Scope

What is Scope

Scope describes the fact that **variables only exist in the block in which they are declared**. For example:

```
1
2  //<---- blank line ---->
3  for (int i = 0; i < array.length; i++) {
4    if (array[i] == 5)
5      break;
6  }
   System.out.println(i);
```

The above code will give an error, because **the variable i only exists in the for loop and is invisible for everyone else**. The right way to do this would be like:

```
1  int i;
2  for (i = 0; i < array.length; i++) {
3    if (array[i] == 5)
4      break;
5  }
6  System.out.println(i);
```

This would successfully print `i` because **the variable was initialized inside the block which* `println()` *is used (It was declared in line one in our second example, as opposed to line two in our first example, where it was declared *inside* the for loop.

Variables with Same Names

Because of scope, we can have **two variables with the same name** in one program:

```
1  public class Cat {
2
3    private int numLegs;
4
5    public void setNumLegs(int numLegs) {
6      this.numLegs = numLegs;
7    } //End setNumLegs()
8
9  } //End Cat()
```

In this example, we have two variables `numLegs`:

1. numLegs variable **in the Cat()** class, the instance variable. This variable was declared on line 3
2. numLegs variable **in the setNumLegs()** method, which exists in that method only. This variable was declared on line 5.

The instance variable in number one only exists in the `Cat()` class. The method variable in number two only exists in `setNumLegs()`. They are two different variables. However, we can use the `Cat()` instance variables in any methods by using `this.variable` as done in line 6.