Programming Languages and Techniques (CIS1200)

Lecture 35

Swing III: Layout and MoD

Chapter 31

Announcements (1)

- Midterm 2
  - Grades and solutions available
  - Regrade requests via Gradescope
    - Opens later today
    - Due by Friday, December 6th
- HW08: TwitterBot*
  - Due tomorrow
  - Practice with I/O and Collections
- HW9: Game Project
  - TAs will give you feedback soon
  - Final Program Due: Monday, December 9th at 11:59pm
  - Grade based on demo with your TA during/after reading days
  - *NO LATE SUBMISSIONS PERMITTED*

* Maybe should be called "XBot" or "TheProjectFormerlyKnownAsTwitterBot"or ”AnywhereButTwitterBot” or…?

Announcements (2)

- Plans for the week of Thanksgiving
  - HW08 due on Tuesday at 11.59pm
  - No recitations this week
  - TA OH till Tuesday will be virtual
  - No OH from Wednesday to Sunday

  - Wednesday, November 27th – Bonus Lecture
    - Come to either lecture (10:15 or noon)
    - Material is not needed for HW or Exams
    - Should be fun!
    - (Will be recorded)

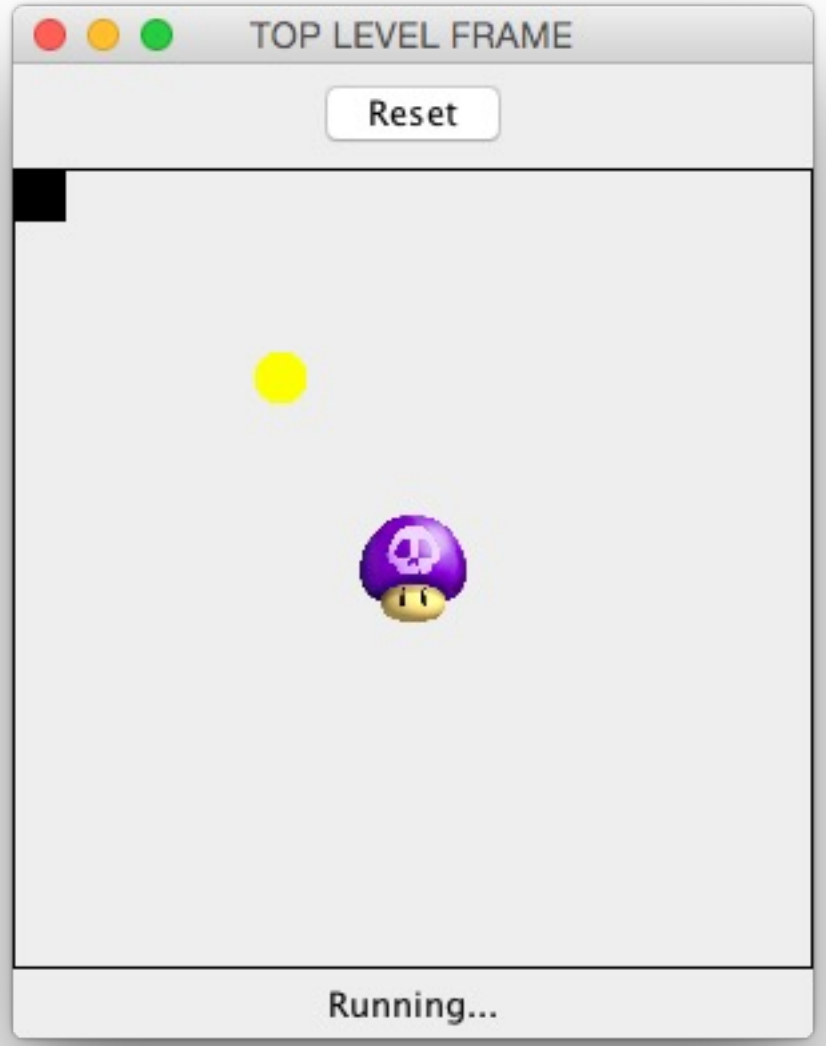
  - No lecture on Friday

Swing Layout Demo

LayoutDemo.java

Mushroom of Doom

How do we put Swing components together to make a complete game?


