An Introduction To Graphical User Interfaces

Part 3: You will learn different ways of having one control access other controls as well as learning the basics of some additional java components. In addition, important non-GUI concepts such anonymous classes/objects, inner classes, friend functions are introduced.

Components Effecting The State Of Other Components

- **Name of the folder containing the full example**: `6controlAffectControls`
  - Access to other controls will occur via the existing Java clases and methods: `getContentPane()`, `getComponent()`.

Components Effecting The State Of Other Components: The `Driver` Class

```
public class Driver
{
     public static final int WIDTH = 800;
     public static final int HEIGHT = 600;
     public static void main(String [] args)
     {
           MyFrame aFrame = new MyFrame();
           aFrame.setSize(WIDTH,HEIGHT);
           aFrame.setVisible(true);
     }
}
```

Components Effecting The State Of Other Components: Class MyFrame

```
public class MyFrame extends JFrame
{
    private JLabel aLabel1;
    private JLabel aLabel2;
    private JButton aButton;
    private MyButtonListener aButtonListener;
```

Components Effecting The State Of Other Components: Class MyFrame (2)

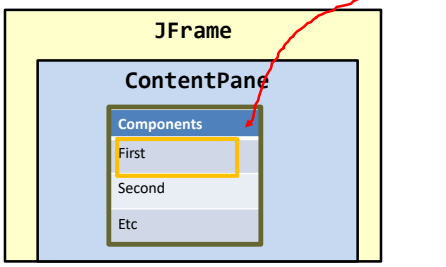
```
public MyFrame()
{
      MyWindowListener aWindowListener =
        new MyWindowListener();
      addWindowListener(aWindowListener);
      aLabel1 = new JLabel("Label 1");
      aLabel2 = new JLabel("Label 2");
      // Component.setBounds(x,y,w,h)
      aLabel1.setBounds(100,100,100,30);
      aLabel2.setBounds(300,100,100,30);
```

Components Effecting The State Of Other Components: Class MyFrame (3)

```
      aLabel1 = new JLabel("Label 1");
      aLabel2 = new JLabel("Label 2");
      aLabel1.setBounds(100,100,100,30);
      aLabel2.setBounds(300,100,100,30);
      aButtonListener = new MyButtonListener();
      aButton = new JButton("Press for multiple effects");
      aButton.addActionListener(aButtonListener);
      aButton.setBounds(150,300,200,50);
      add(aLabel1);
      add(aLabel2);
      add(aButton);
      setLayout(null);
    }
    public JLabel getLabel1() { return aLabel1; }
    public JLabel getLabel2() { return aLabel2; }
}
```

Note: JFrame Containment

- A JFrame actually contains just one GUI component, the content pane.
- GUI widgets that appear to be added to the JFrame are actually added to the content pane (a container in and of itself). Get the components inside the content pane to actually get the widgets that appeared to be added to the JFrame.




```
myFrame.add(aButton)
```

To access controls "added to the frame"

```
aContainer = aFrame.getContentPane()
component = aContainer.getComponent(0)
```

James Tam

Components Effecting The State Of Other Components: Class MyButtonListener

```
public void actionPerformed(ActionEvent e)
{
        JButton aButton = (JButton) e.getSource();
        MyFrame aFrame = (MyFrame)
        aButton.getRootPane().getParent();
        JLabel aLabel1 = aFrame.getLabel1();
        JLabel aLabel2 = aFrame.getLabel2();

        Container aContainer = aFrame.getContentPane();
        // First item added to list, first label
        Component aComponent = aContainer.getComponent(0);
        if (aComponent instanceof JLabel) {
            aLabel1 = (JLabel) aComponent;
            aLabel1.setText("Effect1");
        }
```

James Tam

Components Effecting The State Of Other Components: Class MyButtonListener (2)

```
        // Second item added to list, second label
        aComponent = aContainer.getComponent(1);
        if (aComponent instanceof JLabel) {
            aLabel2 = (JLabel) aComponent;
            aLabel2.setText("Effect1");
        }
}
```

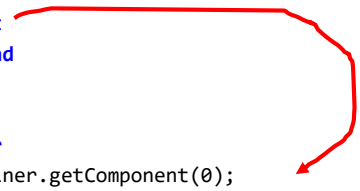
James Tam

Last Example: Critique

- The implementation of the button listener class required knowledge of the implementation of the frame listener class.
  - The order in which controls are added to the frame must be known!
  - What if there are two different authors for these classes?
  - This approach couples the implementation of two classes (changes can introduce errors)

```
// From class MyFrame
 add(aLabel1);  // Added first
 add(aLabel2);  // Added second
 add(aButton);

// From class MyButtonListener
Component aComponent = aContainer.getComponent(0);
if (aComponent instanceof JLabel) {
    aLabel1 = (JLabel) aComponent;
    aLabel1.setText("Effect1");
}
```



Components Effecting The State Of Other Components: Alternate Approach

- **Name of the full online example**: `7controlAffectControlsActionCommand`
  - –Identifying controls via the: `actionCommandAttribute`

The `Driver` Class

```
public class Driver
{
     public static final int WIDTH = 800;
     public static final int HEIGHT = 600;
     public static void main(String [] args)
     {
         MyFrame aFrame = new MyFrame ();
         aFrame.setSize(WIDTH,HEIGHT);
         aFrame.setVisible(true);
     }
}
```

James Tam

Class MyFrame

```
public class MyFrame extends JFrame {
    public static final String B1IDENTIFIER = "1";
    public static final String B2IDENTIFIER = "2";
    private JButton button1;
    private JButton button2;
    private MyButtonListener aButtonListener;

    public MyFrame() {
        MyWindowListener aWindowListener = new
        MyWindowListener();
        addWindowListener(aWindowListener);
        aButtonListener = new MyButtonListener();
```

