Name: ______________________________

CSSE 220—Object-Oriented Software Development

Exam 1, Oct. 1, 2008

This exam consists of two parts. Part 1 is to be solved on these pages. You may use the back of a page if you need more room. Please indicate on the front if you do so. Part 2 is to be solved using your computer. You will need network access to download template code and upload your solution for part 2. Please disable IM, email, and other such communication programs before beginning the exam.

Resources for Part 1: Open book and notes, closed computer.

Resources for Part 2: Open book, notes, and computer. Limited network access. You may use the network only to access your own files, the course ANGEL site and web pages, the textbook's site, Sun's Java website, and Logan Library's Safari Tech Books Online. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.

Parts 1 and 2 are included in this document. You should read over all of the questions before beginning work, but…

You must turn in part 1 before accessing the resources for part 2.

Problem	**Poss. Pts.**	**Earned**
1	15	______
2	9	______
3	4	______
4	12	______
5	4	______
6	8	______
7	9	______
Paper Part Subtotal	**61**	______
Computer Part	**39**	______
Total	100	______

Part 1—Paper Part

The next several questions all refer to a TempReading class. Below is a portion of this class showing its fields, constructor, accessor methods, and some mutator methods. The javadocs and remaining methods are omitted to save space.

```
public class TempReading {
    private int x;
    private int y;
    private double temp;

    public TempReading(int x, int y, double temp) {
        this.x = x;
        this.y = y;
        this.temp = temp;
    }

    public void setLocation(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void setTemp(double temp) {
        this.temp = temp;
    }

    public int getX() {
        return this.x;
    }

    public int getY() {
        return this.y;
    }

    public double getTemp() {
        return this.temp;
    }

    // remaining methods elided ...
}
```

1. (15 points) Below and on the next page are several code snippets, many of which use the TempReading class. For each snippet, first *draw a box-and-pointer diagram* showing the result of executing it. Then *give the output* of the print statement at the end of the snippet.

```
TempReading a =
    new TempReading(10, 20, 32.0);
a.setLocation(5, 6);
System.out.println(a.getX());
```

Output: ____________
Diagram:

```
TempReading b1 =
    new TempReading(0, 0, 98.6);
TempReading b2 = b1;
b1.setTemp(99.7);
System.out.println(b2.getTemp());
```

Output: ____________
Diagram:

```
TempReading[] tr = new TempReading[3];
tr[1] = new TempReading(11, 13, -40.0);
System.out.println(tr[0]);
```

Output: ____________
Diagram:

2. (9 points) For each code snippet below predict its output. (You do *not* have to draw a diagram, but you may if it helps you.)

```
TempReading d1 = new TempReading(3, 4, 54.3);
int x = d1.getX();
int y = d1.getY();
System.out.printf( "(%d,%d) is %5.2f.",
                               x, y, d1.getTemp());
```

Output: ____________

```
TempReading e1 =
    new TempReading(7, 8, 15.71);
int temp = (int) e1.getTemp();
System.out.println(temp);
```

Output: ____________

```
int f = 4;
int g = f + f;
String h = "four";
String i = h + h;
System.out.println(g + i);
```

Output: ____________

3. (4 points) How is an ArrayList different from an array?

4. (12 points) For each for loop below, write down how many times its body will execute, or indicate that we can't tell.

```
for (int i=0; i <= 10; i++) {
    // loop body elided
}
```

Answer: ______

```
for (int i=1; i < 10; i++) {
    // loop body elided
}
```

Answer: ______

```
int data[] = new int[5];
for (int i=0; i<data.length; i++) {
    // loop body elided
}
```

Answer: ______

```
public void someMethod(ArrayList<TempReading> temps) {
    for (TempReading t : temps) {
        // loop body elided
    }
}
```

Answer: ______

5. (4 points) Explain the difference in behavior between

```
public int foo(int i, int j) {
    int r = 0;
    if (i > 0) r++;
    if (j > 0) r++;
    return r;
}
```

and

```
public int foo(int i, int j) {
    int r = 0;
    if (i > 0) r++;
    else if (j > 0) r++;
    return r;
}
```

.

