Lecture 8

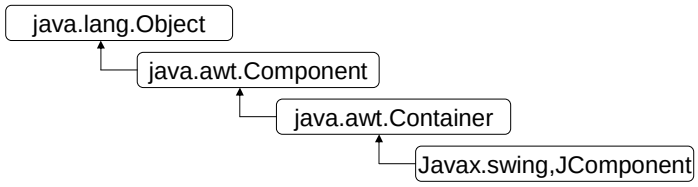
User Interface Components with Swing

Swing Overview (1)

- Classes from package `javax.swing` defines various GUI components — objects with which the user interacts via the mouse, the keyboard or another form of input.
- Some basic GUI components
  - `JLabel`
  - `JTextField`
  - `JCheckBox`
  - `JComboBox`
  - `JList`
  - `JPanel`
- Most of the swing components are written completely in java, so they provide a greater portability and flexibility than the original GUI components from package java.awt
  - Awt components are platform dependent
  - Some swing components are still platform dependent. E.g, `JFrame`

Swing Overview (2)

- Common superclasses of many of the Swing components



- `Component` class
  - Operations common to most GUI components are found in `Component` class.
- `Container` class
  - Two important methods originates in this class
    - `add` — adds components to a `container`.
    - `setLayout` — enables a program to specify the layout manager that helps a `Container` position and size its components.

JLabel

- A `JLabel` object provides text instructions or information on a GUI — display a single line of *read-only* text, an image or both text and image
- Example code (output)
- One thing to be empasized: if you do not explicitly add a GUI component to a container, the GUI component will not be displayed when the container appears on the screen

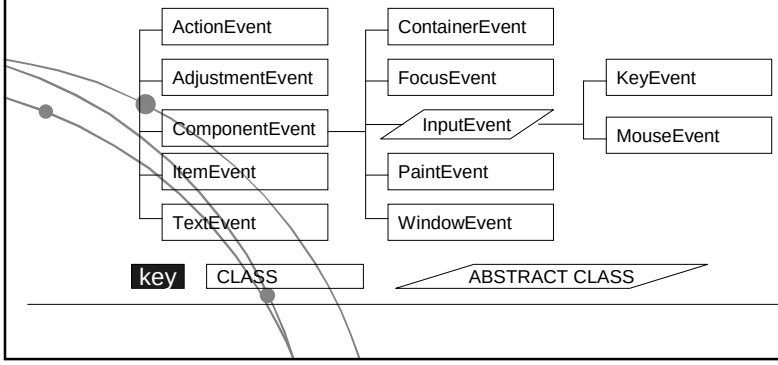
To Make an Interactive GUI Program

- To make an interactive GUI program, you need:
  - Components
    - buttons, windows, menus, etc.
  - Events
    - mouse clicked, window closed, button clicked, etc.
  - Event listeners (interfaces) and event handlers (methods)
    - listen for events to be trigged, and then perform actions to handle them

Event-Handling Model (1)

- Some GUIs are *event driven* — they generate events when the user interacts with the GUI
  - E.g, moving the mouse, clicking a button, typing in a text field, selecting an item from a menu, etc.
  - When a user interaction occurs, an event is sent to the program. Many event types are defined in packages `java.awt.event` and `javax.swing.event`

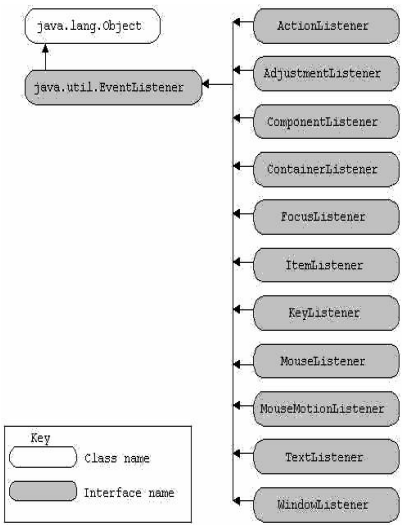
some event classes in package `java.awt.event`



Event-Handling Model (2)

- Three parts of the event-handling mechanism
  - *event source*: the GUI component with which the user interacts
  - *event object*: encapsulated information about the occurred event
  - *event listener*: an object which is notified by the event source when an event occurs, and provides responds to the event
- The programmer must perform two tasks to process a GUI event
  1. register an *event listener*
     - An object of a class that implements one or more of the event-listener interfaces from packages `java.awt.event` and `javax.swing.event`
  2. implement an *event handling method*