James Tam

Class MyFrame (2)

```
        button1 = new JButton("Button1");
        button1.setActionCommand(B1IDENTIFIER);
        button1.setBounds(100,100,100,30);
        button1.addActionListener(aButtonListener);

        button2 = new JButton("Button2");
        button2.setActionCommand(B2IDENTIFIER);
        button2.setBounds(300,100,100,30);
        button2.addActionListener(aButtonListener);

        add(button1);
        add(button2);
        setLayout(null);
    }

}
```

James Tam

Class MyButtonListener

- **Identifying the buttons**

```
public class MyButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        JButton aButton = (JButton) e.getSource();
        MyFrame aFrame = (MyFrame)
        aButton.getRootPane().getParent();
        String temp = aButton.getActionCommand();
        if(temp.equalsIgnoreCase(MyFrame.B1IDENTIFIER))
            aFrame.setTitle("Button 1 pressed");
        else if(temp.equalsIgnoreCase(MyFrame.B2IDENTIFIER))
           aFrame.setTitle("Button 2 pressed");
    }
}
```

James Tam

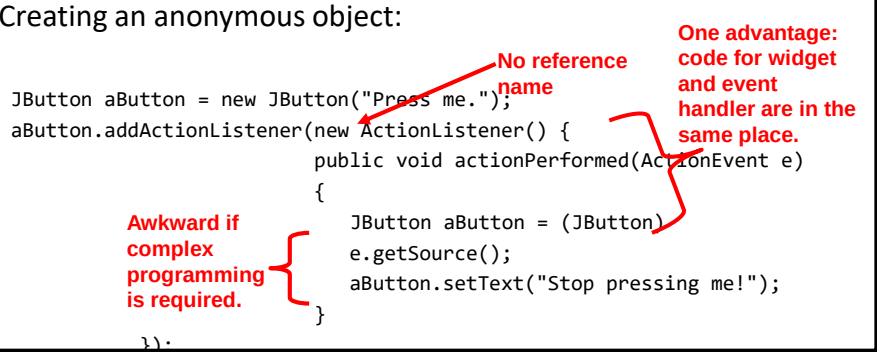
This Version: Critique

- There was one method handles events for all the buttons.
- Inside that method there was a need to 'identify' the source of the event.
  - The method could get very long even though there are few sources of events (buttons)
  - What if the GUI has dozens of buttons or other controls

```
public void actionPerformed(ActionEvent e)
{
   String s = e.getActionCommand();
   if(temp.equalsIgnoreCase(MyFrame.B1IDENTIFIER))
       aFrame.setTitle("Button 1 pressed");
   else if(temp.equalsIgnoreCase(MyFrame.B2IDENTIFIER))
       aFrame.setTitle("Button 2 pressed");
}
```

Anonymous Objects/Anonymous Class

- If an object needs to be created but never directly referenced then it may be candidate for being created as an anonymous object.
- An example of where an anonymous object may be created is an event listener.
- Creating an anonymous object:



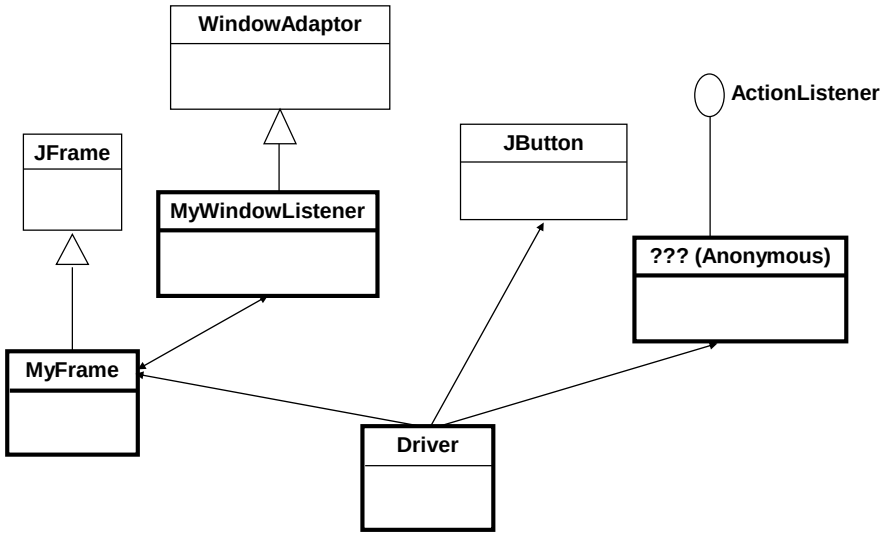
An Example Using Anonymous Class And Object

- **Name of the folder containing the complete example**:

`8controlAffectControlsAnonymousObjectClass`

  - Anonymous objects/classes are used for the listeners (nested inside the container).
  - GUI component are then accessed via the containers accessor methods.

