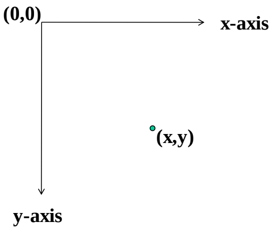
CS 325 - Class 18

- Today
  - Java’s Swing library
  - Basic graphics
  - Basic GUI components
- Announcements
  - Project 3 is due tonight by midnight
  - Start on Project 4

Graphical coordinate system



Graphics: drawing shapes

- We can draw many things including:
  - Lines
  - Rectangles
  - Strings
  - Ovals
- First we try drawing (or “painting”) directly onto the frame itself
  - JFrame has a paint( ) method that currently just paints the background a very light gray color
  - We will define a subclass of JFrame and override this paint( ) method

Example: painting

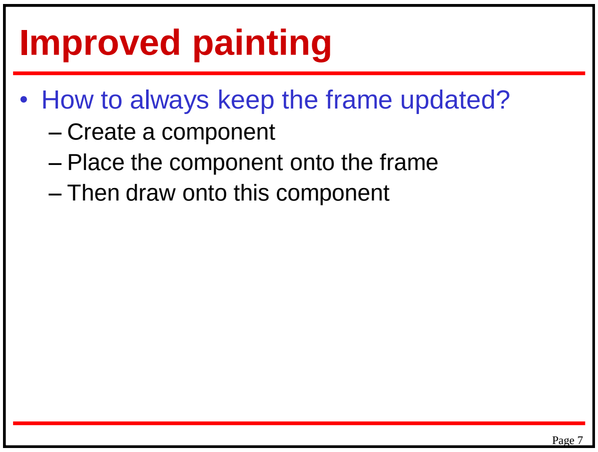
```
public class ExamplePaintFrame extends JFrame {

   public static void main(String[] args) {
         JFrame frame = new ExamplePaintFrame("Paint Frame");
         frame.setDefaultCloseOperation
                                (JFrame.EXIT_ON_CLOSE);
         frame.setSize(300,250);
         frame.setVisible(true);
   }

//  continued on next slide
```

Example: painting (cont.)

```
   public ExamplePaintFrame(String s) { super(s); }
                        // invokes constructor of superclass JFrame

   public void paint(Graphics g) {
         super.paint(g);              // paints the background
         g.drawLine(50, 50,  75, 75);
         g.drawRect(125, 50,  25, 50);
         g.fillRect(200, 50,  50, 25);
         g.drawString("Hello", 50, 150);
         g.drawOval(125, 150,  25, 50);
         g.fillOval(200, 150,  50, 25);
   }
}
```

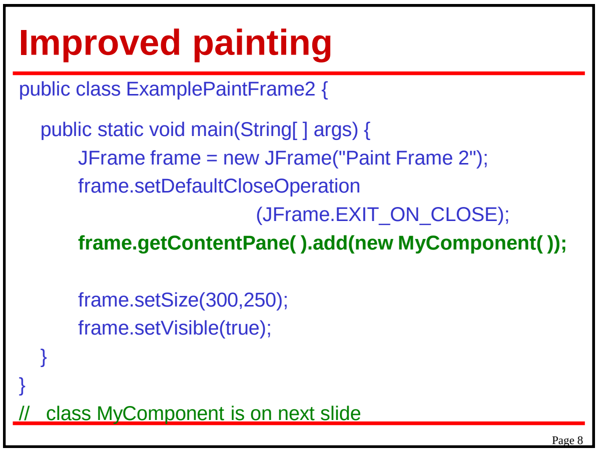
Class Exercise

- Download from **http://cs.ua.edu/325/Summer2007/examples/ExamplePaintFrame.java**
- Compile and run
- Look closely at the graphics
  - Can you figure out the meaning of each parameter value? (Hint: think of the coordinate system)
  - What happens if you move, minimize, or resize the frame?
  - What happens if you cover the frame with another window, and then uncover it?



