PURDUE UNIVERSITY

CS18000: Problem Solving And Object-Oriented Programming

Graphical User Interfaces

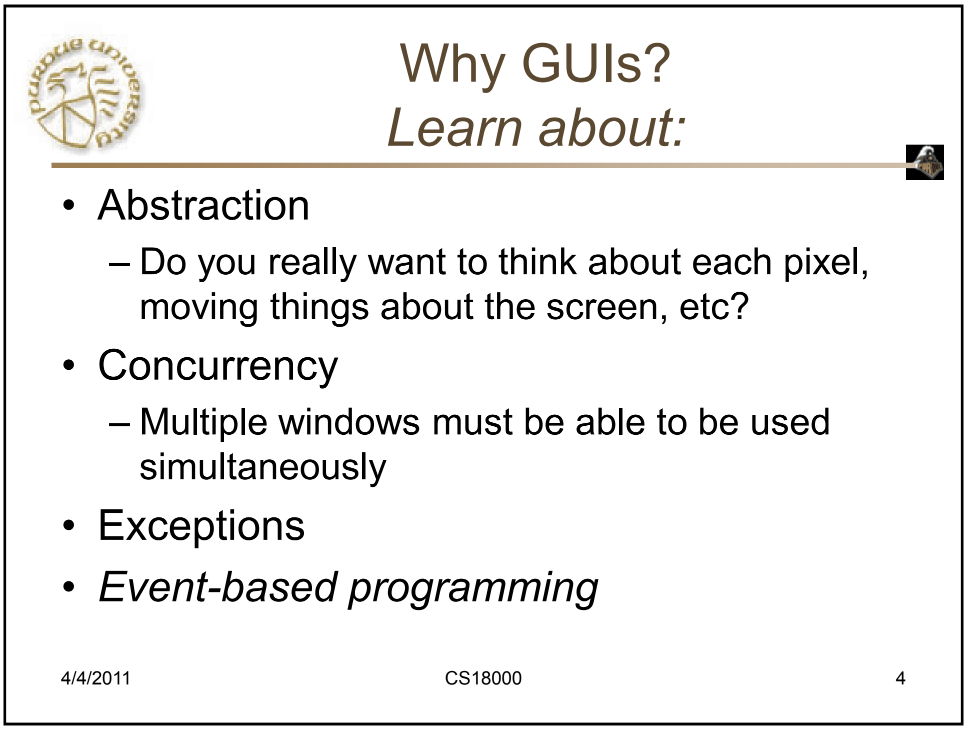
4 April 2011

Prof. Chris Clifton

COMPUTER SCIENCES

GUIs

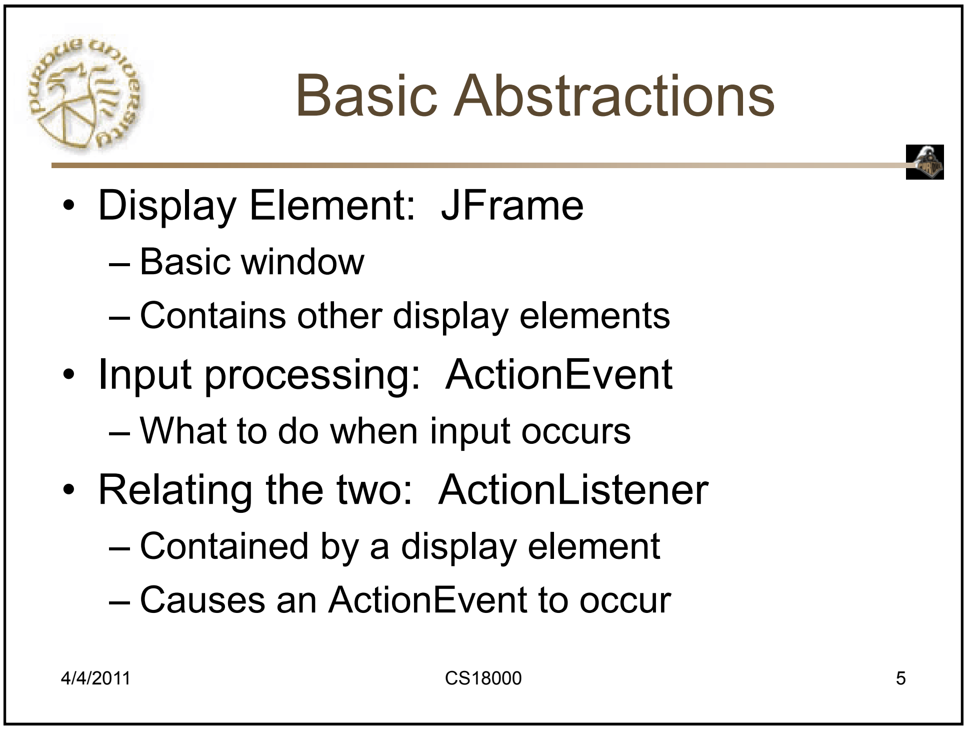
10/20/2010 ©Aditya Mathur. CS 180. 3



Why GUIs?
Learn about:

- Abstraction
  - Do you really want to think about each pixel, moving things about the screen, etc?
- Concurrency
  - Multiple windows must be able to be used simultaneously
- Exceptions
- *Event-based programming*

4/4/2011 CS18000 4

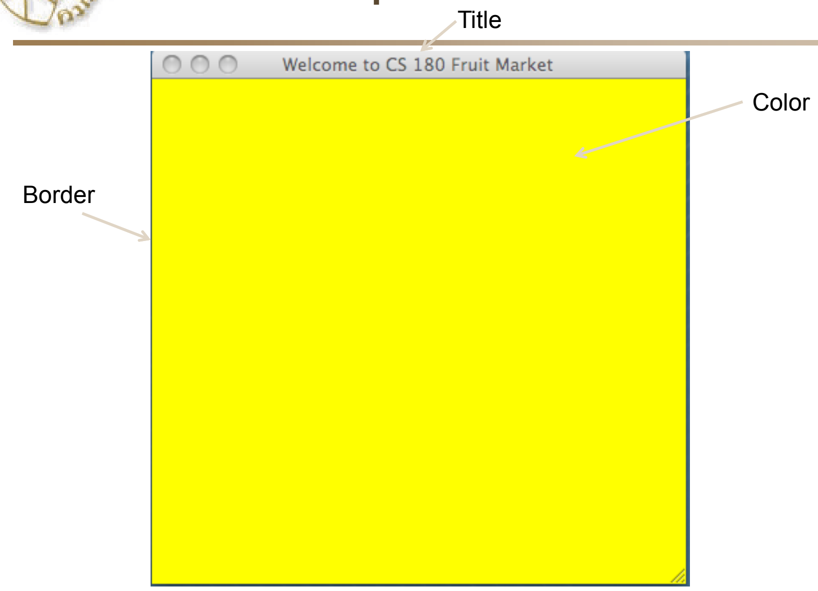


Basic Abstractions

- Display Element: JFrame
  - Basic window
  - Contains other display elements
- Input processing: ActionEvent
  - What to do when input occurs
- Relating the two: ActionListener
  - Contained by a display element
  - Causes an ActionEvent to occur

4/4/2011 CS18000 5

A simple GUI: Frame



6

What is a Frame?

- Empty window
  - Title (if window manager uses it)
  - size
  - location
  - Menu, Cursor, Icon Image, Decorations (close/iconify), …
- Properties
  - What to do when closed
- *Contents*

4/4/2011 CS18000 7

Simple example: Creating a JFrame

```
import javax.swing.*;
public class Frame {
public static void main( String[] args ) {
    JFrame justAFrame = new JFrame("Just a frame");
    justAFr
    justAFr
    justAFr
    justAFr
    System.out.println("Done.");
} }
```

Must Frame extend JFrame?

