Computer Science Department

MHC Mount Holyoke College

CS 101 - Problem Solving and Structured Programming
Lab 6 - TextPlay
Due: November 13/14/15/16

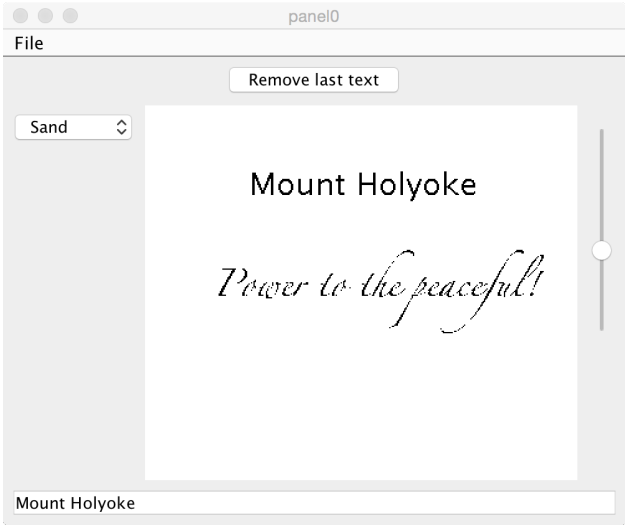
Goals:

- To work with Swing components
- To get more practice converting method signatures to method calls

Introduction to the Assignment

In this lab, you will gain experience writing programs with a modern graphical user interface. You will write a program that manipulates text through a JSlider, a pop-up menu (JComboBox), a JTextField, and a JButton. A picture of the screen can be seen below:

As usual, the center of the screen is the canvas. At the "South" side of the window is a JTextField where the user can write some text to be displayed on the canvas. At the "West" side of the window there is a JComboBox used to select a font. At the "North" side there is a JButton used to remove text from the screen. On the "East" side of the window is a JSlider used to control the font size.



When the user clicks anywhere on the canvas, your program should display the text showing in the JTextField in the selected font and font size at the place where the user clicked. When the user adjusts the JSlider or selects a new font with the JComboBox menu, 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 JButton is pressed, the last text placed on the canvas should be removed. (You can think of this as a one-level undo. Pressing the button a second or third time will not remove additional items.)

Design of the program

The design of this program is fairly straightforward. There is a single class called TextPlay that extends WindowController. The begin method should create the display, including the creation of the various GUI components.

The TextPlay class also needs to handle the mouse clicks on the canvas, using the standard onMouseClick method. It must also handle clicking on the JButton and moving the bubble in the JSlider.

As a quick reminder, remember the basic steps in displaying components:

1. Construct an instance of the component.
2. Initialize it if necessary (like adding items to a JComboBox).
3. Add the component to 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 addAction-Listener.
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.event.*;
import javax.swing.*;
import javax.swing.event.*;
```

should appear at the top of your Java file (in addition to the usual importing of objectdraw and java.awt). These lines inform Java that your program will need access to the Java libraries that support GUI components and events (including event listeners).

At the end of this document, there is a summary of the GUI component types you will use in this assignment.

Set the width of your window to 500 and its height to 420.

Step 1: Adding User's Text to the Canvas

First, place a JTextField in the south part of the window. To do this, construct the JTextField by calling the JTextField constructor, which has this signature:

```
public JTextField (int columns)
```

The columns parameter tells Java roughly how many characters should fit in the field when the user types. Next, construct a Panel and add the TextField to the JPanel. JPanel's constructor has this signature:

```
public JPanel ()
```

JPanel's add method is (use dot notation):

```
public void add (Component c)
```

Remember that a JTextField can be used as a Component so it can passed as a parameter to the add method whose signature is shown above. Then, add the JPanel to the window by calling WindowController's add method (this is different from the add method above):

