CSSE 220

Event Based Programming

Check out *EventBasedProgramming* from SVN

Interfaces - Review

- Interfaces are contracts
  - Any class that ***implements*** an interface **MUST** provide an implementation for all methods defined in the interface.
- Interfaces represent the abstract idea (and what it can do):
  - Discount (calculate discount)
  - Function (get name, evaluate result)
- Classes represent the concrete idea:
  - Fixed Discount, Percentage Discount
  - Add, Multiple, Divide, Subtract, etc.

Interfaces – Review (continued)

- The specific method to use at runtime is decided by late-binding

```
Function add = new Add();
double result = add.evaluate(doubles);
```

The *declared type* of add is **Function**

The *instantiation type* is **Add**

At run-time, Java will use the method implementation of evaluate from the **Add** class thanks to late-binding.

Finish the sentence

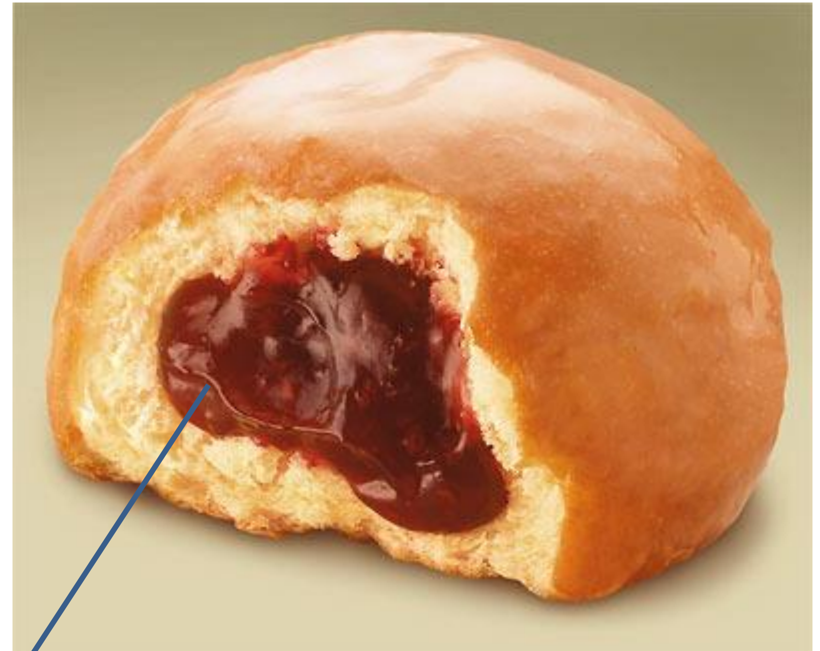
Using interfaces can help reduce ________ between classes.

a. Coupling
b. Cohesion
c. Encapsulation
d. Polymorphism

We need interfaces for event-based programming in Java.

Graphical User Interfaces in Java

- We say what to draw
- Java windowing library:
  - Draws it
  - Gets user input
  - Calls back to us with events
- We handle events



Hmm, donuts

Gooey

Events – What, When, Why, How?

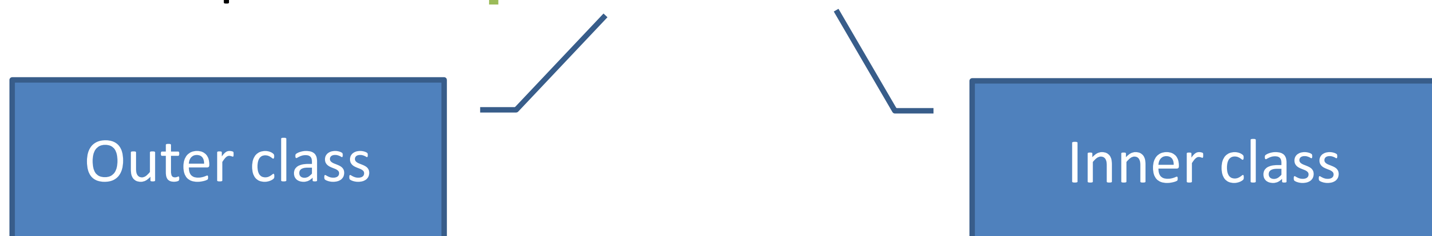
- What:
  - Indication that something has occurred in the application
    - Click, Key Pressed, Window Closed, etc.
- When:
  - Operations that are not required to occur in a specific order
    - User Interaction with screen elements
    - Mouse pressed, mouse released, mouse moved, mouse clicked, button clicked, key pressed, menu item selected, …

Events – What, When, Why, How?

- Why:
  - Handles operations that occur in many different orders
- How:
  - **Implement the interface** corresponding with the event
    - Ex, public class ButtonListener implements ActionListener {}
  - **Create a listener object** AND
  - **Register the event** with **event source** who will call it
    - Ex, button.addActionListener(new ButtonListener());

Inner Classes – What, When, Why, How?