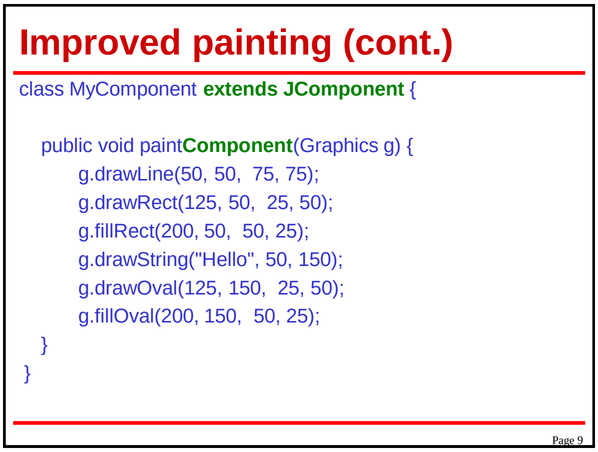
Improved painting

- How to always keep the frame updated?
  - Create a component
  - Place the component onto the frame
  - Then draw onto this component



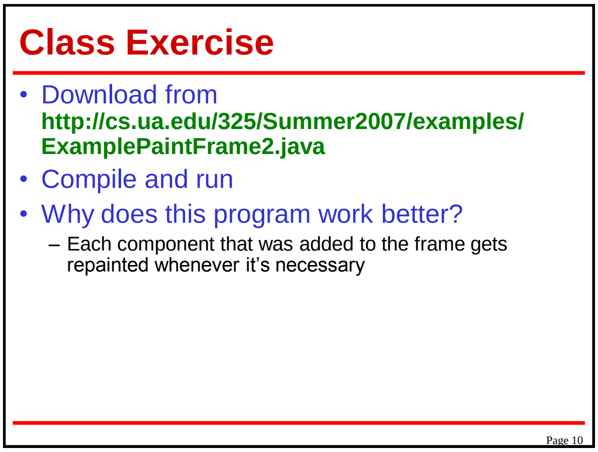
Improved painting

```
public class ExamplePaintFrame2 {

   public static void main(String[ ] args) {
         JFrame frame = new JFrame("Paint Frame 2");
         frame.setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);
         frame.getContentPane( ).add(new MyComponent( ));

         frame.setSize(300,250);
         frame.setVisible(true);
   }
}
//   class MyComponent is on next slide
```



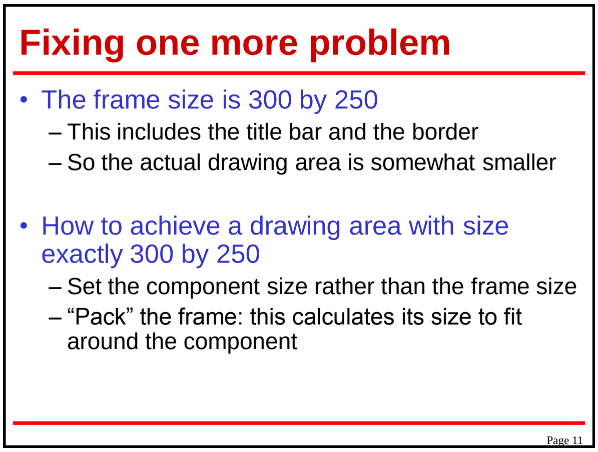
Improved painting (cont.)

```
class MyComponent extends JComponent {

   public void paintComponent(Graphics g) {
         g.drawLine(50, 50,  75, 75);
         g.drawRect(125, 50,  25, 50);
         g.fillRect(200, 50,  50, 25);
         g.drawString("Hello", 50, 150);
         g.drawOval(125, 150,  25, 50);
         g.fillOval(200, 150,  50, 25);
   }
}
```



Class Exercise

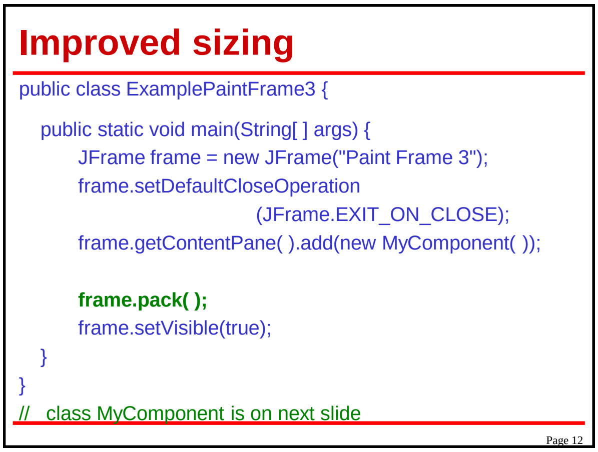
- Download from **http://cs.ua.edu/325/Summer2007/examples/ExamplePaintFrame2.java**
- Compile and run
- Why does this program work better?
  - Each component that was added to the frame gets repainted whenever it’s necessary



