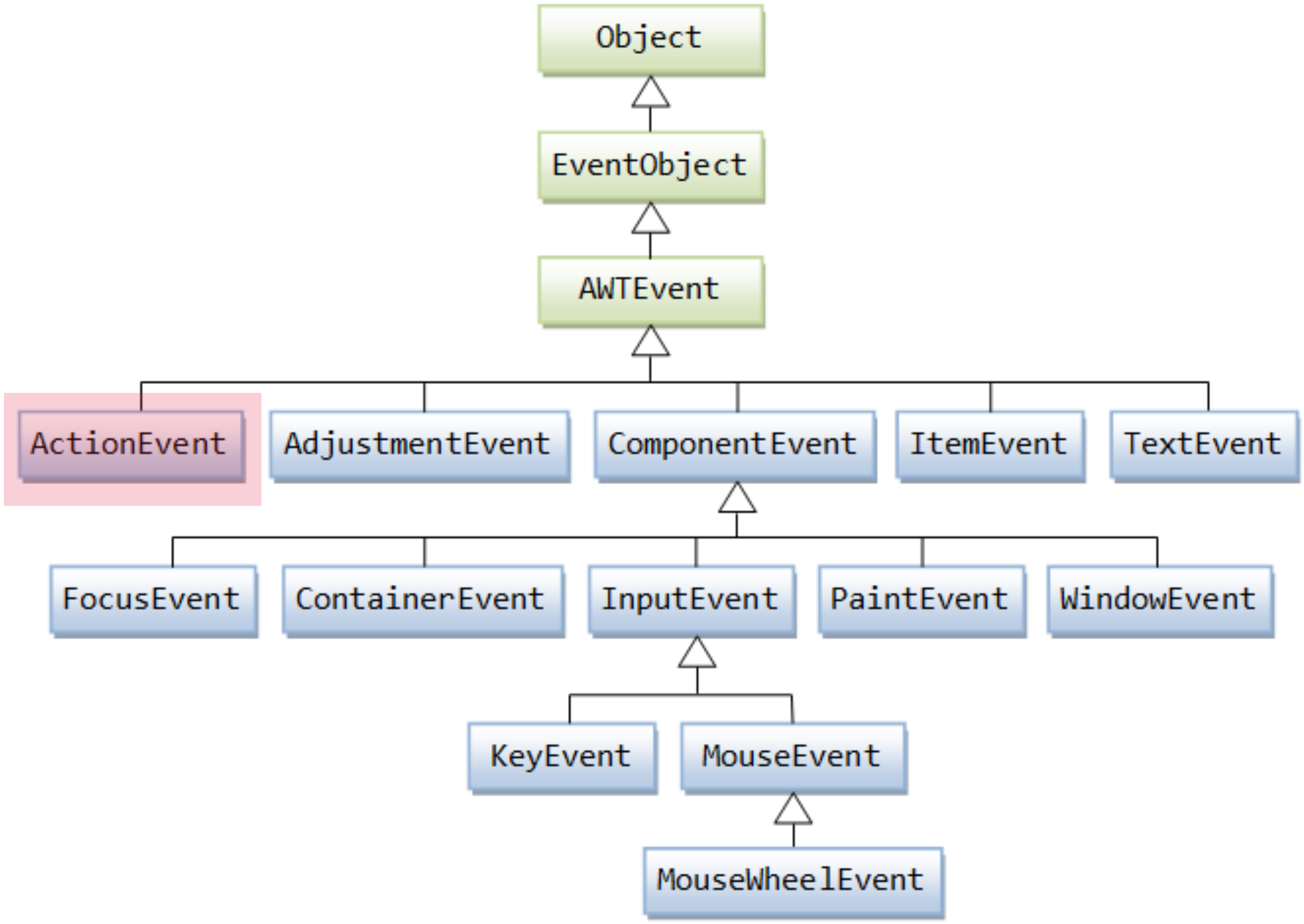
EECS 1720
Building Interactive Systems

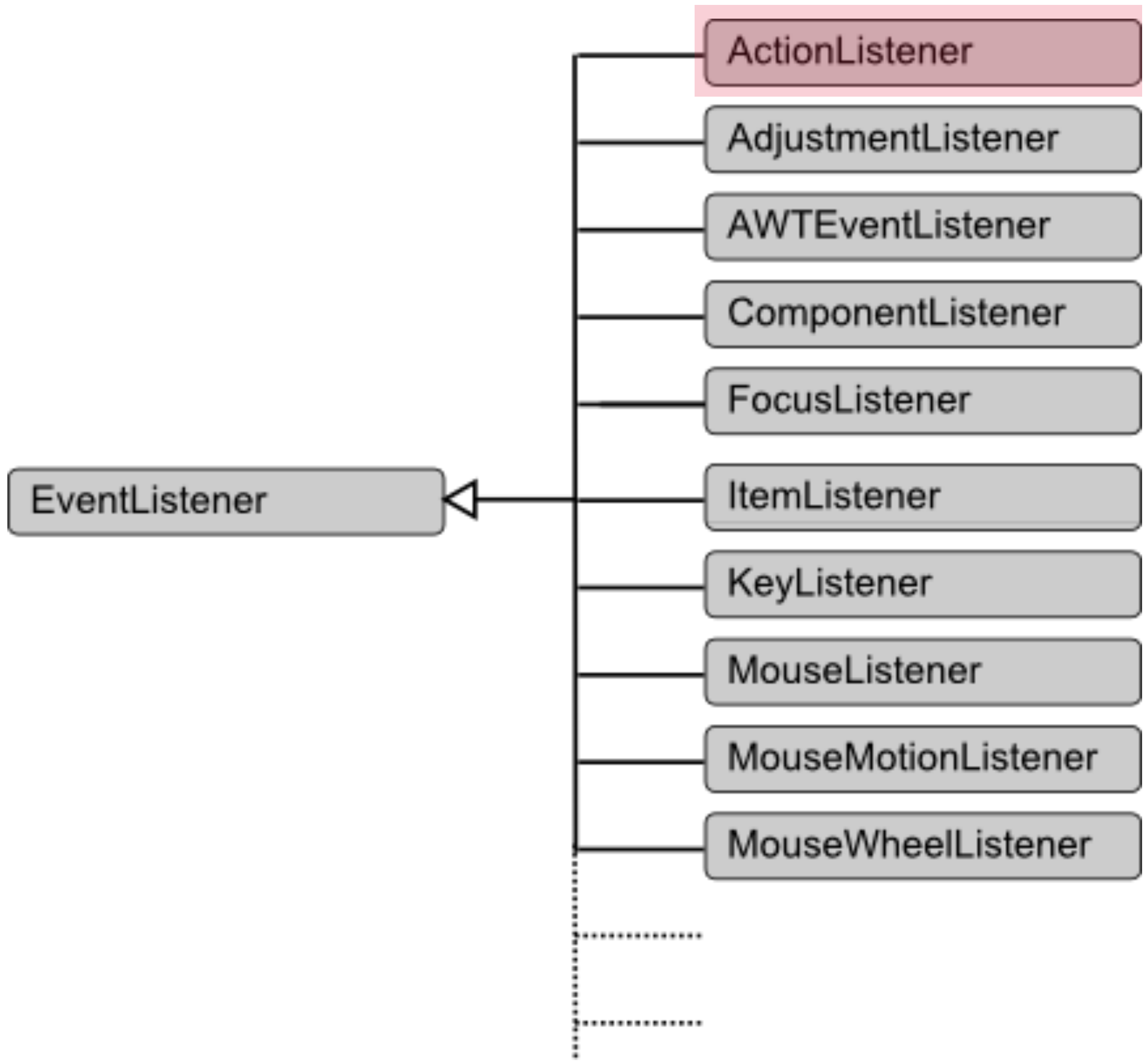
Lecture 16 :: Event Handling [2]

More Examples with ActionEvents

Events


Object
EventObject
AWTEvent
ActionEvent
AdjustmentEvent
ComponentEvent
ItemEvent
TextEvent
FocusEvent
ContainerEvent
InputEvent
PaintEvent
WindowEvent
KeyEvent
MouseEvent
MouseWheelEvent

Event Listeners (interfaces)



