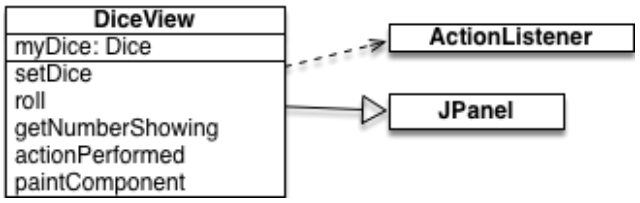
ICT-4361 Homework 4

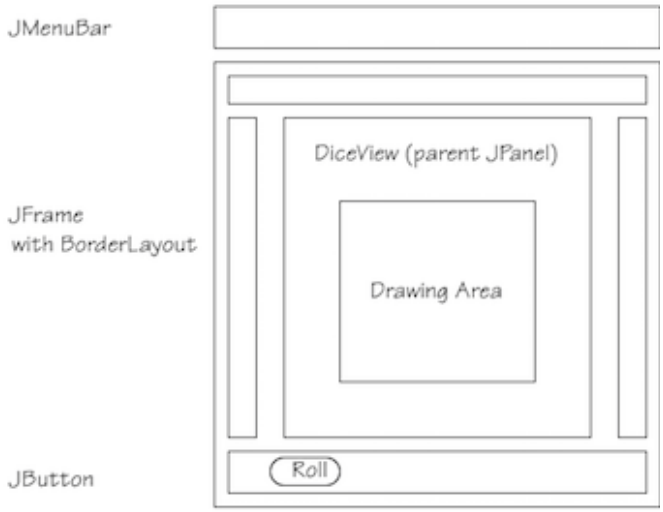
Purpose

This exercise will familiarize you with the basic graphics functionality of Java, using AWT and Swing.
Our approach will be to display the dice throws from our previous homework, and display them graphically.
For this exercise, we will restrict ourselves to displaying one face of 6-side dice. To extend the exercise, think about how you might handle other kinds of dice!



What to Hand In

Please hand in a listing for each program requested, formatted in an easy-to-read style.
Ensure your name, and the name of the file is available in a comment at the top of the file.
Also, ensure that you have a sample of the output from the program.
If your program fails to compile, hand in your error listing as your output.
For each question asked, provide one or two sentences summarizing your answer. Please be both complete and succinct.



Problems

I. Make a graphical dice panel. If properly made, this Dice can be used and embedded in any of your game applications.
Remember, you already have a Dice class, and this panel will leverage it.

1. Create a class called `DiceView` which extends `JPanel` and implements `ActionListener`.
2. Create a private data member to hold an object of type `Dice`
3. Create a setter method for the `Dice` object.
4. Create a public method for the `ActionListener` called `actionPerformed` which does the following:
   - Checks if the ActionEvent action command is "Roll"
   - If so, check the Dice instance to ensure it is non-null
   - If the dice is non-null, `roll()` the dice
   - Call `repaint()`
5. Create a public method for the component called `paintComponent`, which does the following:
   - Gets the size of the `JPanel` and sets local variables width and height
   - Clears the rectangular area of the `JPanel`
   - Draws a rectangle (or rounded rectangle), preferably a having equal sides, to fill most of the area in the `JPanel` (this represents the side of the dice)
   - If the dice is not null, displays the value of the dice within that rectangle
   - If the dice is null, displays the text string "No Dice" in that rectangle

6. Create a main program to test out your JPanel. You may use the following program, or modify it for your own purposes:

```
public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JButton button = new JButton("Roll");
    DiceView dv = new DiceView();
    button.setActionCommand("Roll");
    button.addActionListener(dv);
    Dice d = new Dice();
    // To test what happens with no dice, comment out the next line
    dv.setDice(d);

    frame.add(BorderLayout.CENTER,dv);
    frame.add(BorderLayout.SOUTH,button);
    frame.pack();
    frame.setSize(300,300);
    frame.setVisible(true);
 }
```

Notes

- This separation between the model (Dice) and the view and controller is called the MVC pattern, and is used throughout Java.
- The `JPanel` is found in package javax.swing
- The `ActionListener` interface is found in package `java.awt.event`
- Calling `repaint()` arranges for the component to be redrawn as soon as convenient. The redrawing will invoke paintComponent at the right time.
- The public method for the `ActionListener` has the form: `public void actionPerformed(ActionEvent e)`
- The public method for the painting routine has the form: `public void paintComponent(Graphics g)`. Note that even though Graphics is an abstract class, whatever implementation happens to be used (Graphics2D in this case), the right methods will be called without having to cast the parameter.
- To get the current size of the `JPanel`, you can use the `getSize()` method, which returns a `Dimension` (from package `java.awt`). This has attributes width and height.
- The `Graphic` methods `clearRect`, `drawRect` (or `drawRoundedRect`), and `drawString` should suffice for the actual drawing in the exercise. If you want to draw the pips on the dice, you will want `fillOval`.
- Note that the provided main program has no menus, and no exit button. However, it does arrange to close your application when the frame is closed (this is not the default!).
- This homework exercise is very extensible. Here are a few ideas for extending it:
    - **Add a second dice**. Note that you could add one dice to the EAST and one to the WEST (leaving the CENTER empty). Also note that you will have to add each dice as ActionListeners for the button..
    - **Add an exit button.** To make the button appear side-by-side with the other button, you must create a new JPanel, and put the existing JButton (Roll) in the EAST, and the new JButton (Exit) in the WEST. Then this JPanel is installed into the application the same way the old Roll JButton was, above. You need to attach an ActionListener to the button like so:

```
class ExitHandler implements ActionListener {
  public void actionPerformed(ActionEvent e) {
```

```
            System.exit(0);
        }
    }
```

This could be an inner class if you like.

- **Add a row of dice.** For this one, it is best to create a new class which extends JPanel and puts the dice in a GridLayout or a FlowLayout.
  An instance of this class, which could have an int parameter in its constructor, to tell you how many dice, can be placed in the CENTER, instead of the single dice.
  This is the way panels are nested to form a more complex display
- **Add a menu.** For this one, start simple--a File menu with an Exit button.
  You will use a JMenuBar, create a JMenu titled File, and a JMenuItem titled Exit.
  Your `JMenuItem` Exit pick must add an `ActionListener`, another instance as above in the Exit button case.
  Complete this as follows:
  Add the `JMenu` to the `JMenuBar`; add the `JMenuItem` to the `JMenu`, and then tell the frame to install the `JMenuBar` with `frame.setJMenuBar(menubar)`; (or whatever you called your menu bar).

Evaluation

Criteria	Weight
Create a proper DiceView class	10%
Create a proper data member for dice	10%
Create a proper setter method for the dice	10%
Create a poper actionPerformed method	25%
Create a proper paintComponent method	25%
Create a proper main program and output	20%