Fixing one more problem

- The frame size is 300 by 250
  - This includes the title bar and the border
  - So the actual drawing area is somewhat smaller

- How to achieve a drawing area with size exactly 300 by 250
  - Set the component size rather than the frame size
  - “Pack” the frame: this calculates its size to fit around the component



Improved sizing

```
public class ExamplePaintFrame3 {

   public static void main(String[ ] args) {
         JFrame frame = new JFrame("Paint Frame 3");
         frame.setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);
         frame.getContentPane( ).add(new MyComponent( ));

         frame.pack( );
         frame.setVisible(true);
   }
}
//   class MyComponent is on next slide
```

Improved sizing (cont.)

```
class MyComponent extends JComponent {

   private Dimension preferredSize =
                       new Dimension(300, 250);

   public Dimension getPreferredSize( ) {
         return preferredSize;
   }

   public void paintComponent(Graphics g) {
         …       // first do same code as before
         g.drawRect(2, 2, 295, 245);
   }
}
```

Class Exercise

- Download from **http://cs.ua.edu/325/Summer2007/examples/ExamplePaintFrame3.java**
- Compile and run
- Verify that the new rectangle fits inside the drawing area

Graphical User Interface (GUI)

- We can add many GUI components to our Java/Swing program, including:
  - Labels (JLabel)
  - Buttons (JButton)
  - Textfields (JTextField)
- These components will be illustrated in subsequent sample programs
- Labels are passive, but the other components require listening for events to occur

Example: JLabel

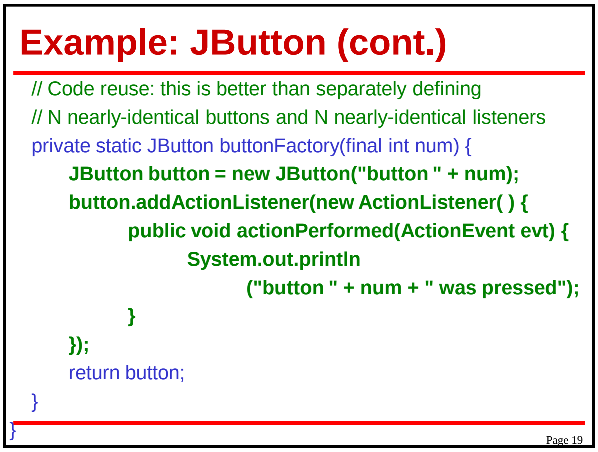
```
public class ExampleJLabel {
   public static void main(String[ ] args) {
         JFrame frame = new JFrame("JLabel example");
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         JPanel panel = (JPanel)frame.getContentPane( );
         panel.setLayout(new GridLayout(2,3));
         panel.add(new JLabel("This is a label"));
         panel.add(new JLabel("Second label"));
         …
         panel.add(new JLabel("This is the last label"));
         frame.pack();
         frame.setVisible(true);
   }
}
```

Class Exercise

- Download from **http://cs.ua.edu/325/Summer2007/examples/ExampleJLabel.java**
- Compile and run
- This program does not do anything
  - No events occur with labels
- Look at the layout of the components
  - Can you figure out the meaning of the parameters passed to GridLayout?