A. Yes, it needs to extend it to use it.
B. Yes, as otherwise it isn't a part of a GUI.
C. No, it can use a JFrame object
D. No, because it has a main()

4/4/2011 CS18000 8

Getting Rid of a Window

- setVisible(false);
  - Window can't be seen
  - But it still exists
- dispose();
  - Window can't be used
  - But can be re-enabled with "pack" or "show"
- Window manager closes window
  - Action determined by DefaultCloseOperation
- setDefaultCloseOperation
  - DO_NOTHING_ON_CLOSE
  - HIDE_ON_CLOSE
  - DISPOSE_ON_CLOSE
  - EXIT_ON_CLOSE

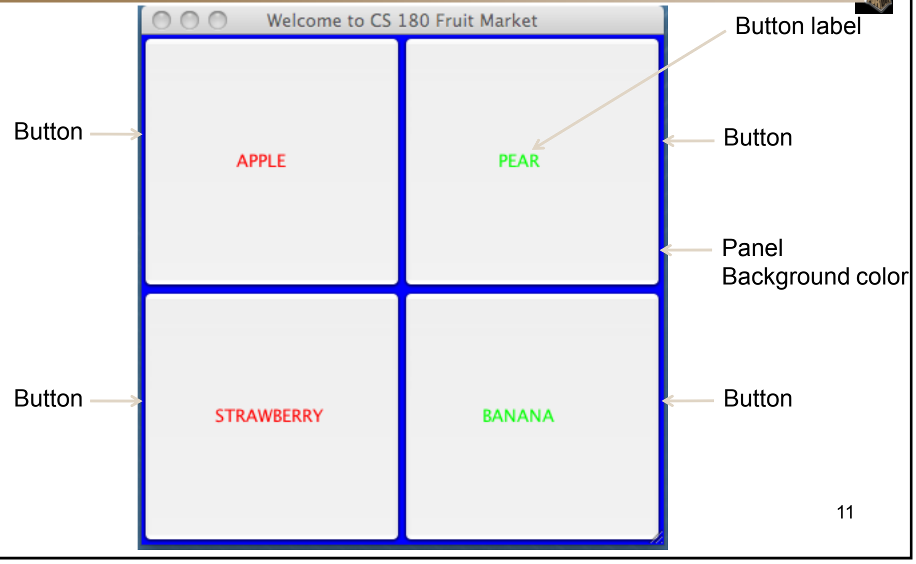
4/4/2011 CS18000 9

Window Contents

- What goes in a window? *Components*
  - JButton
  - JTextArea
  - Widgets
  - …
- Normally create a JPanel and add components to it
  - Then add panel(s) to frame

4/4/2011 CS18000 10

A simple GUI: Frame with a Panel and Four Buttons



11

Window Layout

- Abstraction: Don’t specify details of layout
- Panel has a LayoutManager
  - FlowLayout
  - BoxLayout
  - …
- Layout based on order components added
  - Some LayoutManagers have additional hints
- Default FlowLayout is left-to-right then top-to-bottom

