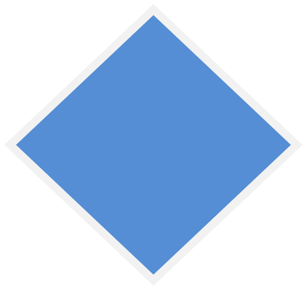
CREATING YOUR OWN CLASSES: PROGRAMMING EXERCISES

1. Create a **Diamond** class that allows users to draw a diamond object anywhere on the screen, make it any width and any height, and set it to any colour. Your **Diamond** class will need to include the following fields, constructors and methods:



FIELDS

```
private int xPos, yPos, width, height;
private Color col;
private Polygon p;
```

The first four fields will set the properties of the diamond, where it will be positioned on the screen (**xPos** and **yPos**) and its dimensions (**width** and **height**). The field named **col** will account for the colour of the diamond. Finally, the field named **p** will represent the polygon object that will need to be created in order to draw a diamond.

CONSTRUCTORS

Create a **no-arg** constructor which initializes the **Diamond** object with default values.

```
public Diamond()
```

METHODS

```
public void drawDiamond(Graphics2D g2)
```

This method must draw a diamond based on the class variables defined by the user.

```
public void fillDiamond(Graphics2D g2)
```

This method must draw a filled-in diamond based on the class variables defined by the user.

```
private void addPoints()
```

This method must add the points to the diamond based on the class variables defined by the user.

```
public void setX(int x)
```

This method sets the x-position of the diamond.

```
public void setY(int y)
```

This method sets the y-position of the diamond.

```
public void setWidth(int w)
```

This method sets the width of the diamond.

```
public void setHeight(int h)
```

This method sets the height of the diamond.

```
public void setColor(Color c)
```

This method sets the colour of the diamond.

```
public int getX()
```

This method returns the x-position of the diamond.

```
public int getY()
```

This method returns the y-position of the diamond.

```
public int getWidth()
```

This method returns the width of the diamond.

```
public int getHeight()
```

This method returns the height of the diamond.

```
public Color getColor()
```

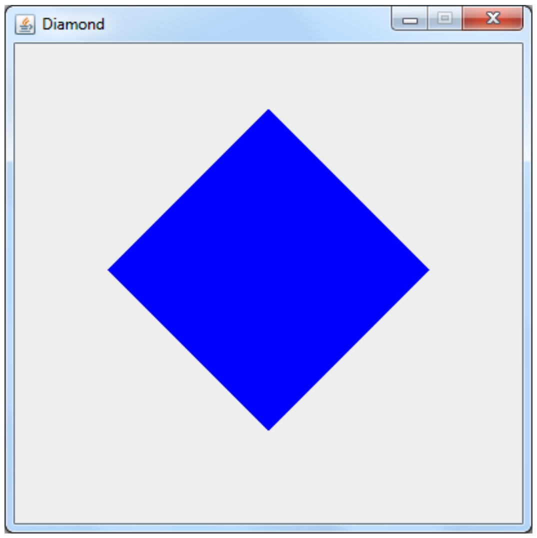
This method returns the colour of the diamond.

Once you have created the **Diamond** class, test the class by creating a program that draws a Diamond object onto the screen. The program itself should look something like this:

```
import javax.swing.*;
import java.awt.*;

public class DiamondProgram extends JFrame {

   private Diamond d;
   private static int FRAME_WIDTH = 300;
   private static int FRAME_HEIGHT = 300;

   public static void main(String[] args) {

      new DiamondProgram();
   }
```

```
    public DiamondProgram() {

       d = new Diamond(FRAME_WIDTH / 2 - 150 / 2,
          FRAME_HEIGHT / 2 - 150 / 2, 150, 150);

       setTitle("Diamond Program");
       setSize(FRAME_WIDTH, FRAME_HEIGHT);
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       setResizable(false);
       setLocationRelativeTo(null);
       setVisible(true);
    }

    public void paint(Graphics g) {

       super.paint(g);
       Graphics2D g2 = (Graphics2D) g;
       g2.setColor(Color.ORANGE);
       d.fillDiamond(g2);
    }
 }
```

Your program output should look something like this:



Save the program and the **Diamond** class in your **UNIT 4** folder.

2. Create a **Parallelogram** class that allows users to draw a three-dimensional parallelogram object anywhere on the screen, make it any width and any height, and set it to any colour. Your **Parallelogram** class will need to include the following fields, constructors and methods:

FIELDS

```
private int xPos, yPos, width, height;
private Color col;
private Polygon p;
```

The first four fields will set the properties of the parallelogram, where it will be positioned on the screen (**xPos** and **yPos**) and its dimensions (**width** and **height**). The field named **col** will account for the colour of the parallelogram. Finally, the field named **p** will represent the polygon object that will need to be created in order to draw a parallelogram.

