CS 101 Spring 2006 Midterm 3 Name: _____ Email ID: _____

This exam is open text book but closed-notes, closed-calculator, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge here:

_____

_____

_____

_____

Page 1	____ / 4
Page 2	____ / 10
Page 3	____ / 17
Page 4	____ / 14
Page 5	____ / 15
Page 6	____ / 10
Page 7	____ / 12
Page 8	____ / 18
Total	**____ / 100**

Note: When an integer type is required use `int`*, when a floating-point type is required use* `double`*. If we don't specify an aspect of the problem (such as what the method or parameter names should be), you can choose it.*

1. (4 points) What section are you in?

____ CS 101-E

____ CS 101-3 (lab 8-9:15 a.m. Thu)

____ CS 101-4 (lab 9:30-10:45 a.m. Thu)

____ CS 101-5 (lab 11-12:15 a.m. Thu)

____ CS 101-6 (lab 12:30-1:45 p.m. Thu)

____ CS 101-7 (lab 2-3:15 p.m. Thu)

____ CS 101-8 (lab 3:30-4:45 p.m. Thu)

____ CS 101-9 (lab 5-6:15 p.m. Thu)

____ CS 101-10 (lab 6:30-7:45 p.m. Thu)

____ CS 101-11 (lab 8-9:15 p.m. Thu)

___ / 4

2. (5 points) What is the best reason for defining mutators for instance variables as opposed to declaring the instances variables to be `public`? (in about 25 words or less).

3. (5 points) Under what condition(s) should something be declared to be a class variable as opposed to an instance variable? (in about 25 words or less).

___ / 10

4. (12 points) Given the following code in the body of *main()*:

```
String a = "B";
String [] b;
String [] c = { "A", "B", "C", "D" };
String [] d = new String [3];
d[1] = "E";
String [] f = d.clone();
String [] g = f;
b = f;
```

Draw a memory diagram after the above code has executed. You may use the shorthand representation for the objects, but you must show ALL references.

5. (5 points) Use the code in the above question to evaluate the following expressions and give the value, or write "causes an error":

   a) `b[0]` _____

   b) `c[2] > c[4]` _____

   c) `c[2][0] > c[4][0]` _____

   d) `f[f.length - 1]` _____

   e) `a == c[1]` _____

___ / 17

6. (9 points) Give the definition of the class *Telephone*. The class has no constructors, one instance variable of type `String` called *number*, and two `static` variables. One is of type `int` called *quantity*; the other is of type `double` called *total*. Besides that, the class has one `static` method *makeFullNumber()*. The method accepts two arguments, a `String` containing a telephone number, and an `int` containing an area code. The method concatenates the two arguments in the following manner: First comes the area code, then a dash, then the telephone number. The method returns the resultant string.

7. (5 points) In **one** statement, declare an array named *a* of ten elements of type `int`, and initialize the elements (starting with the first) to the values 10, 20, 30, … 100, respectively.

___ / 14

8. (15 points) What is the output of the following code?

```
public class ParameterFun {

   private static double x = -1, y = -2;

   public static double funcOne(double x, double y) {
        x = x + y;
        return x;
   }

   public static double funcTwo(double a, double b) {
        x = x + y;
        return x;
   }

   public static double funcThree(double a, double b) {
        a = a + b;
        return a;
   }

   public static void main(String[] args) {
        double a = 1, b = 2;

        b = funcOne(a,b);
        System.out.println("a,b: " + a + "," + b);
        b = funcTwo(a,b);
        System.out.println("a,b: " + a + "," + b);
        b = funcThree(a,b);
        System.out.println("a,b: " + a + "," + b);
        b = funcTwo(a,b);
        System.out.println("a,b: " + a + "," + b);

   }
}
```


___ / 15

9. (10 points) Write a method called *avgLength()* that takes a one-dimensional `String` array as a parameter, and returns the average of all the lengths of the `String` elements as a `double`. Note: elements with a `null` value should not count toward the average.

___ / 10

10. (12 points) Write a method called *find100Sum()* that will search a one-dimensional `int` array (passed as a parameter) to see if any ***three consecutive members*** of the array add up to 100. It returns the index of the first member of a three-member sequence that sums to 100 if it finds such a sequence. If no such sequence is found, -1 should be returned.

___ / 12

11. (18 points) Write a method called *findDistributionOfScores()* that takes a **two-dimensional** `int` array (passed as a parameter called *testScoresBySection)* that contains multiple sections of student scores on a given exam. The method returns an array (we'll call it *totalNumScores*) containing the number of students across all sections who received the particular scores. For example, if your *totalNumScores[100] = 29*, then it signifies that 29 students from all of the sections received a perfect score of 100. You should assume that *testScoresBySection* might be a jagged array, and that each position in *testScoresBySection* contains an integer between -1 and 100 (-1 signifies that the person did not take the exam).

___ / 18