I decided to have a little bit of fun with this program. I made each of the four methods return an integer, and three of the four accept an argument for an offset. Each of the four methods - method1(), method2(), method3(), and method4() - calculate a random number and return it. Methods 2, 3, and 4 can take and integer argument, which gets added to the computed random number between 0 and 30. I designed the program this way so I could link these four methods together in a single line of code. In my main method (driver), I create two integer variables, “a” and “b”. “a” is computed with the following syntax.

```
int a = method2(method3(method4(method1())));
```

The methods appear in the code in the following order, from left to right: Method 2, method 3, method 4, method 1. However, it is method1() that gets called and executed first, so its return value can be passed as the argument for method4(). Method 3 does not get called until methods 1 and 4 have started and finished. And method 2 does not get called until method 3 has finished. “b” is computed with the following.

```
int b = method4(method3(method2(method1())));
```

Like “a”, the innermost method is method1(), so this method gets called first. The methods are called and executed in the order of their names, in ascending order, which is to say, the reverse of how they appear in the code, in descending order. Method 2 is not called until method 1 has finished, and method 3 doesn’t start until method 2 has finished, and so on. The output of this program looks like this.

```
In main method

method2() -> method3() -> method4() -> method1()
In method1()
Exiting method1() with value 11
In method4()
Got offset 11
Exiting method4() with value 66
In method3()
Got offset 66
Exiting method3() with value 154
In method2()
Got offset 154
Exiting method2() with value 165
A = 165
```