TOP LEVEL FRAME
Reset
Running...

JPanel (control_panel)

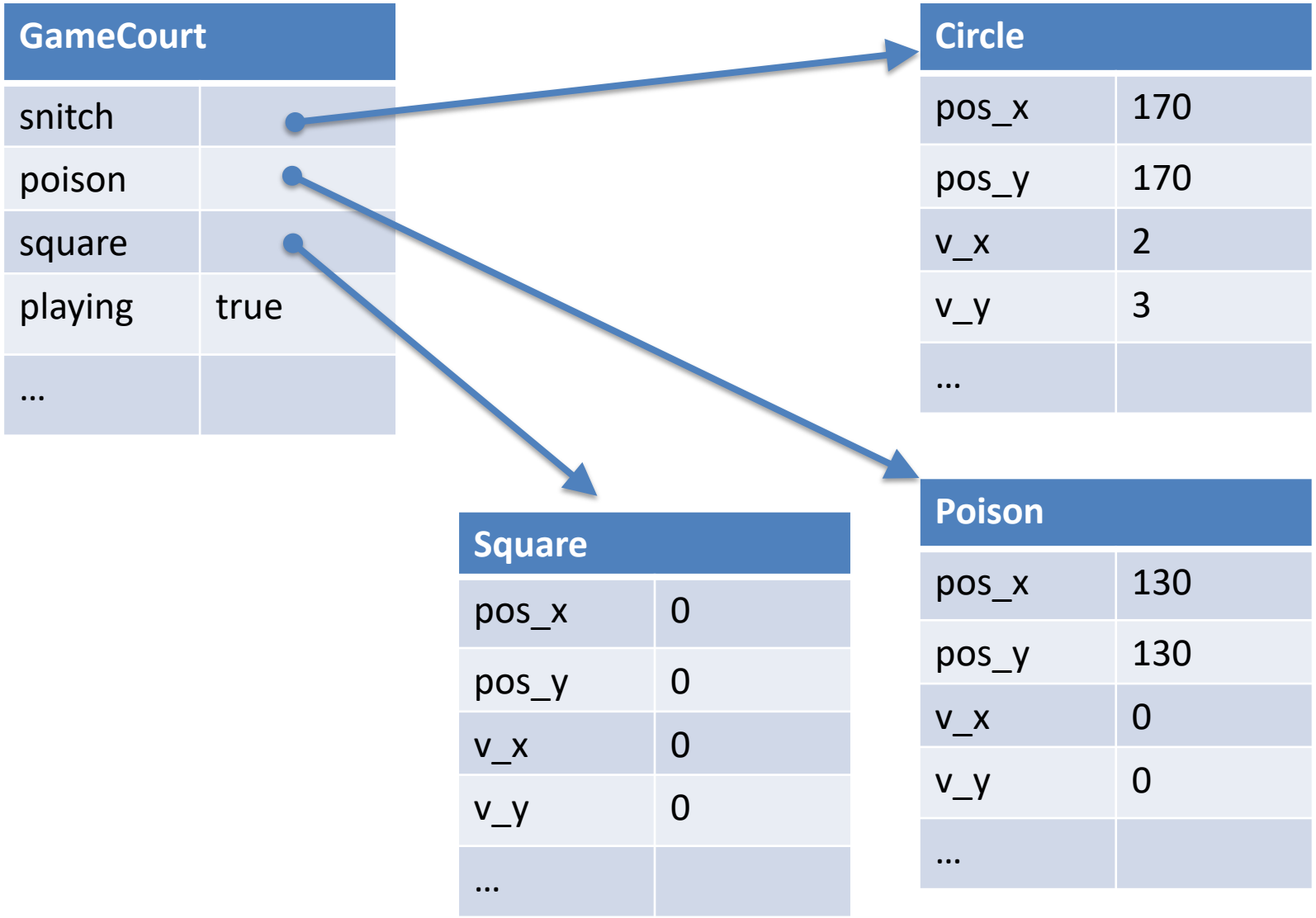
JButton (reset)

