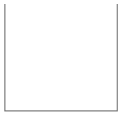
Swing & Java2D Exercise Book

by Shai Almog
vPrise Consulting
שבט תשס"ד
Last Revision March 6, 2004



1 Introduction

2 GUI in Java

3 Getting Started with Swing

4 Java2D and Rendering

5 Creating a File Browser

5.1 Add a column to the table indicating the date the file was last modified

5.1.1 Make use of the java.io.File.lastModified() method. It returns a long value that can be used to create a java.util.Date object.

5.1.2 Extra credit if the date is also rendered as DD/MM/YYYY

5.2 Center the table icon representing the file type.

5.2.1 Use the setHorizontalAlignment() method.

5.3 Hide hidden files.

5.3.1 To detect whether a file is hidden use the File.isHidden() method.

5.4 Color table rows with files larger than 1mb in red.

5.5 Make the tree node selection color (the highlighted background color) yellow.

5.6 Make sure the tables selected row will be painted in pink.

5.7 Add a command line switch that allows the user to place the tree to the right of the table.

5.8 Add a column indicating whether a file is hidden or not.

5.9 Add a column that will feature a check box widget.

5.9.1 The column doesn't have to be editable.

5.9.2 The content of the column can be always checked/unchecked.

5.10 Display files as well as directories within the tree.

5.10.1 Ignore tree selections in which a file is selected.

5.11 Add a column displaying the percentage of the size of the file relatively to the rest of the directory.

5.11.1 If a directory has 3 files sized 2kb, 2kb and 4kb. The percentage column will show: 25, 25, 50.

5.11.2 It is not essential to create a renderer so the values will appear with a % sign.

5.12 Make every odd row background yellow, and every even row light blue. Make sure that selection still functions as expected.

5.13 Have the model change a directory based on the user double clicking on a row in the table.

5.13.1 Add a mouse listener to the table.

5.13.2 To resolve the row matching the appropriate point use the method JTable.rowAtPoint().

5.14 Add a row to the table containing the file: "..". Double clicking on this row (which should always be row 0) should result in going to the parent directory.

5.14.1 Add a mouse listener to the table.

5.14.2 To resolve the row matching the appropriate point use the method JTable.rowAtPoint().

5.15 Make the GUI 50% larger.

5.15.1 Increase the font size.

5.15.2 Increasing the image size is desirable but not essential.

6 Actions, Menus, Buttons and Tool bars

7 Containers

8 Layout managers

8.1 Build an "Icon Layout" that will layout elements in a similar way to the "icon view" in windows explorer.

8.2 Build a top-down version of the flow layout.

8.3 Build a circular flow layout.

8.3.1 It will organize the components in a circle in a similar way to the flow layout.

8.3.2 The radius of the circle may be predetermined for simplicity.

8.4 Build a "folding layout" it will act like a flow layout, however when a flag is set it will place all the components at 0,0 allowing the components to "fold".

8.4.1 Demonstrate this by placing several buttons and when clicking on of them the container should toggle the layout state and revalidate.

8.4.2 The best solution would be to derive from flow layout.

8.5 Use spring layout to build an attractive user registration form with the following attributes: first name, surname, gender, date of birth.

8.5.1 No actual functionality is necessary, only the layout

8.6 Use grid bag layout to build an attractive user registration form with the following attributes: first name, surname, gender, date of birth.

8.6.1 No actual functionality is necessary, only the layout

8.7 Use nested box layouts to build an attractive user registration form with the following attributes: first name, surname, gender, date of birth.

8.7.1 No actual functionality is necessary, only the layout

8.8 Make a "BiDi aware" version of flow layout.

8.9 Layout components in a browser like user interface: view area (text pane), address bar (text field), search button (to the right of the address bar), toolbar & menu bar.

8.9.1 No actual functionality is necessary, only the layout

8.9.2 Extra credit for a status bar at the bottom

8.10 Make a BiDi aware version of border layout that will replace east & west when in right to left mode.

8.11 Adapt the natural grid layout so components placed within it are centered in their cell rather than left/right aligned.