- What:
  - Classes defined **inside** other classes or methods
- When:
  - "Smallish" helper classes
  - Often used for `ActionListener`s
  - Sometimes used for allowing multiple types for an item:
    - Example: `Ellipse2D.Double`



Inner Classes – What, When, Why, How?

- Why:
  - Inner class gets access to the final and static fields and methods of the containing class
- How:

```
public class OuterClass {
    //fields and methods

    class InnerClass {
        //inner class's fields and methods
    }
}
```

Anonymous Classes – What, When, Why, How?

- What:
  - **Anonymous** → no name
  - A special case of inner classes
- When:
  - When you only need one instance
  - The implementation is very small
  - Used for the simplest **`ActionListener`**s…
- Why:
  - Provide the scope necessary for implementing the handler
- How:

```
button.addActionListener(new ActionListener() {
    //implement methods
}
```

Inner Classes and Scope

- **Inner classes can access any variables in surrounding scope**

- Caveats:
  - Local variables must be `final`
  - Can only use instance fields of surrounding scope if we're inside an instance method

- Example:
  - Prompt user for what porridge tastes like

Key Layout Ideas

- JFrame’s add(Component c) method
  - Adds a new component to be drawn
  - Throws out the old one!
- JFrame also has method add(Component c, Object constraint)
  - Typical constraints:
    - BorderLayout.NORTH, BorderLayout.CENTER
  - Can add one thing to each “direction”, plus center
- JPanel is a container (a thing!) that can display multiple components

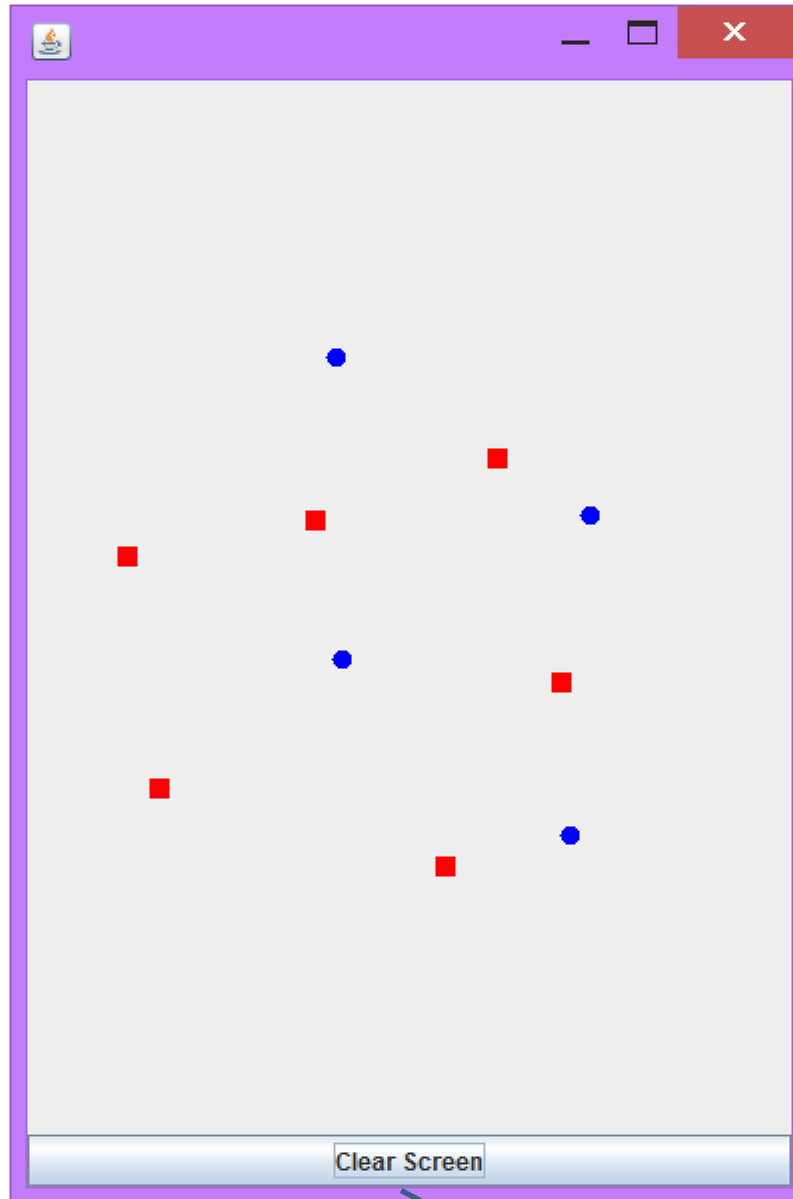
Repaint (and thin no more)

- To update graphics:
  - We tell Java library that we need to be redrawn:
    - `drawComponent.repaint()`
  - Library calls `paintComponent()` when it's ready

- **Don't call `paintComponent()` yourself! It's just there for Java's call back.**

Mouse Listeners

```
public interface MouseListener {
   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);
}
```



So, how do we do this?

Work Time

- LinearLightsOut