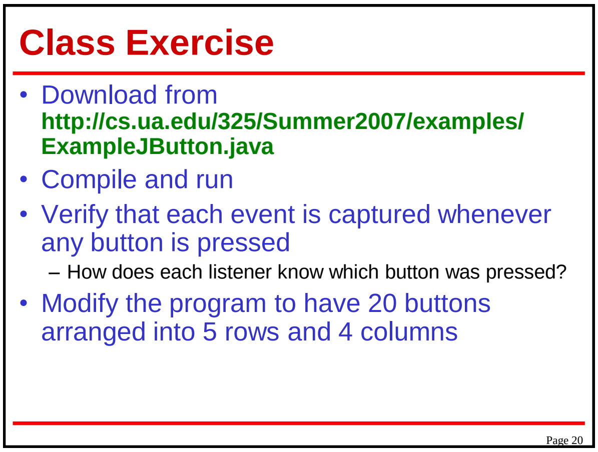
Example: JButton

```
public class ExampleJButton {

   public static void main(String[ ] args) {
         JFrame frame = new JFrame("JButton example");
         frame.setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);
         JPanel content = (JPanel)frame.getContentPane();
         content.setLayout(new GridLayout(2,3));
         for (int k=1; k<=6; k++)
               content.add(buttonFactory(k));      // see next slide
         frame.pack();
         frame.setVisible(true);
   }
```



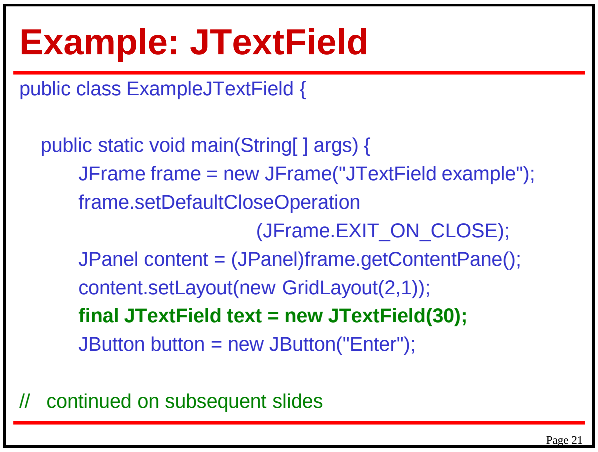
Example: JButton (cont.)

```
  // Code reuse: this is better than separately defining
  // N nearly-identical buttons and N nearly-identical listeners
  private static JButton buttonFactory(final int num) {
      JButton button = new JButton("button " + num);
      button.addActionListener(new ActionListener( ) {
          public void actionPerformed(ActionEvent evt) {
              System.out.println
                  ("button " + num + " was pressed");
          }
      });
      return button;
  }
}
```



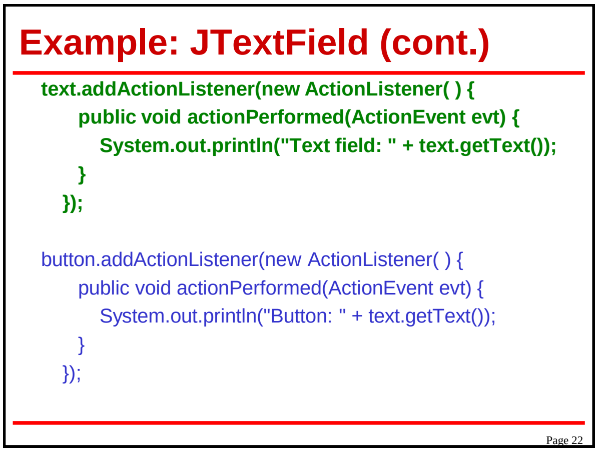
Class Exercise

- Download from **http://cs.ua.edu/325/Summer2007/examples/ExampleJButton.java**
- Compile and run
- Verify that each event is captured whenever any button is pressed
  - – How does each listener know which button was pressed?
- Modify the program to have 20 buttons arranged into 5 rows and 4 columns