An Example Using Anonymous Class And Object (2)



`Driver` Class

```
public class Driver
{
     public static final int WIDTH = 400;
     public static final int HEIGHT = 300;
     public static void main(String [] args)
     {
         MyFrame aFrame = new MyFrame ();
         aFrame.setTitle("Original");
         aFrame.setSize(WIDTH,HEIGHT);
         aFrame.setVisible(true);
     }
}
```

Class MyFrame

```
public class MyFrame extends JFrame
{
    private JLabel aLabel;
    private GridBagLayout aLayout;
    private GridBagConstraints aConstraint;
    private JButton left;
    private JButton right;
```

Class MyFrame (2)

```
    public MyFrame() {
        MyWindowListener aWindowListener =
          new MyWindowListener();
        addWindowListener(aWindowListener);
        aConstraint = new GridBagConstraints();

       left = new JButton("LEFT: Press right button.");
       left.setBackground(Color.lightGray);
```

Class MyFrame (3)

```
left.addActionListener(new ActionListener()
{  // class definition
    public void actionPerformed(ActionEvent e)  {
        // method definition: left button
        JButton left = (JButton) e.getSource();
        MyFrame aFrame = (MyFrame)
          left.getRootPane().getParent();
        String title = aFrame.getTitle();
        aFrame.setTitle("Left pressed");
        right = aFrame.getRight();
        right.setBackground(Color.green);
        left.setBackground(Color.lightGray);
        timeDelay();
        aFrame.setTitle(title);
    } // End method definition
 } // End class definition
); // End of parameter list for addActionListener()
```

James Tam

Class MyFrame (4)

```
right = new JButton("RIGHT: Press left button");
right.setBackground(Color.lightGray);
right.addActionListener(new ActionListener()
{  // Class definition
    public void actionPerformed(ActionEvent e)  {
        // Method definition
        JButton right = (JButton) e.getSource();
        MyFrame aFrame = (MyFrame)
        right.getRootPane().getParent();
        String title = aFrame.getTitle();
        JButton left = aFrame.getLeft();
        aFrame.setTitle("Right pressed");
        left.setBackground(Color.green);
        right.setBackground(Color.lightGray);
        timeDelay();
        aFrame.setTitle(title);
    }
});
```

James Tam

Class MyFrame (5)

```
    private void timeDelay()
    {
        try {
               Thread.sleep(3000);
        }
        catch (InterruptedException e) {
               System.out.println("Problem with pausing of the
                 program");
        }
    }
    public JButton getLeft() {
        return(left);
    }
    public JButton getRight() {
        return(right);
    }
}
```

James Tam

'Friend Functions'

- Some programming languages allow classes to be 'friendly'.
- A method can be declared in class X so it's accessible by another class Y even though the method is outside of the scope of class Y.
- The 'friendly' method of class X allows access to all of the privates & protected parts of X to instances of Y.
- It's used when instances of classes X & Y operate closely.
- Java does not directly allow for friend functions but other languages such as C++ do.
- Does this violate encapsulation?

James Tam

Nested/Inner Classes

- Occurs when one class (body) is defined inside of another class:

```
public class X {
     private class Y {

     }
}
```

Inner class

Outer class

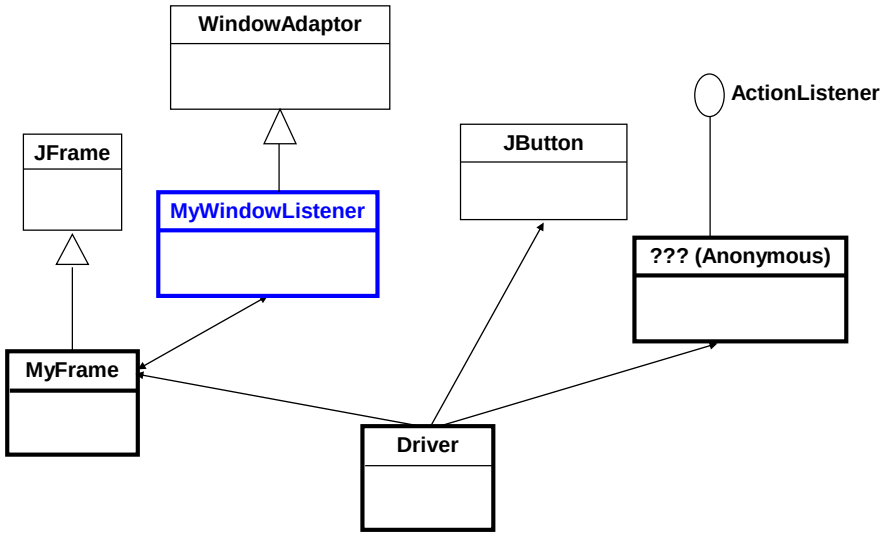
- Why nest class definitions[1]:
  - It is a way of logically grouping classes that are only used in one place.
  - Nested classes can lead to more readable and maintainable code.
  - It increases encapsulation (inner class hidden from all classes except the outer class).
- Similar to declaring anonymous objects, nesting classes may be used when creating event listeners.

1 For more information (last accessed 2021):
http://download.oracle.com/javase/tutorial/java/javaOO/nested.html

Example: Inner Classes

- **Name of the folder containing the full online example**:
`9buttonAlternateInnerClasses`
  - The example again employs an anonymous object/class (not a new concept).
  - What's **new** is the use of a **class definition nested within another class definition**.

Example: **Inner Classes** (2)



The `Driver` Class

```
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Driver
{
     public static final int WIDTH = 300;
     public static final int HEIGHT = 200;
     public static void main(String [] args)
     {
         MyFrame aFrame = new MyFrame();
         aFrame.setSize(WIDTH,HEIGHT);
         JButton aButton = new JButton("Press me.");
```

The Driver Class (2)

```
        aButton.addActionListener(
                new ActionListener() // Anonymous object/class
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                JButton aButton = (JButton) e.getSource();
                                aButton.setText("Stop pressing me!");
                        } // End: Defining method actionPerformed
                }  // End: Defining anonymous object/class
              );  // End: Parameter list for addActionListener

              aFrame.add(aButton);
              aFrame.setVisible(true);
        }
}
```

Class MyFrame: Outline

```
public class MyFrame extends JFrame
{
        // MyFrame's private parts
         public MyFrame()
        {
                :    :
        }
        // Inner class defined within the MyFrame class.
        //  Private because it's only used by the MyFrame class.
        private class MyWindowListener extends WindowAdapter
        {
            public void windowClosing(WindowEvent e)
            {
                :    :
            }
        }
}
```

