APCS :: Lab 11 – Graphical Carnival Game

You will produce a version of the Carnival Game described on the Fall Exam (and below) using GUI components (via the Squint library).

DESCRIPTION

Your GUI can (should) be simple, but it must display all of the information shown on the sample program diagram to the right. **Your program must also contain** a JTextField, a JComboBox, and a JButton. (Note: though this program looks simple, it contains many JComponents that are all working together).

Though it is not required, you will want to use JPanels to make your layout more sensible.

You may layout, color and style your program any way you like. Information on Colors, and Fonts is given below.

PROGRAM DESIGN

In practice, it is a good idea to separate the GUI as much as possible from the “brain” of the program. (This is also called separating the “view” from the “model.”) In this case, we want to separate the layout and function of our GUIManager from the details of the carnival game itself.

The program distribution code includes method stubs and documentation for the Carnival game. For this assignment please adhere to the given structure. In the future you will be expected to be able to envision this separation of view and model on your own.

HELP

Use the Squint textbook! All of the GUI material is covered in the Squint text (pdf available on the blog) to on namely, Chapters 1, 2 and 3. More specific page references are given on the blog.

FAQ:

**Q: How do I make different fonts?**
Like everything in Java, a font is an Object. Many JComponents have a method called `setFont` which accepts an instantiated Font object as an argument. Here is some code with explanations below:

```java
JLabel myLabel = new JLabel("some text here");
Font myFont = new Font("Arial", Font.PLAIN, 16);
myLabel.setFont(myFont);
//or in one step: myLabel.setFont(new Font("Arial", Font.PLAIN, 16));
```

Thus, the Font constructor accepts three arguments: the **name of the font** (a String), the **style of the font** (an int), and a **point size** (an int).

1. **The name of the font** must actually exist on the system on which the program is running ("Arial", "Courier", and "Times" usually work).
2. **The style** argument can be one of several static constants defined in the Font class: Font.PLAIN, Font.BOLD, and Font.ITALIC are typical and have the values 0, 1 and 2 respectively.
3. **The point size** is what you would expect.

You then must use the Font object to set the font of some JComponent, like the JLabel shown in the example above.

**Q: How do I set colors?**

Like everything in Java, a Color is an object. You must construct a Color and then set the color of some JComponent. Here is code with an explanation below:

```java
JPanel P = new JPanel();
JLabel myLabel = new JLabel("Some text here");
P.add(myLabel);
P.setBackground(Color.BLACK);
myLabel.setForeground(new Color(200, 100, 100));
```

The code above shows several things.

1. The Color class comes with some static constants with pre-defined colors. Color.BLACK, Color.BLUE, etc. you can look them up in the documentation.
2. You can also construct a Color object using RGB values.
3. Most JComponents allow you to setBackground and setForeground with varying results. For example, you can’t set the background color of JLabel –
it’s defined to have a transparent background. So you if you want the text to appear in a certain color you must set the foreground color.

**Q: What are the rules of the carnival game again?**

The player starts out with some amount of money (an int)

Each round, the player places a bet and guesses a number 1-6. (Placing a bet means giving the carnie some amount of money in order to play).

The player then rolls three dice and wins or loses some amount of money depending on how many dice show the guessed number.

<table>
<thead>
<tr>
<th>Dice showing guessed number</th>
<th>win/lose amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 dice</td>
<td>-1*bet</td>
</tr>
<tr>
<td>1 die</td>
<td>1*bet</td>
</tr>
<tr>
<td>2 dice</td>
<td>2*bet</td>
</tr>
<tr>
<td>3 dice</td>
<td>3*bet</td>
</tr>
</tbody>
</table>

Thus, if the player’s guess showed on exactly one die every time, the player would break even. Play continues until the player quits the game or runs out of money.