More Swing examples

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming Concepts using Java

Week 12

Connecting multiple events to a listener

- One listener can listen to multiple objects

Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue

```
public class ButtonPanel extends JPanel
                    implements ActionListener{
  // Panel has three buttons
  private JButton yellowButton, blueButton,
                  redButton;

  public ButtonPanel(){
    yellowButton = new JButton("Yellow");
    blueButton = new JButton("Blue");
    redButton = new JButton("Red");
    ...
  }

  public void actionPerformed(ActionEvent evt){
    ...
  }
}
```

Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue
- Make the panel listen to all three buttons

```
public class ButtonPanel extends JPanel
                    implements ActionListener{
  // Panel has three buttons
  private JButton yellowButton, blueButton,
                  redButton;

  public ButtonPanel(){
    yellowButton = new JButton("Yellow");
    blueButton = new JButton("Blue");
    redButton = new JButton("Red");
    // ButtonPanel listens to all three buttons
    yellowButton.addActionListener(this);
    blueButton.addActionListener(this);
    redButton.addActionListener(this);
    add(yellowButton);
    add(blueButton);
    add(redButton);
  }
  ...
}
```

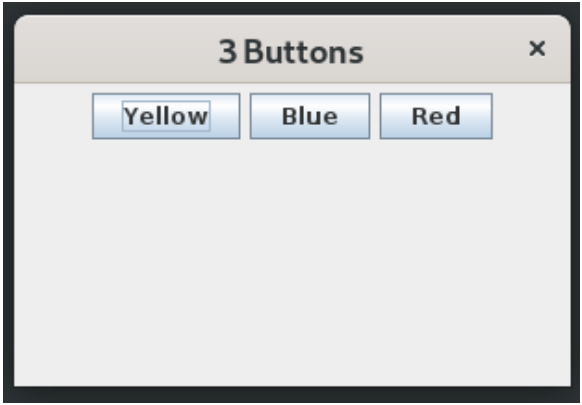
Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue
- Make the panel listen to all three buttons
- Determine what colour to use by identifying source of the event
  - Keep the existing colour if the source is not one of these three buttons

```
public class ButtonPanel extends JPanel
                    implements ActionListener{
  ...
  public void actionPerformed(ActionEvent evt){
    // Find the source of the event
    Object source = evt.getSource();
    // Get current background colour
    Color color = getBackground();

    if (source == yellowButton)
      color = Color.yellow;
    else if (source == blueButton)
      color = Color.blue;
    else if (source == redButton)
      color = Color.red;

    setBackground(color);
    repaint();
  }
}
```

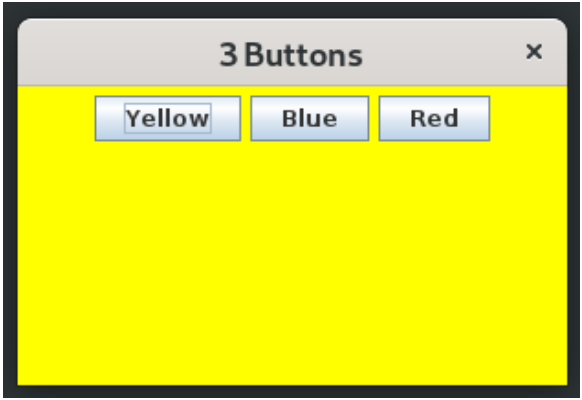
Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue
- Make the panel listen to all three buttons
- Determine what colour to use by identifying source of the event
  - Keep the existing colour if the source is not one of these three buttons
- Output — before any button is clicked