NOTE: The inner class can access the outer class' privates! "Friend"

Definition of class MyWindowListener entirely within definition of class MyFrame

- Listens for events for that window

Class MyFrame (2)

```
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class MyFrame extends JFrame
{
    public MyFrame ()
    {
        MyWindowListener aWindowListener = new
          MyWindowListener();
        this.addWindowListener(aWindowListener);
    }
```

Class MyFrame (3)

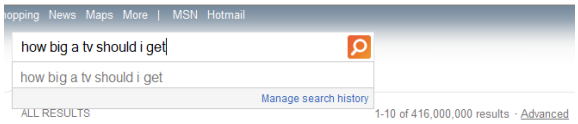
```
    // Inner class defined within the MyFrame class.
    // Private because it's only used by the MyFrame class.
    private class MyWindowListener extends WindowAdapter {
      public void windowClosing (WindowEvent e) {
         JFrame aFrame = (JFrame) e.getWindow();
         aFrame.setTitle("Closing window...");
         delay();
         aFrame.setVisible(false);
         aFrame.dispose();
      }
    }  // End: Definition of class MyWindowListener

    private void delay() {
         try  {
            Thread.sleep(3000); }
         catch (InterruptedException ex)  {
            System.out.println("Pausing of program was interrupted");
      }
} // End: Definition of class MyFrame
```

Proof that the inner class can access the outer class' privates

Types Of Input Text Fields: Short

- `JTextField`: Used to get short user input
  - e.g., entering login or personal information.

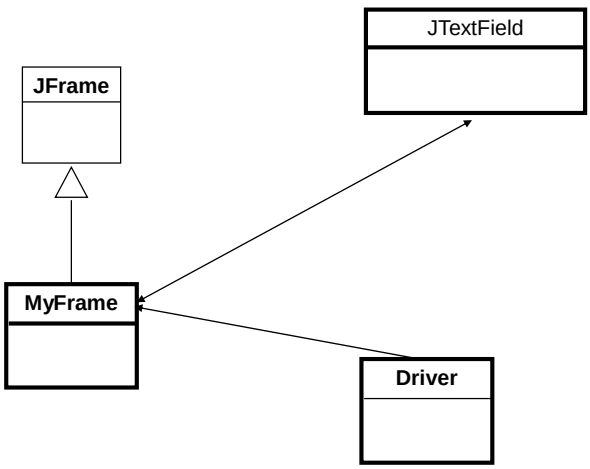


Bing search query

- **Name of the folder containing the full online example**:
  `10textFieldExample`

James Tam

Example: `JTextField` (2)



The Driver Class

```
public class Driver
{
    public static void main(String [] args)
    {
        MyFrame aFrame = new MyFrame();
    }
}
```

James Tam

Class MyFrame

```
public class MyFrame extends JFrame implements
ActionListener
{
    private JTextField text; // New control
    private GridBagLayout aLayout;
    private GridBagConstraints aConstraint;
```

James Tam

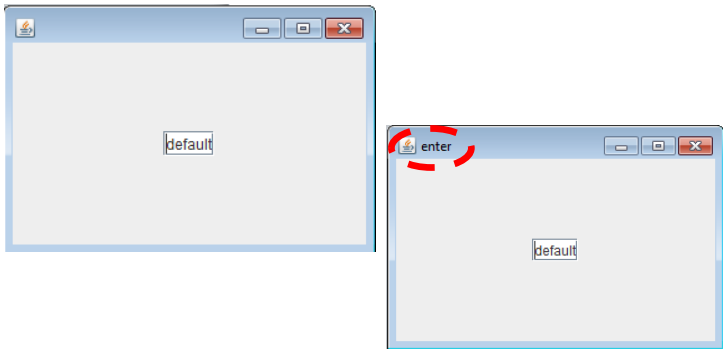
Class MyFrame: Using JTextField

```
public MyFrame()
{
  setSize(300,200);
  setDefaultCloseOperation
    (JFrame.DISPOSE_ON_CLOSE);
  aConstraint = new GridBagConstraints();
  aLayout = new GridBagLayout();
  setLayout(aLayout);
  text = new JTextField("default");
  text.addActionListener(this);
  addWidget(text,0,0,1,1);
  setVisible(true);
}
```

James Tam

Class MyFrame: Reacting To The Event

```
public void actionPerformed(ActionEvent e)
{
    setTitle("enter");
}
}
```



James Tam

Types Of Input Text Fields: Long

- Getting more extensive input
  - e.g., feedback form, user review/comments on a website
  - Requires the use of another control: `JTextArea`

Facebook status update field

- **Name of the folder containing the full online example**:
  `11textAreaExample`

James Tam

The `Driver` Class: Using `JTextArea`

```
public class Driver {
    public static void main(String [] args) {
        JFrame frame = new JFrame();
        frame.setSize(400,250);
        //New control
        JTextArea text = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(text);
        text.setFont(new Font("Times",Font.BOLD, 32));
        for (int i = 0;i < 10; i++)
            text.append("foo" + i + "\n");
        frame.add(scrollPane);
        MyDocumentListener listen = new MyDocumentListener();
        (text.getDocument()).addDocumentListener(listen);
        frame.setVisible(true);
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
}
```

James Tam

The Text Listener: MyDocumentListener

```
public class MyDocumentListener implements DocumentListener {
    public void changedUpdate(DocumentEvent e) {
       System.out.println("updated");
       displayChanges(e);
    }

    public void insertUpdate(DocumentEvent e) {
       System.out.println("insert");
       System.out.println(e.getLength());
       displayChanges(e);
    }

    public void removeUpdate(DocumentEvent e) {
       System.out.println("removed");
       displayChanges(e);
    }
```

