Java For Loop

When you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop:

Syntax

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
```

Example

```
for (int i = 0; i <= 10; i = i + 2) {
  System.out.println(i);

}
```

Nested Loops

It is also possible to place a loop inside another loop. This is called a nested loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Example

```
// Outer loop
for (int i = 1; i <= 2; i++) {
  System.out.println("Outer: " + i); // Executes 2 times

  // Inner loop
```

```
    for (int j = 1; j <= 3; j++) {
      System.out.println(" Inner: " + j); // Executes 6 times (2 * 3)
    }


}
```

Example code 1: Initialization part removed –

```
public static void main(String[] args) {
int i = 0;
for( ; i < 3; i++) {
System.out.println(i);

}
}

//Output:
0
1
2
```

Example code 2: Updation part removed

```
public static void main(String[] args) {
for(int i = 0; i < 3; ) {
System.out.println(i);
i++;
}
}
//Output:
0
1
2
```

Java Break

You have already seen the `break` statement used in an earlier chapter of this tutorial. It was used to "jump out" of a `switch` statement.

The `break` statement can also be used to jump out of a loop.

This example stops the loop when i is equal to 4:

Example

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);


}
```

Java Continue

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

Example

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);


}
```

Break Example

```
int i = 0;
while (i < 10) {
  System.out.println(i);
  i++;
  if (i == 4) {
    break;
  }
}
```

Continue Example

```
int i = 0;
while (i < 10) {
  if (i == 4) {
    i++;
    continue;
  }
  System.out.println(i);
  i++;
}
```