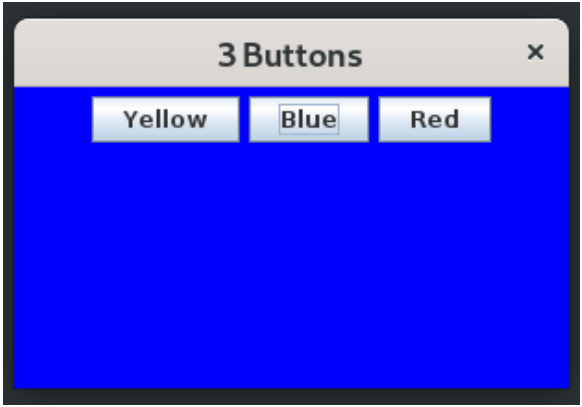
Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue
- Make the panel listen to all three buttons
- Determine what colour to use by identifying source of the event
  - Keep the existing colour if the source is not one of these three buttons
- Output — before any button is clicked . . . and after each is clicked



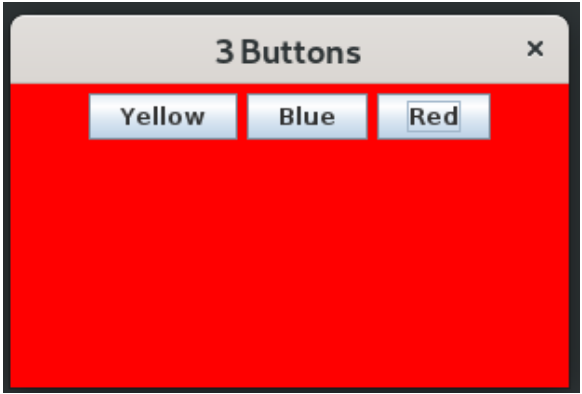
Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue
- Make the panel listen to all three buttons
- Determine what colour to use by identifying source of the event
  - Keep the existing colour if the source is not one of these three buttons
- Output — before any button is clicked . . . and after each is clicked



Connecting multiple events to a listener

- One listener can listen to multiple objects
- A panel with three buttons, to paint the panel red, yellow or blue
- Make the panel listen to all three buttons
- Determine what colour to use by identifying source of the event
  - Keep the existing colour if the source is not one of these three buttons
- Output — before any button is clicked . . . and after each is clicked



Multicasting: multiple listeners for an event

- Two panels, each with three buttons, Red, Blue, Yellow

```
import ...
public class ButtonPanel extends JPanel
                    implements ActionListener{
  private JButton yellowButton, blueButton,
                  redButton;

  public ButtonPanel(){
    yellowButton = new JButton("Yellow");
    blueButton = new JButton("Blue");
    redButton = new JButton("Red");

    ...

    add(yellowButton);
    add(blueButton);
    add(redButton);
  }
  ...
}
```

Multicasting: multiple listeners for an event

- Two panels, each with three buttons, Red, Blue, Yellow
- Clicking a button in either panel changes background colour in both panels

```
import ...
public class ButtonPanel extends JPanel
                    implements ActionListener{
  private JButton yellowButton, blueButton,
                  redButton;

  public ButtonPanel(){
    yellowButton = new JButton("Yellow");
    blueButton = new JButton("Blue");
    redButton = new JButton("Red");

    ...

    add(yellowButton);
    add(blueButton);
    add(redButton);
  }
  ...
}
```

Multicasting: multiple listeners for an event

- Two panels, each with three buttons, Red, Blue, Yellow
- Clicking a button in either panel changes background colour in both panels
- Both panels must listen to all six buttons
  - However, each panel has references only for its local buttons
  - How do we determine the source of an event from a remote button?

```
import ...
public class ButtonPanel extends JPanel
                    implements ActionListener{
  private JButton yellowButton, blueButton,
                  redButton;

  public ButtonPanel(){
    yellowButton = new JButton("Yellow");
    blueButton = new JButton("Blue");
    redButton = new JButton("Red");

    ...

    add(yellowButton);
    add(blueButton);
    add(redButton);
  }
  ...
}
```

Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...

```
import ...
public class ButtonPanel extends JPanel
                    implements ActionListener{
  private JButton yellowButton, blueButton,
                  redButton;

  public ButtonPanel(){
    yellowButton = new JButton("Yellow");
    blueButton = new JButton("Blue");
    redButton = new JButton("Red");

    yellowButton.setActionCommand("YELLOW");
    blueButton.setActionCommand("BLUE");
    redButton.setActionCommand("RED");

    add(yellowButton);
    add(blueButton);
    add(redButton);
  }
```

Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`

```
public class ButtonPanel extends JPanel
                    implements ActionListener{
  ...
  public void actionPerformed(ActionEvent evt){
    Color color = getBackground();
    String cmd = evt.getActionCommand();

    if (cmd.equals("YELLOW"))
      color = Color.yellow;
    else if (cmd.equals("BLUE"))
      color = Color.blue;
    else if (cmd.equals("RED"))
      color = Color.red;

    setBackground(color);
    repaint();
  }
  ...
}
```

Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel

```
public class ButtonPanel extends JPanel
                    implements ActionListener{
  ...

  public void addListener(ActionListener o){
    // Add a commmon listener for all
    // buttons in this panel
    yellowButton.addActionListener(o);
    blueButton.addActionListener(o);
    redButton.addActionListener(o);
  }
}
```

Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel
- Add both panels to the same frame

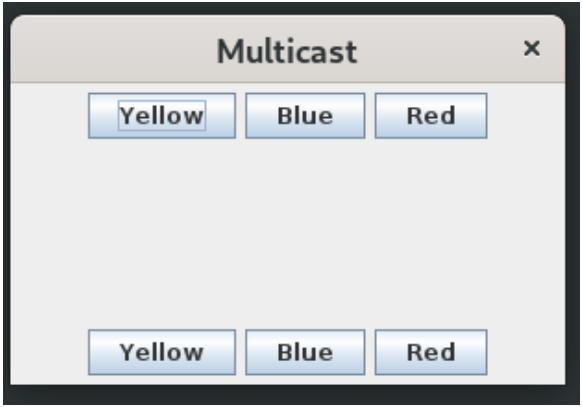
```
public class ButtonFrame extends JFrame
                    implements WindowListener{
  private Container contentPane;
  private ButtonPanel b1, b2;

  public ButtonFrame(){
    ..
    b1 = new ButtonPanel();   // Two panels
    b2 = new ButtonPanel();

    // Each panel listens to both sets of buttons
    b1.addListener(b1);  b1.addListener(b2);
    b2.addListener(b1);  b2.addListener(b2);

    contentPane = this.getContentPane();
    // Set layout to separate out panels in frame
    contentPane.setLayout(new BorderLayout());
    contentPane.add(b1,"North");
    contentPane.add(b2,"South");
  }
```

Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel
- Add both panels to the same frame
- How it works

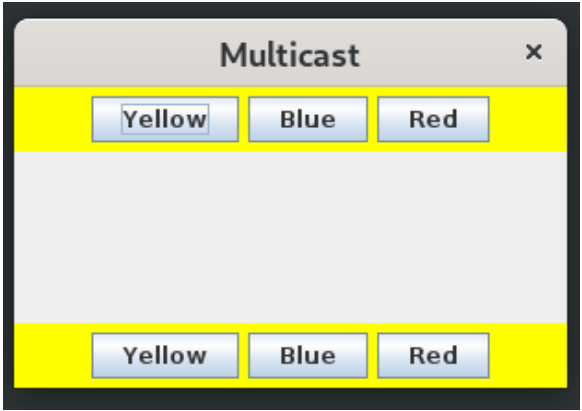
Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel
- Add both panels to the same frame
- How it works