CONSTRUCTORS

Create a **no-arg** constructor which initializes the **Parallelogram** object with default values.

```
public Parallelogram()
```

METHODS

```
public void drawParallelogram(Graphics2D g2)
```

This method must draw a parallelogram based on the class variables defined by the user.

```
public void setX(int x)
```

This method sets the x-position of the parallelogram.

```
public void setY(int y)
```

This method sets the y-position of the parallelogram.

```
public void setWidth(int w)
```

This method sets the width of the parallelogram.

```
public void setHeight(int h)
```

This method sets the height of the parallelogram.

```
public void setColor(Color c)
```

This method sets the colour of the parallelogram.

```
public int getX()
```

This method returns the x-position of the parallelogram.

`public int getY()`

This method returns the y-position of the parallelogram.

`public int getWidth()`

This method returns the width of the parallelogram.

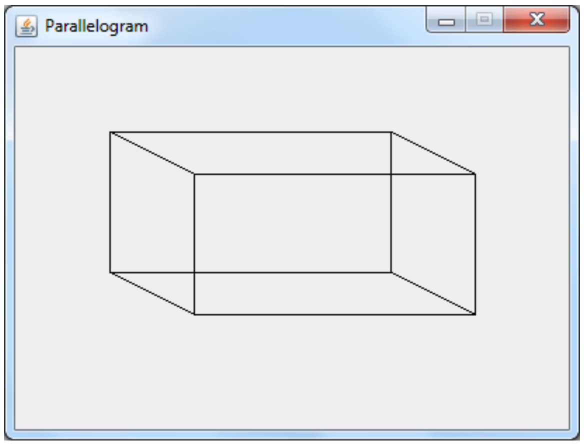
`public int getHeight()`

This method returns the height of the parallelogram.

`public Color getColor()`

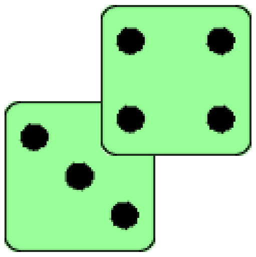
This method returns the colour of the parallelogram.

Once you have created the class, declare and initialize an instance of the **Parallelogram** class in a program so that your output looks something like this:



Save the program and the **Parallelogram** class in your **UNIT 4** folder.

3. Create a **Dice** class that allows the user to create dice and generates a random roll. Your **Dice** class will need to include the following fields, constructors and methods:



FIELDS

```
private ImageIcon imgDice;
private Random rnd;
private int rollValue, width, height;
```

The **ImageIcon** object called **imgDice** will store one of seven dice images which you can find in the **Classroom Rosters** folder. We will also need a **Random** object to randomly generate a number between 1 and 6. Finally, we will need three **int** variables: one to store the value of the roll, the width of the dice image, and the height of the dice image.

CONSTRUCTORS

Create a **no-arg** constructor which initializes the **ImageIcon** object to the blank die image (**die0.gif**), initializes a **Random** object, and the three **int** variables.

```
public Dice() {

   imgDice = new ImageIcon("images\\die0.gif");
   rnd = new Random();
   rollValue = 0;
   width = imgDice.getIconWidth();
   height = imgDice.getIconHeight();
}
```

METHODS

```
public void roll()
```

This method must randomly generate 1 of 6 possible numbers, set the **ImageIcon** to the corresponding image (i.e. if the number 3 is randomly generated, a 3 should be rolled), and assigns the value of the roll.

```
public int getDiceWidth()
```

This method must return the width of the dice image.

```
public int getDiceHeight()
```

This method must return the height of the dice image.

```
public Image getDice()
```

