Announcements

• Lab 4 in recitation this week – due next week.
• Lab 5 started next week in recitation.
• Exam 3 after spring break.
New Relationship: Composition

- Whole-part relationship
- The “source” is responsible for creating the “target”
- The lifetime of the target is linked to the lifetime of the source

Composition

- In Java code:
  ```java
  public class Source {
      private Target _target;
      public Source() {
          _target = new Target();
      }
  }
  ```
User Interactive Components

- `javax.swing.JButton`
- Create one and place it on a graphical container
- Note that its default functionality is none
- We need to tell the button how to react when clicked upon

Events

- All things that a user does to interact with the system can be considered events
- These events are noticed by the computer and reacted to by various programs that are running on the system
Events

• If we want the components of our program to react to user events, we need to create event handlers
• These event handlers know what to do when an event has been observed

Events

• Clicks on a button are ActionEvents
• We create a listener (ActionListener) to react to those events
• Use method addActionListener on a JButton to indicate which listener(s) are to be notified when an event is observed
**Observer**

- The way Java handles events conforms to the Observer design pattern

- Design Patterns are formal ways to describe general solutions to common problems

**ActionListener**

- All objects that will react to events and want to be registered observers need to be ActionListeners
- However, as we saw, there is no way to create an ActionListener object because ActionListener is not a class, it is an interface
Interfaces

- Another type that a user can define in Java
- Interfaces give the declarations of capabilities without giving implementations of those capabilities

Implementing an Interface

- Classes that a programmer defines can implement an interface
- In order to do this, the class must provide definitions for all the capabilities the interface has left undefined
Realization Relationship

• Implementing an interface is an example of the Realization Relationship from our UML relationships

Realization

• In Java code:

```java
public class Source implements Target {
}
```
Notes about UML

• Note that a box that represents an interface in UML only has two sections, a section for the name and a section for the capabilities
• Interfaces do not have properties, so no section needed

Code Example

• GraphicsExample3 in Lecture Code repository
• Note that there is a UML diagram in the project that shows the realization relationship
GraphicsExample3

• Creates JFrames when the user clicks the button
• Let’s create a button that does something else
  – Suggestions?

Exam 2 Stats

• Min: 37 (1 total F)
• Median: 90
• Average: 87.76
• Max: 100 (26; 73 total A’s)
To Work on now...

• Write out a plan (in English) as to how to create the program suggested last lecture:

• A shape on the screen whose movement is controlled by a button.

How do I do this?

• Read the description
• Get ready to do your work
• THINK before you code
  – Doodle, sketch, brainstorm
  – Organize your thoughts
What do we need to do? (Brainstorming)

- Make a package
- Make a JFrame
- Make DrawingCanvas
- Make shape
- Create a button
- Need an ActionListener
- What does the button do?

The example

- Code is in GraphicsExample4 in the Lecture Code repository