SI@UCF Java Course Test #2 Solutions
June 27, 2018

1. (15 pts) If we assign the softest possible sound to the intensity of 1, then we come up with a formula to calculate the intensity of a sound based on its decibel level. The formula is as follows:

$$Intensity = 10^{dB/10}$$

where dB is the decibel level of the sound. Complete the program below so that it prints out a chart of each integer decibel level from 0 to 140 (jet engine) and its corresponding intensity (a real number). (Note: Intensity represents how many times "louder" than the softest possible sound.)

Useful Math class method:

```
// Returns a raised to the power b.
static double pow(double a, double b)

import java.util.*;

public class decibels {

    public static void main(String[] args) {


        System.out.println("Decibels\tIntensity");
        for (int i=0; i<=140; i++)
            System.out.println(i+"\t\t"+(Math.pow(10, i/10.0)));

        // Grading: 0 pts - header
        //          4 pts - for loop
        //          1 pt print, 2 pts i, 8 pts intensity


    }
}
```

2. (20 pts) An exercise given in class but never collected dealt with a row of students receiving their exams back. A student would become upset if either of his neighbors (to his direct left or direct right) scored higher than he scored by 10 or more points. Given an array that stores each student's score in order of the row (index 0 is the left-most student, and the last valid index is the right-most student), write a static method that returns the number of upset students on the row corresponding to the input array. (Note: the left most and right most students only have one neighboring student, not two.) Complete the method below to solve the problem. **Remember, there should be NO println's in this method!!!** (Note: for any array, id.length is an expression equal to the length of the array, where id is the name of the variable referencing the array.)

```
public static int numUpset(int[] rowScores) {


    int res = 0;                                      // 1 pt

    for (int i=0; i<rowScores.length; i++) {          // 2 pts

        boolean upset = false;
        if (i>0 && rowScores[i-1] >= rowScores[i]+10) // 6 pts
            upset = true;

        if (i<rowScores.length-1 &&
            rowScores[i+1] >= rowScores[i]+10)        // 6 pts
            upset = true;

        if (upset) res++;                             // 4 pts
    }

    return res;                                        // 1 pt

    // Note for grading: this can be accomplished w/o the
    // boolean variable. Map points accordingly.



}
```

3. (25 pts) Imagine creating a beverage class. A beverage has a name and how much liquid (in oz.) is can carry and how much liquid is left in it. In the beginning, how much is left will be equal to the capacity of the beverage. For example, we can construct a can of coke to have the name "Coke" and to start off with 12 ounces. There are several operations that we can "do" to beverages. We can drink part of them, or throw them away, for example. We can also combine two beverages (but only if their names are the same!) into one. On the next page will be an incomplete specification of the beverage class. Fill in all of the methods that are not defined in the blank space provided based on the comments explaining their function.

```
public class beverage {

  private String name;
  private int capacity;
  private int numOunces;

  // Creates a new beverage with the name s, a capacity
  // of c and makes sure it's filled.
  public beverage(String s, int c) {
      name = s;             // 2 pts
      capacity = c;         // 2 pts
      numOunces = c;        // 1 pt
  }

  public boolean drink(int howmuch) {
    if (howmuch > numOunces)
      return false;
    numOunces -= howmuch;
    return true;
  }

  // Throws away the remaining drink that is left.
  public void dumpOut() {
      numOunces = 0; // 2 pts
  }

  // Fills in howmuch number of ounces in the current
  // beverage if there's room and returns true. Returns
  // false otherwise.
  public boolean refill(int howmuch) {
      if (howmuch+numOunces > capacity) // 2 pts
          return false;  // 2 pts
      numOunces += howmuch; // 2 pts
      return true; // 2 pts
  }

  // if other does NOT have the same name as this object
  // then null is returned. Otherwise, a new beverage
  // is created that has a capacity that is the sum of the
  // two capacities and the numOunces is the sum of the
  // two containers and the name is the same.
  public beverage combine(beverage other) {
      if (!name.equals(other.name)) // 2 pts
          return null;              // 1 pt
      beverage temp =
         new beverage(name,capacity+other.capacity);//3 pts
      int toDrink = capacity - numOunces;
      toDrink = toDrink + other.capacity - numOunces;
      temp.drink(toDrink); // 3 pts for all of this
      return temp; // 1 pt
  }
}
```

4. (15 pts) In Java, we can concatenate two Strings (or a String and anything else) with the plus sign. Consider the problem of writing a ***recursive*** method that takes in a String and returns a new String that is the result of reversing the input string. In your method, you may use the following string methods:

```
// Returns the length of this string.
public int length()

// Returns the substring of this string starting at index start.
public String substring(int start)

public static String rev(String s) {


    if (s.length() <= 1) return s;

    return rev(s.substring(1)) + s.charAt(0);

    // Grading: base case = 4 pts
    //          return = 1 pt
    //          rec call = 2 pts
    //          s.substring(1) = 3 pts
    //          + = 2 pts
    //          s.charAt(0) = 3 pts




}
```

5. (20 pts) Write a code segment in the draw function to create an 5 x 7 grid of black squares, each 50 pixels per side, where the squares are NOT filled in. The squares will represent the calendar for the month of February, not on a leap year where February 1st is a Sunday. The first row should have text drawn in each square with a single letter for the day of the week ('S', 'M', 'T', 'W', 'T', 'F', 'S'). The subsequent rows should have the corresponding dates written in them, with the second row containing the numbers 1 through 7, the third rom, the numbers 8 through 14 and so forth.

```
class Month extends JComponent {

    // Our constants.
    final public static int SCRSIZE = 500;
    final public static int SQSIZE = 50;
    final public static int OFFX = 45;
    final public static int OFFY = 35;
    final public static int OFFYLET = 55;
    final public static int WEEKS = 4;
    final public static int DAYS = 7;
    final public static String DAYNAMES = "SMTWTFS";

    public Month() {
    }

    public void paintComponent(Graphics g) {

        g.setColor(Color.WHITE);
        g.fillRect(0, 0, SCRSIZE, SCRSIZE);
        g.setColor(Color.BLACK);
        g.setFont(new Font("TIMESNEWROMAN", Font.PLAIN, 20));
        int day = 1;

        for (int y=0; y<WEEKS+1; y++) {                              // 2 pts
            for (int x=0; x<DAYS; x++) {                             // 2 pts

                // Be Flexible here with position, 6 pts
                g.drawRect(OFFX+x*SQSIZE, OFFY+y*SQSIZE, SQSIZE, SQSIZE);

                // 5 pts for this case.
                if (y > 0) {
                    g.drawString(""+day, OFFX + x*SQSIZE+SQSIZE/3,
                                         OFFYLET+y*SQSIZE+SQSIZE/3);
                    day++;
                }
                // 5 pts for this case.
                else {
                   g.drawString(""+DAYNAMES.charAt(x),
                                OFFX+x*SQSIZE+SQSIZE/3,
                                OFFYLET+y*SQSIZE+SQSIZE/3);
                }
            }
        }
    }
}
```

6. (5 pts) In which state did the first California Pizza Kitchen open? **California** **(Give to All)**