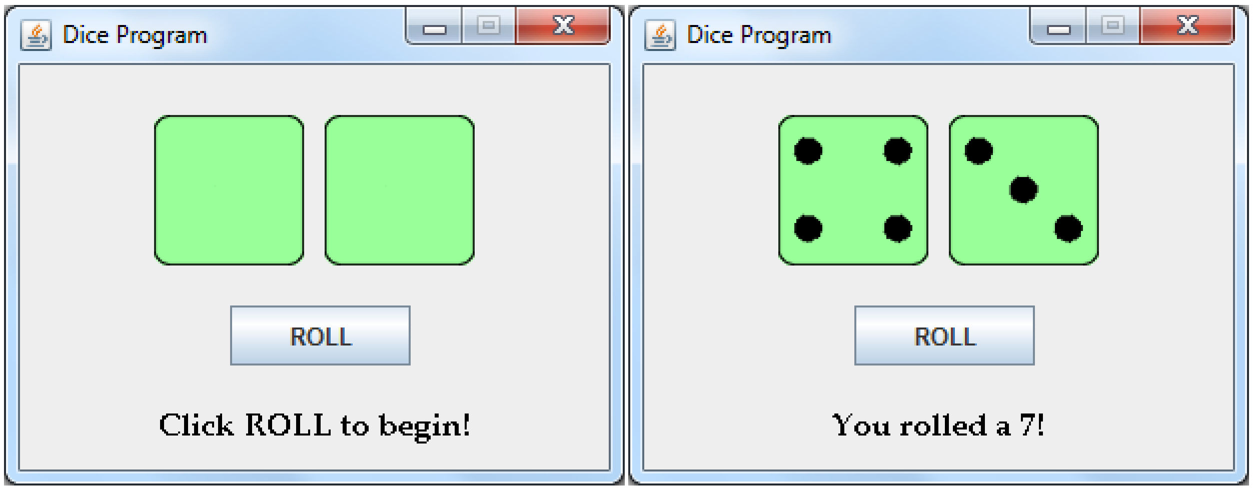
This method must return the dice image that was randomly generated.

```
public int getRollValue()
```

This method must return the value of the roll.

Once you have created the **Dice** class, test the class by creating a program that uses two dice and outputs a random roll every time the user clicks the **ROLL** button and outputs the value of the roll as a String (you will need to use the **drawString**() method included in the **Graphics2D** class.

Your program output should look something like this:



Click ROLL to begin!

You rolled a 7!

Save the project as **Dice** in your **UNIT 4** folder.

4. Create a **Star** class that allows users to draw a star object anywhere on the screen, make it any width and any height, and set it to any colour. Your **Star** class will need to include the following fields, constructors and methods:

FIELDS

```
private int xPos, yPos, width, height;
private Color col;
private Polygon p;
```

The first four fields will set the properties of the star, where it will be positioned on the screen (**xPos** and **yPos**) and its dimensions (**width** and **height**). The field named **col** will account for the colour of the star. Finally, the field named **p** will represent the polygon object that will need to be created in order to draw a star.

CONSTRUCTORS

Create a **no-arg** constructor which initializes the Star object with default values, and two more constructors: the first which allows users to set the location and dimension of the star, and the second which allows users to set the location, the dimensions and the colour.

```
public Star()
public Star(int x, int y, int w, int h)
public Star(int x, int y, int w, int h, Color c)
```

METHODS

```
public void drawStar(Graphics2D g2)
```

This method must draw a star based on the class variables defined by the user.

```
public void fillStar(Graphics2D g2)
```

This method must draw a filled-in star based on the class variables defined by the user.

```
public void setX(int x)
```

This method sets the x-position of the star.

```
public void setY(int y)
```

This method sets the y-position of the star.

```
public void setWidth(int w)
```

This method sets the width of the star.

```
public void setHeight(int h)
```

This method sets the height of the star.

```
public void setColor(Color c)
```

This method sets the colour of the star.

```
public int getX()
```

This method returns the x-position of the star.

```
public int getY()
```

This method returns the y-position of the star.

```
public int getWidth()
```

This method returns the width of the star.

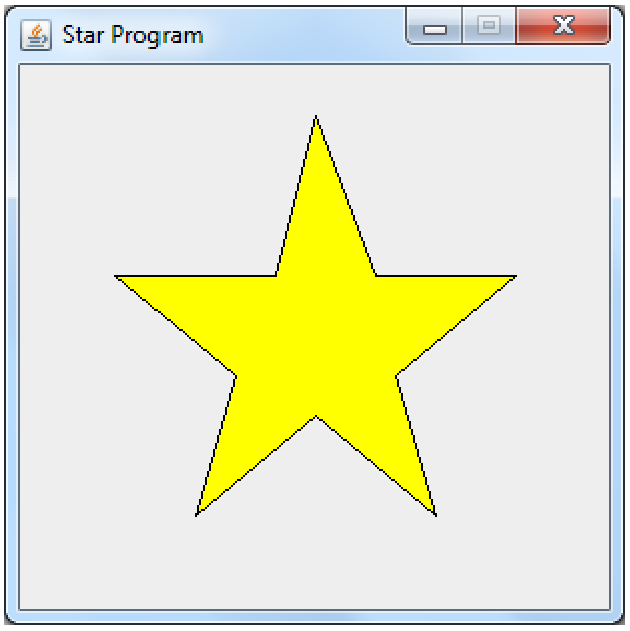
```
public int getHeight()
```

This method returns the height of the star.

```
public Color getColor()
```

This method returns the colour of the star.

Once you have created the class, declare and initialize an instance of the **Star** class in a program so that your output looks something like this:



Save the program and the **Star** class in your **UNIT 4** folder.