

Name: _____

CSSE 220—Object-Oriented Software Development

Exam 1, Sept. 30, 2009

This exam consists of two parts. Part 1 is to be solved on these pages. Ask your instructor for scratch paper if you need more room. 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: You may use a single sheet of $8\frac{1}{2} \times 11$ inch paper with notes on both sides. Notes may be printed but must not be smaller than 4 point.

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	6	_____
6	5	_____
7	9	_____
Paper Part Subtotal	60	_____
Computer Part	40	_____
Total	100	_____

Part 1—Paper Part

The next several questions all refer to a LapTime 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 LapTime {  
    private int trial;  
    private int car;  
    private double speed;  
  
    public LapTime(int trial, int car, double speed) {  
        this.trial = trial;  
        this.car = car;  
        this.speed = speed;  
    }  
  
    public void setAttempt(int trial, int car) {  
        this.trial = trial;  
        this.car = car;  
    }  
  
    public void setSpeed(double speed) {  
        this.speed = speed;  
    }  
  
    public int getTrial() {  
        return this.trial;  
    }  
  
    public int getCar() {  
        return this.car;  
    }  
  
    public double getSpeed() {  
        return this.speed;  
    }  
  
    // remaining methods elided ...  
}
```

1. (15 points) Below are several code snippets, many of which use the LapTime 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.

```
LapTime a =  
    new LapTime(1, 24, 148.0);  
a.setSpeed(127.0);  
System.out.println(a.getCar());
```

Output: _____
Diagram:

```
LapTime lap1 =  
    new LapTime(0, 0, 98.6);  
LapTime x = lap1;  
lap1.setAttempt(2, 8);  
System.out.println(x.getTrial());
```

Output: _____
Diagram:

```
LapTime[] tr = new LapTime[5];  
tr[3] = new LapTime(4, 48, 157.3);  
System.out.println(tr[2]);
```

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.)

```
LapTime d1 = new LapTime(6, 12, 203.4);
int x = d1.getCar();
int y = d1.getTrial();
System.out.printf("(%d,%d) is %6.2f.",
                  x, y, d1.getSpeed());
```

Output: _____

```
LapTime e1 =
    new LapTime(7, 0, 155.61);
int sp = (int) e1.getSpeed();
System.out.println(sp);
```

Output: _____

```
int f = 3;
int g = f * f;
String h = "six";
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=2; i <= 7; i++) {  
    // loop body elided  
}
```

Answer: _____

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

Answer: _____

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

Answer: _____

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

Answer: _____

5. (6 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 method below.

```
/**
 * Transforms the given input string by reversing its characters and swapping
 * the case of all letters.
 *
 * For example, reverseAndSwapCase("Hello") yields "OLLEh".
 *
 * @param input
 * @return a transformed string
 */
public static String reverseAndSwapCase(String input) {
    // body code elided
}
```

List arguments that would constitute a good test set for this method. For each argument you list, say briefly why it should be 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 = Rectangle(5, 10, 15, 20);
```

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

```
Rectangle r;  
r.setX(15);
```

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

```
ArrayList<Double> scores = new ArrayList<Double>();  
Scanner sc = new Scanner(System.in);  
while (true) {  
    System.out.print("Enter a score, or 'e' to exit: ");  
    String input = sc.next();  
    if (input == "e") {  
        break;  
    }  
    scores.add(Double.parseDouble(input));  
}
```

When you run the code at the left, it doesn't stop when you enter "e". *Circle* the bug in the code and *describe* what's wrong:

Turn in part 1 before beginning part 2.

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.

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 the problems 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. (28 points) Implement the `Triangle` class. The parameters passed to the `Triangle` constructor are as shown in Figure 1.

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 scale the triangle by a given factor,
- implement a method to rotate the triangle by a given angle,
- implement a method to calculate the area, and
- implement a method to draw the polygon on a given `Graphics2D` object.

JUnit tests for this problem are in `TriangleTest`.

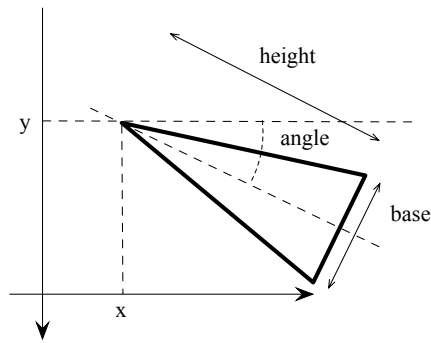


Figure 1: Example showing the meaning of the parameters passed to the Triangle constructor.

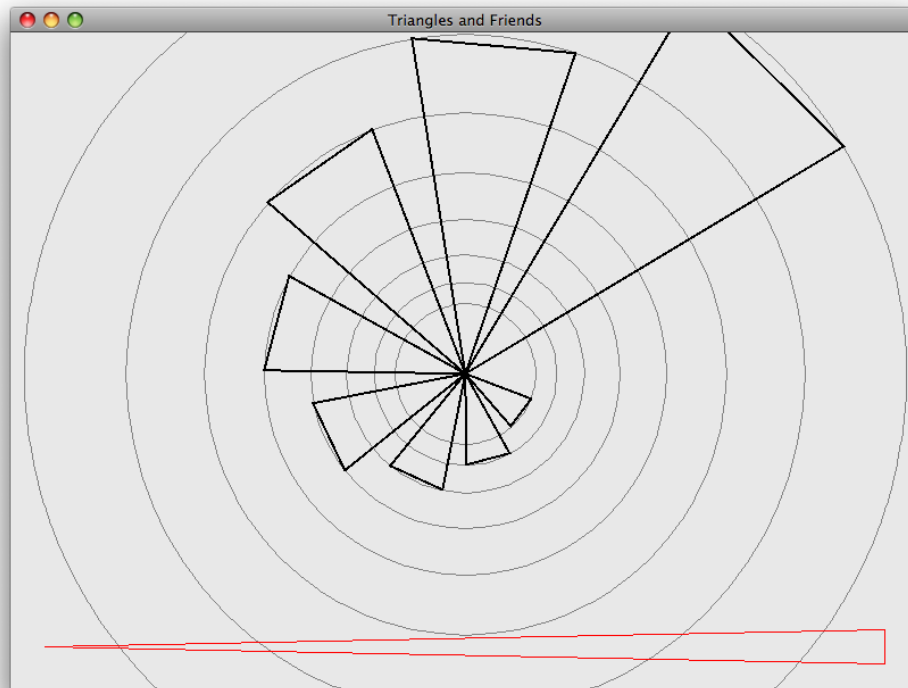


Figure 2: Example drawing of Triangles

Code that uses the `drawOn()` method starts in `TriangleViewer`. You only need to implement `Triangle`. `TriangleViewer` and `TriangleComponent` are completed already. A correct implementation `Triangle` will result in an image like in Figure 2

JUnit Test or Graphics	Points	Earned
ArrayFillerTests (3 each)	12	_____
TriangleTests (4 each)	12	_____
Triangle graphics		
location	4	_____
size	4	_____
orientation	8	_____
Computer Subtotal	40	_____