James Tam

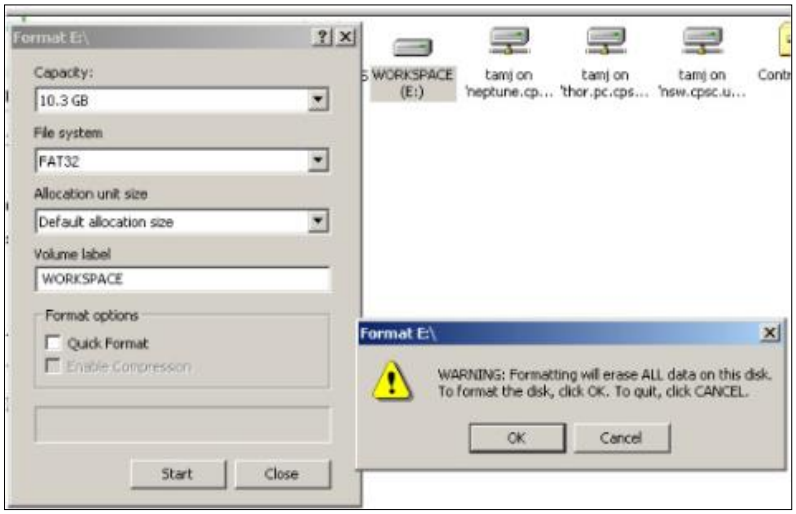
The Text Listener: MyDocumentListener (2)

```
    public void displayChanges(DocumentEvent e)
    {
       Document d = e.getDocument();
       try
       {
           String s = d.getText(0,d.getLength());
           System.out.println(s);
       }
       catch (BadLocationException ex)
       {
           System.out.println(ex);
       }
    }
}
```

James Tam

Dialog Boxes And "User-Friendly Design"

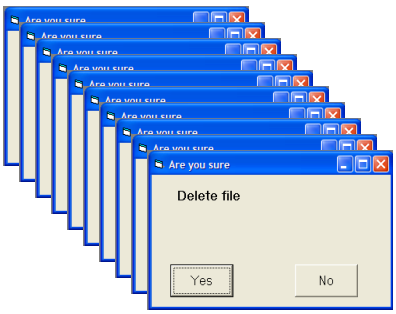
- Note: used *sparingly* dialog boxes can communicate important information or to prevent unintentional and undesired actions.



James Tam

Dialog Boxes And "User-Friendly Design" (2)

- They interupt the regular use of the program so make sure they are only used sparingly
  - …they can easily be over/misused!)

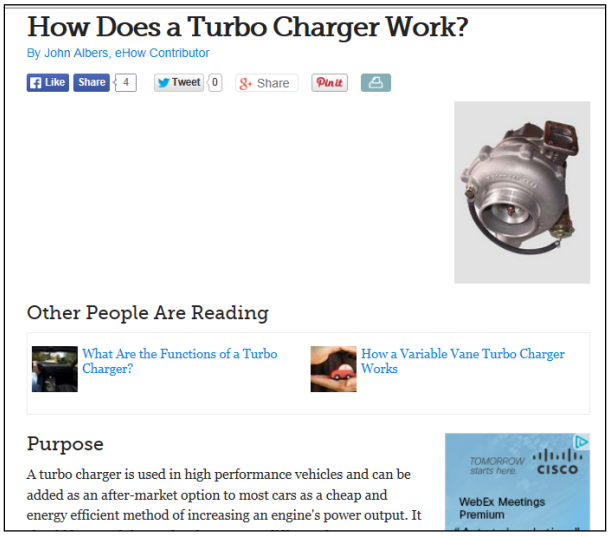



```
[csb 481 16 ]> rm *
rm: remove file1 (yes/no)? y
rm: remove file2 (yes/no)? y
rm: remove file3 (yes/no)? y
rm: remove file4 (yes/no)? y
rm: remove file5 (yes/no)?
```

James Tam

Dialogs Are Frequently Used Online

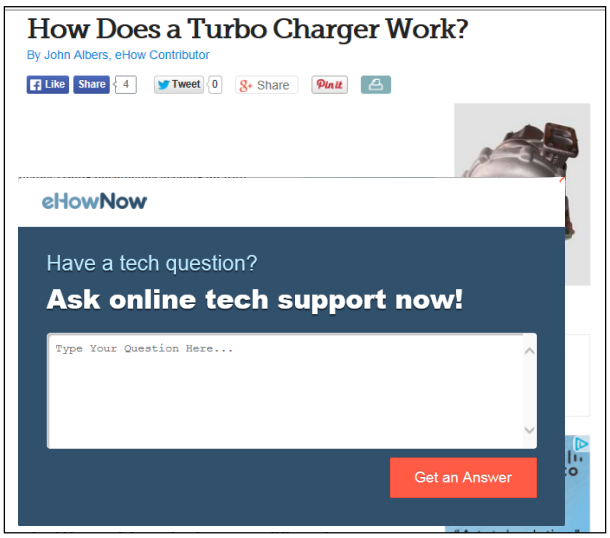
- Great! I've got the info that I need.



James Tam

Dialogs Are Frequently Used Online

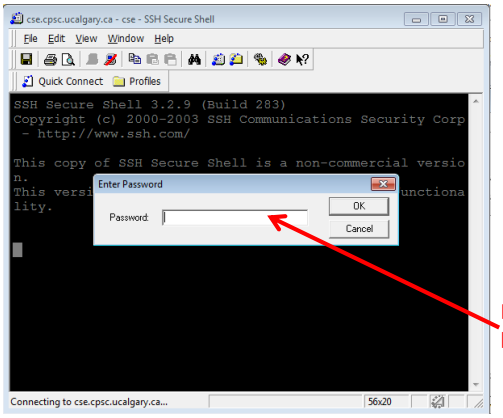
- Hey I was reading that!



