Computer Programing in Java

Lecture 7
Java Loops

م.م حسين عجام

Loops

- Loops can execute a block of code as long as a specified condition is reached.

- Loops are handy because they save time, reduce errors, and they make code more readable.

Java While Loop

- The while loop loops through a block of code as long as a specified condition is true:

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

- In the example below, the code in the loop will run, over and over again, as long as a variable (i) is less than 5:

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

The Do/While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

Example

- The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

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
}
```

- **Statement 1** is executed (one time) before the execution of the code block.
- **Statement 2** defines the condition for executing the code block.
- **Statement 3** is executed (every time) after the code block has been executed.

Example

- The example below will print the numbers 0 to 4:

```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

Example explained

- Statement 1 sets a variable before the loop starts (int i = 0).
- Statement 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.
- Statement 3 increases a value (i++) each time the code block in the loop has been executed.

Example

- This example will only print even values between 0 and 10:

```
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)
  }
}
```

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:

```
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:

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

Break and Continue in While Loop

- You can also use break and continue in while loops:
- Break Example

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

Continue

- Continue Example

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