

# Loops Using For

by Sophia



## WHAT'S COVERED

In this lesson, you will learn about patterns that generate for loops when coding an algorithm. This lesson focuses on the basic or traditional version of the for loop. Specifically, this lesson covers:

### Table of Contents

- [1. Reviewing the for Loop](#)
- [2. Uses of the for Loop](#)
  - [2a. Counting Certain Elements in an Array](#)
  - [2b. Computing Totals](#)

## 1. Reviewing the for Loop

You learned that a for loop is a repetitive control structure. A for loop allows use of a loop that is executed a specific number of times

If you recall, the structure of the for loop looks like this:

➞ EXAMPLE

```
for(<initialization>; <condition>; <increment/decrement>) {  
    <statement(s)>  
}
```



### HINT

In the example above, the <initialization>, <condition>, <increment/decrement> and <statement(s)> terms with outside arrows are just for information purposes; these are not keywords or actual code. These are just to explain what goes into each of these parts of a for loop.

The <initialization> may declare a numeric variable (int or long) that is used as the loop counter and sets its initial value. If the variable has been declared before the loop, the <initialization> just sets the initial value. If the variable is declared and initialized here, the variable is only accessible in the body of the loop. If the variable is declared before the loop, its value is accessible outside of the loop (as well as inside the body of

the loop).



#### CONCEPT TO KNOW

It is important to remember that this initialization happens only once at the start of the loop; the variable is initially declared and used to refer to each of the elements in the iterable object (such as a list).

The `<condition>` is a boolean expression (evaluating to true or false based on the result of comparison). The loop executes as long as this expression evaluates as true. The loop stops when this expression becomes false.

There is a semicolon after the `<initialization>` and the `<condition>`.

The `<increment/decrement>` is an expression that defines how the value of the loop variable is modified at the end of each iteration. The most common expression used is the loop variable followed by the **increment operator** (`++`), so the loop variable's value is incremented by one after each iteration. It is also possible to change the variable by other values or show an example of the variable, but incrementing by one is the most common.



#### CONCEPT TO KNOW

The syntax of a for loop is similar to the while loop in that there is a for loop statement and then the body of the loop.



#### TERM TO KNOW

##### Increment Operator ( `++` )

The `++` operator increases the value in the variable immediately to the left of the operator by one.

---

## 2. Uses of the for Loop

There are many scenarios where the for loop is valuable when programming in Java. These include, but are not limited to:

- Counting elements within a list,
- Computing totals based on a list, or
- Finding the largest and smallest elements within the list.

Any time that each element in an array needs to be reviewed, it is a perfect time to use a for loop.

### 2a. Counting Certain Elements in an Array

To count the number of even values in an array, write the following code saved in a file named `ForCountEven.java`.

Note the setup of the for loop:

```
import java.util.Arrays;
```

```
class ForCountEven {
```

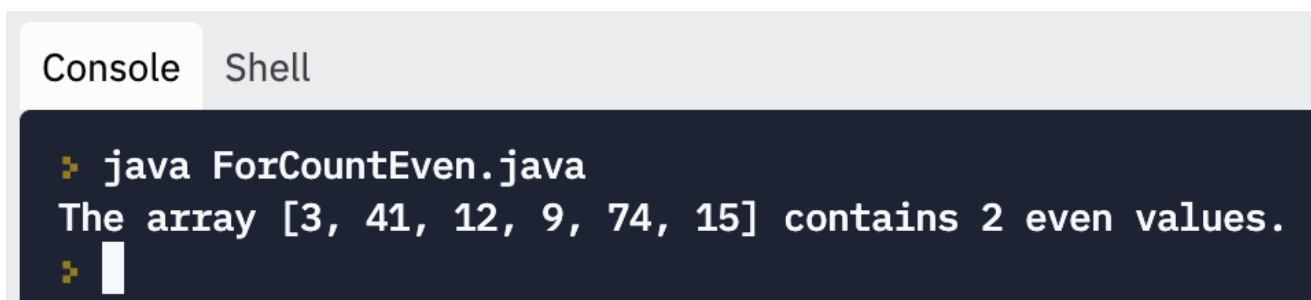
```

public static void main(String[] args) {
    int[] numbers = {3, 41, 12, 9, 74, 15};
    // Counter initialized to 0
    int evenCount = 0;

    // Loop variable starts at 0, since first element index in array is 0.
    // Remember the last index is 1 less than length of array.
    for(int index = 0; index < numbers.length; index++) {
        if(numbers[index] % 2 == 0) {
            evenCount++;
        }
    }
    // Display result
    System.out.print("The array " + Arrays.toString(numbers) + " ");
    System.out.println("contains " + evenCount + " even values.");
}
}

```

Running this program should produce the following output:



```

Console  Shell
❯ java ForCountEven.java
The array [3, 41, 12, 9, 74, 15] contains 2 even values.
❯

```

## 2b. Computing Totals

For loops are also useful in adding up the values in an array. This can be an important step in carrying more complex calculations, such as finding an average.

Another similar loop that computes the total of an array of numbers is as follows:

```

import java.util.Arrays;

class ForSum {
    public static void main(String[] args) {
        int[] numbers = {3, 41, 12, 9, 74, 15};
        // Sum initialized to 0
        int sum = 0;
        // Loop variable starts at 0, since the first element index in array is 0.
        // Remember that last index is 1 less than length of array.
        for(int index = 0; index < numbers.length; index++) {
            sum += numbers[index];
        }
    }
}

```

```
// Display result
System.out.print("The sum of the values in the array ");
System.out.print(Arrays.toString(numbers));
System.out.println(" = " + sum + ".");
}
}
```

The output from this program should look like this:

```
java ForSum.java
The sum of the values in the array [3, 41, 12, 9, 74, 15] = 154.
```



#### BIG IDEA

The variable `sum` was set to zero before the loop starts. In this loop, the iteration variable was used in the body of the loop to access each item in the array in turn and add it cumulatively to the `sum` variable. As the loop executes, the `sum` variable accumulates the sum of the elements; a variable used this way is sometimes called an **accumulator**.

Similar to the `while` loop, the `break` and `continue` statements can be used in `for` loops, and they will work in the same way. The `break` statement will immediately end the `for` loop and execute the first statement after the end of the loop. The `continue` statement will end the current iteration of the loop and go back to the start of the loop again to evaluate the expression's condition.



#### SUMMARY

In this lesson, you reviewed the **for loop and its uses** in more detail. This included its use as a **definite loop**. You used the `for` loop to both count and total elements of a list. You learned how a `for` loop can be used to **count the largest or smallest element in an array** as well. You learned that a `for` loop can also use the `continue` and `break` statements to change the flow of the loop. Finally, you learned about **computing totals**.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source [cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf](https://cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf)

It has also been adapted from “Python for Everybody” By Dr. Charles R. Severance. Source [py4e.com/html3/](https://py4e.com/html3/)



#### TERMS TO KNOW

##### Increment Operator ( ++ )

The `++` operator increases the value in the variable immediately to the left of the operator by one.

