SI@UCF Java GUI Test #2 Solutions

7/22/2015

1) (15 pts) Consider creating a MyColor class that contains a red, green and blue component, each of which are integers ranging from 0 to 255, inclusive. We want MyColor to implement Comparable, so that an array of MyColor objects can be sorted (using Java's built in sort) as follows: sort by the sum red + green + blue, smallest to largest. For any ties, first break the tie by red (smaller red coming first), then green (smaller green coming first). This will break all ties. For example, the color r = 30, g = 40, b = 50 will come before r = 30, g = 41, b = 49, but after the color r = 30, g = 41, b = 48. Complete the compareTo method for this class below:

```
public class MyColor implements Comparable<MyColor> {

    private int red;
    private int green;
    private int blue;

    public int intensity() {
        return red+green+blue;
    }

    public int compareTo(MyColor other) {


        if (this.intensity() != other.intensity())
            return this.intensity() - other.intensity)();

        if (this.red != other.red)
            return this.red - other.red;

        return this.green - other.green;


        // Grading - 5 pts for each tiebreaker.

    }
}
```

2) (20 pts) A Sudoku puzzle, a 9 x 9 grid of integers, must have each value from 1 through 9 appear exactly once on each row, among other requirements. Complete the method below so that it takes in a complete Sudoku board (guaranteed to only be filled with entries in between 1 and 9, inclusive), and returns true if and only if each row has the values 1 through 9 appear once each. Follow the prototypes given below, where the method validRow checks the validity of one row and the first method, validRows, calls validRow to ensure the validity of all the rows.

```
public static boolean validRows(int[][] board) {

    for (int i=0; i<9; i++) {           // 2 pts
        if (!validRow(board[i]))        // 3 pts
            return false;               // 1 pt
    return true;                        // 1 pt

}

public static boolean validRow(int[] row) {


    int[] freq = new int[10];       // 2 pts
    for (int i=0; i<9; i++)         // 2 pts
        freq[row[i]]++;             // 3 pts

    for (int i=1; i<10; i++)        // 2 pts
        if (freq[i] != 1)           // 2 pts
            return false;           // 1 pt

    return true;                    // 1 pt


}
```

3) (15 pts) Write a method that takes in an array of Strings, all of which are solely composed of uppercase letters and prints out a list of the unique strings in the array, with their frequencies (how many times each string occurred in the array), in alphabetical order. Please use a TreeMap to complete this task. For example, if the input array was {"B", "B", "F", "A", "F", "F", "C", "A"} then your method should print:

```
A    2
B    2
C    1
F    3
```

Complete the method shown on the next page. (Note: The strings in the array may have more than one character each, even though this isn't the case in the sample above.)

```
public static void printFreqInOrder(String[] names) {


  TreeMap<String,Integer> freq = new TreeMan<String,Integer>();
  for (int i=0; i<names.length; i++) {

    if (freq.containsKey(names[i])) {
      int val = freq.get(names[i]);
      freq.put(names[i], val+1);
    else
      freq.put(names[i], 1);
  }

  while (freq.size() > 0) {
    String name = freq.firstKey();
    System.out.println(name+"\t"+freq.get(name));
    freq.remove(name);
  }

  // Grading - 8 pts for building map
  //           7 pts for printing data in alpha order
}
```

4) (15 pts) An n-bracket design of stars has 2n-1 rows where the first row has n stars, the second n-1 stars, etc. decreasing until the n^{th} row which has 1 star, followed by increasing rows of stars.

Complete the void **recursive** function below so that it prints out an n-bracket design of stars, where n is the given input parameter. (Hint: The base case is when n = 1.)

```
public static void nBracket(int n) {

    if (n == 1) System.out.println("*");
    else {
        for (int i=0; i<n; i++) System.out.print("*");
        System.out.println();
        nBracket(n-1);
        for (int i=0; i<n; i++) System.out.print("*");
        System.out.println();
    }
}
```

Grading - 4 pts base case, 3 pts for manually printing first and last line, 5 pts for recursive call situated correctly.

5) (15 pts) Consider the Oval class below. The Oval class defines the x and y coordinates of an oval within the JFrame.

```
class Oval {
    private int x;
    private int y;

    public Oval(int myx, int myy) {
        x = myx;
        y = myy;
    }
}
```

Given an ArrayList of Oval objects, complete the draw method below so that it draws the ovals. (Assume this snippet of code is properly invoked with all the appropriate support code, and the ovals ArrayList is properly initialized.)

```
ArrayList<Oval> ovals = new ArrayList<>();

public void paint(Graphics g) {
        this.draw(g, ovals);
}

public void draw(Graphics g, ArrayList<Oval> ovals) {


     for (Oval o : Ovals)                    // 5 pts
     {
          g.drawOval(o.x, o.y, 10, 10);      // 10 pts
     }




}
```

6) (15 pts) Consider the method below, where Oval is the class from the previous question, ovals is the same ArrayList as in the previous question, and MAX_X is the size of the JFrame in the X direction, and MAX_Y is the size of the JFrame in the Y direction.

```
public void secretMethod() {
        for (Oval o : ovals) {
            if (o.x != MAX_X/2 || o.y != MAX_Y/2)
            {
                if (MAX_X/2 < o.x) o.x--;
                else               o.x++;
                if (MAX_Y/2 < o.y) o.y--;
                else               o.y++;
            }
        }
}
```

(a) Assuming this method is called repeatedly like the update method was in the posted Java games and each Oval has a random X and Y position in the JFrame, what will the secretMethod do to the ovals eventually?

The method will eventually make every oval converge in the center of the JFrame, since the if statement will continue to be triggered until both the X and Y of the given oval is equal to the center, given by MAX_X/2 and MAX_Y/2. (5 pts)

(b) What will happen if the condition, on the left side of the if statement's or, is removed?

The ovals will eventually converge in the center of the JFrame in a horizontal line. (5 pts)

(c) What will happen if the condition, on the right side of the if statement's or, is removed?

The ovals will eventually converge in the center of the JFrame in a vertical line. (5 pts)

7) (5 pts) The MTV Video Music Award Nominees were recently announced. On which television network will the associated awards show be televised?

MTV (Give to all - 5 pts)