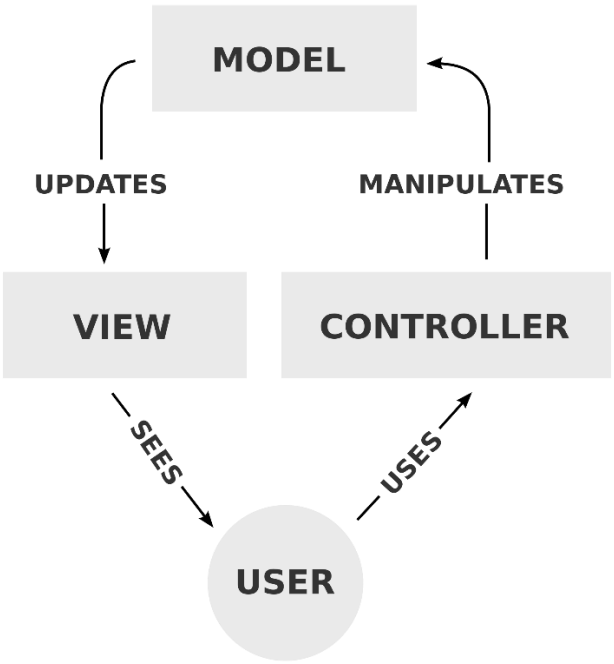
Introduction

This assignment builds upon the Dice quiz / project by providing a Graphical User Interface. It provides an opportunity to learn an important pattern known as the Model-View-Controller (MVC) pattern which is a prominent design pattern. A *design pattern* is a solution to a frequently occurring problem. In this case the problem is how to maintain a clean separation between the user interface and the data that the user manipulates via the user interface.

Model-View-Controller

MVC pattern is important for the design of desktop application with GUIs and dynamic web sites that retrieve data from databases and present the data to different client devices. Here is a basic diagram of the interactions between the layers:



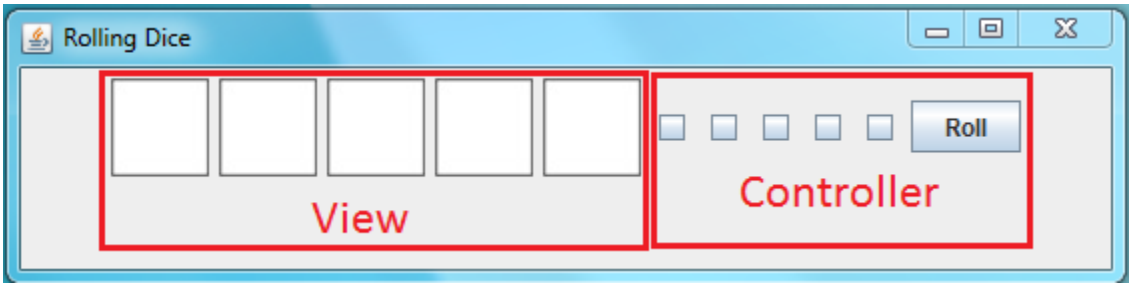
(You can read more about MVC here)

The value of MVC is that the model is generally very stable, whereas the view of the data is constantly changing due to new user interface requirements. If properly implemented MVC permits the model (and parts of the controller) to remain unchanged, while the view changes dramatically.

Rolling Dice == Dice with a GUI

We are going to put a Java Swing GUI on the non-graphical Simple Dice project. Here is the application when it first launches:

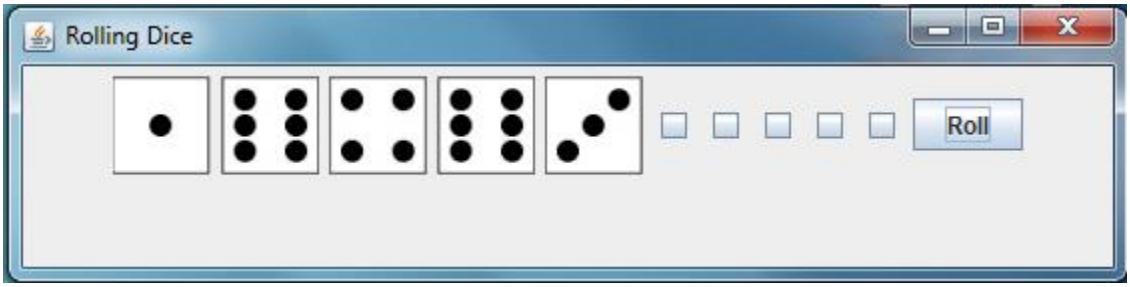
What is visible is the view and aspects of the controller:



