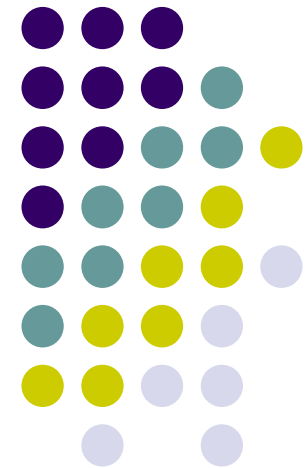


1st week discussion

# ICS 52: Introduction to Software Engineering

---

Instructor: Prof. Dan Frost  
TA: Tiago Proenca  
tproenca@ics.uci.edu  
09.30.2009



# Containment Hierarchy



## **Top-level container**

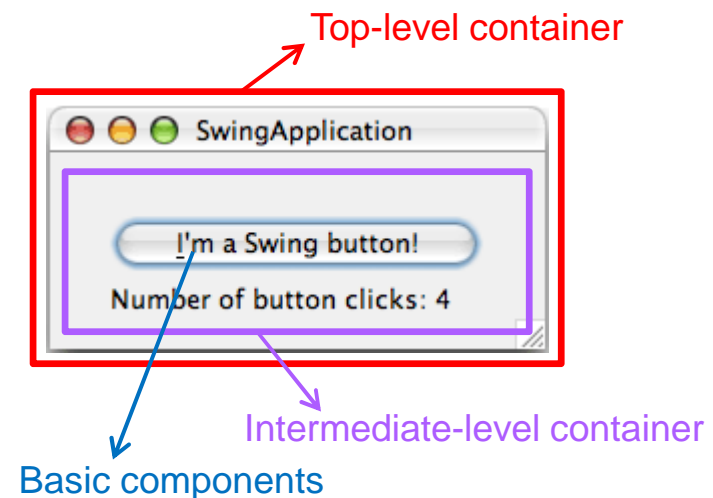
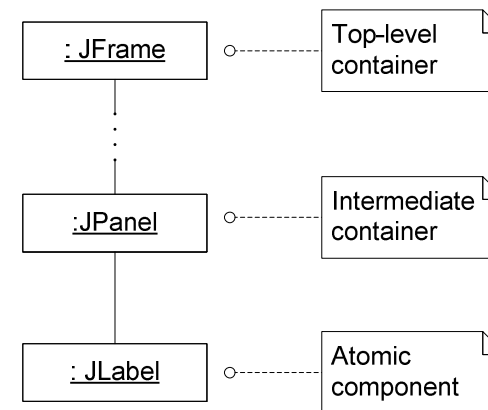
- contains all of the visual components of a GUI
- provides the screen real estate used by application.
- Define look and feel
- Example: JFrame, JDialog

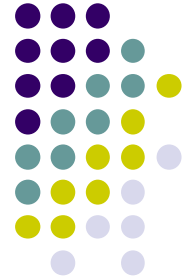
## **Intermediate containers**

- used to organize and position GUI components
- Layout management
- Example: JPanel

## **Basic components**

- present information to or get information from user.
- Examples: button, label, text field, editor pane, combo box.





# Layout Managers

- Responsible for positioning and sizing components added to a container.
- Each container is associated with a `LayoutManager`



