Week 6 - Wednesday

COMP 2000

Last time

- What did we talk about last time?
- Finished layout
- Action listeners

Questions?

Project 2

Action Listeners

ActionListener interface

- What can listen for a **JButton** to click?
- Any object that implements **ActionListener**
- **ActionListener** is an interface like any other with a single abstract method in it:

```
void actionPerformed(ActionEvent e);
```

- We need to write a class with such a method
- We will rarely need to worry about the **ActionEvent** object
- But it does have a **getSource()** method that will give us the **Object** (often a **JButton**) that fired the event

Adding an action listener

- The reason we brought up anonymous inner classes is that we can use this syntax to make an **`ActionListener`** object right when we need it, for a button

```
JButton button = new JButton("Push me!");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        button.setText("Ouch!"); // arbitrary code
    }
}); // ugly: parenthesis for end of method call
```

- It's ugly, but it works

Things you might do in an action listener

- Call arbitrary methods
- **`setText()`** sets the text on many widgets
- **`getText()`** gets the text from widgets so you can do something with it
- Both **`setText()`** and **`getText()`** apply to:
  - **`JButton`**
  - **`JLabel`**
  - **`JTextField`**
  - **`JTextArea`**
- **`setIcon()`** sets the icon on many widgets
  - **`JButton`**
  - **`JLabel`**
- **`setEnabled()`** can be used to enable and disable buttons

Weird rules

- Using lambdas looks cleaner, but the same anonymous inner classes are being created
- When you write code in the method of an anonymous inner class
  - You can refer to member variables and methods in the anonymous inner class (if any)
  - You can refer to member variables and methods in the surrounding object (even private ones)
  - You can generally read the values of local variables, but you **cannot** change them
- Don't worry too much about all this
- Just write your action listeners and come see me if you have problems

Make the calculator work

- Using this information about action listeners, we should be able to make calculator GUI we created functional
- Buttons **0-9** and **.** should add the appropriate symbol to the display **`JTextField`**
- Buttons **+**, **-**, *, and **/**
  - Should parse what's in the **`JTextField`** into a **`double`** and store it in a member variable
  - Store the appropriate operation in a member variable (maybe as a **`char`**?)
  - Should clear the **`JTextField`**
- Button **=**
  - Should parse what's in the **`JTextField`** into a double
  - Perform the operation that was stored earlier with this value and the value stored earlier
  - Should put the result back in the **`JTextField`**

Sounds

Sounds

- In COMP 1600, we played sounds using the **`StdAudio`** library
- There are many Java libraries for playing sound
- We're not going to focus deeply on the multimedia playback facilities provided by Swing or other Java libraries
- But playing a sound now and then is useful
- Especially because you have to for Project 2

Clip

- One relatively straightforward way to play an audio file is to use the **`Clip`** interface
- We can retrieve an appropriate object that implements **`Clip`** by calling the static **`getClip()`** method on the **`AudioSystem`** class
- **`Clip`**, **`AudioSystem`**, and some of the other classes we'll mention are in the **`javax.sound.sampled`** package

```
Clip clip = AudioSystem.getClip();
```

Opening a sound file

- Once you have a **`Clip`** object, you can open an audio file in three steps
  - Create a **`File`** object with the path of the audio file
  - Use the **`AudioSystem.getAudioInputStream()`** method to create an **`AudioInputStream`**
  - Open the **`AudioInputStream`** with the **`Clip`**

```
File file = new File("gunshot.wav");
AudioInputStream stream =
AudioSystem.getAudioInputStream(file);
clip.open(stream);
```

- The **`AudioSystem`** library supports **`.au`** and **`.wav`** files (and maybe others) but not **`.mp3`** or **`.flac`**

Playing the file

- Once you have the audio opened, you can play it with a couple of methods on the **`Clip`** object
  - **`start()`** will play the sound once
  - **`loop(times)`** will loop the sound **`times`** times (or **`Clip.LOOP_CONTINOUSLY`** to play forever)

```
clip.loop(10); // Plays 10 times
```

- When the sound is playing, you can stop it with the **`stop()`** method
- Note that the sound will stop playing if the program ends
- That's one reason why this stuff makes more sense in a GUI program, where the program continues on

Quiz

Upcoming

Next time…

- Recursion

Reminders

- Read Chapter 19
- Keep working on Project 2