6. (8 points) *Unit Testing*: Consider the documentation for the isLeapYear() method below.

```
/**
 * Calculates whether the given year is a leap year on the Gregorian
 * calendar.
 *
 * From Wikipedia: The Gregorian calendar, the current standard calendar in
 * most of the world, adds a 29th day to February in all years evenly
 * divisible by 4, except for centennial years (those ending in -00) which
 * are not themselves divisible by 400. Thus 1600, 2000, and 2400 are leap
 * years, but 1700, 1800, 1900, 2100, 2200, and 2300 are not.
 *
 * @param year
 * @return whether the given year is a leap year
 */
public static boolean isLeapYear(int year) {
    // body code elided
}
```

Give a list of different arguments that would constitute a good test set for the method. For each argument you list, say briefly why you would include it in the unit test.

7. (9 points) For each of the code snippets below, first *circle* the bug, then *describe* what is wrong with the circled code.

```
Rectangle r = (5, 10, 15, 20);
```

Circle the bug in the code above and *describe* what's wrong:

```
Rectangle r;
r.translate(15, 25);
```

Circle the bug in the code above and *describe* what's wrong:

```
Scanner sc = new Scanner(System.in);
ArrayList<Integer> nums = new ArrayList<Integer>();
while (true) {
    System.out.print("Enter a number, or 'q' to quit: ");
    String input = sc.next();
    if (input == "q") {
        break;
    }
    int n = Integer.parseInt(input);
    nums.add(n);
}
```

Circle the bug in the code at left and *describe* what's wrong:

Name: ________________________________

Part 2—Computer Part

Resources for Part 2: Open book, notes, and computer. Limited network access. You may use the network only to access your own files, the course ANGEL site and web pages, the textbook's site, Sun's Java website, and Logan Library's Safari Tech Books Online. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.

You must turn in the preceding pages
before accessing the resources for part 2.

Instructions. You must actually get these problems working on your computer. Almost all of the credit for this problem will be for code that actually works. There are several different small methods to write, so you can get a lot of partial credit by getting some of them to work. If you get every part working, comments are not required. If you do not get a method to work, comments may help me to understand enough so I can give you (a small amount of) partial credit.

After you have handed in part 1, **begin part 2 by checking out the project named *Exam1* from your course SVN repository**. (Ask for help immediately if you are unable to do this.)

When you have finished the problems, and more frequently if you wish, you should **submit your code by committing it to your SVN repository**. We will check commit logs, so you must be careful not to commit anything after the end of the exam. For grading, we will ensure that the included JUnit tests have not been changed.

Problem Descriptions

C1. (12 points) Implement the four static methods in ArrayFiller according to their javadoc descriptions. The methods fill arrays or ArrayLists with a given sequence of numbers. For each method you must use just one loop. That is, you'll use four loops, one in each method. The program comments include "TODO, Prob. C1" items to indicate where to put your work.

JUnit tests for this problem are in ArrayFillerTest.

C2. (20 points) Implement the Polygon class. A polygon is a closed curve made up from line segments that join the polygon's corner points. You must complete the "TODO, Prob. C2" items in the program comments, including:

- study the given javadocs and decide on what field or fields to use,
- implement the constructor,
- implement a method to add a corner point to the polygon,
- implement methods to calculate the circumference and area (see javadocs and formula below), and
- implement a method to draw the polygon on a given Graphics2D object.