# FlowLayout

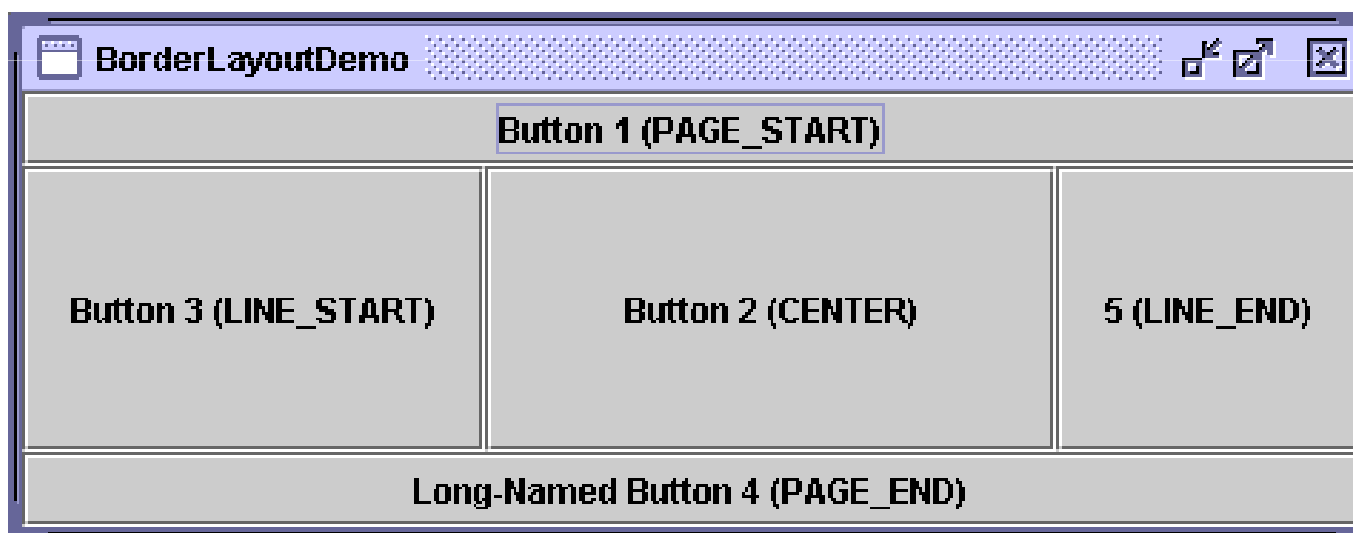
- lays out components left to right, top to bottom.



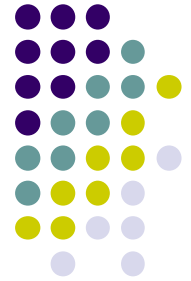


# BorderLayout

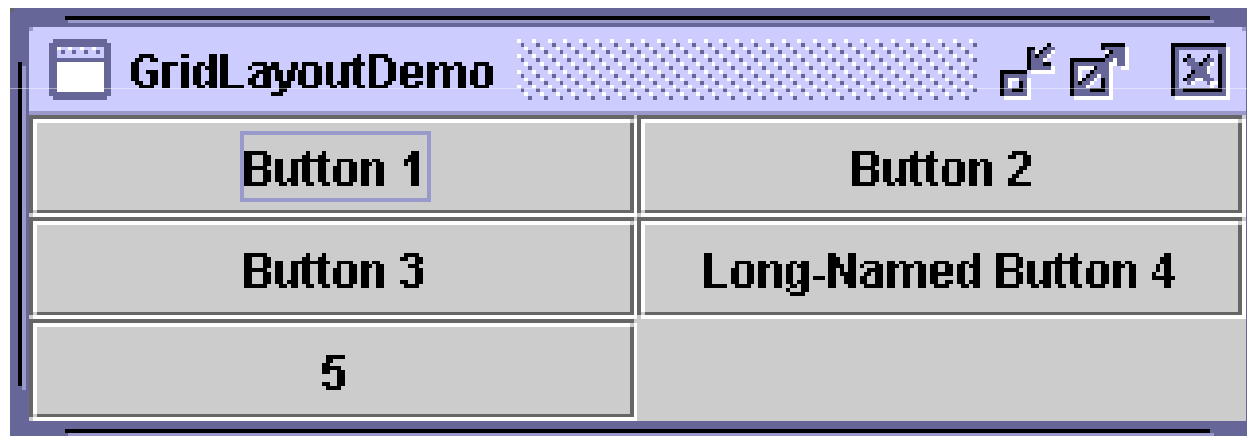
- lays out up to five components, positioned “north,” “south,” “east,” “west,” and “center.”