Event-listener interface of package `java.awt.event`

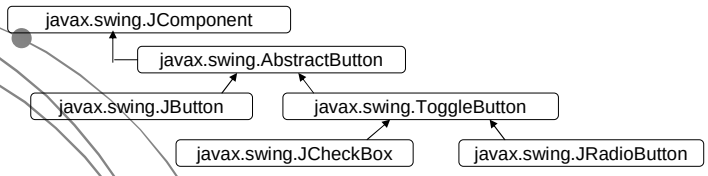


`JTextField` and `JPasswordField`

- They are single-line areas in which text can be entered by the user from the keyboard or text can simply be displayed
- When the user types data into them and presses the *`Enter`* key, an action event occurs. If the program registers an event listener, the listener processes the event and can use the data in the text field at the time of the event in the program.
- Eample code (output)

Buttons

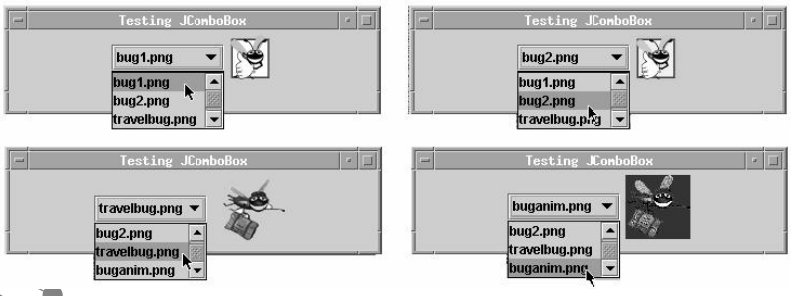
- A *button* is a component the user clicks to trigger a specific action
- There are several types of buttons in Java, all are subclasses of `AbstractButton`
  - *command buttons:* is created with class `JButton.` It generates `ActionEvent`
  - *toggle buttons*: have on/off or true/false values
  - *check boxes*: a group of buttons. It generates `ItemEvent`
  - *radio buttons*: a group of buttons in which only one can be selected. It generates `ItemEvent`



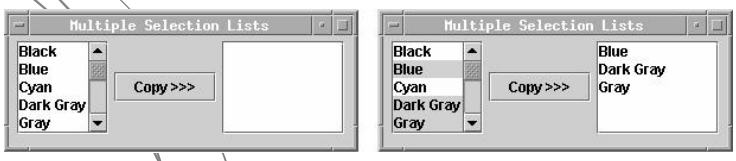
- Example Code of JButton (output)

More Examples …

- `JComboBox`: a drop-down list provides a list of items from which the user can make a selection. It generates `ItemEvent`



- `JList`: a list supports both *single selection* and *multiple-selection*. It generates `ListSelectionEvent`



Layout Management

- Let's see an example first

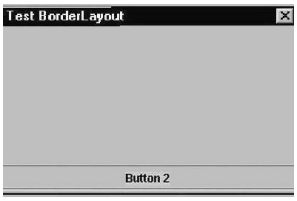
- All components in a container are positioned by a *layout manager*. Buttons in the above example are managed by the *flow layout manager*, which is the default layout manager for a panel.
  - The default manager lines the components horizontally until there is no more room and then start a new row of components
  - After resizing the container, the layout manager reflows the components automatically
  - The default is to center the components in each row. You can choose aligning them to the left or to the right of the container.

```
panel.setLayout(new FlowLayout(FlowLayout.LEFT));
```

- Other managers: see http://java.sun.com/docs/books/tutorial/uiswing/layout/visual.html.

Layout Management — Using Panel

- Potential problem with BorderLayout:
  - The button is stretched to fill the entire southern region of the frame
  - If you add another button to the southern region, it would just displace the first button



- Solution – use additional *panels*.
  - It acts as containers for interface elements and can themselves be arranged inside a larger panel
  - Use flow layout by default
  - To fix the problem of BorderLayout
    1. Create a panel
    2. Add components to the panel
    3. Add the panel to the larger container


```
JPanel p = new JPanel();
p.add(button1);
p.add(button2);
P.add(button3);
frame.add(panel, BorderLayout.SOUTH);
```

Supplemental Reading

- A visual Index to the Swing Components

http://java.sun.com/docs/books/tutorial/uiswing/components/components.html

- **Creating a GUI with JFC/Swing**

http://java.sun.com/docs/books/tutorial/uiswing/index.html

- **Building a User Interface**

http://java.sun.com/developer/onlineTraining/new2java/divelog/part1/
http://java.sun.com/developer/onlineTraining/new2java/divelog/part2/index.jsp
http://java.sun.com/developer/onlineTraining/new2java/divelog/part3/
http://java.sun.com/developer/onlineTraining/new2java/divelog/part4/
http://java.sun.com/developer/onlineTraining/new2java/divelog/part5/