Tutorial 7

In this tutorial you will be looking at exercises related to the topic of graphical user interfaces through the use of Java Swing and AWT packages.

Problems

1. This problem focuses on layout managers. For this exercise you need to create a very simple media player interface, similar to that in Figure 1. Your application should make use of a BorderLayout Manager. The Songlist should be a JList, The volume control is an instance of VolumePanel. The complete code for the VolumePanel class can be found online. The navigation controls at the bottom is an instance of NavigationPanel. The provided code has a simple "Play" and "Stop" button. You need to re-write this code to obtain the pictorial navigational control shown below. Pictures are provided for you to use as button icons. To add an image icon to a button make use of the following code:

   new ImageIcon(getClass().getResource("path to the gif files"))

![Simple Media Player](attachment:image.png)

Figure 1: Simple Media Player

a. Add the missing code to the GUIProblem1 class to set the layout manager to BorderLayout, and to populate the window with the required components. Also add the songs to the songlist. (Add some of your favourite songs as well, if you want).

b. Rewrite the NavigationPanel class so that it contains all 5 buttons with pictorial icons rather than text labels like "Play" or "Stop". Note you need to set the backgrounds of the buttons to black if you want them to appear like the demo window.

c. Optional: Add vertical scroll bars to the song list.
2. The purpose of this exercise is to get some practice with writing action listeners for a simple button-based GUI. The demonstration program runs a Tic Tac Toe game. There is a model class, called \texttt{TicTacToeModel} that represents the game and rules. The model class is complete and will require no modification for the basic requirements.

There is also a \texttt{TicTacToeGUI} class that represents the window and interface. The model does not know anything about the interface. Also, the model uses a single array to store the game positions. The GUI, on the other hand, "thinks" in terms of rows and columns on the game board. There is code that translates between the model's "positions" and the GUI's "rows" and "columns".

The \texttt{TicTacToeGUI} class creates a frame with 9 buttons that are meant to represent the X's and O's game. Your task is to write the action listeners and update() method so that when the user clicks on a button it gets labelled with an "X" and the defending component (the game model), automatically adds an “O” in a defending position.

Specific Requirements and Suggestions

a. Add Action Listeners: Add action listeners to the buttons so that they can react when clicked.

b. Write the actionPerformed() method: This method should determine which button was clicked and then invoke the models' \texttt{XPlays(row,column)} method to make a move.

c. Write the update() method: Each button must be properly labelled based on the state of the game model. It should use the game model's \texttt{symbolForPosition(int row, int column)} method to obtain the correct label for the button. Use the JButton's \texttt{setText()} method to set the label of the button.

d. Show a win in colour (Optional): Modify the code so that the buttons that form a winning row, column, or diagonal appear with a different background colour.

3. Create a simple Slide Show application. This should consist of a window that has a \texttt{JPanel} which uses a \texttt{CardLayout} to represent the slides (one at a time) and two arrow buttons to navigate across the slides forwards and backwards. Note the following:

a. for the buttons use the standard \texttt{BasicArrowButton} objects that are available in Java in the \texttt{javax.swing.plaf.basic} package.

b. the \texttt{JPanel} and \texttt{CardLayout} are made into instance variables so that it is possible to access them from the event handlers

c. the main window is set to use a \texttt{FlowLayout}, the default was \texttt{BorderLayout}.

d. use the \texttt{BorderFactory.createLineBorder()} to make a nice black border around our panel.
4. Create a recipe catalogue similar to the one depicted in Figure 2 below. The basic components must include two `JLists`, one representing the categories of the recipes and another representing the recipe names. The other two components are two text areas, which can be used to enter text related to the `ingredients` and to the `directions` respectively. All components can have associated scroll-bars. Make correct use of the `GridBagLayout` and correctly set the dimensions of each component such that resizing has no effect on the overall design.

![Figure 2: Recipe Catalogue](image)

5. Write a class that implements a calculator. Functionality to include:
   a. basic mathematical functions
   b. decimal point
   c. equals: when pressed generates a result
   d. cancel: when pressed clears all the input

Some issues:
- how to handle operator precedence. For example, how will the result of \(4 + 3 \times 2\) be computed? Suggestion: it is easier to treat all operators as having equal precedence and process them from left to right.
- how to handle the entering of multiple decimal points. Typically the program can accept the first decimal point and ignores the rest.
- when the display is 0 and the user enters 0, the display should not change. However if the display is nonzero and the user enters 0 then 0 is appended to the current display.

6. **Extend** the calculator class to allow the user to enter a number using the keyboard. The class needs to implement the `KeyListener` interface and define the `keyTyped()` method.