Event Listeners

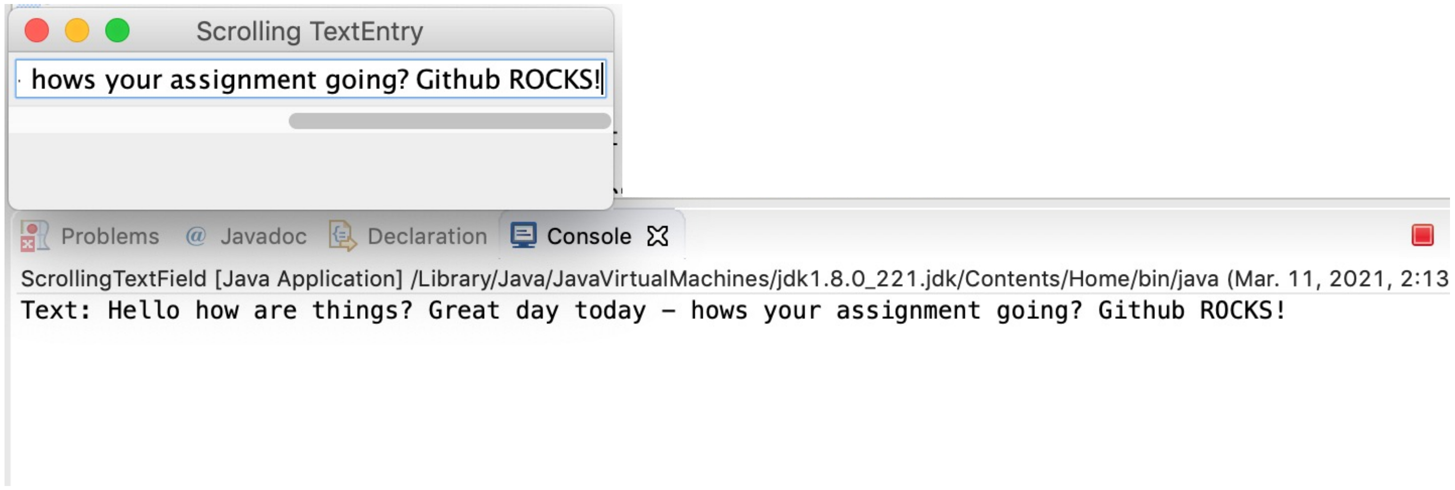
User Action	**Event Triggered**	**Event Listener interface**
Click a `Button, JButton`	**`ActionEvent`**	**`ActionListener`**
Open, iconify, close `Frame, JFrame`	`WindowEvent`	`WindowListener`
Click a `Component, JComponent`	`MouseEvent`	`MouseListener`
Change texts in a `TextField, JTextField`	`TextEvent`	`TextListener`
Type a key	`KeyEvent`	`KeyListener`
Click/Select an item in a `Choice, JCheckbox, JRadioButton, JComboBox`	`ItemEvent,` **`ActionEvent`**	`ItemListener,` **`ActionListener`**

Examples of GUI controls with ActionEvents

- ScrollingText
- CheckBox
- CheckBoxRadio
- ComboBox
- JList
- JMenu

Scrolling Text (with ActionEvent)

- Output textfield on enter/clicking away -> actionEvent



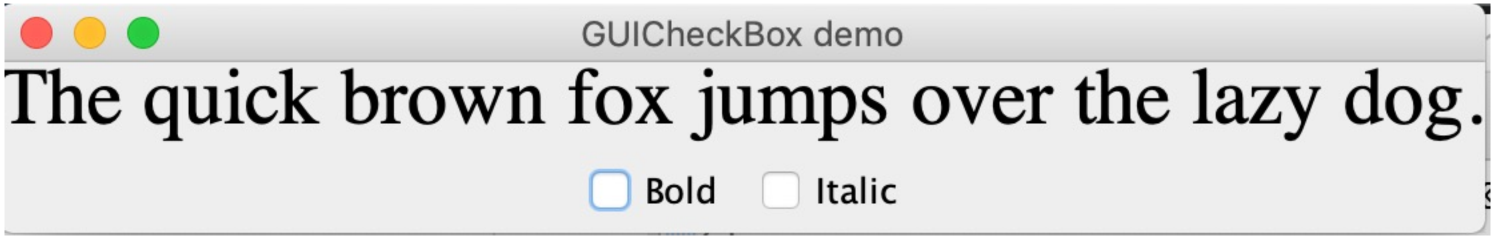
```
hows your assignment going? Github ROCKS!
```

```
ScrollingTextField [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java (Mar. 11, 2021, 2:13
Text: Hello how are things? Great day today - hows your assignment going? Github ROCKS!
```

```java
public class ScrollingTextField extends JFrame {

     private final JTextField textField;
     private JScrollBar scrollBar;

     public ScrollingTextField(String title)  {
          super(title);
          this.textField = new JTextField();
          this.textField.addActionListener(new MyActionListener());
          this.scrollBar = new JScrollBar(JScrollBar.HORIZONTAL);
          JPanel panel = new JPanel();
          panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

          BoundedRangeModel brm = textField.getHorizontalVisibility();
          scrollBar.setModel(brm);
          panel.add(textField);
          panel.add(scrollBar);

          this.add(panel, BorderLayout.NORTH);
          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          this.setSize(300, 100);
          this.setVisible(true);
     }

     private class MyActionListener implements ActionListener {
          public void actionPerformed(ActionEvent e) {
                    System.out.println("Text: " + textField.getText());
          }
     }
     public static void main(String[] args) {
          ScrollingTextField frame  =  new ScrollingTextField("Scrolling TextEntry");
     }
}
```

