- We are using a program structure that utilizes JFrames and JPanels. The main method creates a JFrame object, which is a container for the JPanel. The JPanel can be thought of as our canvas, on which we will draw all of our graphics.

- The following is a typical main method for this structure:

```
public static void main(String[] args) {
    JFrame window = new JFrame("Title!");
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setBounds(new Dimension(800,800); //( w, h)

    MyPanel panel = new MyPanel();
    panel.setFocusable(true);
    panel.grabFocus();

    window.add(panel);
    window.pack();
    window.setVisible(true);
}
```

- The JFrame will frequently call the paintComponent(Graphics g) method of the JPanel. Thus, the paint method is the place where we should put our graphics code. Note we cast g to Graphics2D g2. We will use g2 to do our drawing.

- If you want to change the color of your "pen" , use the g2.setColor(COLOR) method. You can feed it Color.BLUE or Color.RED, for many preset colors, or make your own color, using new Color(int r, int g, int b). These ints must lie in the range [0,255].

- You can draw shapes quickly using the following. You can replace draw with fill.

  - `g2.drawRect(x,y,w,h)`
  - `g2.drawOval(x,y,w,h)`
  - `g2.drawLine(x1,y1,x2,y2)`

- Alternately, you can make objects to represent the shapes you are drawing:

  - `Rectangle box = new Rectangle( x, y, width, height)`
  - `Ellipse2D.Double oval = new Ellipse2D.Double(x, y, width, height)`
  - `Line2D.Double line = new Line2D.Double(x1, y1, x2, y2)`

  Then draw or fill them using g2.draw(shape) or g2.fill(shape).
  This approach can have benefits, as the objects have methods defined that can be useful down the road (intersections, for example).

- You can display a String on the canvas using g2.drawString(String text, int x, int y). You can change the font with g2.setFont(new Font("Comic Sans", Font.PLAIN, 24))

- A polygon can be created by specifying the coordinates of its vertices. These are stored in two arrays of integers, and then passed into the constructor of the Polygon class. For example:

```
int[] xVals = {200, 100, 300};
int[] yVals = {400, 250, 100};
Polygon polly = new Polygon(xVals, yVals, 3);
g2.draw(polly); //
```

  This creates polygon with vertices at (200, 400), (100, 250), and (300, 100). Note that the third parameter of the Polygon constructor is the number of points.

  See API Doc for Polygon for more information on the Polygon class.

- An arc can be drawn or filled. An arc is a sector of an ellipse.
  The drawArc method takes in 6 parameters, all ints.

```
g2.drawArc(x, y, width, height, startAngle, arcAngle)
```

  The first 4 parameters define an ellipse. The startAngle defines where in the ellipse to start drawing, with 0 degrees being at 3 o'clock. The arcAngle defines how many degrees to go from the startAngle, with positive angles going in a counterclockwise direction.
  The arc shown has a startAngle of 0, and a arcAngle of 90.

  Here is a summary of the methods:

  http://mathbits.com/MathBits/Java/Graphics/GraphingMethods.htm