THE GRAPHICS CONTEXT

AP Computer Science
Your screen can be filled with many windows
- The upper left corner of each window is (0,0)

Each window can contain one or more panels
- A panel is a rectangular region
- The upper left corner of each panel is (0,0)

Classes Needed for Graphics
- Same as those needed for a GUI
Step #1: Create a subclass of JPanel

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

public class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        //can put code here to draw things
    }
}
```

PaintComponent will be called automatically when the window is refreshed...you don’t need to call it yourself.
The call to the super class clears the background.
Step #2: Create an instance of the panel class and add it to the application window

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

public class MyExample extends JFrame {
    private JPanel thePanel; // could also use private MyPanel thePanel;

    public MyExample() {
        thePanel = new MyPanel();
        Container pane = getContentPane();
        pane.add(thePanel, BorderLayout.CENTER);
    }

    public static void main(String[] args) {
        MyExample theGUI = new MyExample();
        theGUI setSize(300, 500);
        theGUI.setVisible(true);
        theGUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```
Methods that can be used in the Panel class

- Dimensions of the Panel
  - getWidth()
  - getHeight()

- Background Color of the Panel
  - setBackground(Color.red)

- Refreshing the Screen
  - repaint()
    - Erases the current drawing and then calls paintComponent()
ACCESSING THE GRAPHICS CONTEXT

- Graphics g = getGraphics();
  - Used in the panel class
  - Eliminates the need for writing the paintComponent() method
  - Allows you to draw multiple objects without erasing previous objects
  - WARNING: You CANNOT call repaint() or getGraphics() from a constructor...the window/panel must be displayed before these methods can be used
DRAWING METHODS

- **Lines**
  - `g.drawLine(x1, y1, x2, y2);`

- **Rectangles and Squares**
  - `g.drawRect(x, y, width, height);`
  - `g.fillRect(x, y, width, height);`
  - `g.drawRoundRect(x, y, width, height, arcWidth, arcHeight);`

- **Ovals and Circles**
  - `g.drawOval(x, y, width, height);`
    - `x` and `y` specify the upper left corner of the rectangle that surrounds the oval
    - *Making width and height equal will draw a circle*
  - `g.fillOval(x, y, width, height);`
**DRAWING METHODS (CONT)**

- **Arcs**
  - `g.drawArc(x, y, width, height, startAngle, arcAngle);`
    - `x` and `y` are the upper left corner of the surrounding rectangle
    - 3 o’clock is considered 0 degrees
    - A positive `arcAngle` moves that far counterclockwise from the `startAngle`
    - A negative `arcAngle` moves that far clockwise from the `startAngle`
  - `g.fillArc(x, y, width, height, startAngle, arcAngle);`

- **Polygons**
  - `g.drawPolygon(int x[ ], int y[ ], int n);`
    - `n` is the number of sides
    - The two arrays provide pairs of coordinates
Words
- `g.drawString(thestring, x, y);`
  - `(x, y)` is the base of the first character

Color
- `Color myColor = g.getColor();`
- `g.setColor(new Color(r, g, b));`
  - Can use `Color.red` (or any other standard word)
  - `r, g,` and `b` must be between 0 and 255
ADD MENUS

- Declare private variables for the menu bar, each menu, and each menu item
  - private JMenuBar menubar;
  - private JMenu filemenu;
  - private JMenuItem one, two;

- Add the menus in the constructor
  - menubar = new JMenuBar();
  - filemenu = new JMenu("File");
  - one = new JMenuItem("Option 1")
  - two = new JMenuItem("Option 2");
  - menubar.add(filemenu);
  - filemenu.add(one);
  - filemenu.add(two);
  - setJMenuBar(menubar);

- Use the ActionListener for each menuitem with private inner classes (just like you do for buttons)
  - one.addActionListener(new MyMenu());