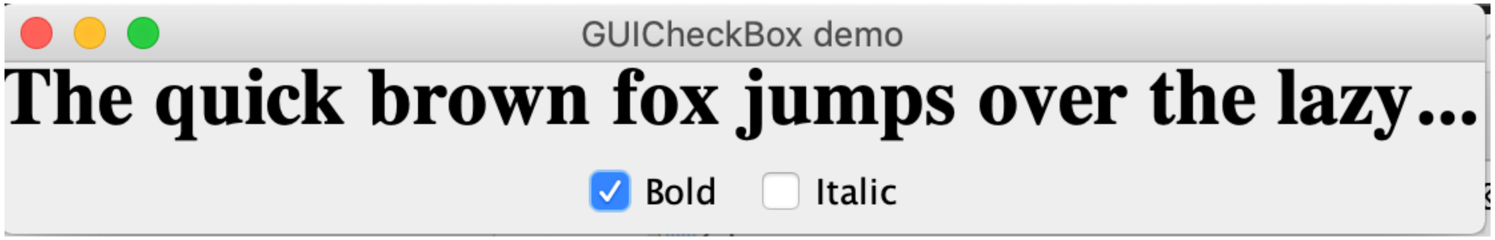
Check box (with ActionEvent)

- Change font properties using different checkboxes:


GUICheckBox demo

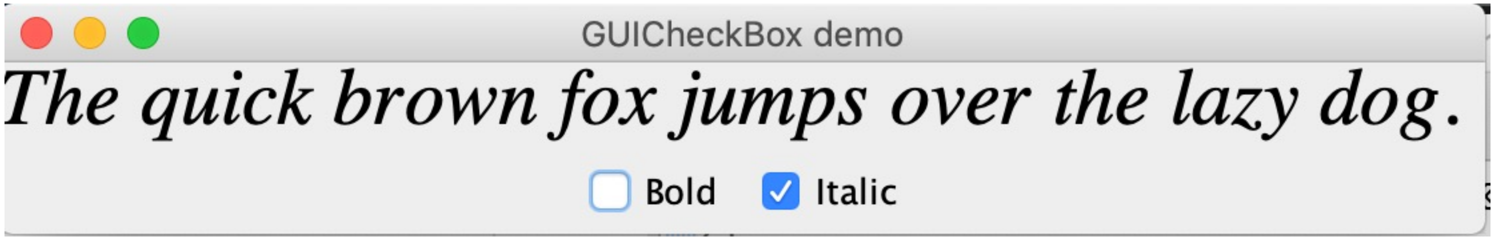
The quick brown fox jumps over the lazy dog.

☐ Bold ☐ Italic


GUICheckBox demo

The quick brown fox jumps over the lazy...