8.12 Adapt the natural grid layout so it will take into consideration the insets of components within it.

8.13 Add a "vertical span" feature to the natural grid layout allowing a cell to span vertically.

8.14 Add a "horizontal span" feature to the natural grid layout allowing a cell to span horizontally.

8.15 Add a "zoom" feature to the natural grid layout that will treat all sizes as x% larger than they really are.

8.15.1 This can be accomplished by sizing everything to larger values according to the zoom factor.

9 Java Beans

10 Keyboard

11 Advanced Swing

11.1 Design a utility that downloads the content of http://www.yahoo.com/ in an offline thread and updates a JLabel with this content.

11.1.1 The update operation should occur on the Swing thread, while the download operation should occur on a separate thread.

11.1.2 This can be accomplished using any of the methods for thread control.

11.2 Create a tool that spawns 100 threads, each of which generate 100 random numbers and place them into a text area.

11.2.1 The update operation should occur on the Swing thread, while random number generation should occur on a separate thread.

11.2.2 This can be accomplished using any of the methods for thread control.

11.3 Create a tool that spawns a thread that counts from 0 to 1,000,000 and then back down to 0 repeatedly. The tool should update the GUI on its current status.

11.3.1 The counting thread is in an infinite loop.

11.3.2 The update operation should occur on the Swing thread, while the counting operation should occur on a separate thread.

11.3.3 This can be accomplished using any of the methods for thread control.

11.4 Create a proxy to the ListModel interface that will allow it to have a "title" object on top. Demonstrate this functionality by having a list containing the elements: "Red", "Green", "Blue" and the proxy will add the word "Colors" into the list before them thus producing: "Colors", "Red", "Green", "Blue".

11.4.1 A proxy interface and an abstract proxy class are desired but not essential for this exercise.

11.5 Create a proxy for the tree model that hides all of the folders matching the name "hidden". Use the file system tree model to demonstrate how a directory named "hidden" is indeed hidden.

11.5.1 A proxy interface and an abstract proxy class are desired but not essential for this exercise.

11.6 Create a ComboBoxModel proxy that allows us to place a separator in a specific location. So if we have a combo box containing: "Red", "Green", "Blue", "Fucia", "Caramel", "Nut" and we would want to create a: new SeparatorComboProxy(model, 3) which would result in: "Red", "Green", "Blue", "-------", "Fucia", "Caramel", "Nut". The combo model should disallow the selection of the separator row.

11.6.1 Disallowing selection is easy in the combo box model since it contains support for selection. This can be disabled by simply doing nothing.

11.6.2 A proxy interface and an abstract proxy class are desired but not essential for this exercise.

11.7 Create a reflection proxy that traces all the calls to the underlying model and logs them using System.out. Apply this proxy to at least two different models.

11.8 Create a proxy that converts a tree model to a table model. It should receive a node within the tree and display the content of that node within a single column in a table model. Demonstrate this functionality by using the file system tree model to create a simple file browser without a "real" table model.

11.9 Create a proxy that converts a TableModel and a ListSelectionModel into a ComboBoxModel. The value of the list entry should be mapped to a value of one of the columns in the table model. Changes in the combo selection should update the table. Demonstrate this functionality on the table model.

11.9.1 You can access the list selection model of a JTable via a getter method in JTable.

11.9.2 The selection in a combo box is within the combo box model while the selection in the table maps to a list selection model.

11.10 Create a template that orders elements in a form. Nest it together with a template that places elements in the bottom of the screen. Create a demo in which a user can fill out a form and select ok/cancel operations when the form is completed.

11.11 Create a proxy table model that validates input. Invalid input should be rejected.

11.11.1 All input is sent to the table via the setValueAt, method.

11.12 Adapt the Swing worker class to make use of a thread pool rather than spawn a different thread every time.

11.12.1 Changes to the semantics of the swing worker class are valid.

11.13 Layer proxies on the table, for performing a simple operation (such as logging). Try to layer as much proxies as possible and inspect performance.

11.13.1 Disable the actual functionality of the proxy in order to determine whether this is the cause of performance degradation. See how many proxies can be created.

11.14 Create a table model containing random data with 50,000,000 rows and 20 columns. Apply layers of proxies and verify that its performance is still similar to the performance of a 200 row table model.

11.15 Adapt the Swing worker class to make use of the new java concurrency package and simplify its implementation.

12 Eye Candy

13 Look & Feel

14 Tables

15 Table Rendering

15.1

16 Text Components

17 i18n

17.1

18 Performance

19 Deployment

20 Desktop Integration

21 Transferable

22 Undo