James Tam

Dialog Boxes (If There Is Time)

- Typically take the form of a small window that 'pops up' during program execution.



Part of the login 'dialog'

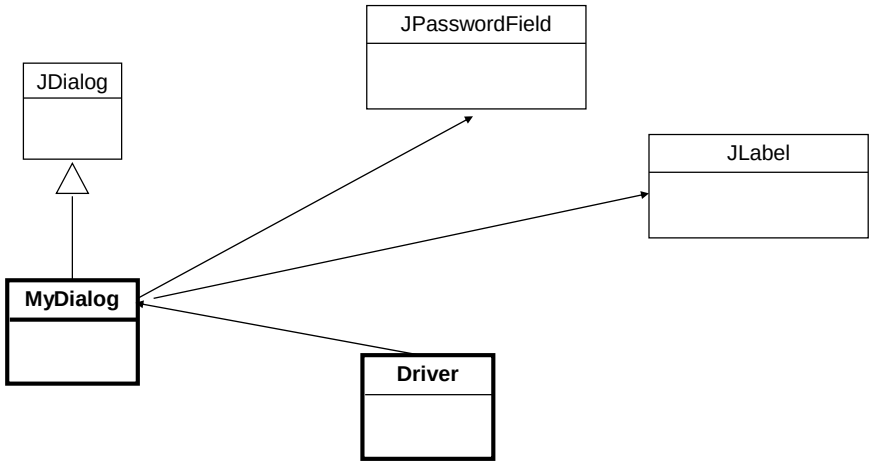
James Tam

JDialog Example

- **Name of the folder containing the full online example**:
  `12dialogExample`

James Tam

Example: JDialog (2)



The Driver Class

```
public class Driver
{
    public static void main(String [] args)
    {
        MyDialog aDialog = new MyDialog();
        aDialog.setBounds(100,100,300,200);
        aDialog.setVisible(true);
    }
}
```

James Tam

Class MyDialog

```
public class MyDialog extends JDialog implements ActionListener
{
    private static final int MATCH = 0;
    private static final String ACTUAL_PASSWORD = "123456";
    private JPasswordField aPasswordField;
    private JLabel aLabel;

    public MyDialog() {
        aLabel = new JLabel("Enter password");
        aLabel.setBounds(50,20,120,20);
        aPasswordField = new JPasswordField();
        aPasswordField.setBounds(50,40,120,20);
        aPasswordField.addActionListener(this);  //Event handler
        setLayout(null);
        addControls();  // Slide #2
        setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    }
```

James Tam

Class MyDialog (2)

```
    public void addControls()
    {
        add(aLabel);
        add(aPasswordField);
    }
```

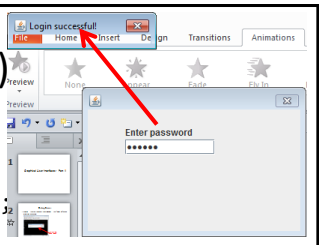
James Tam

Class MyDialog (3)

```
public void actionPerformed(ActionEvent e) {
    Component aComponent = (Component) e.getSource();
    if (aComponent instanceof JPasswordField) {
        JPasswordField aPasswordField =
          (JPasswordField) aComponent;
        String passWordEntered = new
          String(aPasswordField.getPassword());
        if (passWordEntered.compareTo(ACTUAL_PASSWORD)
                                        == MATCH)
            loginSuccess();  //Slide #4
        else
            loginFailed() //Slide #5
    }
}
```

James Tam

Class MyDialog (4)



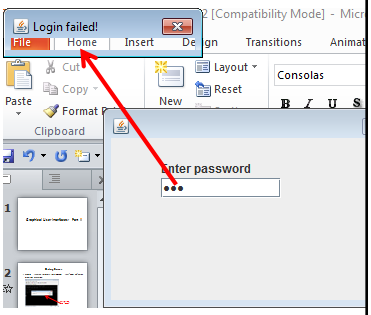

```
public void loginSuccess() {
    JDialog success = new JDialog();
    success.setTitle("Login successful!");
    success.setSize(200,50);
    success.setVisible(true);
    cleanUp(success);
}

public void cleanUp(JDialog popup) {
    try
        Thread.sleep(3000);
    catch (InterruptedException ex)
        System.out.println("Program interrupted");
    this.setVisible(false);
    this.dispose();
    popup.setVisible(false);
    popup.dispose();
    System.exit(0);  // Dialog cannot end whole program
}
```

James Tam

Class MyDialog (5)

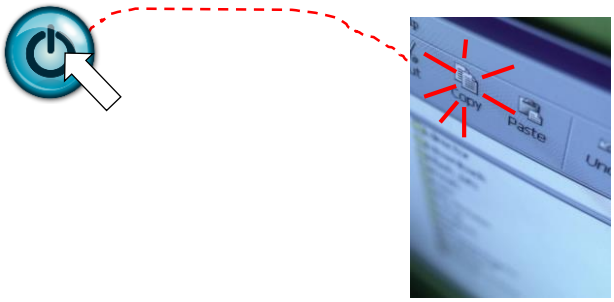
```
public void loginFailed()
{
    JDialog failed = new JDialog();
    failed.setTitle("Login failed!");
    failed.setSize(200,50);
    failed.setVisible(true);
    cleanUp(failed);
}
public void cleanUp(JDialog popup) {
    try
        Thread.sleep(3000);
    catch (InterruptedException ex)
        System.out.println("Program interrupted");
    this.setVisible(false);
    this.dispose();
    popup.setVisible(false);
    popup.dispose();
    System.exit(0);  // Dialog cannot end whole program
}
```



James Tam

Controls Affecting Other Controls

- As previously shown this is not an uncommon occurrence

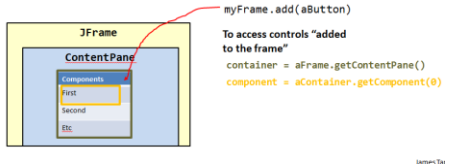


- The code to react to the event allows for easy access to the control that raised the event.