✓ Bold ☐ Italic

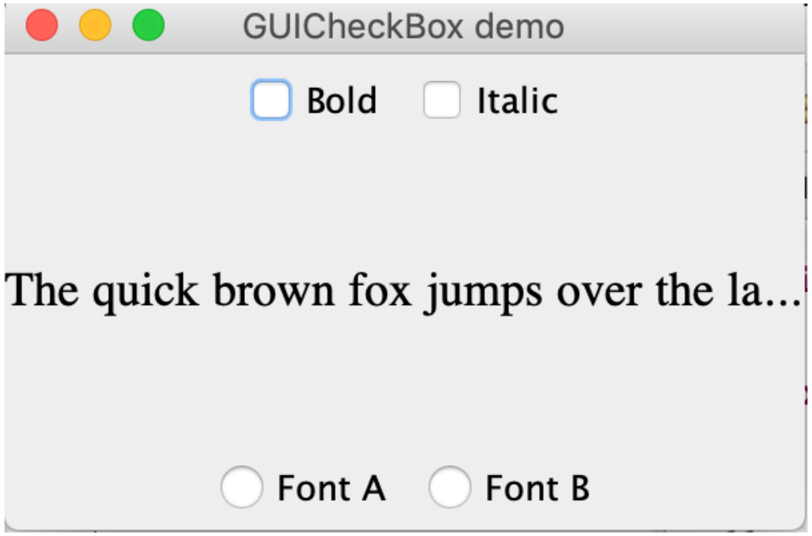

GUICheckBox demo

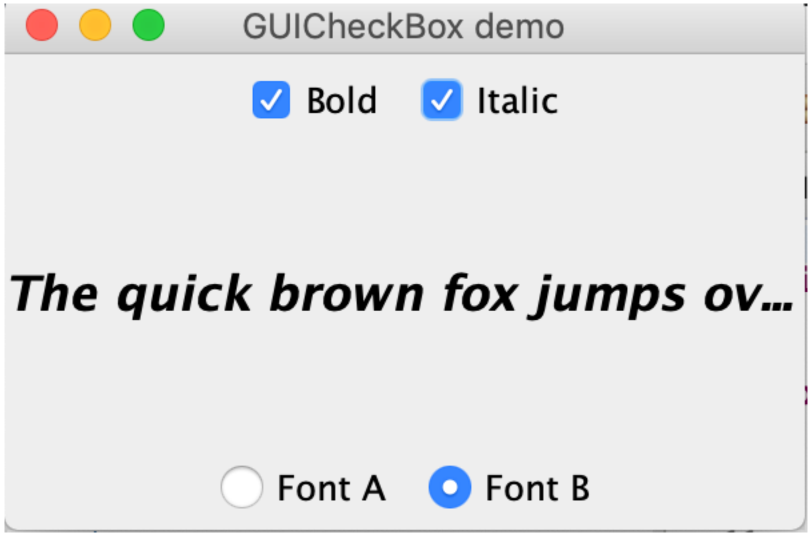
The quick brown fox jumps over the lazy dog.

☐ Bold ✓ Italic

```java
public class GUICheckBox extends JFrame {
      private static final int FONTSIZE = 30;
      private int DEFAULT_WIDTH = 300;
      private int DEFAULT_HEIGHT = 200;
      private JLabel label;
      private JCheckBox bold;
      private JCheckBox italic;

      public GUICheckBox(String title) {
            super(title);
            this.label = new JLabel("The quick brown fox jumps over the lazy dog.");
            this.label.setFont(new Font("Serif", Font.PLAIN, FONTSIZE));
            this.add(label, BorderLayout.CENTER);

            JPanel buttonPanel = new JPanel(); // JPanel not class field
            this.bold = new JCheckBox("Bold");
            this.bold.addActionListener(new MyCheckBoxListener());
            buttonPanel.add(bold);

            this.italic = new JCheckBox("Italic");
            this.italic.addActionListener(new MyCheckBoxListener());
            buttonPanel.add(italic);
            this.add(buttonPanel, BorderLayout.SOUTH);

            this.setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
            this.setResizable(true);
            this.pack();
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setVisible(true);
      }
      private class MyCheckBoxListener implements ActionListener {
            public void actionPerformed(ActionEvent event) {
                  int mode = 0;
                  if (bold.isSelected()) // can access fields in outer class
                        mode += Font.BOLD;
                  if (italic.isSelected())
                        mode += Font.ITALIC;
                  label.setFont(new Font("Serif", mode, FONTSIZE));
            }
      }
      public static void main(String[] args) {
            GUICheckBox frame = new GUICheckBox("GUICheckBox demo");
      }
}
```

