Ch. 6

More practice with methods

```
import java.util.Scanner;
// What is printed by the following program?
public class Test{
   public static void main(String[] args){
      Scanner keyboard = new Scanner(System.in);
      int a = keyboard.nextInt();
      int b = keyboard.nextInt();
      System.out.println(mystery(a, b));
   }

   public static int mystery(int x, int y){
      int result = 1;
      for(int i = 1; i <= y; i++){
         result = result * x;
      }
      return result;
   }
}
```

The user enters: 2 3
The user enters: 3 4

```
// Complete the method below so that it prints out
// what is in the green box.

public class Test{
   public static void main(String[] args){
      int num = 4;
      for(int i = 1; i <= num; i++){
         mystery(i, num);
      }
   }

   public static void mystery(int x, int y){
      // your code goes here

   }
}
```

Output:
4
43
432
4321

```
// What is printed by the following program?

public class Test{
   public static void main(String[] args){
      System.out.println(mystery(5));
   }

   public static int mystery(int x){
      int result = 1;
      for(int i = 1; i <= x; i++){
         result = result * i;
      }
      return result;
   }
}
```

```
public class MethodCalls{
   public static void main(String[] args){
      System.out.println(func1(10));
   }
   public static int func1(int x){
      System.out.println("func1: " + x);
      int y = func2(2 * x);
      return y;
   }
   public static int func2(int a){
      System.out.println("func2: " + a);
      return func3((a + 3));
   }
   public static int func3(int x){
      System.out.println("func3: " + x);
      return (x + 1);
   }
}
```

What is printed by this program?

```
public class MethodCalls{
   public static void main(String[] args) {
      func1(func2(func3(5)));
   }
   public static void func1(int x){
      System.out.println("func1 " + x);
   }
   public static int func2(int x){
      System.out.println("func2 " + x);
      return 2*x;
   }
   public static int func3(int x){
      int y = func2(x);
      System.out.println("func3 " + x);
      return y;
   }
}
```

What is printed by this program?

Method Overloading

- Two methods with the same signature, but different parameters

```
public static int sum(int x, int y){
   return x + y;
}
public static int sum(int x, int y, int z){
   return x + y + z;
}
public static int sum(int x, double y){
   return x + (int)y;
}
```

```
// What is printed by this program?
public class OverloadTest{
   public static void main(String[] args){
      int x = weirdSum(2, 3, 4);
      int y = weirdSum(5, 6);
      int z = weirdSum(6, 7);
      System.out.println(weirdSum(x, y, z));
   }

   public static int weirdSum(int a, int b){
      return (a + b + 5);
   }

   public static int weirdSum(int a, int b, int c){
      return (a + b + c + 10);
   }
}
```

```
// What is printed by this program?
public class OverloadTest2{
   public static void main(String[] args){
      int x = weirdSum(1, 2, 3);
      int y = weirdSum(3, 4);
      int z = weirdSum(4, 5);
      System.out.println(weirdSum(x, y, z));
   }

   public static int weirdSum(int a, int b){
      return weirdSum(a, b, 3);
   }

   public static int weirdSum(int a, int b, int c){
      return (a + b + c + 10);
   }
}
```

Recall: conversion

- **Type casting** occurs when we force one data to be of another type of data
  - Ex:
```
int twoPi = (int)(Math.PI*2);
    // twoPi has a value of 6
```
- **Implicit conversion** occurs when going from lower data types to higher data types
  - Ex:
```
double x = twoPi;
  // x has a value of 6.0
```

```
// What if we call a method using implicit conversion?
public class Conversion{
   public static void main(String[] args){
      int y = 3;
      double x = func(y);
   }

   public static double func(double a){
      return 2*a;
   }
}
```

```
// When method overloading doesn't work
public class Conversion2{
   public static void main(String[] args){
      System.out.println(sum(2.0, 3));
      System.out.println(sum(2, 3.0));
      System.out.println(sum(2, 3));
   }

   public static double sum(double a, int b){
      return a + b;
   }

   public static double sum(int a, double b){
      return a + b;
   }

}
```

Compiler error. Since both the 2 and 3 could be converted to double, the call is ambiguous.

Using String with Scanner

```
// This program simply prints all words of a string on
// their own line.
import java.util.Scanner;
public class StringScanner{
   public static void main(String[] args){
      String str = "This string has five tokens.";
      Scanner strScanner = new Scanner(str);
      while(strScanner.hasNext()){
         System.out.println(strScanner.next());
      }
   }
}
```

```
// This program simply prints all words of a string on
// their own line.
import java.util.Scanner;
public class StringScanner{
   public static void main(String[] args){
      String str = "1 2 5 7 9";
      Scanner strScanner = new Scanner(str);
      while(strScanner.hasNext()){
         System.out.println(square(strScanner.nextInt()));
      }
   }

   public static int square(int x){
      return x*x;
   }
}
```

Global Variables

- A variable that has scope throughout the entire program
- 2 types in Java:
  - Static global variables − the one we'll be focused on
  - Instance variables − we'll get to this in ch. 7

```
import java.util.Scanner;
public class GlobalDemo{
   static int sqr;
   public static void main(String[] args){
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Enter the num to be squared");
      sqr = keyboard.nextInt();
   }

   public static void square(){
      sqr = sqr*sqr;
   }
}
```

```
// What if we want a global Scanner? No problem.
import java.util.Scanner;
public class GlobalDemo{
   static Scanner keyboard;
   public static void main(String[] args){
      keyboard = new Scanner(System.in);
      System.out.println("Enter your first name");
      String nm = keyboard.next();
      goodDay(nm);
   }

   public static void goodDay(String n){
      System.out.println("Enter the day of the week.");
      String day = keyboard.next();
      System.out.println("Happy " + day + ", " + n);
   }
}
```

Unit Testing

- The process of individually testing a small part or unit of a program, typically a method
  - Testbench − A program used to test correctness of another program
  - Test vector − set of values used to unit test

```
// This program demos the use of Unit Tests – using
// assert.
public class UnitTest{
    public static void main(String[] args){
        System.out.println("Testing started.");
        assert(square(3) == 9) : "Assertion square(3) failed";
        assert(square(4) == 16) : "Assertion square(4) failed";
        assert(square(5) == 25) : "Assertion square(5) failed";
        System.out.println("Testing complete.");
    }

    public static int square(int x){
        return x*x;
    }
}
```

To use assert, the program must be run at the command line. Use option -ea

```
// This program demos the fibonacci sequence
public class Test{
    public static void main(String[] args){
        System.out.println(fib(8));
    }

    public static int fib(int x){
        int num = 0;
        int num2 = 1;
        int fibonacci;
        for (int i = 0; i < x; i++)
        {
            fibonacci = num + num2;
            num = num2;
            num2 = fibonacci;
        }
        return num;
    }
}
```

```
// How many times does the following snippet print "testing"?
int i = 1;
if (i < 20 ) {
    while (i <= 40) {
        System.out.println("testing");
        i = i + 1;
    }
}
```