i=0, the first if statement succeeds, and cause a break to label one and then prints the statement. When i=1, the second if statement succeeds, and cause a break to label two and then prints the statements. When i=2, the third if statement succeeds, and cause a break to the to label three and then prints all the three statements.

❖ Continue Statement

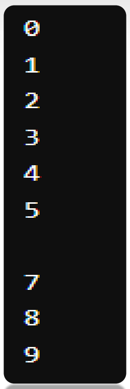
The continue statement pushes the next repetition of the loop to take place, hopping any code between itself and the conditional expression that controls the loop.

Here is an example:



```
import java.io.*;

 public class Topperworld{
    public static void main(String[] args)
    {
        for (int i = 0; i < 10; i++) {
            if (i == 6){
                System.out.println();
                // using continue keyword
                // to skip the current iteration
                continue;
            }
            System.out.println(i);
        }
    }
}
```

Output:



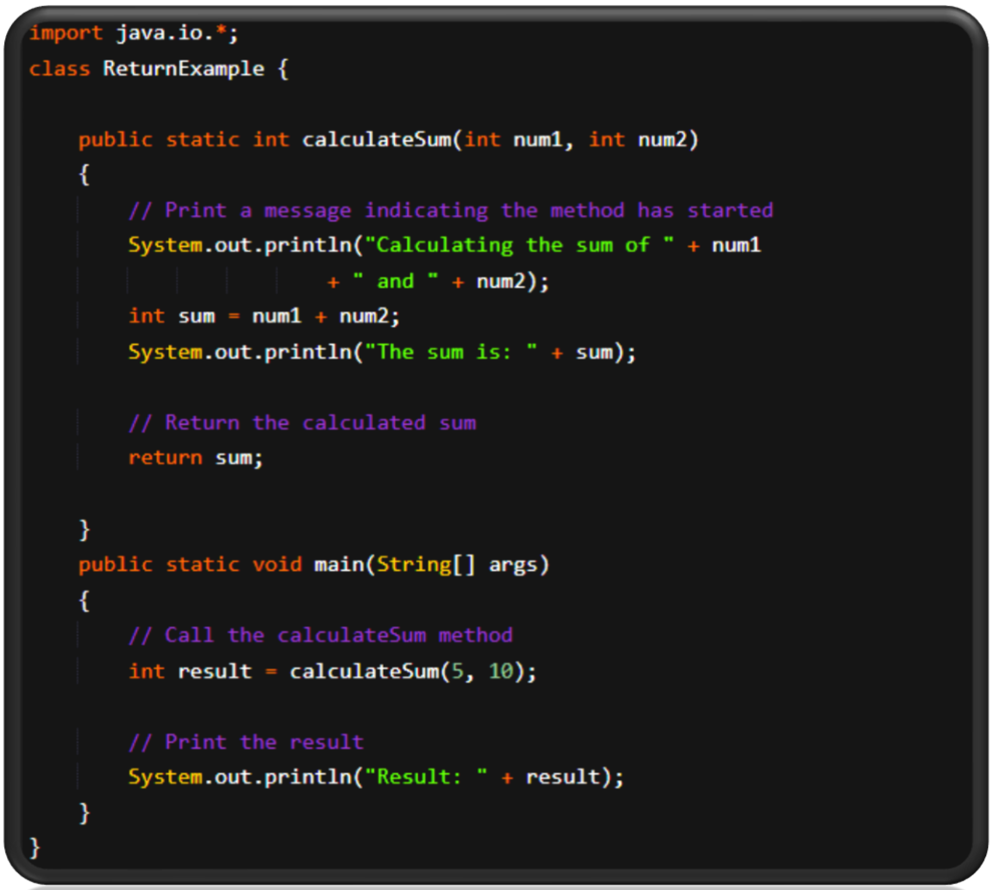
```
0
1
2
3
4
5

7
8
9
```

In the program, when the value of i is 6, the compiler encounters the continue statement, and then 6 is skipped.

❖ Return Statement

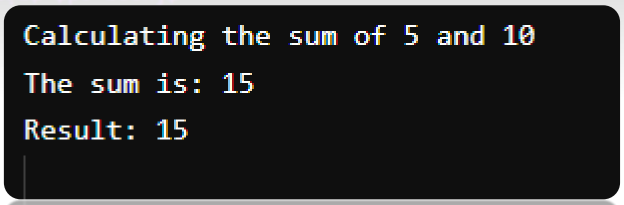
The "**return**" keyword can help you transfer control from one method to the method that called it. Since the control jumps from one part of the program to another, the return is also a jump statement.

- "return" is a reserved keyword means we can't use it as an identifier.
- It is used to exit from a method, with or without a value.



```java
import java.io.*;
class ReturnExample {

    public static int calculateSum(int num1, int num2)
    {
        // Print a message indicating the method has started
        System.out.println("Calculating the sum of " + num1
                        + " and " + num2);
        int sum = num1 + num2;
        System.out.println("The sum is: " + sum);

        // Return the calculated sum
        return sum;

    }
    public static void main(String[] args)
    {
        // Call the calculateSum method
        int result = calculateSum(5, 10);

        // Print the result
        System.out.println("Result: " + result);
    }
}
```

Output:



```
Calculating the sum of 5 and 10
The sum is: 15
Result: 15
```