Programming Languages and Techniques (CIS120)

Lecture 34

April 13, 2016

Swing II: Inner Classes and Layout

Event handling in Java vs. OCaml

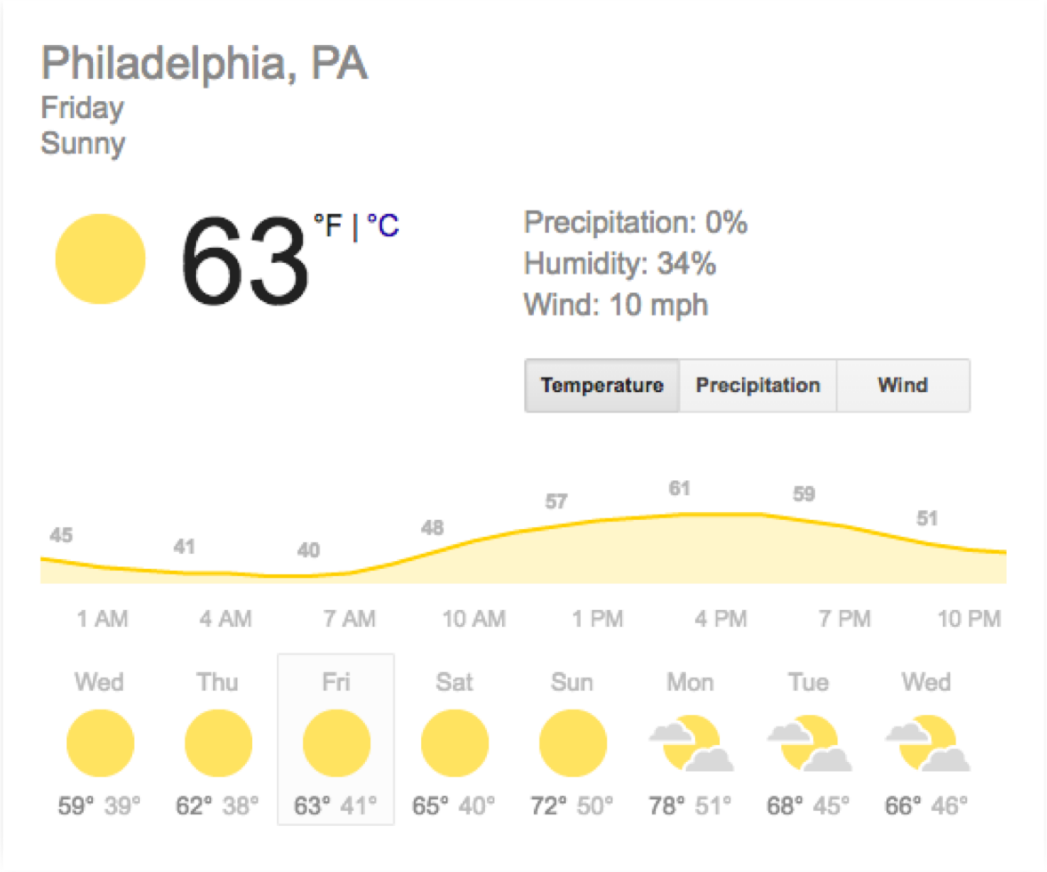
```
class ButtonListener implements ActionListener {
    private LightBulb bulb;
    public ButtonListener (LightBulb b) {
        bulb = b;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        bulb.flip();
        bulb.repaint();
    }
}

// somewhere in run …
LightBulb bulb = new LightBulb();
JButton button = new JButton("On/Off");
button.addActionListener(new ButtonListener(bulb));
```

Which version do you prefer? Why?