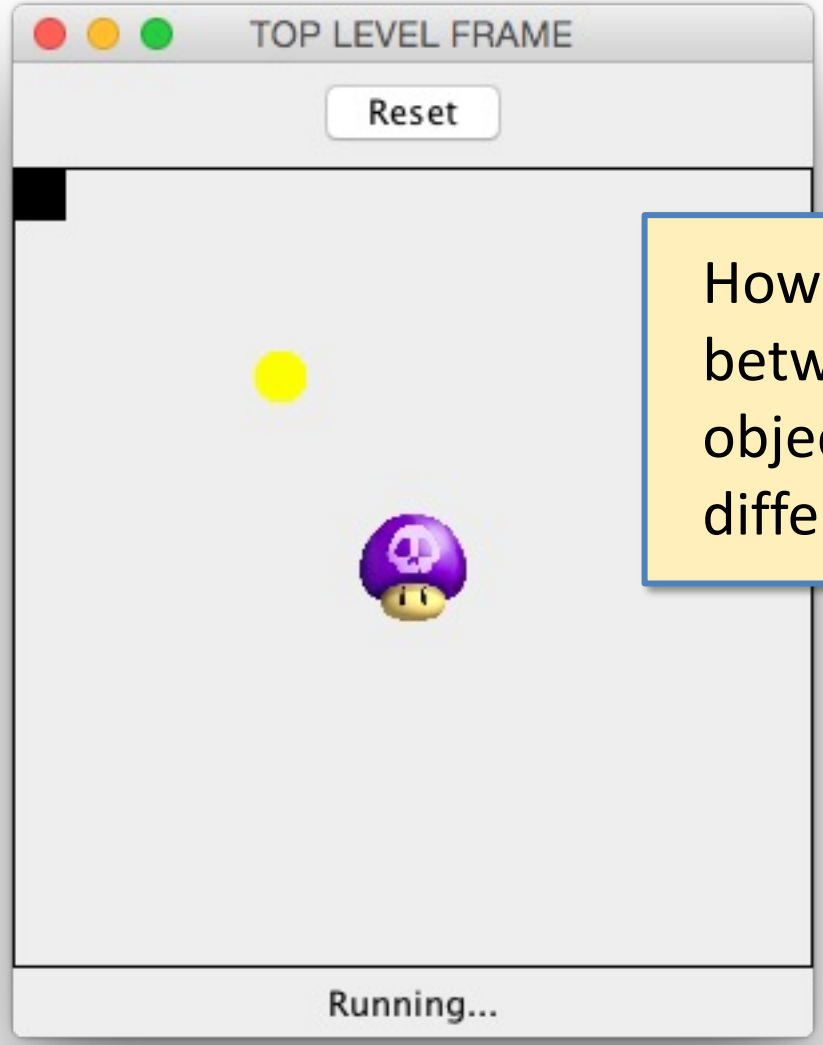
GameCourt, subclass of JPanel (court)

JPanel (status_panel)

JLabel (status)

Game State


GameCourt
snitch
poison
square
playing
true
…
Circle
pos_x
170
pos_y
170
v_x
2
v_y
3
…
Square
pos_x
0
pos_y
0
v_x
0
v_y
0
…
Poison
pos_x
130
pos_y
130
v_x
0
v_y
0
…



How can we share code between the game objects, but show them differently?

Abstract Classes

- An abstract class provides an *incomplete* implementation:
  - some methods are marked as abstract
  - those methods must be overridden to create instances

```
public abstract class AbstractClass {
    private int x = 0;
    public int m() {
        return frob(frob(x));
    }
    abstract int frob(int x);
}

class ConcreteClass extends AbstractClass {
    @Override
    int frob(int x) {
        return x * 120;
    }
}
```

Keyword "abstract" marks methods without implementations.

A subclass overrides the abstract method with an implementation.

36: It is possible to fill in __??__ with code so that, when run, the variable ac will contain an object of type AbstractClass.