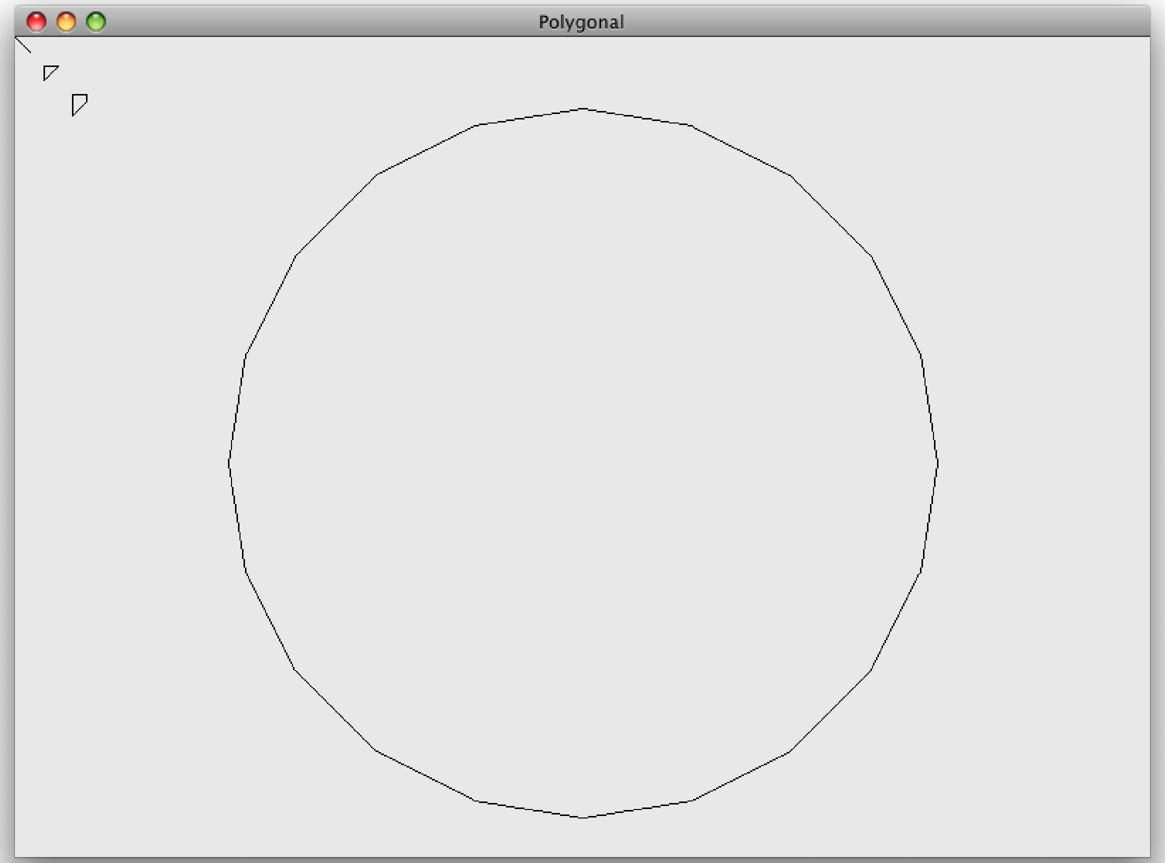

Figure 1: Example drawing of Polygons

The area of a polygon with corners $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ is:

$$\left|\frac{1}{2}\left(x_0y_1 + x_1y_2 + \cdots + x_{n-1}y_0 - y_0x_1 - y_1x_2 - \cdots - y_{n-1}x_0\right)\right|$$

Each x-component is multiplied by the y-component of the subsequent point and the products are summed. Then each y-component is multiplied by the x-component of the subsequent point and the products are subtracted from the previous sum. The answer is the absolute value of this total divided by 2.

Hints:

- Recall that the Point2D class includes accessor methods getX() and getY().
- Notice that Point2D also includes a method distance() that takes another Point2D and calculates the distance between the two.

JUnit tests for this problem are in PolygonTest.

Code that uses the drawOn() method starts in PolygonViewer. A correct implementation will produce an image like in Figure 1

C3. (**Bonus** 10 points) Implement the areMatching() method in the ListMatcher class (see "TODO, Prob. C3"). This method checks whether the two given lists contain the same number of the same values, ignoring the order. That is, if the first list contains two 3s then the second list must also contain exactly two 3s or they don't match.

Here are some examples:

- The list 1, 2, 3 matches 3, 2, 1
- The list 1, 2, 3 matches 1, 3, 2
- The list 1, 1, 3 matches 1, 3, 1
- The list 1, 2, 3 does not match 3, 2, 1, 1
- The list 1, 2, 3 does not match 3, 2, 1, 4
- The list 1, 2, 3, 4 does not match 3, 2, 1

Your implementation **must not change either list passed to it**.

JUnit tests for this problem are in ListMatcherTest.

JUnit Test or Graphics	**Points**	**Earned**
ArrayFillerTests (3 each)	12	______
PolygonTests (2 each)	20	______
Polygon graphics	7	______
ListMatcherTests (1 each, up to 10)	10 bonus	______
Computer Subtotal	**39**	______