keep iClickers out

CS1110 Lecture 23, 18 Nov 2010

Listening to GUIs

Reading for today: Ch 17.4.1, 17.4.4. [Optional: Ch. 12.1 & 12.1.1 (interfaces)]

For next lecture: Ch 10 (exceptions)

If a staff member isn't at their office hours, email djg17@cornell.edu, llee@cs.cornell.edu, dcv22@cornell.edu, and the missing person immediately! We might be able to locate a sub --- but you have to tell us there's a problem!

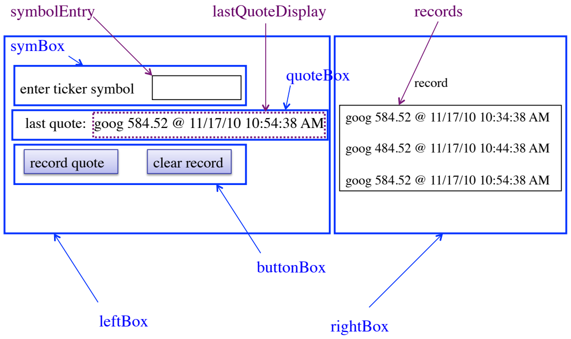
A7 (try to make the suggested milestones timeline! Also, typo on pg 3, 6 lines from bottom should be Breakout.main(new String[] {"2","3"}); A6 officially due tonight.

No labs next week.

There are labs on last week of classes, but they will be office hours plus an optional exercise on exceptions (covered on final).

Final: Friday Dec 10th, 9-11:30am, Statler Auditorium.
Register conflicts on CMS assignment "final exam conflicts" *by Monday November 30th*.

Blue: "invisible" organization elements; purple: fields


symbolEntry
lastQuoteDisplay
records
symBox
enter ticker symbol
quoteBox
record
last quote:
goog 584.52 @ 11/17/10 10:54:38 AM
goog 584.52 @ 11/17/10 10:34:38 AM
goog 484.52 @ 11/17/10 10:44:38 AM
goog 584.52 @ 11/17/10 10:54:38 AM
record quote
clear record
buttonBox
leftBox
rightBox

In class StockQuoteGUI extending JFrame, we could do either

(o1) buttonBox.add(new JButton("record quote"));

or

(o2) create **field** JButton rb= new JButton("record quote")
buttonBox.add(rb);

Which, if any, is the best option?

A. o1 and o2 are equal, because they both have the same effect

B. o1, because there's no reason that we need to store the name (on the tab) of the JButton anywhere

C. o2, because we might need to refer to that JButton

D. o2 except with rb being a static variable

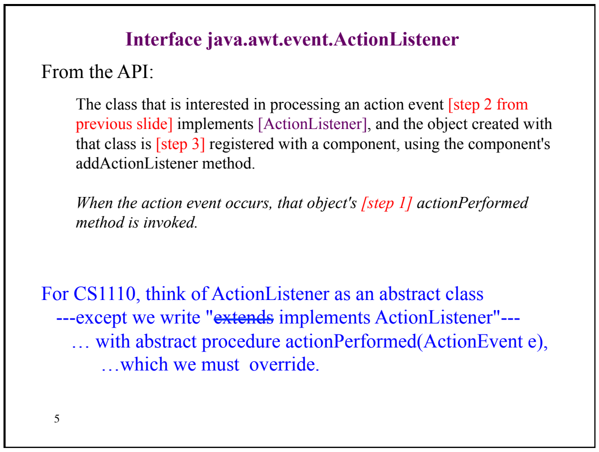
E. o2 except with rb being a local variable

Responding to events (in Java)

- An event is a mouseclick, a mouse movement into or out of a window, a keystroke, etc.

- To “listen to” (and hence react to) an event:
  1. Write a method in some class that will listen to the event.
  2. Let Java know that the method is defined in the class
  3. Register as a *listener* an instance-of-the-class-that contains-the-method with the component where the event could happen (e.g. a JButton)

The reason for this tripartite structure is that we want to use the ActionListener interface.

