Test 1: CompSci 6

75 Minute Exam

September 24, 2008

Name (print): ________________________________

Honor Acknowledgment (signature): ________________________________

DO NOT SPEND MORE THAN 15 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later.

Before starting, make sure your test contains 12 pages. The last page is blank and can be used as scratch paper, but all pages must be turned in.

None of the Java programs or program segments should have syntax errors unless stated. If you think there is a syntax error or you do not understand what a question is asking, then please ask. Import statements may not always be shown.

Do not discuss this test with anyone until the test is handed back.

<table>
<thead>
<tr>
<th></th>
<th>value</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>10 points</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>8 points</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>12 points</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>15 points</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>25 points</td>
<td></td>
</tr>
<tr>
<td>Extra Credit</td>
<td>5 points</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>70 points</td>
<td></td>
</tr>
</tbody>
</table>
**PROBLEM 1:**  
(Maximum impact: 10 points)

**Part A:** (5 points)
Complete the method `allDifferent` that returns true only if all three of the components of the given `Color` have the different values. For example, a call to `allDifferent` with a color that has red, green, and blue component values (3, 128, 255) should return true; while a color with component values of (128, 200, 128) should return false.

```java
public boolean allDifferent (Color c)
{
}
```

**Part B:** (5 points)
Complete the method `max3` that takes three numbers and returns the one with the largest value (if more than one of the values are equally the largest, then return the first such value). Your implementation may not use any conditionals or loops and **must** call the method `Math.max` at least once and use its result in determining the result of this function.

```java
public int max3 (int a, int b, int c)
{
}
```
PROBLEM 2: (Understanding Java: (8 points))

For each of the following questions below, please answer briefly in complete sentences.

1. Describe the purpose of the constructor method of a class and two differences between the constructor and any other class method.

2. Describe the conceptual differences between an (object) class and an (object) instance. Give two examples of each that emphasize those differences.
PROBLEM 3:  (Old and In the Way (12 points))

The method `getMax` below is intended to return the name of the oldest person in a list of `Person` objects, where `Person` is declared simply as

```java
public class Person {
    public String name;
    public int age;
}
```

You will be asked four questions about the method. Currently `getMax` does not work because of compilation problems and perhaps logic problems (the line numbers are not part of the code).

```java
String getMax (ArrayList<Person> people) {
    int max = people.get(0).age;
    for (int k = 1; k < people.size(); k++) {
        String name;
        if (people.get(k).age > max) {
            max = people.get(k).age;
            name = people.get(k).name;
        }
    }
}
```

Part A  As written `getMax` generates the compilation error below when compiled.

```java
Person.java:15: missing return statement
```

Describe in one sentence why this error is generated and how to fix the problem by adding a single `return` statement to the method `getMax`. After adding the statement, the method will still not be correct, but the error will be fixed. You should add the single most logical line of code consistent with the current version of the program (you can add it the code above).
**Part B** After fixing the error in Part A, the following compiler error is generated. (Assuming you added the “right” single statement in Part A).

Person.java:14: cannot find symbol: variable name

Describe in one sentence why this error is generated and how to fix this error by moving one line of code. Which line should be moved and where should it be moved to so that the compiler error is not generated. Do not alter the line, just move it (you can show where to move it in the code on the previous page).

**Part C** After fixing both previous errors, the program will still generate the following compiler error.

Person.java:15: variable name might not have been initialized

Given that name is assigned a value within the loop, briefly describe the situation in which the compiler is correct and it would not be given a value. How can the problem be fixed by modifying one line of code (again, you can show the modified code on the previous page)?

**Part D** Now that the code actually compiles, are you certain that it will work correctly in all possible cases? Why or why not?
PROBLEM 4:  (Order doesn’t matter: 15 points)

Part A: (8 points)
Complete the method, `createFromFile`, for the `Canvas` class that reads a data file and adds the specified number of `BouncingBall` or `Racer` objects to its `myMovers` list.

Assume that the text file being read is in the format shown below: the name of the type of `Mover` to create followed by a number indicating the number of objects of that type of shape to create. Each number of movers should be created as described in their original classwork and should be treated independently, e.g., each set of racers should start on the right side of the canvas and be sized to fit the canvas’s height regardless of how many other sets of racers may be created later according to the data file. Likewise all bouncing balls should start in the center and move in a random initial direction.

