Eric Roberts
CS 106A

Handout #22
April 18, 2012

Parameter Problem Set

Due: Wednesday, April 25, 5:00 P.M. There are no late days on this problem set.

Name (*please print*) _____

Section Leader _____

Download the `ParameterProblemSet` starter folder for the assignment directory, update the Word document with your answers, open the dummy project in Eclipse, and then submit your answer just as you would for any other assignment.

This problem set is designed to make sure that you understand how parameter passing works, which does not always come across in the assignments. This problem set is intended to be a pencil-and-paper exercise rather than a set of programming problems, although you are free to use the computer to check your work. Problems of this sort, however, will definitely appear on the midterm and final exam, so it is important to make sure that you can solve them without electronic assistance.

Special note: The best entries in each of the short answer questions will qualify for tickets in the final drawing for a special grand prize.

1. True/False questions

For each of the following statements below, indicate whether it is true or false in Java. None of these are in any sense "trick questions" for which the answer depends on thinking up some special case. If you understand what's going on with methods, the answers will be entirely straightforward.

1a) The value of a local variable named `i` has no direct relationship with that of a variable named `i` in its caller. _____

1b) The value of a parameter named `x` has no direct relationship with that of a variable named `x` in its caller. _____

1c) Assigning a new value to a parameter changes the value of the corresponding variable in the caller's argument list. _____

1d) Local variables of type `int` are automatically initialized to 0 when a method is called. _____

1e) Predicate methods always return a value of the primitive type `boolean`. _____

2. Short answer

Many beginning students find it hard to understand the relationships among variables in the different methods that make up a program. Often, students tell me that it would be easier if a particular variable name, such as `total` or `i`, always contained the same value in any part of the program in which that variable appeared. In your own words, describe what would be wrong with that approach?

Answer to question 2

3. Short answer

Suppose that you need to explain the concept of arguments and parameters to a sibling (or, more likely, a parent) with no background in programming at all. What everyday, real-world example could you use to make it clear why a procedure (or any generalized notion of a set of steps for carrying out a particular task) might want to include some form of parameterization?

Answer to question 3

4. Short answer

In a few sentences, describe the difference between local variables and instance variables. Make sure that your answer includes any differences in (a) how those variables are declared and (b) how long the values of those variables persist.

Answer to question 4

5. Tracing method execution

For each of the programs that follow, show what output is produced by the program when it runs. Each of these problems is taken from a past midterm exam.

5a)

```
/*
 * File: Mystery.java
 * ------------------
 * This program doesn't do anything useful and exists only to test
 * your understanding of method calls and parameter passing.
 */

import acm.program.*;

public class Mystery extends ConsoleProgram {

   public void run() {
      ghost(13);
   }

   private void ghost(int x) {
      int y = 0;
      for (int i = 1; i < x; i *= 2) {
         y = witch(y, skeleton(x, i));
      }
      println("ghost: x = " + x + ", y = " + y);
   }

   private int witch(int x, int y) {
      x = 10 * x + y;
      println("witch: x = " + x + ", y = " + y);
      return x;
   }

   private int skeleton(int x, int y) {
      return x / y % 2;
   }
}
```

Answer to question 5a

5b)

```
/*
 * File: Hogwarts.java
 * -------------------
 * This program doesn't do anything useful and exists only to test
 * your understanding of method calls and parameter passing.
 */

import acm.program.*;

public class Hogwarts extends ConsoleProgram {

   public void run() {
      bludger(2001);
   }

   private void bludger(int y) {
      int x = y / 1000;
      int z = (x + y);
      x = quaffle(z, y);
      println("bludger: x = " + x + ", y = " + y + ", z = " + z);
   }

   private int quaffle(int x, int y) {
      int z = snitch(x + y, y);
      y /= z;
      println("quaffle: x = " + x + ", y = " + y + ", z = " + z);
      return z;
   }

   private int snitch(int x, int y) {
      y = x / (x % 10);
      println("snitch: x = " + x + ", y = " + y);
      return y;
   }
}
```

Answer to question 5b

5c)

```
/*
 * File: Valentine.java
 * --------------------
 * This program doesn't do anything useful and exists only to test
 * your understanding of method calls and parameter passing.
 */

import acm.program.*;

public class Valentine extends ConsoleProgram {

   public void run() {
      str = "OX";
      int n = 5;
      println(heart(valentine(n, str), n));
   }

   private String valentine(int n, String word) {
      String str = "";
      while (n > 0) {
         str += word;
         n--;
      }
      println("valentine: str = " + str + ", n = " + n);
      return str;
   }

   private String heart(String word, int n) {
      println("heart: str = " + str + ", n = " + n);
      return word.substring(0, n);
   }

   private String str;
}
```

Answer to question 5c