James Tam

Ways Of Accessing Other Controls

1. Via Java Swing containment
   - Example to illustrate with `JButton` control: `6controlAffectControls`



   - Stylistically acceptable (of course!)
   - Can be challenging to track down specific container/method

James Tam

Ways Of Accessing Other Controls (2)

2. Implementing the listener class as a nested inner class.
   - (Recall that if one class is defined inside the definition of another class that the inner class is within the scope of the outer class and as a consequence it can access private attributes or methods).
   - JT's $0.02: take care that you don't employ this technique too often and/or to bypass encapsulation/information hiding.

```
public class MyFrame extends JFrame {
    private JLabel a Label;
      ...
    private class MyWindowListener extends extends
      WindowAdapter {
          public void windowClosing  (WindowEvent e) {
              aLabel.setText("Shutting down");
          }
    } // End definition for inner window listener class
}  // End definition for outer frame class
```

James Tam

Ways Of Accessing Other Controls (3)

3. Adding the control as an attribute of the control that could raise the event.
   - Once you have access to the container then you can use accessor methods to get a reference to all the GUI components contained within that container.
   - The previously mentioned example (#6) illustrated this:

```
public class MyFrame extends Jframe {
     private JLabel aLabel1;
     private JLabel aLabel2;
       ...
     public JLabel getLabel1 () { return aLabel1; }
     public JLabel getLabel2 () { return aLabel2; }
}
```

   - JT's $0.02:
     - Replaces Java's containment with a simpler has-a relation that you created

James Tam

Ways Of Accessing Other Controls (4)

- Note: adding one control as an attribute of another control need not be limited only to actual 'containers' such as `JFrame` or `JDialog`
- Example (button event changes a label)

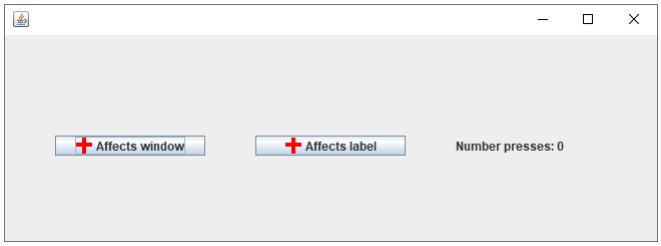
```
public class MyButton extends JButton {
     private JLabel aLabel;
     ...
     public Jlabel getLabel() {  return(aLabel); }
}

public class MyButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        MyButton aButton = (MyButton) e.getSource();
        JLabel aLabel = aButton.getLabel();
    }
}
```

James Tam

Adding Graphics To Java Controls

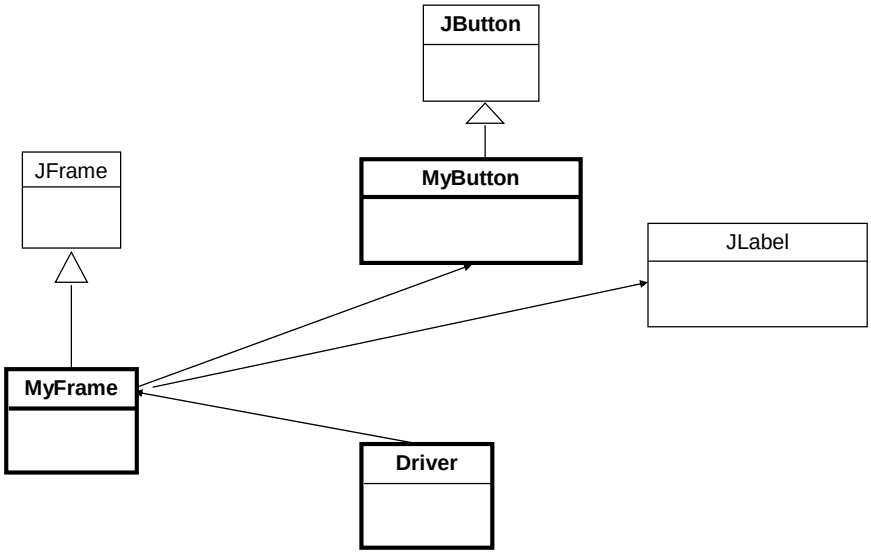
- **Name of the folder containing the full online example**: `13addingPicturesToControlsTrackingState`



- This program allows the state of the GUI (number of button presses) to be tracked and displayed.

