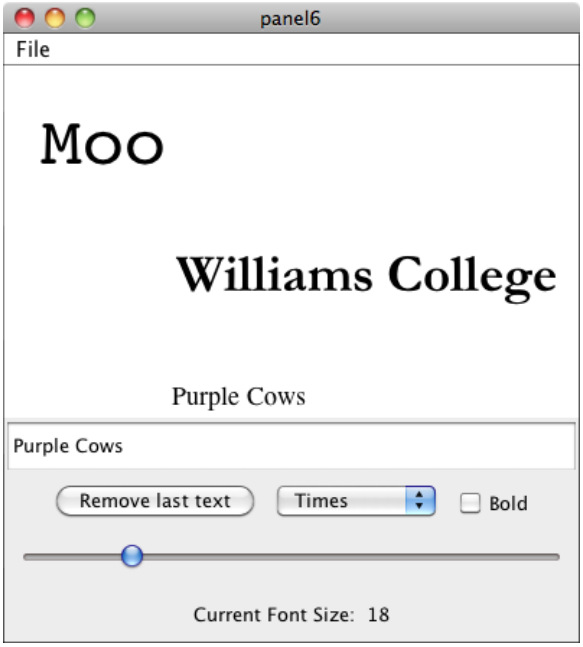
Lab 5

TextPlay

Objective To gain experience using GUI components and listeners.

Note You may work with a partner on this lab.

The Scenario You will write a program that manipulates text through a `JSlider`, a couple of `JLabels`, a pop-up menu (`JComboBox`), a `JTextField`, a `JButton`, and a `JCheckBox`. A picture of the screen can be seen below:



As usual, the center of the screen is the `canvas`. At the "South" end of the screen is a `JPanel` that holds a `JTextField` where the user can write some text to be displayed on the `canvas`, a subpanel that contains three components (used to select a font, make text bold, and remove text from the screen), a `JSlider` used to control the font size, and below that a label showing the current font size. The main"South" panel should use a `GridLayout` with four rows and one column as its layout manager. The subpanel uses a simple `FlowLayout`. The components included in the subpanel are a `JButton` that is used to remove the last item written on the screen, a pop-up menu (`JComboBox`) that allows the user to choose from among several fonts (we used "Courier", "Verdana", "Arial", and "Times"), and a `JCheckBox` to indicate whether text should appear in bold.

When the user clicks anywhere on the `canvas`, your program should display (at the place where the user clicked) the text showing in the `JTextField`. It should be displayed on the canvas in the selected font, font size, and bold-ness. When the user adjusts the `JSlider`, selects a new font with the `JComboBox` menu, or checks/unchecks the "Bold" `JCheckBox`, the last text placed on the `canvas` should change its size or font accordingly. Changing the text in the `JTextField` has no impact on items displayed on the `canvas`.

If the "Remove last text" `JButton` is pressed, the last text placed on the canvas should be removed. (You may think of this as a one-level undo. Pressing the button a second or third time will not remove additional items.)

How to Proceed

First, download a copy of the starter folder from the handouts page and open it in BlueJ. Before beginning, be sure that you have your GUI cheat sheet in front of you. (You can also find it on the CS134 web site.) As a quick reminder, remember that the basic steps in displaying components are:

1. Construct an instance of the component.
2. Initialize it if necessary (like adding items to a `JComboBox`).
3. Add the component to a `JPanel` or to the content pane of the window controller.

To react to events generated by the user interacting with a GUI component, remember these steps:

1. Add the window controller as a listener to the component, calling a method like `addActionListener`.
2. Declare that the window controller class implements the listener interface for that component, such as `ActionListener`.
3. Implement the method that is called when the user interacts with the component, such as `actionPerformed`.

Note that the statements

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
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).

Here is a suggestion on how to decompose the development of this program into simpler steps.

1. Include code in your `begin` method to add a `JPanel` at the "South" end of the program window. This `JPanel` should use a `GridLayout` having 4 rows and 1 column. Place a `JTextField` in the panel. Make it so that if the user clicks anywhere in the `canvas` then whatever is showing in the `JTextField` is displayed as a `Text` item on the canvas. Make sure that successive clicks on the canvas insert new `Text` items on the screen (showing whatever is currently in the `JTextField`).

   You will want a variable associated with the `Text` item most recently displayed so that you can change its font or font size later. You will associate this name with the current text item in your `onMouseClick` method.

   Since your program does not react to the user typing in the `JTextField`, it is not necessary to associate a listener with this component.

2. Now that you can display text items, start working on adding the components that will let you modify them.

   Start with the `JButton`.

   Add a `JButton` to the panel so that it contains the words "Remove last text". In order to allow your `WindowController` extension to respond when the button is pressed, add the phrase "`implements ActionListener`" to the header of the class. Call the button's `addActionListener` method in your `begin` method to inform the button that your `WindowController` wants to be notified when the button is pressed, and add an `actionPerformed` method to your class that performs the appropriate action (removing the text from the canvas) when the button is pressed.

   Make sure your program behaves reasonably if the user clicks the button before actually putting any text on the canvas.