CheckBox + RadioButtons





```
public class GUICheckBoxRadio extends JFrame {

    private static final int FONTSIZE = 18;
    private static final String DEFAULTFONT = "";
    private int DEFAULT_WIDTH = 300;
    private int DEFAULT_HEIGHT = 200;
    private JLabel label;
    private JCheckBox bold;
    private JCheckBox italic;
    private ButtonGroup group;
    private JRadioButton buttonA;
    private JRadioButton buttonB;

    public GUICheckBoxRadio(String title) {
        // …
        JPanel stylePanel = new JPanel(); // JPanel not class field
        this.bold = new JCheckBox("Bold");
        this.bold.addActionListener(new MyCheckBoxListener());
        stylePanel.add(this.bold);


        this.italic = new JCheckBox("Italic");
        this.italic.addActionListener(new MyCheckBoxListener());
        stylePanel.add(this.italic);
        this.add(stylePanel, BorderLayout.NORTH);

        this.group = new ButtonGroup();
        this.buttonA = new JRadioButton("Font A");
        this.buttonA.addActionListener(new MyRadioButtonListener());
        this.buttonB = new JRadioButton("Font B");
        this.buttonB.addActionListener(new MyRadioButtonListener());

        group.add(this.buttonA);
        group.add(this.buttonB);

        // …
```

```
private class MyCheckBoxListener implements ActionListener {

     public void actionPerformed(ActionEvent event) {
          int mode = 0;

          if (bold.isSelected()) // can access fields in outer class
                    mode += Font.BOLD;
          if (italic.isSelected())
                    mode += Font.ITALIC;

          // ensure only mode is changed, other properties of the current font are retained
          label.setFont(new Font(label.getFont().getName(), mode,
                                                  label.getFont().getSize()));
     }
}
```

```
// inner class to do listening

private class MyRadioButtonListener implements ActionListener {

     public void actionPerformed(ActionEvent actionEvent) {

          JRadioButton aButton = (JRadioButton) actionEvent.getSource();

          System.out.println("Selected: " + aButton.getText());

          if (aButton==buttonA) {
               System.out.println("Setting to MonoSpaced");
               label.setFont(new Font("MonoSpaced", label.getFont().getStyle(),
                                                  label.getFont().getSize()));
          }

          else if (aButton==buttonB) {
               System.out.println("Setting to SansSerif");

               label.setFont(new Font("SansSerif", label.getFont().getStyle(),
                                                  label.getFont().getSize()));
          }

     }
}
```

Note we can cast a source to a type of object if we are sure that the source is the right “actual” type

YORK UNIVERSITÉ UNIVERSITY

```
// inner class to do listening

private class MyRadioButtonListener implements ActionListener {

     public void actionPerformed(ActionEvent actionEvent) {

          if (actionEvent.getSource() instanceof JRadioButton) {

               // can safely cast
               JRadioButton aButton = (JRadioButton) actionEvent.getSource();

               if (aButton==buttonA) {
                    System.out.println("Setting to MonoSpaced");
                    label.setFont(new Font("MonoSpaced", label.getFont().getStyle(),
                                                  label.getFont().getSize()));
               }
               else if (aButton==buttonB) {
                    System.out.println("Setting to SansSerif");
                    label.setFont(new Font("SansSerif", label.getFont().getStyle(),
                                                  label.getFont().getSize()));
               }
          }

     }
}
```

If we are unsure, we can check the type using **instanceof** first!

YORK UNIVERSITÉ UNIVERSITY

Models

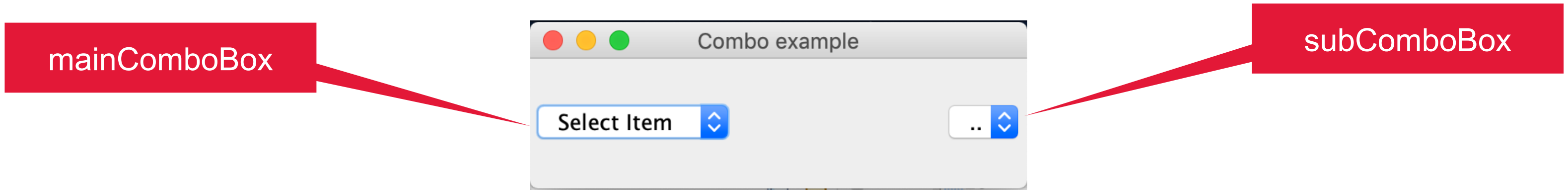
- Most swing components have models associated with them
    - Store component's entire 'state'
    - Some components have multiple models

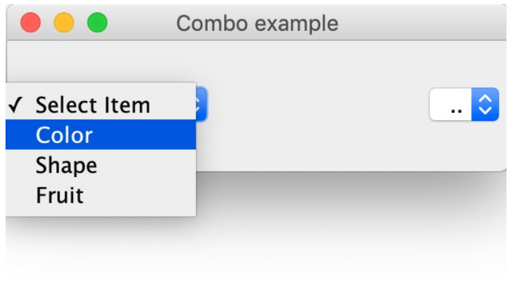
- JButton →
    - ButtonModel (pressed, enabled, etc.)
- JComboBox →
    - ComboBoxModel (holds combo list, what is selected, etc.)
- JList →
    - ListModel (holds contents),
    - ListSelectionModel (track list's current selection)