1. Java
2. OCaml

```
let bulb, bulb_flip = make_bulb ()
let onoff,_, nc = button "On/Off"
;; nc.add_event_listener (mouseclick_listener bulb_flip)
```

Announcement



Philadelphia, PA
Friday
Sunny

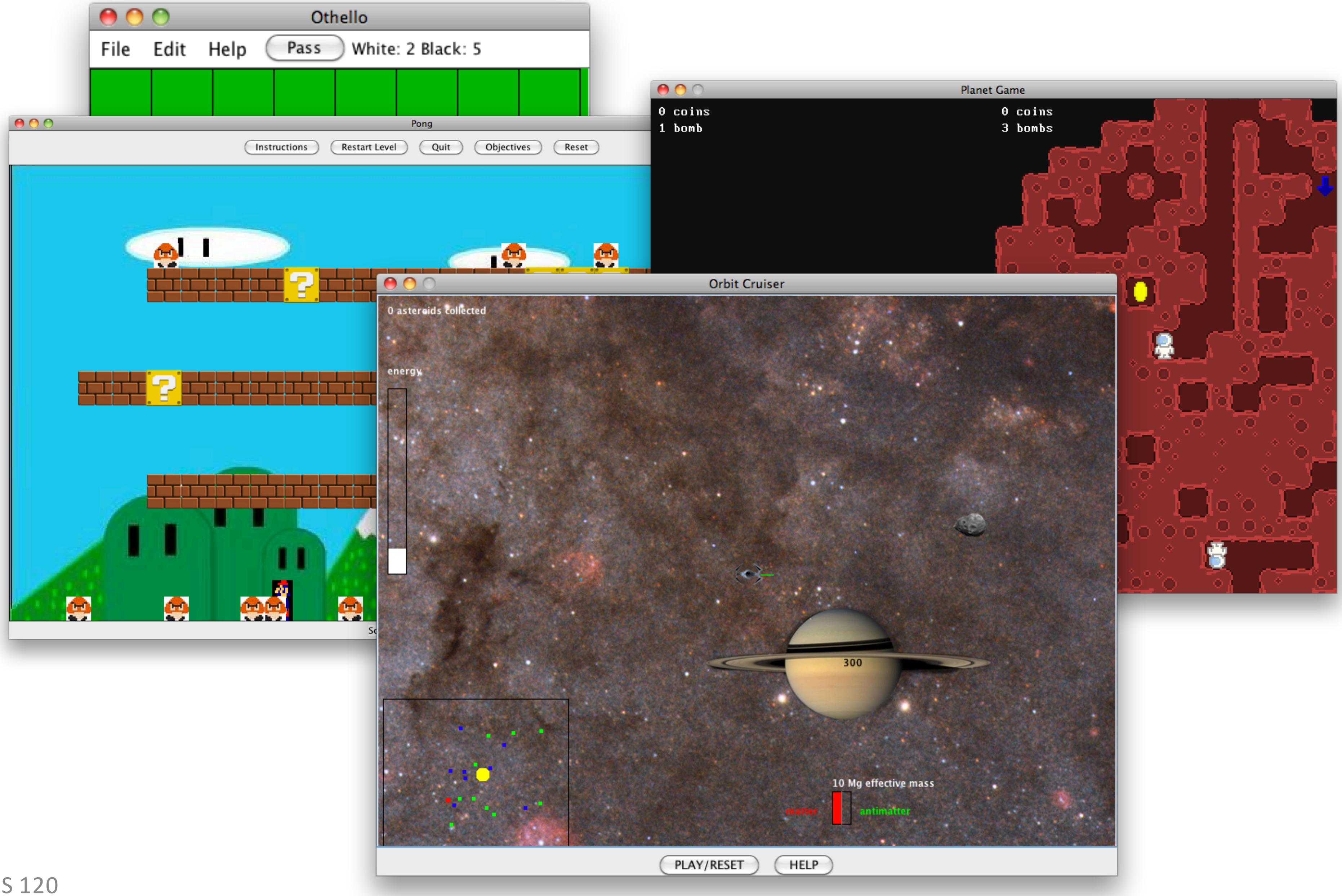
63 °F | °C

Precipitation: 0%
Humidity: 34%
Wind: 10 mph

Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed
59° 39°	62° 38°	63° 41°	65° 40°	72° 50°	78° 51°	68° 45°	66° 46°