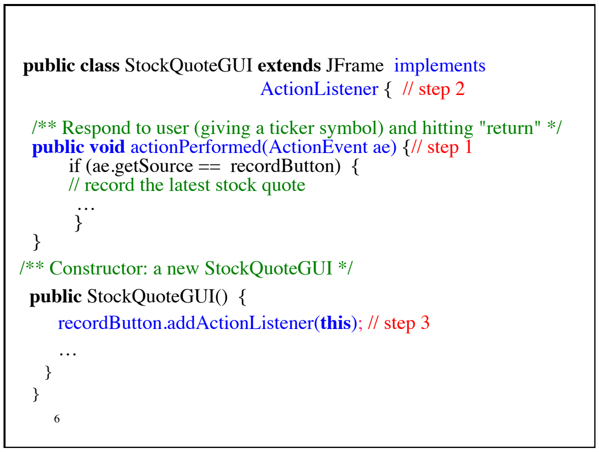


Interface java.awt.event.ActionListener

From the API:

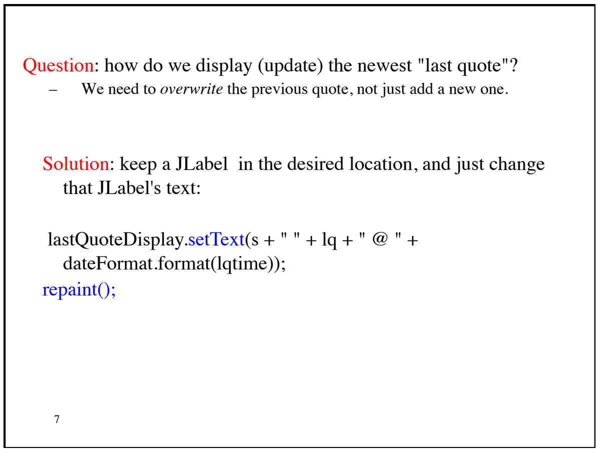
> The class that is interested in processing an action event [step 2 from previous slide] implements [ActionListener], and the object created with that class is [step 3] registered with a component, using the component's addActionListener method.
>
> *When the action event occurs, that object's [step 1] actionPerformed method is invoked.*

For CS1110, think of ActionListener as an abstract class
---except we write "~~extends~~ implements ActionListener"---
… with abstract procedure actionPerformed(ActionEvent e),
…which we must override.



```
public class StockQuoteGUI extends JFrame  implements
                                    ActionListener {  // step 2

  /** Respond to user (giving a ticker symbol) and hitting "return" */
  public void actionPerformed(ActionEvent ae) {// step 1
        if (ae.getSource ==  recordButton)  {
        // record the latest stock quote
          …
         }
  }
/** Constructor: a new StockQuoteGUI */
 public StockQuoteGUI()  {
      recordButton.addActionListener(this); // step 3
      …
   }
 }
```

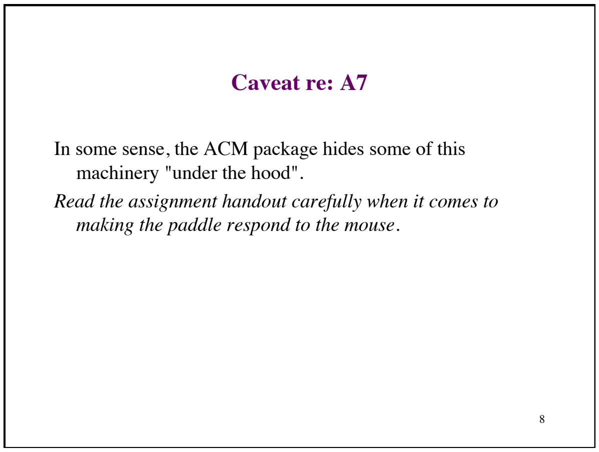


Question: how do we display (update) the newest "last quote"?

- We need to *overwrite* the previous quote, not just add a new one.

Solution: keep a JLabel in the desired location, and just change that JLabel's text:

```
lastQuoteDisplay.setText(s + " " + lq + " @ " +
   dateFormat.format(lqtime));
repaint();
```



Caveat re: A7

In some sense, the ACM package hides some of this machinery "under the hood".

Read the assignment handout carefully when it comes to making the paddle respond to the mouse.