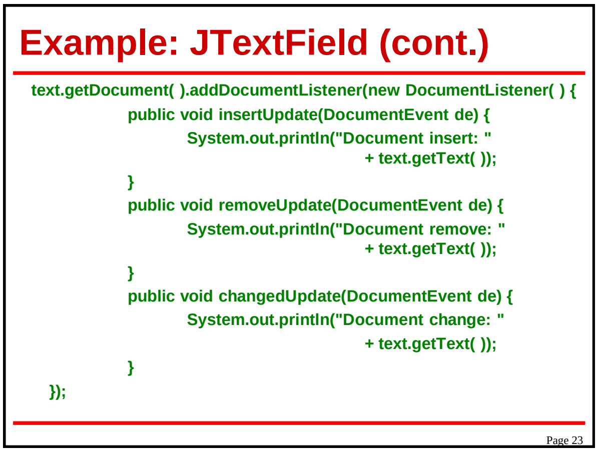
Example: JTextField

```
public class ExampleJTextField {

   public static void main(String[ ] args) {
        JFrame frame = new JFrame("JTextField example");
        frame.setDefaultCloseOperation
                              (JFrame.EXIT_ON_CLOSE);
        JPanel content = (JPanel)frame.getContentPane();
        content.setLayout(new GridLayout(2,1));
        final JTextField text = new JTextField(30);
        JButton button = new JButton("Enter");

//   continued on subsequent slides
```



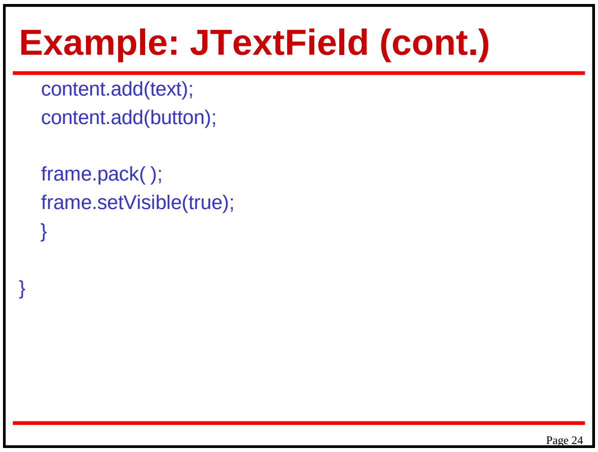
Example: JTextField (cont.)

```
   text.addActionListener(new ActionListener( ) {
        public void actionPerformed(ActionEvent evt) {
           System.out.println("Text field: " + text.getText());
        }
      });

   button.addActionListener(new ActionListener( ) {
        public void actionPerformed(ActionEvent evt) {
           System.out.println("Button: " + text.getText());
        }
      });
```



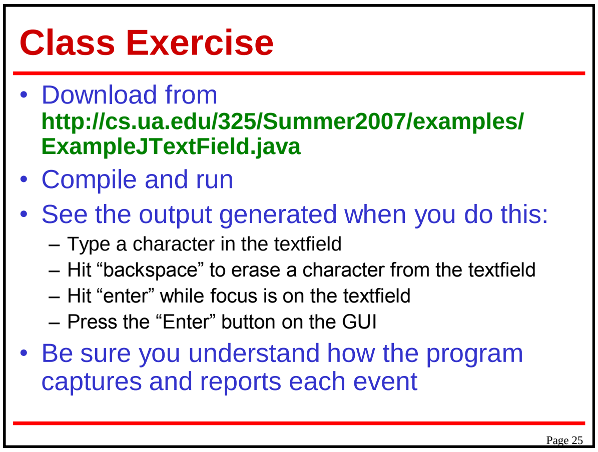
Example: JTextField (cont.)

```
  text.getDocument( ).addDocumentListener(new DocumentListener( ) {
               public void insertUpdate(DocumentEvent de) {
                       System.out.println("Document insert: "
                                                + text.getText( ));
               }
               public void removeUpdate(DocumentEvent de) {
                       System.out.println("Document remove: "
                                                + text.getText( ));
               }
               public void changedUpdate(DocumentEvent de) {
                       System.out.println("Document change: "
                                                + text.getText( ));
               }
     });
```



Example: JTextField (cont.)

```
   content.add(text);
   content.add(button);

   frame.pack( );
   frame.setVisible(true);
   }

}
```



Class Exercise

- Download from **http://cs.ua.edu/325/Summer2007/examples/ExampleJTextField.java**
- Compile and run
- See the output generated when you do this:
  - Type a character in the textfield
  - Hit “backspace” to erase a character from the textfield
  - Hit “enter” while focus is on the textfield
  - Press the “Enter” button on the GUI
- Be sure you understand how the program captures and reports each event