- Wear sunscreen this weekend
- No class Friday

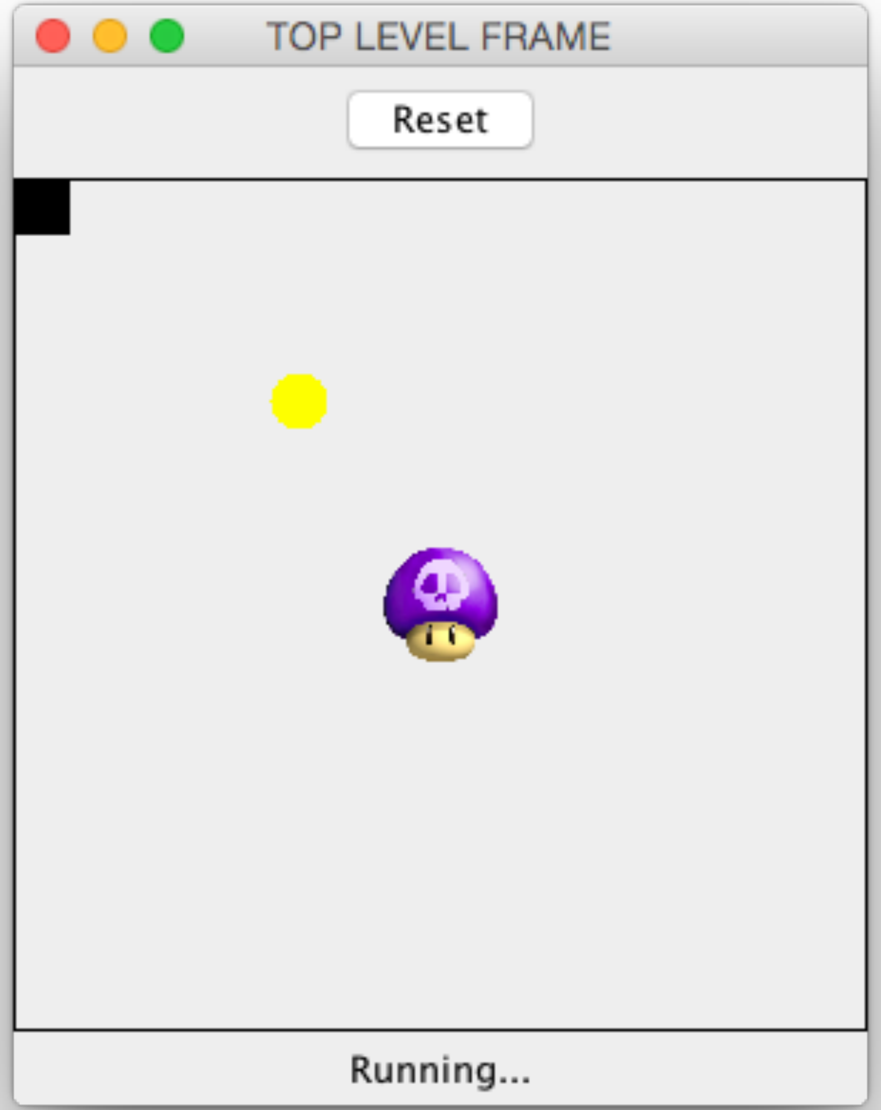
HW9: Game Project Available now



Game project

- Game Design Proposal Milestone Due: (12 points)
  Tuesday April 19th at 11:59pm
  - (Should take about 1 hour)
  - Submit proposal.txt
  - **Must** discuss your ideas with any TA BEFORE you submit
  - STRONGLY encouraged to check in before Tuesday
- Final Program Due: (88 points)
  Tuesday April 26th at 11:59pm
  - Submit zipfile online, submission *only* checks if your code compiles
- Grade based on demo with your TA during reading days
  - Make sure that you test your program in Moore 100, especially if you use outside libraries
  - Grading rubric on the assignment website
  - Recommendation: don't be too ambitious.