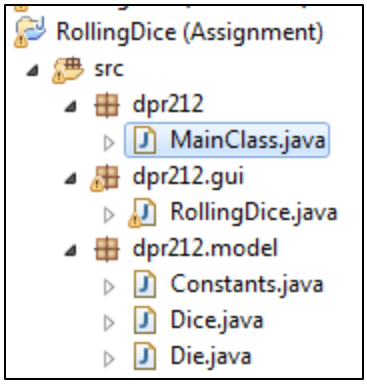
The dice and die objects live behind the scenes and are the application's model.

You can guess what clicking the roll button does, but what are the check boxes for? When a check box is checked, the corresponding die is "locked" and will not change in future rolls. Unchecking it permits the die to change.

Here is what a typically roll looks like:



There are three packages defined: 'dpr212', 'dpr212.gui' and ' dpr212.model' since we want a clean separation between the GUI part of the program and the Model part of the program. The GUI depends on the Model and is just a view of the state of the model; the Model never depends on GUI.



The packages permits us to use the Java visibility keywords `public, protected, private` more fully in order to enforce the GUI and Model separation.

Technical Discussion

Hint: All of your code will go in the `RollingDice` class, specifically in its constructor. There are directions in the code as to what to do. You will have to create an anonymous `ActionListener` object so review BJLO2 pp. 508 - 509. (BJLO pp. 480 - 481). Every `ActionListener` must have a `actionPerformed()` method. The only thing this method has to do is roll the dice!

Review the Model class's: `Constants` (there's only one constant), `Dice`, and `Die` (these classes are exactly the same as found in the Dice project.)

Let's review the one GUI class `RollingDice`. When you run the `MainClass` in the 'dpr212' package all you get is an empty window. Your job for this assignment is to create and add the components needed for the GUI. You will be creating/adding five `JLabel` components (labels are usually text objects, but in Swing a label can be image based), five `JCheckbox` components, and one `JButton` component.

Notice that `savedRoll` is an array in the starter program.

```
JLabel [] savedRoll;
```

If you don't make it an array you must reference five different `JLabel` objects by name (`savedRoll1`, `savedRoll2`, etc.) which makes programming very difficult. So here is the trick to initializing the label objects:

```
// get space for the array
savedRoll = new JLabel[Constants.MAX_DICE];
// initialize array elements with JLabel objects
for (int i=0; i<Constants.MAX_DICE; i++) {
      // initialize the array element with JLabel object
      savedRoll[i] = new JLabel(emptyIcon);
```

```
    }
```

You must add code that does the following:

1. Create needed JComponents for the GUI.
2. Hook up an event listener to the roll button.
3. Add the components to the panel to make them visible.

Enhancement #1

You may have noticed that in the console window the values of the die faces are displayed:

```
New roll:
At 0 show value 5
At 1 show value 2
At 2 show value 3
At 3 show value 1
At 4 show value 4
```

If you are a very careful observer you will note that the graphical dies are out of sync with the printed values. What's going on? Well the die images are locked but the model die are not. The enhancement is to lock the model die also by passing the check box information to the model level.

With this enhancement the printed output on the way to the coveted Yahtzee will look like this:

```
New Roll:
At 0 show value 1
At 1 show value 2
At 2 show value 6
At 3 show value 1
At 4 show value 5
New Roll:
At 0 show value 1
At 1 show value 1
At 2 show value 6
At 3 show value 1
At 4 show value 2
New Roll:
At 0 show value 1
At 1 show value 1
At 2 show value 5
At 3 show value 1
At 4 show value 1
New Roll:
At 0 show value 1
At 1 show value 1
At 2 show value 1
At 3 show value 1
At 4 show value 1
```

Assignment

Download RollingDice.zip from Canvas and import it into Eclipse. When you have completed the assignment submit `RollingDice.java` and a comment on "what I learned."