```
public void add (Component c, int position)
```

The position may be any of BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST, or BorderLayout.WEST. Since we want the textfield in the south, you would use BorderLayout.-SOUTH in this case.

Run your program and check that the textfield shows up where you expect.

Now, 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. You will define the normal onMouseClick method to have this behavior. 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.

Run your program and test that it has all of this functionality.

Step 2: Removing the User's Text from the Canvas

Next, follow similar steps to add a JButton to the display so that it contains the words "Remove last text". Refer to the GUI cheatsheet at the end of this handout to find the constructors and methods that you need to create and display a JButton. Run your program to make sure that the button appears.

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 should be notified when the button is pressed. addActionListener has this signature:

```
public void addActionListener (ActionListener al)
```

Finally, add an actionPerformed method to your class that performs the appropriate action (removing the text from the canvas) when the button is pressed. The actionPerformed method that you declare must have this signature:

```
public void actionPerformed (ActionEvent e)
```

To remove the user's text from the canvas, you can call this method on the Text object:

```
public void removeFromCanvas ()
```

Make sure your program behaves reasonably if the user clicks the button before actually putting any text on the canvas. (Hint: Remember that a variable that has a class for its type will have a value of null before it has been assigned to. You can also use null on the right hand side of an assignment to allow a variable to "forget" the value it had.)

Step 3: Changing the Font of the Text

The next step is to create the JComboBox menu that will allow you to choose the font in which the Text item is displayed. Refer to the GUI cheat sheet at the end of this handout for the con-

structor and methods used with JComboBox. The demo program uses Courier, Sand, Times and Zapfino. Feel free to change the list of fonts to others available on the lab Macs.

If you do this on your Mac, you may get a pop-up window with a rather cryptic message when you compile: "TextPlay.java uses unchecked or unsafe operations. Recompile with -Xlint:unchecked for details." You do not need to be concerned about this. To keep this message from popping up each time you compile, you need to change your preferences as follows:

- Open the BlueJ menu and select Preferences...
- Select the Miscellaneous tab
- Uncheck "Show compiler warning when unsafe collections are used."

Add the JComboBox menu to the west side of the window and tell it that your program will be the listener. The JComboBox requires an ActionListener just like JButton does, so you do not need to add an "implements" clause. You do need to modify your actionPerformed method to react to menu selections made by the user. 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 JComboBox object. Depending on which value getSource() returns, you should either remove the last text from the canvas or change its font.

To change the font, you can call the method

```
public void setFont(String fontName)
```

of the Text class to change the font used by an existing Text object.

Step 4: Changing the Font Size

The last control we want you to add is the JSlider to control the font size. Start by adding a JSlider to the east side of the window.

Look at the GUI cheat sheet at the end of this handout to determine what type of listener a JSlider needs. (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.) Define the appropriate listener method so that when the bubble in the slider is moved, the size of the text changes. The possible font sizes should range from 10 to 48.

To change the font size, you can call the method

```
public void setFontSize (int size)
```

of the Text class to change the font size used by an existing Text object.

Step 5: Test with new Text

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. Avoid duplicating the code used to set the font and font size by creating a private method that you can call from several places.

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.

Extra credit: Nicer layout

Placing each GUI component in its own area of the display limits us to using only 4 components and also results in a less pleasing display. You can create a better display by placing multiple components in a single JPanel and then placing that JPanel in one of the 4 sections of the screen. For extra credit, you can experiment with JPanels and Layout Managers to get a more pleasing user interface, such as the one shown to the right.

Grading

10	Step 1: Adding user's text to canvas
10	Step 2: Removing user's text
10	Step 3: Changing the font
10	Step 4: Changing the font size
10	Step 5: Creating next text properly
10	Compiles without error
5	Comments
5	Variable names
4	Constant declarations
4	Indentation
5	If statements and while loops
5	Choosing between instance variables and local variables
5	Parameters
5	Good use of private methods
2	Folder Naming
100	**Total**
5	**Extra credit: better layout**

Turning in Your Work

Your work will be automatically collected from your dev/cs101/Lab6 folder at the time that it is due. Please be sure your files are in the right place.

Over the course of the semester, you may have up to 5 late days total. Use them wisely! To request a late day, fill out the Google form on the course website so that we do not collect your assignment when it is due.

Swing GUI Cheat Sheet

There are many more type of GUI components and many more methods on the types shown below. For more information, please refer to the online API available at http://java.sun.com/javase/6/docs/api/

General reminders

When the listener method is called, you can find out which component sent the event by calling getSource() on the event.