- *NO LATE SUBMISSIONS PERMITTED*


TOP LEVEL FRAME
Reset
Running...

How to have first-class computation?

```
class ButtonListener implements ActionListener {
    private LightBulb bulb;
    public ButtonListener (LightBulb b) {
        bulb = b;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        bulb.flip();
        bulb.repaint();
    }
}

// somewhere in run …
LightBulb bulb = new LightBulb();
JButton button = new JButton("On/Off");
button.addActionListener(new ButtonListener(bulb));
```

```
let bulb, bulb_flip = make_bulb ()
let onoff,_, bnc = button "ON/Off"
;; bnc.add_event_listener (mouseclick_listener bulb_flip)
```

Inner Classes

Anonymous Inner Classes

- Define a class and create an object from it all at once, inside a method

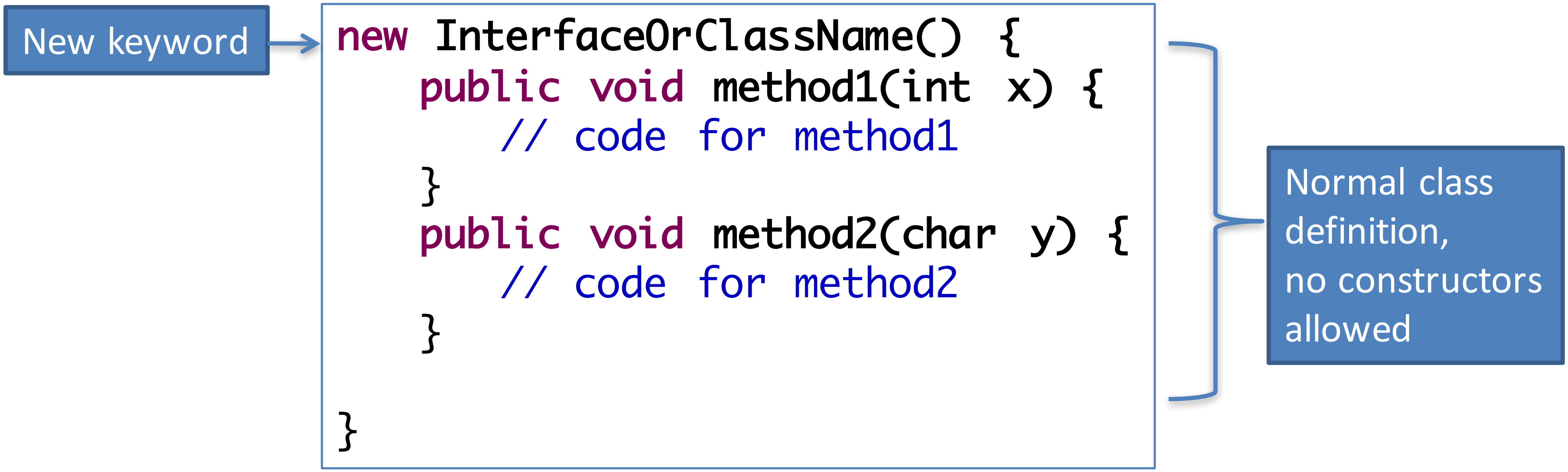
Puts button action right with button definition

```
final LightBulb bulb = new LightBulb();
JButton button = new JButton("On/Off");

button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        bulb.flip();
        bulb.repaint();
    }
});
```

Can access fields and methods of outer class, as well as final local variables

Anonymous Inner class

- New *expression* form: define a class and create an object from it all at once



New keyword

```
new InterfaceOrClassName() {
    public void method1(int x) {
        // code for method1
    }
    public void method2(char y) {
        // code for method2
    }

}
```

Normal class definition, no constructors allowed

Static type of the expression is the Interface/superclass used to create it

Dynamic class of the created object is anonymous! Can't refer to it.

