April 26: Intro to Graphics

CMPSCI 121, Spring 2012

Introduction to Problem Solving with Computers

Prof. Learned-Miller

Logistics

- Tuesday’s lecture is required.
  - Bring an id. It’s worth 5 points.
- Review for final is in Monday’s section.
  - A review sheet will be handed out, but possibly not before the review session.
- I will send out final make-up times via email.

Graphics

- The good old days:
  1. Clear the whole screen.
  2. Start drawing stuff.
  3. You are completely in control.
     1. No multi-tasking.
     2. No multiple windows to get in the way.
     3. No "repainting".

Graphics today

- Complex interaction with the windowing system.
- Complex interactions with other processes running on the same computer:
  - Word processor
  - Spread sheets
  - etc.

Challenges

- How do we make animations happen at a fixed speed?
- Windowing systems:
  - How do we save a window's contents when it is "minimized"?
  - How do we repaint a window when it is brought to the front?
- Internet games: how to keep game going during internet "glitches"?
- How do we "wake up" the computer when a button is pressed?

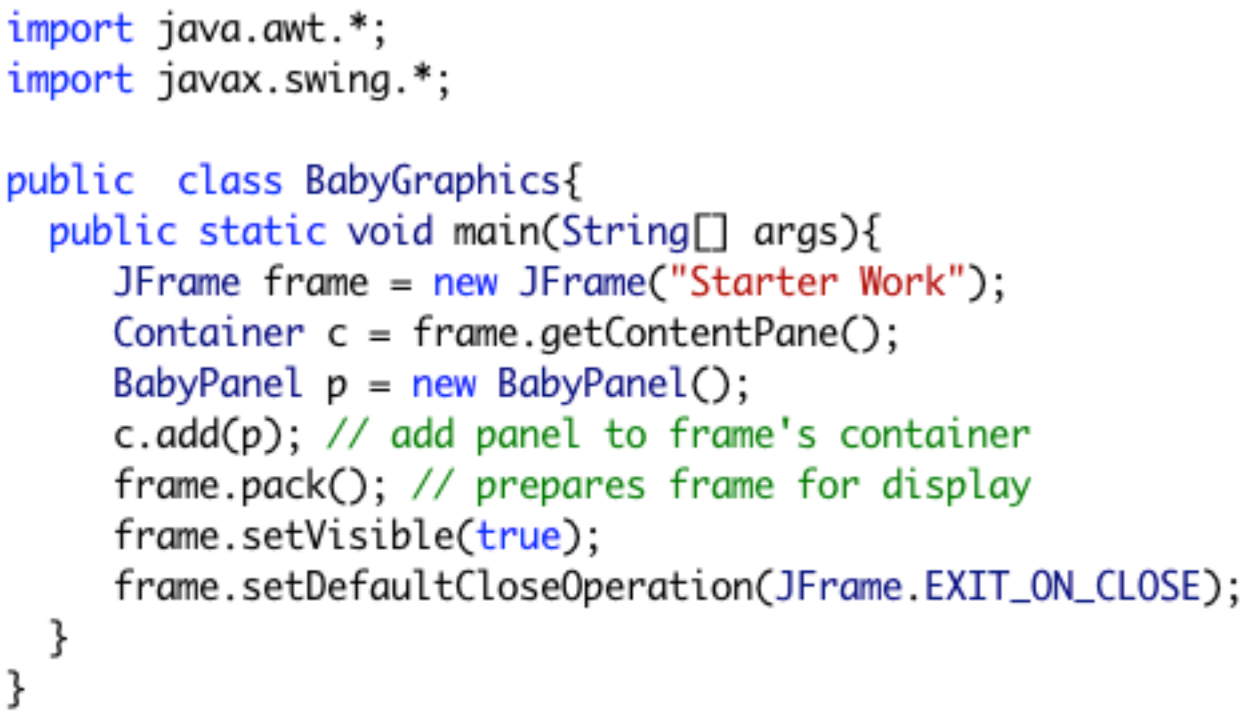
Strategy for Working with Graphics

- "Borrow and modify"
  - Find a program that does something like (but not too much like) what you want, for example:
    - Draws something once
    - Animates something
    - Uses a slider bar.
    - Uses a text editing window.
  - Modify it.
- Careful not to violate copyright or other rights!
- Cite source when in doubt.

