Assignment 10
Due SATURDAY April 10

1. **An ID Card**

   Design an ID card with your picture (or any picture you get off the web or draw in Paint) in the center, your name on the top, and your personal information (height, weight, eye color, address) split left and right. The bottom section of the card should display "Java Programmer ".

   Use a JLabels for the NORTH, CENTER and SOUTH. You should use JPanel for the information in the WEST and EAST. You might consider a GridLayout manager 3 x 1 for those panels.

   The ID card should be the size of a typical driver's license (250 x 150), place it at position (0,0) and do not allow resizing.

Include all the files that you need to run the program

2. **Faces**

Create a frame with three buttons in the NORTH section of the frame. The buttons should be labeled with the names of three of your favorite TV or movie characters such as Bart, Homer, and Marge or perhaps Sleepy, Dopey, and Grumpy. Use a JPanel for the buttons

Place a label in the CENTER section of the frame. It would be nice if, when you click a button, a picture of the corresponding character appears on the label. However, button-clicking is a topic for later. For the present, the constructor should place a randomly chosen image, of one of the three characters in the CENTER section of the frame. Because the image is randomly chosen, the same picture does not show each time the frame is instantiated.

In the SOUTH section of the frame include a quotation from the character or caption about the character. The quote should be on a label.

Include a main(...) method that instantiates your frame.

If you need a picture, there are pictures of Homer, Marge, and Bart and some other characters on the course web page under "OtherStuff." Or grab one from the web , or use your own from myHill.

3. *IMPORTANT: This will help with the next two problems*

Java has a Point class with two public variables

int x and int y

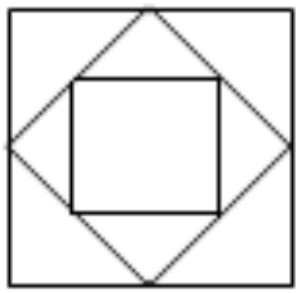
You can create a Point object as

Point a = new Point (50, 100)

You can access the coordinates as a.x and a.y because they are public. So, here, a.x is 50 and a.y is 100. You don't need getters.

IF YOU HAVE YOUR OWN POINT CLASS, YOU CAN USE THAT. HOWEVER, IF YOU WANT TO USE JAVA'S CLASS DO NOT HAVE YOUR POINT CLASS IN THE SAME FOLDER.

For this problem, you will draw a set of squares nested one inside the other so that each inner square should have its four corners at the **midpoints** of the previous squares. Here is a picture drawing 3 nested squares:



To do this, create a class TunnelVision that extends JPanel.
Here is the basic structure:

```
public class TunnelVision extends JPanel
{

    Point a,b,c,d;          //the corners of the square
                            // a Point object has public fields x and y
    int count;              // the number of squares to be drawn
```

```
    public TunnelVision(int count)  // constructor
   {
                //The constructor should  initialize the four Point objects.
                // You can use any set of coordinates for the first square.
                //For example (0,0), (0,200), (200,200) and (200,0).
                //The points are hard-wired into your program
   }

  public paintComponent(Graphics g)  // override PaintComponent from JPanel
   {
      super.paintComponent(g);
      draw (count, g);       // You must pass the Graphics object to draw.
                             // draw(...) does the work
   }

  void draw(int count, Graphics g)
   {
       // draws the squares on the panel
      // use a loop to draw the picture.
    }

}
```

Use the following class to display the panel in a frame

```
public class Framer
{
  public static void main(String[] args)
  {
    Scanner input = new Scanner(System.in);
    System.out.print("How many rectangles: ");
    int num = input.nextInt();

    JFrame f = new JFrame();
    f.setBounds(?,?,?,?);     // you figure that out

    JPanel p = new TunnelVision(num);
    f.add(p);  // add panel to center of frame

    f.setVisible(true);
     f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

4. Repeat problem 3 but in this case draw the picture recursively. Yikes! Recursion!!!!

Call the Panel class TunnelVision1 instead of TunnelVision

Now implement the draw(…) method recursively—no loop

Now draw should look like:

public void draw(Point a, Point b , Point c, Point d, int count, Graphics g)

Each time draw is called count should be decremented so that the recursion stops when count equals 0.

Use this main class:

```
public class Framer1
{
   public static void main(String[] args)
  {
     Scanner input = new Scanner(System.in);
     System.out.print("How many rectangles: ");
     int num = input.nextInt();

    JFrame f = new JFrame();
    f.setBounds(?,?,?,?);     // you figure that out

    JPanel p = new TunnelVision1(num);
    f.add(p);         // add panel to center of frame

    f.setVisible(true);
     f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```