Like first-class functions

- Anonymous inner classes are the real Java equivalent of Ocaml first-class functions
- Both create "delayed computation" that can be stored in a data structure and run later
  - Code stored by the event / action listener
  - Code only runs when the button is pressed
  - Could run once, many times, or not at all
- Both sorts of computation can refer to variables in the current scope
  - OCaml: Any available variable
  - Java: only instance variables (fields) and variables marked final

Quiz

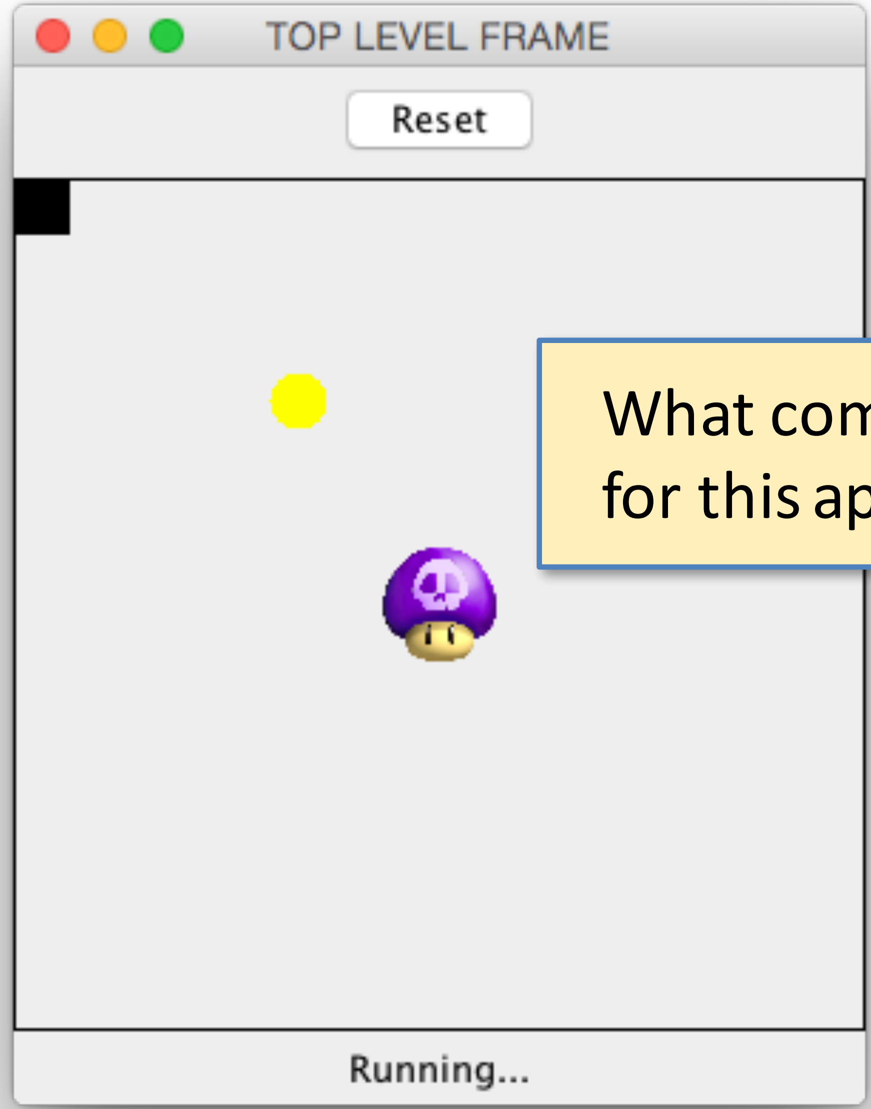
```java
public class Demo {
    private JLabel label1 = new JLabel("a label");

    void m(JLabel label2) {
        JLabel label3 = new JLabel("another label");

        JButton button = new JButton("button");
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                label1.setText("label1"); // 1
                label2.setText("label2"); // 2
                label3.setText("label3"); // 3
            }

        });
    }
}
```

Which reference is allowed?

1. label1 only
2. label2 only
3. label3 only
4. all are
5. none are

Swing Programming Demo

Layout



What components would you use for this app?