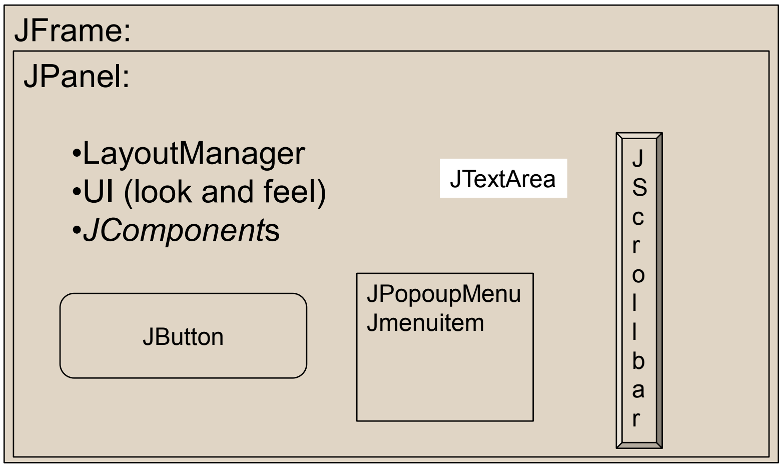
Creating a Window

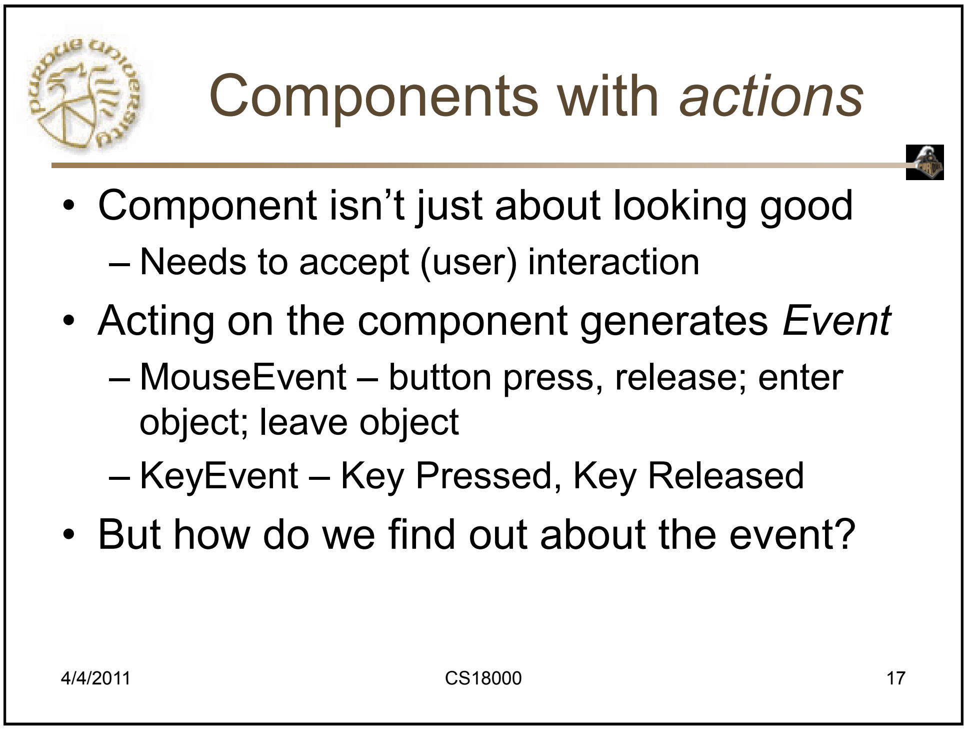
- Create a Frame
  - *Optionally set size, position, on-close action*
- Create a panel
  - *Choose layout methodology*
  - Create and add components to the panel
- Add panel to frame
- *(optional) pack() frame to set window size*
- Set visible

Next: Components with *actions*

- Component has listener
  - waits for event to occur
  - executes method when it does
- Easy to use
  - simply define appropriate methods
- But need to be careful
  - Think like concurrent programs

Window:

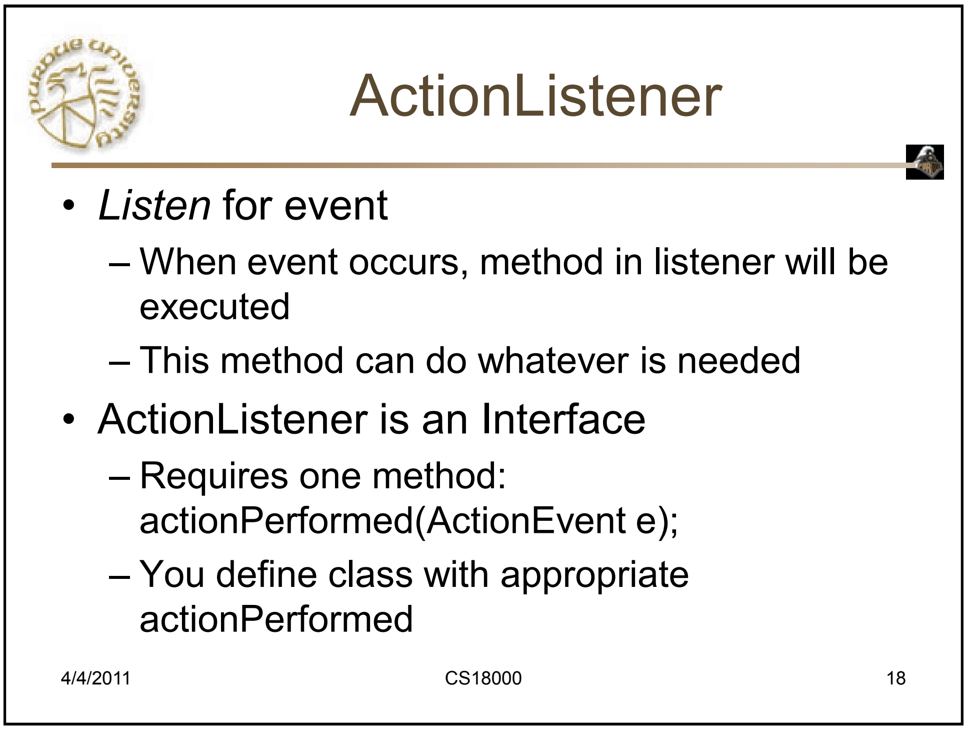




Components with *actions*

- Component isn't just about looking good
  - Needs to accept (user) interaction
- Acting on the component generates *Event*
  - MouseEvent – button press, release; enter object; leave object
  - KeyEvent – Key Pressed, Key Released
- But how do we find out about the event?

4/4/2011 CS18000 17



ActionListener

- *Listen* for event
  - When event occurs, method in listener will be executed
  - This method can do whatever is needed
- ActionListener is an Interface
  - Requires one method: actionPerformed(ActionEvent e);
  - You define class with appropriate actionPerformed

4/4/2011 CS18000 18

Simple listener

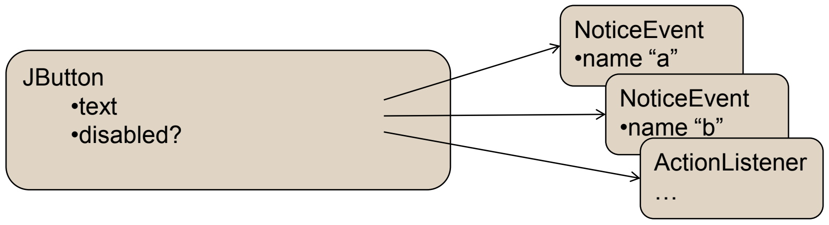
```
class NoticeEvent implements ActionListener {
    private String listenerName = null;
    public NoticeEvent(String name) {
        listenerName = name;
    }
    public void actionPerformed(ActionEvent e) {
        System.out.println("Listener " + listenerName +
          ":  " + e.getActionCommand() +
          " occurred at " + e.getWhen() +
          " with parameters " + e.paramString() );
} }
```

Using a listener

The JButton object stores elements as a:

A. Linked List, because we don't know how many it may have
B. Array, because it is simpler.
C. Who cares? We just need to add and remove listeners!
D. Linked List, because not all the listeners are the same.

– Some objects may have multiple listeners



Example

```
JFrame window = new JFrame("Examp");
Jbutton button = new JButton("Example");
ActionEvent a = new NoticeEvent("ex");
button.addActionListener(a);
JPanel panel = new JPanel();
panel.add(button);
window.add(panel);
window.pack();
window.setVisible(true);
```

Question: Are GUIs really concurrent?

- It seems like things are “just happening”
  - Similar to concurrent threads
- Program doesn’t end without window finishing
  - Similar to concurrent threads
- Is it really the same?
  - Exercise: Test and find out