James Tam

Example: Adding Pictures To Controls (2)



The Driver Class

```
public class Driver
{
    public static void main(String [] args)
    {
        MyFrame aFrame = new MyFrame();
        aFrame.setVisible(true);
    }
}
```

James Tam

Class MyFrame

```
public class MyFrame extends JFrame
{
    public static final String DEFAULT_LABEL_STRING = "Number
      presses: ";
    public static final int WIDTH = 700;
    public static final int HEIGHT = 300;
    private MyButton frameButton;
    private MyButton labelButton;
    private JLabel aLabel;
    private int numPresses;

    public MyFrame()
    {
        numPresses = 0;
        initializeControls();
        initializeFrame();
    }
```

James Tam

Class MyFrame (2)

```
public void addControls() {
    add(frameButton); //Calls to super class method
    add(labelButton);
    add(aLabel);
}

public JLabel getLabel() {
    return(aLabel);
}

public int getNumPresses() {
    return(numPresses);
}

public void incrementPresses() {
    numPresses++;
}
```

James Tam

Class MyFrame (3)

```
public void initializeFrame()
{
    setSize(WIDTH,HEIGHT);
    setLayout(null);
    addControls();
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

James Tam

Class MyFrame (4)

No path provided: location is the same directory as the program

```
    public void initializeControls() {
        ImageIcon anIcon = new ImageIcon("IconPic.gif");
        frameButton = new MyButton("Affects
                                   window",anIcon,this);
        frameButton.setBounds(50,100,150,20);
        frameButton.addActionListener
          (new FrameButtonListener()); // Frame events only
        labelButton = new MyButton("Affects label",anIcon,this);
        labelButton.setBounds(250,100,150,20);
        labelButton.addActionListener
          (new LabelButtonListener());  // Label events only
        aLabel = new JLabel(DEFAULT_LABEL_STRING +
                            Integer.toString(numPresses));
        aLabel.setBounds(450,100,150,20);
    }
}
```

James Tam

Class MyButton

Each instance will have a reference to a Java GUI widget (label, frame etc.)

Image reference passed onto the appropriate super class constructor

```
public class MyButton extends JButton
{
    private Component aComponent;

    public MyButton(String s,
                    ImageIcon pic,
                    Component aComponent)
    {
        super(s,pic);
        this.aComponent = aComponent;
    }

    public Component getComponent()
    {
        return(aComponent);
    }
}
```

James Tam

Class To Change Label: LabelButtonListener

```
public class LabelButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent anEvent)
    {
       MyButton aButton = (MyButton) anEvent.getSource();
       MyFrame aFrame = (MyFrame) aButton.getComponent();
          // Method of MyButton
       aFrame.incrementPresses();  // Frame stores count
       JLabel aLabel = aFrame.getLabel();
       String s = MyFrame.DEFAULT_LABEL_STRING;
       int currentPresses = aFrame.getNumPresses();
       s = s + Integer.toString(currentPresses);
       aLabel.setText(s); // Label displays current count
    }
}
```

"Number presses: "

"Number presses: "<#

James Tam

Class To Update Frame: FrameButtonListener

```
public class FrameButtonListener implements ActionListener
{
    // Assumes screen resolution is at least 1024 x 768
    private final static int MAX_X = 1023;
    private final static int MAX_Y = 767;

    // Time in milliseconds
    private final static int DELAY_TIME = 2500;
```

James Tam

Class To Update Frame: FrameButtonListener (2)

```
public void actionPerformed(ActionEvent anEvent)
{
   MyButton aButton = (MyButton) anEvent.getSource();
   JFrame aFrame = (JFrame) aButton.getComponent();
   aFrame.setTitle("Don't you click me! I'm in a bad
                    mood!!!");
   Random aGenerator = new Random();
   // Control randomly "runs away" based on screen size
   int x = aGenerator.nextInt(MAX_X);
   int y = aGenerator.nextInt(MAX_Y);
   aFrame.setLocation(x,y); // Move control to new location
   aButton.setBackground(Color.RED);  // Control is angry
   pause();
   aFrame.setTitle("");  // Angry text is gone
}
```

This class

MAX_X = 1023;

MAX_Y = 767;

James Tam

Class To Update Frame: FrameButtonListener (3)

```
private void pause()  // Give user time to note GUI changes
{
   try
   {
       Thread.sleep(DELAY_TIME);
   }
   catch (InterruptedException ex)
   {
       ex.printStackTrace();
   }
}
}
```

This class

DELAY_TIME = 2500

James Tam

After This Section You Should Now Know

- Different ways of having events from one Java `Component` after other `Component`:
  - Using Java's built in containment classes and methods of access.
  - Declaring the `Component` as an attribute of the container class (and then access the `Component` via an accessor method).
  - Implementing the listener ability in the container class for the control.
  - Declaring the container as an attribute of the component that is the source of the event.
- Anonymous objects/classes as well as an inner class
  - How to declare of define each one
  - How they can be used in conjunction with event listeners.
  - (Inner classes) how this approach for defining a class is similar to defining friend functions

After This Section You Should Now Know (2)

- Methods of reacting to events from different controls
  - Using the `actionCommandAttribute` to differentiate between controls that caused the event to be raised.
- Basic method and attributes for setting the physical properties (e.g. size) and visual characteristics (e.g. color) of a Java `Component`.
- The basic use of the following Java controls: `JTextField`, `JTextArea`, `JScrollPane`, `Jdialog` (along with cautions about their over use), `JPasswordField`.
- How to augment a Java `Component` with an image (via the Java `ImageIcon` class).

References

- Books:
  - "*Java Swing*" by Robert Eckstein, Marc Loy and Dave Wood (O'Reilly)
  - "*Absolute Java*" (4th Edition) by Walter Savitch (Pearson)
  - "*Java: How to Program*" (6th Edition) by H.M. Deitel and P.J. Deitel (Pearson)
- Websites:
  - Java API specifications (last accessed 2021): https://docs.oracle.com/en/java/javase/11/docs/api/index.html
  - Java tutorials (last accessed 2021): http://download.oracle.com/javase/tutorial/uiswing/
  - Java tutorial, layout (last accessed 2021): http://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

James Tam

Copyright Notice

- Unless otherwise specfied, all images were produced by the author (James Tam).

James Tam