```
public void actionPerformed (ActionEvent event) {
    Object theButton = event.getSource();
    if (theButton == framedCircleButton) {
        // Create a framed circle
    }
}
```

If a method returns a String, remember to compare the result using the equals method, not ==:

```
aMenu.getSelectedItem ().equals ("A value")
```

GUI Components

To construct a font use the Font constructor:

```
public Font (String name, int style, int size)
```

Style can be one of Font.BOLD, Font.ITALIC, Font.PLAIN, or Font.BOLD + Font.ITALIC.

The specific components we have considered:

Adding components to the window

First, construct a JPanel and add the component to the JPanel:

```
public JPanel ( )
public void add (Component c)
```

Then add the JPanel to the window:

```
public void add (Component c, int position)
```

Pass the JPanel as the Component parameter above. The position may be any of BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST, or BorderLayout.WEST.

JButton

Constructor:

```
public JButton (String s)
```

Listener Interface:

```
ActionListener
```

Adding the listener:

```
public void addActionListener (ActionListener al)
```

Listening Method:

```
public void actionPerformed (ActionEvent e)
```

JComboBox

Constructor and Initialization:

```
public JComboBox ( )
public void addItem (Object item)
```

General Methods:

To find out which item was selected, use:

```
public Object getSelectedItem ( )
```

If you wish to treat the value returned from this method as a String, you can call the toString() method:

```
String text = menu.getSelectedItem().toString();
```

Listener Interface:

```
ActionListener
```

Adding the listener:

```
public void addActionListener (ActionListener al)
```

Listening Method:

```
public void actionPerformed (ActionEvent e)
```

JSlider

Constructor:

```
public JSlider (int orientation,
             int minimum, int maximum, int initialValue)
```

where orientation is one of JSlider.HORIZONTAL or JSlider.VERTICAL

General Methods:

To set the position of the bubble, use:

```
public void setValue (int newVal)
```

To find out the current value, use:

```
public int getValue ( )
```

Listener Interface:

```
ChangeListener
```

Adding the listener:

```
public void addChangeListener (ChangeListener al)
```

Listening Method:

```
public void stateChanged (ChangeEvent e)
```

JTextField

Constructor:

To create a field, use the following, passing as a parameter the number of characters the field should be able to hold:

```
public JTextField (int columns)
```

To create a field of a particular size, use:

```
public JTextField (int columns)
```

General Methods:

To find out the current value, use:

```
public String getText ( )
```

Information needed for extra credit only

The following information will be useful if you do the extra credit. You do not need to use the information described in the remainder of this handout for the required portion of the assignment.

JPanel

To use a layout manager other than FlowLayout with a JPanel, call:

```
public void setLayout (LayoutManager lm)
```

LayoutManager may be any of the layout managers listed below. When using FlowLayout, Grid-Layout or BoxLayout, use the following add method to add components:

```
public void add (Component c)
```

Layout Managers

BorderLayout (Default for WindowController)

Constructor:

```
public BorderLayout ()
```

FlowLayout (Default for JPanel)

Constructor:

```
public FlowLayout ()
```

GridLayout

Constructors:

```
public GridLayout (int rows, int cols)
public GridLayout (int rows, int cols, int colSpacing, int rowSpacing)
```

BoxLayout

Constructors:

```
public BoxLayout (Container target, int axis)
```

where axis can be either X_AXIS or Y_AXIS. target should be the same component whose layout you are changing, like:

```
somePanel.setLayout (new BoxLayout (somePanel, X_AXIS));
```