# GridLayout



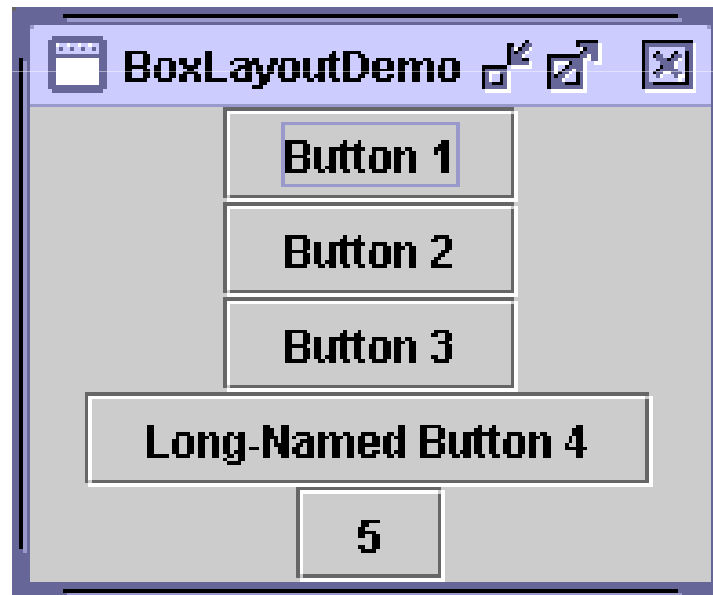
- lays out components in a two-dimensional grid.





# BoxLayout

- lays out components in either a single horizontal row or single vertical column



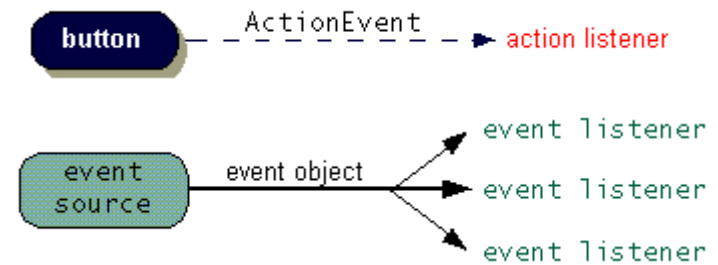
# Other Layouts



- <http://java.sun.com/docs/books/tutorial/uiswing/layout/layoutlist.html>

# Events

- A way of informing the system that something has happened
- Involves two things:
  - *Events*: signals generated by
    - mouse clicks or double clicks
    - menu selections
    - text entered etc.



- *Event Listeners*: Interfaces which are designed to capture and process certain types of events;
  - register with component;
  - informed when component generates an applicable event.
  - respond by performing some appropriate action.



## Implementing an Event Handler



All event handlers require three pieces of code:

1. declaration (extend/implement a listener interface)

```
public class MyClass implements ActionListener {
```

2. registration of an instance of the event handler class as a listener

```
someComponent.addActionListener(instanceOfMyClass);
```

3. providing code that implements the methods in the listener interface in the event handler class

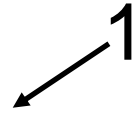
```
public void actionPerformed(ActionEvent e) { ...//code that reacts to the action... }  
}
```

# To detect when the user clicks an onscreen button

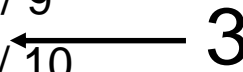
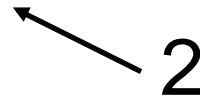


- A program must have an object that implements the `ActionListener` interface
- The program must register this object as an action listener on the button (the event source), using the `addActionListener` method.
- When the user clicks the onscreen button, the button fires an action event.
- This results in the invocation of the action listener's `actionPerformed` method (the only method in the `ActionListener` interface)