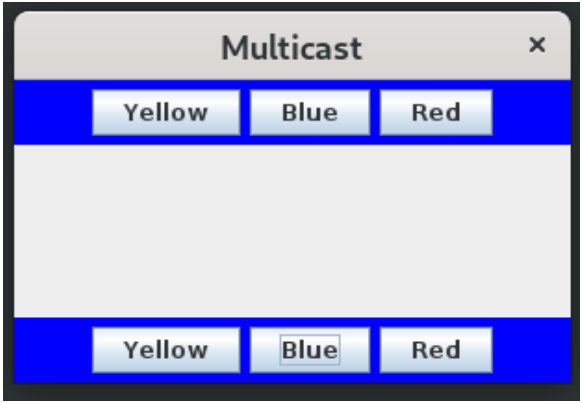
Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel
- Add both panels to the same frame
- How it works



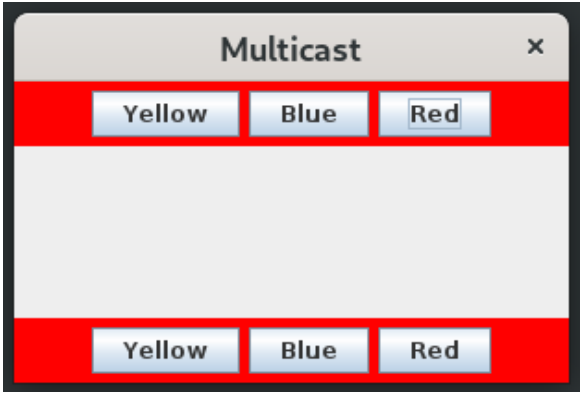
Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel
- Add both panels to the same frame
- How it works



Multicasting: multiple listeners for an event

- Associate an `ActionCommand` with a button
  - Assign the same action command to both `Red` buttons, ...
- Choose colour according to `ActionCommand`
- Need to add both panels as listeners for each button
  - Add a public function to add a new listener to all buttons in a panel
- Add both panels to the same frame
- How it works



Other elements – checkboxes

- `JCheckbox`: a box that can be ticked

Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green

```
import ...
public class CheckBoxPanel extends JPanel
                    implements ActionListener{
  private JCheckBox redBox;
  private JCheckBox blueBox;

  public CheckBoxPanel(){
    redBox = new JCheckBox("Red");
    blueBox = new JCheckBox("Blue");

    ...
  }
}
```

Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`

```
import ...
public class CheckBoxPanel extends JPanel
                    implements ActionListener{
  private JCheckBox redBox;
  private JCheckBox blueBox;

  public CheckBoxPanel(){
    redBox = new JCheckBox("Red");
    blueBox = new JCheckBox("Blue");

    redBox.addActionListener(this);
    blueBox.addActionListener(this);

    ...
  }
}
```

Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not

```
import ...
public class CheckBoxPanel extends JPanel
                    implements ActionListener{
  private JCheckBox redBox;
  private JCheckBox blueBox;

  public CheckBoxPanel(){
    redBox = new JCheckBox("Red");
    blueBox = new JCheckBox("Blue");

    redBox.addActionListener(this);
    blueBox.addActionListener(this);

    redBox.setSelected(false);
    blueBox.setSelected(false);

    add(redBox);
    add(blueBox);
    ...
  }
```

Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not
- `isSelected()` returns current state

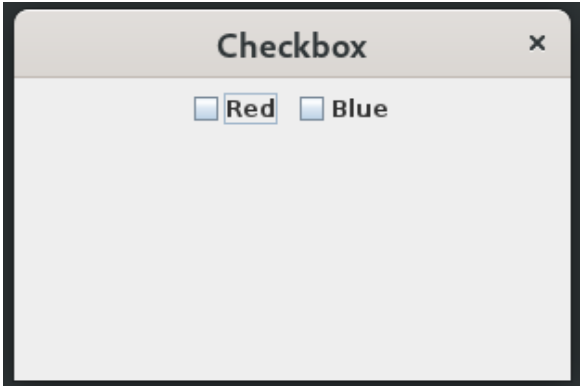
```
public class CheckBoxPanel extends JPanel
                        implements ActionListener{
  ...

  public void actionPerformed(ActionEvent evt){

    Color color = getBackground();

    if (blueBox.isSelected())
      color = Color.blue;
    if (redBox.isSelected())
      color = Color.red;
    if (blueBox.isSelected() &&
        redBox.isSelected())
      color = Color.green;

    setBackground(color);
    repaint();
  }
}
```

Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not
- `isSelected()` returns current state
- Rest similar to basic button example

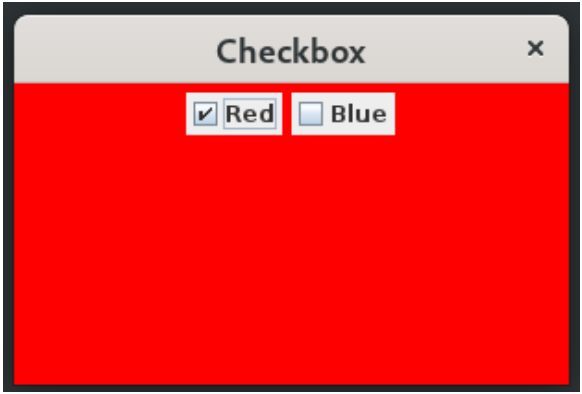
Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not
- `isSelected()` returns current state
- Rest similar to basic button example



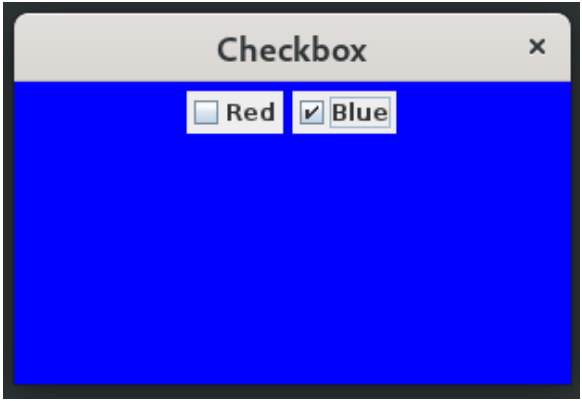
Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not
- `isSelected()` returns current state
- Rest similar to basic button example



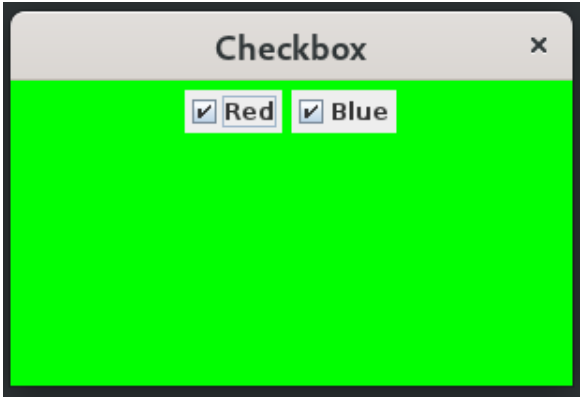
Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not
- `isSelected()` returns current state
- Rest similar to basic button example



Other elements – checkboxes

- `JCheckbox`: a box that can be ticked
- A panel with two checkboxes, `Red` and `Blue`
  - Only `Red` ticked, background red
  - Only `Blue` ticked, background blue
  - Both ticked, background green
- Only one action — click the box
  - Listener is again `ActionListener`
- Checkbox state: selected or not
- `isSelected()` returns current state
- Rest similar to basic button example



Summary

- Swing components such as buttons, checkboxes generate high level events
- Each event is automatically sent to a listener
  - Listener capability is described using an interface
  - Event is sent as an object — listener can query the event to obtain details such as event source, action label, ... and react accordingly
- Association between event generators and listeners is flexible
  - One listener can listen to multiple objects
  - One component can inform multiple listeners
- Must explicitly set up association between component and listener
  - Events are "lost" if nobody is listening!
- Swing objects are the most aesthetically pleasing, but useful to understand how GUI programming works across other languages