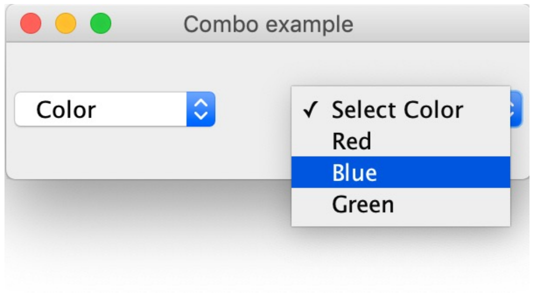
Modifying a UI control’s Model

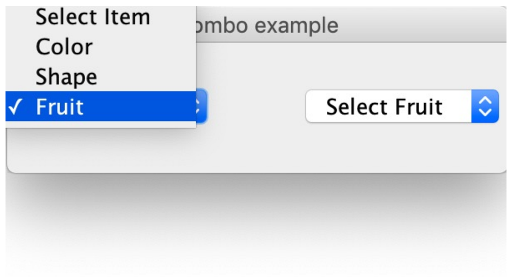
- Note: normally set the model at instantiation
  - E.g. JButton takes a String, or String + Icon, etc
    - We manipulate aspects of the state separately using mutators, so do not explicitly work with ButtonModel
  - E.g. JComboBox takes a String[] to set the options list
    - Using dropdown to select changes which option currently selected
    - How to change all of the options ? We can create and set the model with a DefaultComboBox() model
- setModel(…) is a mutator enabled for some objects allowing the entire model to be replaced for a UI object

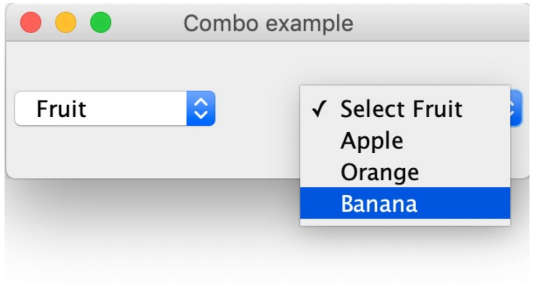
ComboBox Demo











```
public class ComboBoxExample extends JFrame {

      private JComboBox mainComboBox;
      private JComboBox subComboBox;

      // options strings (for subComboBox)
      private String[] colourItems = { "Select Color", "Red", "Blue", "Green" };
      private String[] shapeItems = { "Select Shape", "Circle", "Square", "Triangle" };
      private String[] fruitItems = { "Select Fruit", "Apple", "Orange", "Banana" };

      public ComboBoxExample(String title) {

            super(title);
            Container pane = this.getContentPane();

            String[] items = { "Select Item", "Color", "Shape", "Fruit" };
            mainComboBox = new JComboBox(items);
            mainComboBox.addActionListener(new MyEventHandler());

            pane.add(mainComboBox, BorderLayout.WEST);

            subComboBox = new JComboBox(); // initialized with empty string
            pane.add(subComboBox, BorderLayout.EAST);

            // setup and show frame
            this.setSize(300, 100);
            this.setResizable(true);

            //this.pack();
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setVisible(true);

      }


      public static void main(String[] args) {
            ComboBoxExample frame = new ComboBoxExample("Combo example");
      }

      // listener next page
}
```

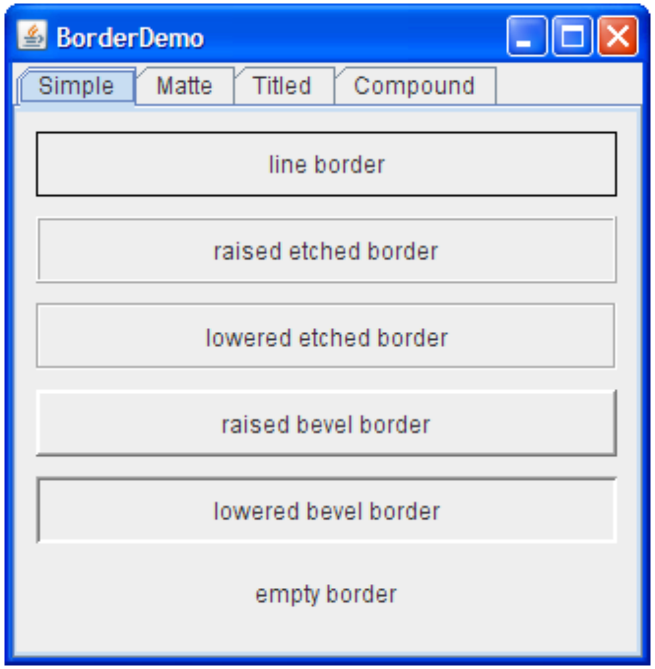
```java
private class MyEventHandler implements ActionListener {


      public void actionPerformed(ActionEvent e) {
            String item = (String) mainComboBox.getSelectedItem();

            // SIMPLE APPROACH

            switch (item) {
            case "Color":
                  subComboBox.setModel(new DefaultComboBoxModel(colourItems));
                  break;

            case "Shape":
                  subComboBox.setModel(new DefaultComboBoxModel(shapeItems));
                  break;

            case "Fruit":
                  subComboBox.setModel(new DefaultComboBoxModel(fruitItems));
                  break;

            default:
                  subComboBox.setModel(new DefaultComboBoxModel());
            }

      }


}
```