0

True

0%

False

0%

```
public abstract class AbstractClass {
    private int x = 0;
    public int m() {
        return frob(frob(x));
    }
    abstract int frob(int x);
}

// somewhere in main:
Abstract Class ac = new AbstractClass __??__;
```

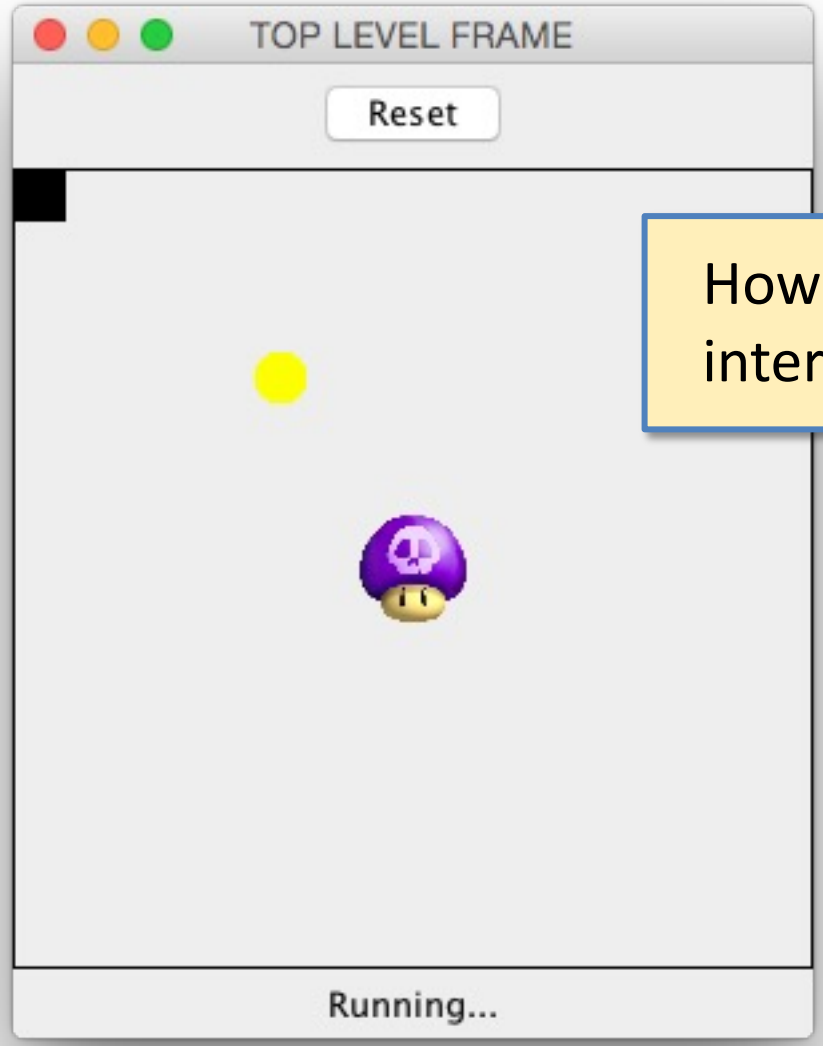
True or False: It is possible to fill in the hole marked __??__ so that, when run, the variable ac will contain a new object of type AbstractClass.

```
public abstract class AbstractClass {
    private int x = 0;
    public int m() {
        return frob(frob(x));
    }
    abstract int frob(int x);
}

// somewhere in main:
AbstractClass ac = new AbstractClass () {
    @Override
    int frob(int x) { return 0; }
};
```

Answer: True – use an anonymous inner class!

Updating the Game State: timer

```
void tick() {
  if (playing) {
    square.move();
    snitch.move();
    snitch.bounce(snitch.hitWall());      // bounce off walls...
    snitch.bounce(snitch.hitObj(poison)); // ...and the mushroom

    if (square.intersects(poison)) {
       playing = false;
       status.setText("You lose!");
    } else if (square.intersects(snitch)) {
       playing = false;
       status.setText("You win!");
    }
    repaint();
  }
}
```



How does the user interact with the game?