Strategy continued

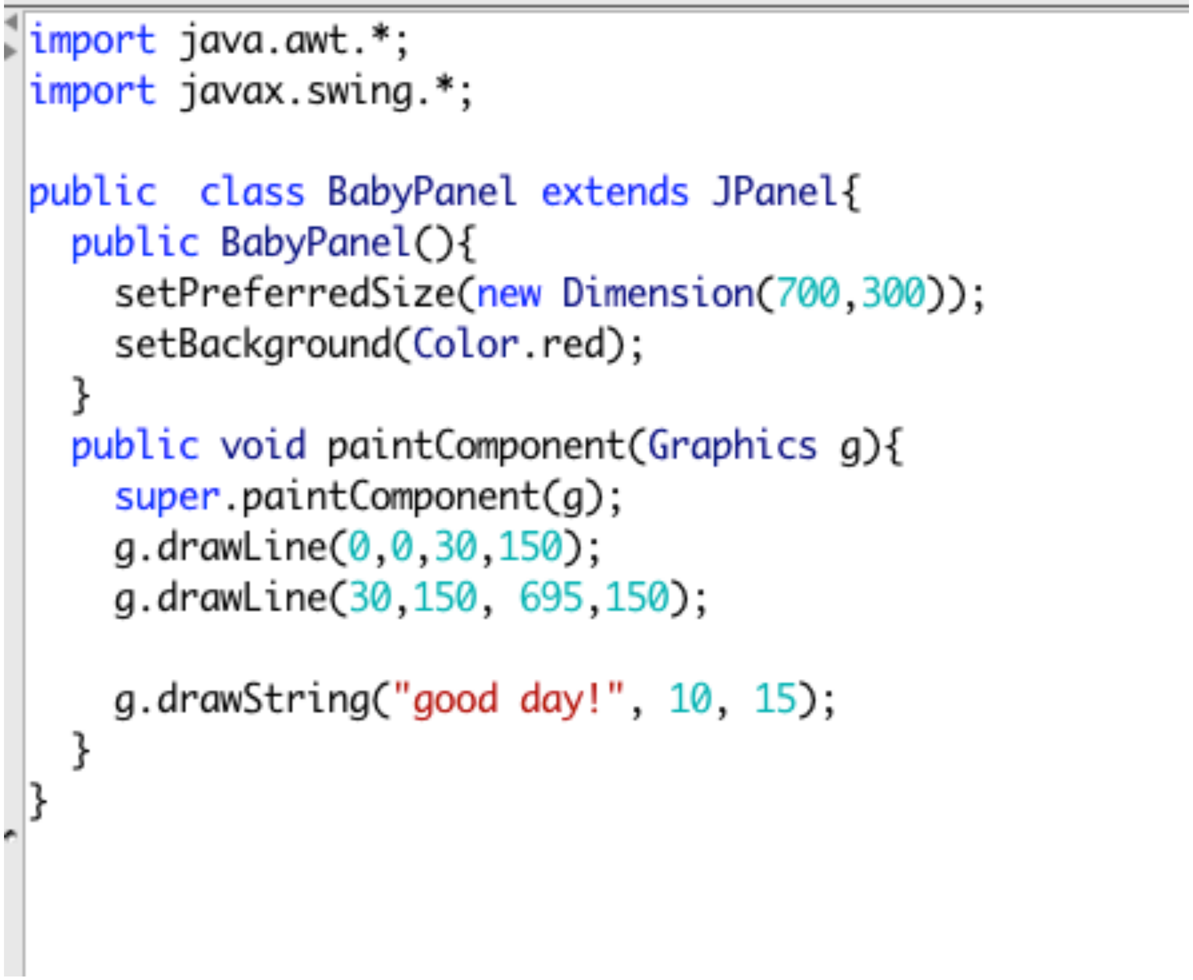
- If you want to change colors
  - find out where colors are set, and change that part of the code.
- If you want to change what is drawn:
  - find that part of the code and change it
- If you want to add more buttons
  - find that part of the code and add more of the same stuff

Simple graphics



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

public  class BabyGraphics{
  public static void main(String[] args){
      JFrame frame = new JFrame("Starter Work");
      Container c = frame.getContentPane();
      BabyPanel p = new BabyPanel();
      c.add(p); // add panel to frame's container
      frame.pack(); // prepares frame for display
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Simple graphics



