

Variable Scope - Revisited

We discussed variable scope earlier in our study of the `while`, `do/while`, and `for` loops. Now that we are defining and using custom methods, a few more points are warranted. The general rule is that a variable has scope, or visibility, from the point where it is declared to the end of the block where it is declared, and examples were given in Chapter 3.

When defining a method, however, it sometimes happens that a programmer declares a variable and picks a name for it that matches a variable name in another method. It is also possible that one of these methods calls the other. When this happens, the variables are treated as separate variables. Figure 1 shows a code segment including a `main()` method and a method called `testMethod()`. Both methods include the declaration of `int` variable named `i`. In each case, the scope of the variable is shown in a thick line.

```
public static void main(String[] args)
{
    int i = 123;
    System.out.println(i); // print 1
    testMethod();
    System.out.println(i); // print 3
}

public static void testMethod()
{
    int i = 456;
    System.out.println(i); // print 2
    return;
}
```

The diagram illustrates the scope of the variable `i` in two methods. In the `main()` method, the variable `i` is declared and assigned the value 123. Its scope is indicated by a thick line that starts at the declaration and ends at the closing brace of the method. In the `testMethod()` method, the variable `i` is declared and assigned the value 456. Its scope is also indicated by a thick line that starts at the declaration and ends at the closing brace of the method. A box labeled "Different variables" has two arrows pointing to the `i` labels in both methods, emphasizing that these are two distinct variables.

Figure 1. Variable scope revisited

Within the `main()` method, `testMethod()` is called. A variable `i` is declared and assigned 456 in `testMethod()`; however, this has no influence on the variable `i` in `main()`. Three print statements appear and their sequence of execution is numbered. The output generated is

```
123
456
123
```

The third line above emphasizes that the assignment of the variable `i` in `testMethod()` did not change the content of the variable with the same name in `main()`.