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):
  - Measurable objects (return a measure)
  - NumberSequences (get the next number, reset)
- Classes represent the concrete idea:
  - Country, Bank Account
  - AddOne, PowersOfTwo.

Interfaces – Review (continued)

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

```
Sequence sequence = new PowersOfTwo();
System.out.println(sequence.next());
```

The *declared type* of operation is **Sequence**

The *instantiation type* is **PowersOfTwo**

At runtime, Java will use the method implementation of next() from the **PowersOfTwo** class, thanks to late-binding.

Finish the sentence

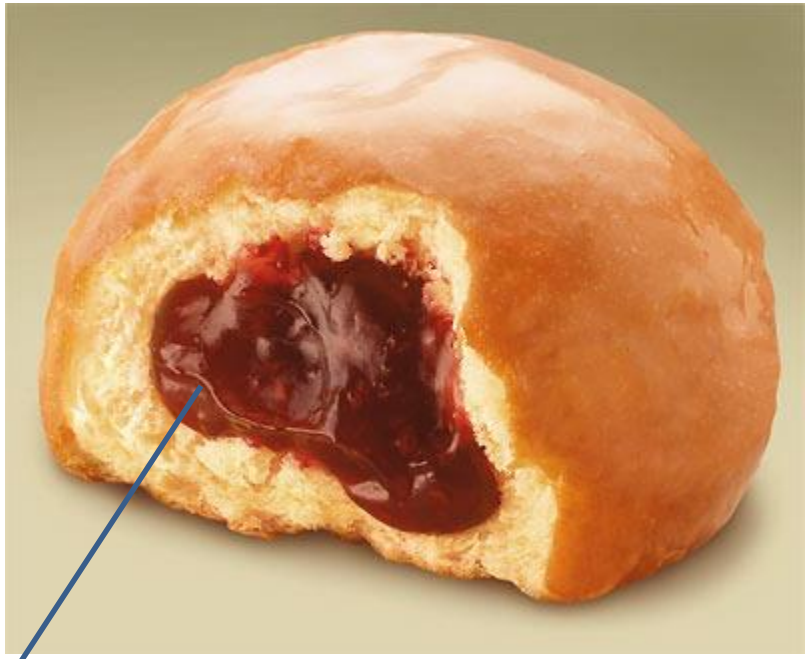
Using interfaces can help reduce _____ between classes.

1. Coupling
2. Cohesion
3. Encapsulation
4. 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

Handling Events

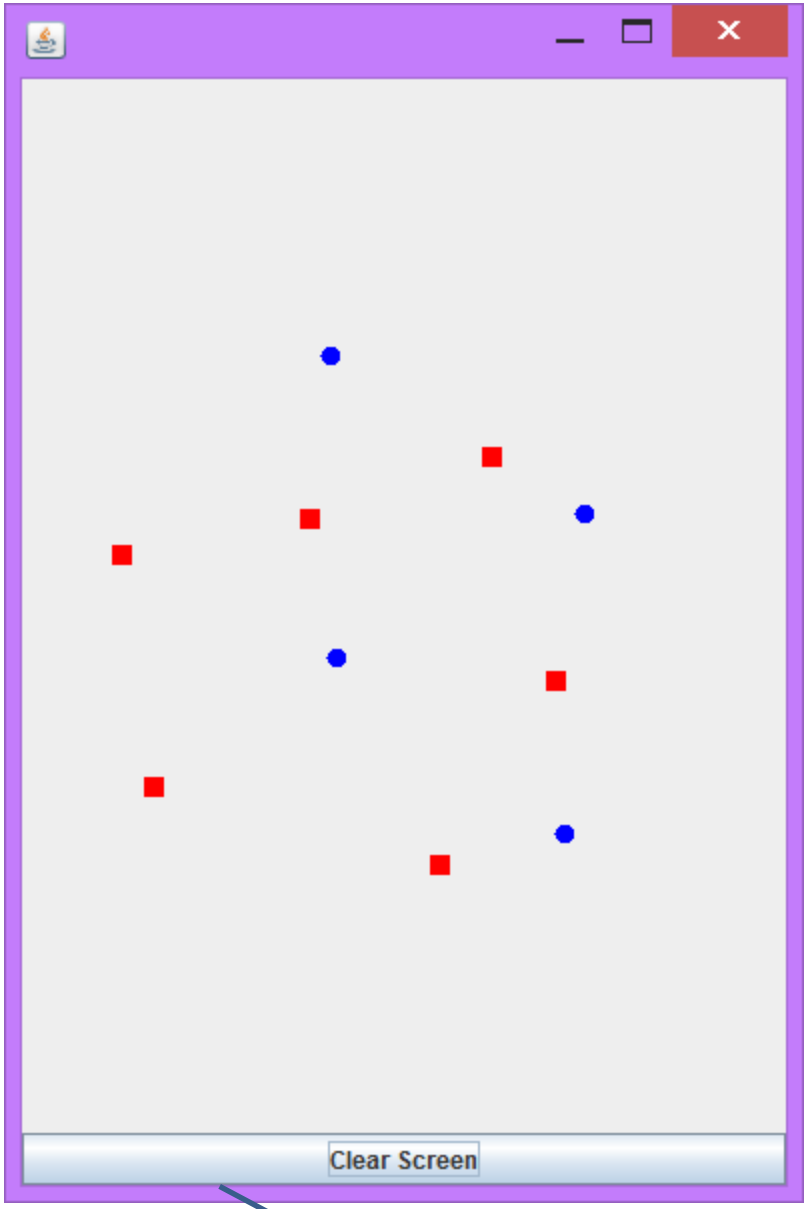
- Many kinds of events:
  - Mouse pressed, mouse released, mouse moved, mouse clicked, button clicked, key pressed, menu item selected, …
- We create **event listener objects**
  - that implement the right **interface**
  - that handle the event as we wish
- We **register** our listener with an **event source**
  - Sources: buttons, menu items, graphics area, …

Q1

Draw a blue circle on left-click, red square on right-click

Each 20x20, centered on click

Clear screen button does what it says.



So, how do we do this?

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

Q2,3

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);
}
```

Q5

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.**

Q4

Using Inner Classes

- Classes can be defined **inside** other classes or methods
- Used for “smallish” helper classes

- Example: `Ellipse2D.Double`

- Often used for `ActionListener`s…
- Add to Breakfast program?

Q6

Anonymous Classes

- Sometimes very small helper classes are only used once
  - This is a job for an anonymous class!

- **Anonymous** → no name
- A special case of inner classes

- Used for the simplest `ActionListener`s…

Inner Classes and Scope

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

- Caveats:
  - Can only use instance fields of surrounding scope if we're inside an instance method

- Example:
  - Prompt user for what porridge tastes like

Work Time

- LinearLightsOut