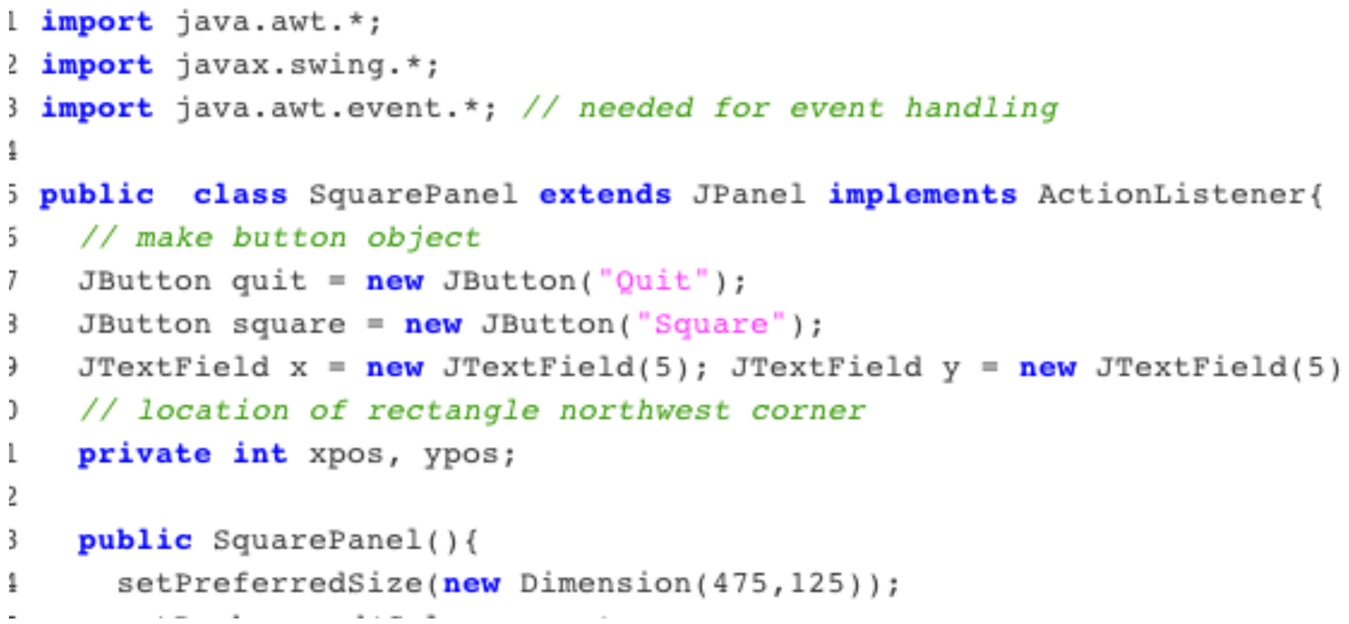
```
import java.awt.*;
import javax.swing.*;

public  class BabyPanel extends JPanel{
  public BabyPanel(){
    setPreferredSize(new Dimension(700,300));
    setBackground(Color.red);
  }
  public void paintComponent(Graphics g){
    super.paintComponent(g);
    g.drawLine(0,0,30,150);
    g.drawLine(30,150, 695,150);

    g.drawString("good day!", 10, 15);
  }
}
```

The paintComponent() method

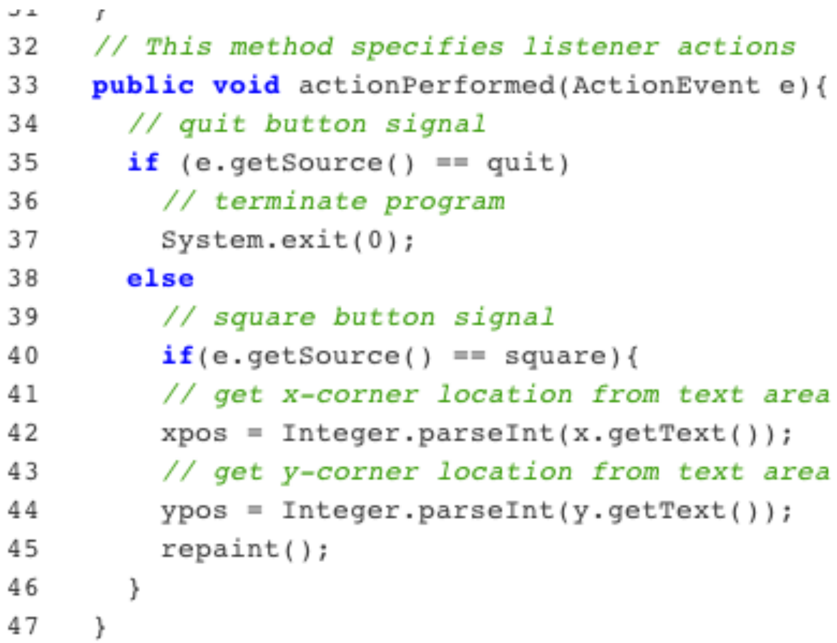
- paintComponent() is where the drawing happens
- method of JPanel and subclasses
- Do not call directly!
  - paintComponent() is called by the system (weird!)
  - Call repaint() to force a call to paint()

Actions and Listeners

- Want to be able to add "actionable items" to interfaces:
  - Buttons
  - Sliders
  - Keystrokes
  - Mouse movements
- Steps:
  - Create button
  - Add button to panel
    - Does graphics part, but button doesn't know where to send it's information.
  - Tell button to send its info to panel:
    - myButton.addActionListener(panel);



```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*; // needed for event handling

public  class SquarePanel extends JPanel implements ActionListener{
  // make button object
  JButton quit = new JButton("Quit");
  JButton square = new JButton("Square");
  JTextField x = new JTextField(5); JTextField y = new JTextField(5)
  // location of rectangle northwest corner
  private int xpos, ypos;

  public SquarePanel(){
    setPreferredSize(new Dimension(475,125));
```



```
}
// This method specifies listener actions
public void actionPerformed(ActionEvent e){
  // quit button signal
  if (e.getSource() == quit)
    // terminate program
    System.exit(0);
  else
    // square button signal
    if(e.getSource() == square){
    // get x-corner location from text area
    xpos = Integer.parseInt(x.getText());
    // get y-corner location from text area
    ypos = Integer.parseInt(y.getText());
    repaint();
  }
}
```

DrJava