



Computer Science 252

Problem Solving with Java

The College of Saint Rose
Spring 2014

Topic Notes: Java's Swing GUI

Here's a program with a few "buttons" which are used to drop a ball at varying speeds. The buttons are ugly!

See Example: `FallingBallUglyButtons`

Real, modern programs do not use buttons that look like this...

We can add some nicer looking buttons to our program using Java Swing graphical user interface components:

See Example: `FallingBallButtons`

Putting aside the fact that the buttons here are in very odd places and are quite huge, they at least look nicer and show a graphical reaction when pressed, much like the buttons we see and use all the time in modern programs.

See the detailed comments in the example to see how this is accomplished.

We can make the buttons look nicer and all live together at the bottom of the window by using a `JPanel` to group them.

See Example: `FallingBallButtonsPanel`

Again, see the comments in the example to see the details.

We could approach the speed selection for the ball in a somewhat different manner using a `JComboBox`, often referred to as a dropdown menu.

See Example: `FallingBallComboButton`

We can enhance this a bit to react to changes in the `JComboBox` as well by also making our class the action listener for the `JComboBox`.

See Example: `FallingBallComboReact`

Our next Swing component is the `JSlider`, which looks a lot like a scroll bar in some cases.

See Example: `FallingBallSlider`

There are many more Java Swing components and associated event handlers. Some of these will come up in future examples, and include:

- `JLabel` – the equivalent of an `Objectdraw Text` object, except that it does not live in a `DrawingCanvas`.

- `JTextField` – a small editable text box.
- `JTextArea` – a large editable text box.
- radio buttons and check boxes, menus, and much more.

There is much more than can be done with `JPanels` and layout managers to arrange the Swing components within a window just how you want them.

Note that these can be used within `Objectdraw` without a canvas being created by having your class extend `Controller` instead of `WindowController` and without `Objectdraw` by having your class extend `JApplet` and providing an `init` method instead of a `begin` method.

You will only be responsible for those components, interfaces, and event handling mechanisms that specifically arise in our class examples, labs, and other assignments.