1. Clicking Reset button restarts the game
2. Holding arrow key makes square move
3. Releasing key makes square stop

Adapters

MouseAdapter

KeyAdapter

Two interfaces for mouse listeners

```
interface MouseListener extends EventListener {
   public void mouseClicked(MouseEvent e);
   public void mouseEntered(MouseEvent e);
   public void mouseExited(MouseEvent e);
   public void mousePressed(MouseEvent e);
   public void mouseReleased(MouseEvent e);
}
```

```
interface MouseMotionListener extends EventListener {
    public void mouseDragged(MouseEvent e);

    public void mouseMoved(MouseEvent e);
}
```

Lots of boilerplate

- There are seven methods in the two interfaces.
- We only want to do something interesting for three of them.
- Need "trivial" implementations of the other four to implement the interface…

```
public void mouseMoved(MouseEvent e)   { }
public void mouseClicked(MouseEvent e) { }
public void mouseEntered(MouseEvent e) { }
public void mouseExited(MouseEvent e)  { }
```

- Solution: MouseAdapter class…

Adapter classes

- Swing provides a collection of abstract event adapter classes
- These adapter classes implement listener interfaces with empty, do-nothing methods
- To implement a listener class, we extend an adapter class and override just the methods we need
- Another example: MouseListener and MouseMotionListener
  - Seven methods in two separate interfaces
  - Suppose we only need to override three of them

```
private class MyMouseListener extends MouseAdapter {
   public void mousePressed(MouseEvent e) { … }
   public void mouseReleased(MouseEvent e) { … }
   public void mouseDragged(MouseEvent e) { … }
}
```

KeyListener interface

```
interface KeyListener extends EventListener {

   public void keyPressed(KeyEvent e)
   // Invoked when a key has been pressed.

   public void keyReleased(KeyEvent e)
   // Invoked when a key has been released.

   public void keyTyped(KeyEvent e)
   // Invoked when a key has been typed.
}
```

KeyAdapter class

```
class KeyAdapter implements KeyListener {

   public void keyPressed(KeyEvent e) { return; }
   // Invoked when a key has been pressed.

   public void keyReleased(KeyEvent e) { return; }
   // Invoked when a key has been released.

   public void keyTyped(KeyEvent e) { return; }
   // Invoked when a key has been typed.
}
```

Using the Keyboard

- The "**Focus**" determines which JComponent is notified when a keyboard event occurs

- During set up

```
setFocusable(true);     // Enable key events
addKeyListener(…);      // Register reactions to events
```

- Once the component is visible

```
// Make sure that this component has the keyboard focus
requestFocusInWindow();
```

Updating the Game State: keyboard

```
setFocusable(true);
addKeyListener(new KeyListener() {
  public void keyPressed(KeyEvent e) {
     if (e.getKeyCode() == KeyEvent.VK_LEFT)
          square.v_x = -SQUARE_VELOCITY;
     else if (e.getKeyCode() == KeyEvent.VK_RIGHT)
          square.v_x = SQUARE_VELOCITY;
     else if (e.getKeyCode() == KeyEvent.VK_DOWN)
          square.v_y = SQUARE_VELOCITY;
     else if (e.getKeyCode() == KeyEvent.VK_UP)
          square.v_y = -SQUARE_VELOCITY;
     }

     public void keyReleased(KeyEvent e) {
          square.v_x = 0;
          square.v_y = 0;
     }

    public void keyTyped(KeyEvent e) { }
});
```

Make square's velocity nonzero when a key is pressed

Make square's velocity zero when a key is released

do nothing

Updating the Game State: keyboard

```
setFocusable(true);
addKeyListener(new KeyAdapter() {
  public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_LEFT)
        square.v_x = -SQUARE_VELOCITY;
    else if (e.getKeyCode() == KeyEvent.VK_RIGHT)
        square.v_x = SQUARE_VELOCITY;
    else if (e.getKeyCode() == KeyEvent.VK_DOWN)
        square.v_y = SQUARE_VELOCITY;
    else if (e.getKeyCode() == KeyEvent.VK_UP)
        square.v_y = -SQUARE_VELOCITY;
    }

    public void keyReleased(KeyEvent e) {
        square.v_x = 0;
        square.v_y = 0;
    }
});
```

Make square's velocity nonzero when a key is pressed

Make square's velocity zero when a key is released