# An Example



```
public class ButtonClickExample extends JFrame // 1
    implements ActionListener { // 2
    JButton b = new JButton("Click me!"); // 3
    public ButtonClickExample() { // 4
        b.addActionListener(this); // 5
        getContentPane().add(b); // 6
        pack(); // 7
        setVisible(true); } // 8
    public void actionPerformed(ActionEvent e) { // 9
        b.setBackground(Color.RED); } // 10
    public static void main(String[] args) { // 11
        new ButtonClickExample(); } } // 12
```





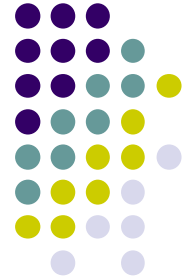
# The for-each loop

- Before Java 1.5

```
void cancelAll(Collection<TimerTask> c) {  
    for (Iterator<TimerTask> i = c.iterator(); i.hasNext();) {  
        i.next().cancel();  
    }  
}
```

- After Java 1.5

```
void cancelAll(Collection<TimerTask> c) {  
    for (TimerTask t : c) {  
        t.cancel();  
    }  
}
```



# The for-each loop

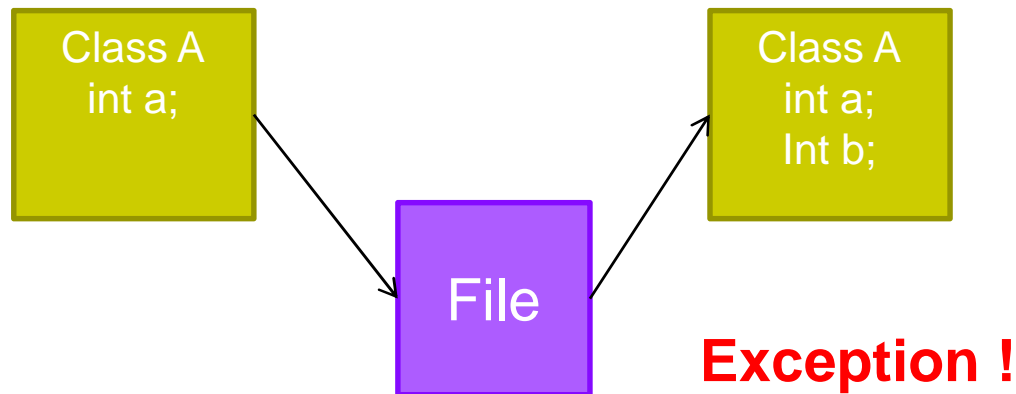
- It hides iterators, so it is not usable to remove or replace elements
- More information:
  - <http://java.sun.com/j2se/1.5.0/docs/guide/language/foreach.html>

# Questions from today

## SerialVersionUID



- The serialVersionUID is a universal version identifier for a Serializable class. Deserialization uses this number to ensure that a loaded class corresponds exactly to a serialized object. If no match is found, then an [InvalidClassException](#) is thrown.
- Example:



# Guidelines for serialVersionUID



- always include it as a field, for example: "private static final long serialVersionUID = 7526472295622776147L;" include this field even in the first version of the class, as a reminder of its importance
- do not change the value of this field in future versions, unless you are knowingly making changes to the class which will render it incompatible with old serialized objects
- new versions of Serializable classes may or may not be able to read old serialized objects; it depends upon the nature of the change;
- <http://java.sun.com/developer/technicalArticles/Programming/serialization/>



# Cool Eclipse shortcuts

- How to organize the imports?
  - CTRL+SHIFT+O
- How to easily write a loop?
  - Example... type “for” and press CTRL+Space
- How to find a class?
  - CTRL+SHIFT+T
- How to format your code?
  - CTRL+SHIFT+F



# Cool Eclipse Shortcuts

- How to find who is calling some method?
  - CTRL+SHIFT+G
- How to show the hierarchy of a class?
  - F4
- How to extract a method?
  - ALT+SHIFT+M
- How to rename using refactor?
  - ALT+SHIFT+R