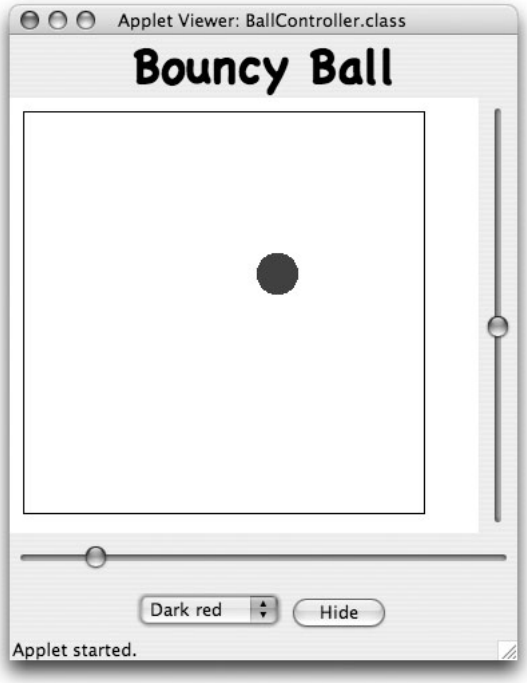
Programming Exercise
Follow the Bouncing Ball

Objective: To gain experience using Swing components and listeners.

The Scenario. In this lab, we would like you to get practice in the use of Swing components. So we would like you to create a program that includes two `JSlider`s, a `JLabel`, a pop-up menu (`JComboBox`), and a `JButton`.

A picture of the screen can be seen below:



As usual, the main part of the screen is the `canvas`. When the program begins, a ball will start bouncing around the canvas. (We will provide a class containing the code to make the ball bounce.) The user uses the various Swing components to change the appearance and speed of the ball. The changes should take effect immediately.

At the "North" side of the screen is a `JLabel` identifying the program as "Bouncy Ball". At the "East" side is a slider (`JSlider`) which controls the vertical velocity of the ball. At the "South" side of the screen is a panel (`JPanel`) which holds a slider which controls the horizontal velocity, a pop-up menu (`JComboBox`) which allows the user to change the ball's color, and a `JButton`, which is used to show/hide the ball. If the `JButton` is pressed, the ball should be hidden and the button's label should be changed to "Show". Pressing it again should show the bouncing ball and change the label back to "Hide". The ball should continue to move while it is hidden and you should be able to change the ball's properties while it is hidden.

How to Proceed First a brief Swing review: To display a component, you must create the component and add it to the display. To accept input, you must attach a listener to the component, declare that the listening class `implements` the listener interface, and define the listening method required by that interface.

Next, examine the two incomplete classes we have provided to help you start this lab: `BallController.java` and `BouncyBall.java`. `BallController` is the class to which you will add the Swing components. `BouncyBall.java` contains the class that implements the bouncing ball. We have already provided the `begin` and `run` methods

to get the ball to bounce. You will need to add methods that allow its behavior to change as the components are used.

Note that the statements `import java.awt.*;`, `import java.awt.event.*;`, `import javax.swing.*;`and `import javax.swing.event.*;` appear at the top of our starter .java file. These lines inform Java that your program will need access to the Java libraries that support GUI components and events (including event listeners).

To complete your implementation, proceed as follows:

1. Include code in your `begin` method to add a label (`JLabel`) centered at the "NORTH" end of the program window. The label should say "Bouncy Ball". To set the font of the label, you need to create a new font using:

   ```
   new Font (String fontName, int style, int size);
   ```

   The style can be `Font.BOLD`, `Font.ITALIC`, `Font.PLAIN`, or `Font.BOLD + Font.ITALIC`. The font used in the demo is "Chalkboard". You can use any other font available on your machine.

   This component does not accept input so you do not need a listener.

2. Now work on a component that accepts user input by adding the vertical speed slider to the "EAST" side of the canvas. The slider is a `JSlider` representing the range of vertical speeds in pixels per second. The demo ranges from 40 to 600 pixels/second with an initial value of 40 pixels/second.

   In order to allow your `WindowController` extension to "listen" to the `JSlider`, make sure that your class header ends with the phrase "`implements ChangeListener`" and that you invoke the method `addChangeListener(this)` on your `JSlider`. Define the `stateChanged` method so that every time the `JSlider` is "adjusted", the vertical speed of the ball changes.

   You also need to add a method `setVerticalSpeed` in the `BouncyBall` class to make the ball change. Note that `BouncyBall` expects speeds in pixels/millisecond, but the slider gives you speeds in pixels/second. Divide the slider value by 1000.0 (a double representing the number of milliseconds in a second) to convert the value inside `setVerticalSpeed`. Look at how the speed parameters are used in the `BouncyBall` constructor and mimic that code. Recall as well that you need to multiply the speed passed in by the length of the pause (`PAUSETIME`) to scale the speed appropriately.

   Finally, `BouncyBall` uses negative speeds to move up (and to move left). The slider always gives you a positive value in the range. Before changing the speed of the ball, remember to check if the ball currently has a negative speed. If it does, multiply the slider speed by -1 so the ball continues to move in the same direction, just at a different speed.

3. Notice that there is more than one component below the canvas on the display. You need to create a `JPanel` to hold the collection of components. The panel should have the horizontal slider in one row and the remaining components in a second row. To do this, you need to set the new panel's layout (using `setLayout`) to a `GridLayout` with 2 rows and 1 column.

   You will add the horizontal slider first to the new panel so that it occupies the top row. To allow the second row to contain more than one component, you need to create a panel for the second row. This panel can use the default `FlowLayout` manager.

4. Add the horizontal speed slider to the "SOUTH" panel. Its implementation is quite similar to the vertical speed slider.

5. Add the color menu to the panel created to contain the second row. The color menu is a `JComboBox` whose entries list color names. See the GUI cheat sheet for the constructor and the method used to add choices to the `JComboBox`. Be sure to add to the class header that your program "implements" `ActionListener`. (When you have a class which implements several interfaces, you simply separate

the interfaces with commas. Do NOT include the keyword `implements` more than once in the class header!) Also, don't forget to tell the `JComboBox` that your program will be the listener.

Add code to the appropriate method to handle `ActionEvent`s. You will need to add a `setColor` method to the `BouncyBall` class to change the ball's color.

6. Next add the hide/show `JButton`. When the user clicks on the button, the ball should hide and the label should change to "Show" using `setLabel`. The opposite behavior should occur if the user clicks again. You can find out what the current label is by calling `getLabel` on the button.

Submitting Your Work Before turning in your work, be sure to double check both its logical organization and your style of presentation. Make your code as clear as possible and include appropriate comments describing major sections of code and declarations.