GUIs part 1
Ch 6.1-6.3

Formatting Hard Disk.
This will erase all information on your computer!
Highlights

- Make windows

- Put stuff in windows
Sequential vs. Event driven

So far, we have done sequential programming. This means we start at the beginning of main() and go one line at a time.

```java
public static void main(String[] args) {
    System.out.println("Hello");
    System.out.println("Goodbye");
}
```

(See: Sequential.java)
Sequential vs. Event driven

Exceptions no longer obeyed this rule:
- You could hop to the catch block no matter where the exception was thrown from (inside the try block)

This was our first experience of event driven programming, the user could change the order our program ran in

(See: SemiSequential.java)
Sequential vs. Event driven

Graphic User Interfaces (GUIs) are almost completely event driven programming
- The user is very much in control of what code is run

This means you will make methods that you never actually call anywhere (weird)

There are many different ways (classes) to do GUIs, but we will use the built in “swing”
The basis of GUIs are windows (or frames)

To make a window, you need to make a JFrame object

- Need to import javax.swing.JFrame

```java
JFrame window = new JFrame();
```

In order to actually see it, we need to run the setVisible() method on the JFrame (arg: true)

```java
window.setVisible(true);
```

(See: BasicWindow.java)
There are a few other methods that you should normally use to setup the JFrame window:

setSize(int width, int height) - Makes the window size width by height pixels

setDefaultCloseOperation() - Sets up what clicking the X on top does (default behavior hides the program!)

(See: ClosingWindow.java)
You can make buttons that do things when pressed

To make a button, you make a JButton object:

```java
JButton pressMe = new JButton("Press me!");
```

And then add that to your JFrame instance:

```java
window.add(pressMe);
```

(We will not cover layouts, but I will use FlowLayout)

```java
window.setLayout(new FlowLayout());
```

(See: WorthlessButton.java)
The previous code example is rather worthless since the button does not do anything

In order to get a useful button you need to implement the ActionListener interface

So far we have not strayed too far from sequential programming, but now we will (Note: main() returning does not end the program)
Let's make a class that implements the ActionListener interface:

```java
public class RandomDigit implements ActionListener
```

Then we need to override the `actionPerformed()` method:

```java
@override
public void actionPerformed(ActionEvent event)
```

(See: RandomDigit.java)
Link JButton to ActionListener

Now we can react when the user presses our button (JButton pressMe)

Finally, we need to link our ActionListener to our JButton:

```java
ActionListener randomNumber = new RandomDigit();
pressMe.addActionListener(randomNumber);
```

pressMe instance of JButton

(See: UsefulButton.java)
Oh, hi! I'm here from the Internet.

\ What are you doing!? Gluing captions to your cats.
JLabel

JLabel is simply some text in a window that does not interact with the user, used for:
- Helping the user navigate your GUI
- Convey information

To do this, you simply make a JLabel object and then add it to your window (like a JButton):

```java
JLabel digits = new JLabel("Press this button to get a digit");
window.add(digits);
```

(See: LabelsAndButtons.java)
JTextField

To make a box where the user can enter text, create a JTextField object (and add it to JFrame)

```java
JTextField text = new JTextField("Type here");
window.add(text);
```

JTextField has some useful methods:
- `setText(String)` - sets the text in the box
- `getText()` - returns the String of text in the box

(See: TextGUI.java)
Make it stop!

(See: MakeItStop.java)
MouseListener

ActionListeners work when you press a button, but it is much more fun to work with a mouse.

You can implement MouseListener to get information from the mouse:

```java
public class MouseThingy implements MouseListener
```

This works very similar to ActionListener, except with more methods.
MouseListener

The methods you need to override are:

mousePressed() - when you press down
mouseReleased() - when you let go of click
mouseClicked() - a full click (no drag)
mouseEntered() - mouse goes on top of your window (from outside the window)
mouseExit() - when the mouse cursor leaves your window

(See: TellMeMouseMoveCoordinates.java)