A sample data file representing forty different shapes is shown below:

```
ball 13
racer 7
racer 10
ball 10
```

Complete the `Canvas` method `createFromFile` below:

```
public void createFromFile (Scanner input)
{
    // implementation goes here
}
```
Part B: (7 points)

Complete the method, `reverseFile`, that reads a data file into a list of `String` objects and then reverses the order of that list. Thus, at the end of the method, the last word read from the file should be the first word in the list, i.e., at index 0; the second to last word should be the second word in the list; etc.

In writing your method, you may use the following method, `swap`, that, given two indices within the collection, swaps the values at those positions.

```java
public void swap (ArrayList<String> words, int first, int second)
{
    String tmp = words.get(first);
    words.set(first, words.get(second));
    words.set(second, tmp);
}
```

Complete the method `reverseFile` below:

```java
public void reverseFile (Scanner input)
{

```
PROBLEM 5:  (Right on Target: (25 points))

Part A: (10 points)
Write the paint method of a subclass of Mover such that draws itself as a number of concentric ovals centered at the same position. They should be drawn from largest to smallest so that they are all visible and look like an archery target. The distance between each should be equal and each oval should be brighter than the one drawn before it, starting from the shape’s given color. The number of ovals drawn is passed as a parameter to the constructor as numRings.

For example, a target of three ovals should be drawn such that largest oval’s size is the same as that given for the entire target, the inner circle’s size is two-thirds of that given for the target, and the center circle’s size is one-third of that given for the target. Your paint method should work for any target with a positive number of ovals.

Complete the subclass declared below (declaring and initializing any needed instance variables).

```java
public class Target extends Mover {
    public Target (Point center, Dimension size, Point velocity, Color color, int numRings) {
        super(center, size, velocity, color);
    }

    public void paint (Graphics pen) {

    }

    // update method not used
}
```
Part B: (6 points)

Write the update method of a subclass of Mover such that it expands its size based on its velocity instead of changing its position until its size is the same or larger than that of the bounds.

For example, if its size was 100x50 pixels and its velocity was (2, 2), then its dimensions would double each time update was called. After the first call, its size would be 200x100; after the second call, its size would be 400x200; and on and on, until one of the dimensions grew to be larger than the corresponding bounds dimension.

Complete the class Grower started below.

```java
public class Grower extends Mover
{
    // no instance variables needed

    public Grower (Point center, Dimension size, Point velocity, Color color)
    {
        super(center, size, velocity, color);
    }

    public void paint (Graphics pen)
    {
        // implementation not shown
    }

    public void update (Dimension bounds)
    {
```
**Part C: (9 points)**

Complete the method `createMoversInACircle` for a `Canvas` class to create the given number of `BouncingBall` objects such that they are all the same size and are positioned in a circular pattern with each touching the one beside it (i.e., tangent to each other) like the marks on a clock (if those marks were circles). In this case, their position and size should be set so that they are spread evenly around the perimeter of a circle whose diameter is the minimum of the width and height of the canvas.

Recall that the formula for computing the circumference, or perimeter, of a circle is $2 \times \text{Math.PI} \times \text{radius}$. To calculate the the x and y position, multiply the radius of the circle by `Math.cos(angleInRadians)` and `Math.sin(angleInRadians)`, respectively and cast the result to an int value.

For example, given 4 objects to create and a canvas whose size is 100x200 pixels, you should create 4 attractors that are 78x78 pixels in size and centered at $(50, 0)$, $(100, 50)$, $(50, 100)$, and $(0, 50)$, respectively. Use the constants defined at the top of the method below to set their velocity and color respectively.

Complete the method `createMoversInACircle` below:

```java
public void createMoversInACircle (int numToCreate)
{
    final Point BALL_VELOCITY = new Point(0, 0);
    final Color BALL_COLOR = Color.BLUE;
```
PROBLEM 6 : (Find the Counterfeit (extra credit - 5 points))

Part A: (2 points)
You are given nine gold coins but you suspect one of the coins is counterfeit and is lighter than real gold. To prove your suspicion, you decide to weigh the coins. However, looking around you find that you only have a standard balance, i.e., one that compares the weights of two quantities. There is an easy algorithm to determine which coin is fake by comparing each coin against another using, in the worst case, four separate balance measurements. Describe an algorithm that allows you to find the fake coin with only two measurements.

Part B: (3 points)
Now, you are given six bags of coins, each bag with a large number of coins in it. However, only one of the bags has genuine gold coins in it. The other bags have counterfeit coins in them. It is known that each coin of false gold weighs 100 grams and that each real coin weighs 110 grams apiece. This time, you find that you have only a simple scale that displays the weight in grams. Again, there is a easy algorithm to determine the real bag of gold that requires five weighings in the worst case. Describe an algorithm to find the bag of real gold with only one weighing?