3. At this point, there are only two GUI components in the panel you have placed in the south quadrant of your display: the `JTextField` and the `JButton`. Since this panel uses a 4x1 `GridLayout`, there is no room left in the second row to add extra components. To make it possible to add extra components in that row of this `GridLayout JPanel`, you should create a new subpanel to hold the `JButton` you already created and the `JComboBox` component you will create in the next step. Then, add the existing `JButton` to the subpanel rather than to the original panel. After this is done, test the program again. It should work the same except the `JButton` will no longer be stretched to fill the entire width of the window.

4. The next step is to create the `JComboBox` menu that will allow you to choose the font in which the `Text` item is displayed.

   See the GUI cheat sheet for the constructor and the method used to add choices to the `JComboBox` menu. Then add the `JComboBox` menu to the subpanel and tell it that your program will be the listener. The `JComboBox` requires an `ItemListener` so you need to extend the "implements" clause to also include `ItemListener`. (When you have a class that implements several interfaces, you simply separate the interfaces with commas. Do NOT include the keyword `implements` more than once in the class header!)

   You will also need to define an `itemStateChanged` method to react to menu selections made by the user.

   To change the font of an existing `Text` object, you can call the method

   ```
   setFont(String fontName)
   ```

5. Now add a `JCheckBox` component to allow the user to set the text created to bold. We didn't cover `JCheckBox`es in lecture, but they are very similar to `JButton`s: they require their listeners to implement the `ActionListener` interface, they have an `addActionListener` method, and they call the listener's `actionPerformed` method whenever the user checks/unchecks the box. To get the current state of a check box (i.e., whether or not it is currently checked), you invoke the accessor method

   ```
   boolean isSelected()
   ```

   on the `JCheckBox` object.

   Since both the button and the check box now invoke your `actionPerformed` method when they are selected, you will need to modify that method to distinguish the source of the event and respond accordingly. To do this, use the `getSource()` method on the event parameter to find out which GUI component the user interacted with. `getSource()` will return either the `JButton` object or the `JCheckBox` object. Depending on which value `getSource()` returns, you should either remove the last text from the canvas or change whether or not it is bold.

6. The last control we want you to add is the `JSlider` to control the font size. As shown on the first page of the handout, we also want you to place a `JLabel` component under the `JSlider` to display the current font size.

   Start by adding a `JSlider` to your main panel just as you added the `JTextField` and subpanel. Place it after the subpanel. Look at the GUI cheat sheet to determine what type of listener a `JSlider` needs.

   Define the appropriate listener method so that when the "thumb" in the scrollbar is moved, the size of the text changes. The possible font sizes should range from 10 to 48.

7. The `JLabel` beneath the `JSlider` should always display the current value represented by the `JSlider`. As shown on the first page of the handout, we want you to display this value preceded by the string "Current Font Size: ". This can be done as usual using the concatenation operator (+) in constructing the string to be inserted as the parameter of the `setText` method of the `JLabel`.

8. Finally, make sure that when the user clicks on the `canvas` the new `Text` item drawn has all the characteristics selected for font and font size.

Be careful that your program does not crash if you manipulate the controls after removing the last text item from the canvas or before adding the first text item to the canvas.

Extensions

As always, feel free to add any other features to your program. Here is one suggestion: Add the capability of changing the color of your text. There are two ways of doing this:

1. Add a `JComboBox` menu with the names of colors (e.g., black, red, blue, green, yellow). Make "`this`" a listener for that choice menu. Update your `itemStateChanged` method to change the color of the text when the user uses the color menu.
2. A more flexible way to control the text color would be to use a set of three sliders with values ranging from 0 to 255. You could do this by adding three more `JSlider`s to your display.

Submitting Your Work

Once you have saved your work in BlueJ, please perform the following steps to submit your assignment:

- First, return to the Finder. You can do this by clicking on the smiling Macintosh icon in your dock.
- From the "Go" menu at the top of the screen, select "Connect to Server...".
- For the server address, type in "afp://Guest@fuji" and click "Connect".
- A selection box should appear. Select "Courses" and click "Ok".
- You should now see a Finder window with a "cs134" folder. Open this folder.
- You should now see the drop-off folders for the three lab sections. Drag your "Lab5TextPlay" folder into the appropriate lab section folder. When you do this, the Mac will warn you that you will not be able to look at this folder. That is fine. Just click "OK".
- Log off of the computer before you leave.

You can submit your work up to 11 p.m. on Wednesday if you're in the Monday night lab; up to 6 p.m. on Thursday if you're in the Tuesday morning lab; and up to 11 p.m. on Thursday if you're in the Tuesday afternoon lab **although it is likely that you will complete this during your lab period**. If you submit and later discover that your submission was flawed, you can submit again. We will grade the latest submission made before your lab deadline. The Mac will not let you submit again unless you change the name of your folder slightly. It does this to prevent another student from accidentally overwriting one of your submissions. Just add something to the folder name (like the word "revised") and the re-submission will work fine.

Grading Guidelines

As always, we will evaluate your program for both style and correctness. Here are some specific items to keep in mind and focus on while writing your program:

Style

- Descriptive comments
- Good names

- Good use of constants
- Appropriate formatting
- Good use of private and local variables
- Good use of private methods

Correctness

- Displays appropriate text at click point (text should have currently selected font, size, and boldness)
- Changes the font (and modifies the size in the label, when appropriate)
- Removes the last text
- Does not generate null pointer errors