Aside: Using Borders..

- Typically used in JPanels/JLabels & custom subclasses of JComponent
- Uses *BorderFactory* class to create borders

```
// BorderFactory.createEmptyBorder();
// BorderFactory.createEmptyBorder(top,left,bottom,right);

JPanel pane = new JPanel();
pane.add(new JButton(“button 1”));
pane.add(new JButton(“button 2”));
pane.setBorder( BorderFactory.createEmptyBorder(10,40,10,40) );

// other kinds of borders?  Many others also
pane.setBorder( BorderFactory.createLineBorder(Color.BLUE) );
pane.setBorder( BorderFactory.createBevelBorder(BevelBorder.RAISED) );
pane.setBorder( BorderFactory.createRaisedBevelBorder() );
pane.setBorder( BorderFactory.createEtchedBorder() );

// can apply to other components (e.g. JButtons, etc.)
// look for whether a component has a setBorder method
```

More examples (see BorderDemo.java from "How to use Borders" from the java tutorials)



line border

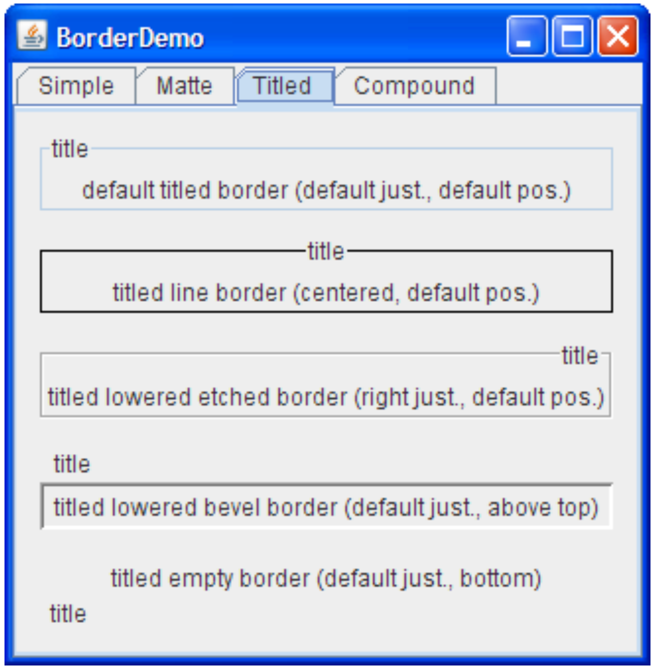
raised etched border

lowered etched border

raised bevel border

lowered bevel border

empty border



title

default titled border (default just., default pos.)

title

titled line border (centered, default pos.)

title

titled lowered etched border (right just., default pos.)

title

titled lowered bevel border (default just., above top)

titled empty border (default just., bottom)

title

https://docs.oracle.com/javase/tutorial/uiswing/components/border.html

Empty Border..

- Can be used to achieve the equivalent of “insets”
    - `pane.setBorder( BorderFactory.createEmptyBorder(10,40,10,40) );`
- i.e.
    - The border will fill the space around the contents that are added to pane

Spacers/RigidArea

- Can create spacer objects (JPanels, JLabels etc.. With no content)

```
// assume pane is a reference to content pane (using default layout)
JPanel spacer = new JPanel();
spacer.setPreferredSize(new Dimension(100,50));
pane.add(spacer, BorderLayout.NORTH);
```

- OR.. can create an invisible box component that is always a fixed size

```
// assume pane is a reference to content pane (using default layout)
pane.add(Box.createRigidArea(new Dimension(50,50)), BorderLayout.NORTH);
```