Project Report

Ahmet Kutluğ Alperen Kars

Comp132, Section 01

Öznur Özkasap

May 18, 2022

**Overview**

Having a poor aesthetic quality, this project claims no more than being functional. In general overview, both the interface and the methods are easy to understand and get used. The project has 3 unvisual, 10 visual classes. To begin with the unvisual classes,

**App.java**

*App* class is mainly responsible for storing the data through 4 databases, all of which are static as this class is a general database of all the project:

**groups** keeps the track of *GroupData* objects

**groupsnd** keeps the track of *Group* objects

**users** keeps the track of *User* objects

**database** keeps the track of username-password pairs that belong to *User* objects

**GroupData.java**

*GroupData* class is a backend information storage for visual Group objects. Fields of this class contain images that are posted in the group, members of the group and further relevant information about it.

**User.java**

*User* class is the fundamental component of the project. Its fields contain all the personal information as stated in the instructions as well as *Content* databases for storing the content paths, contents, and their captions. It has several search methods used for finding a specified GroupData, Group and Content, consecutively. Basic group and user-related operations (joining a group, following a user) are also placed in this class, but not visually.

**VISUAL CLASSES**

**Content.java**

*Content* class is the minimal component of the visual part of the project. It is a subclass of *JPanel,* and has several informational fields (creator, title, time, image, text) as well as GUI fields. When necessary, all the *Content* object is placed within any panel-frame. While creating Content.java, GroupLayout was maybe the best option since it includes numerous unrelated panels and areas, thus selected. It includes two panels, in one of which the image is placed and the text in the other. In the right-hand side panel, there is also a pink button visible only to the *Content*’s creator and allows her/him to edit the content when clicked. *Content* objects are created by created by *filechooser* objects.

**filechooser.java**

This is a subclass of JFrame class, implements ActionListener, contains 3 constructors and 1 method, and each of them are used for different purposes.

filechooser(User user) this constructor is useful when a *Content* object will be shared in the homepage. *filechooser* object created by calling this constructor would have 2 buttons “open”, “save”, 1 textField and 1 textArea. When “open” is clicked, a JFileChooser object is created through the concrete method *actionPerformed* and user is enabled to choose the content to be shared. After selection, user can either add optional caption by writing it into the textField or share the content by clicking on “save”.

filechooser(String k,User user) this constructor is designed for editing posts. Its general implementation is similar to the previous one, just with the minor difference of “Back” button, which enables user to go back without changing anything. The current content’s components are updated with the newly given ones.

filechooser(User user, String groupName) this constructor is for in-group activities. It is called when a user wants to share a content in a group. It has the same implementation with the first constructor.

**boş.java**

This is the main page for all users, unique to each of them. It is a subclass of JFrame and has its owner *user*. BorderLayout was suitable for this class since it provides more organized view. Placed within the frame, there are 2 panels: One of them is used for placing the *Content* objects and the other is for buttons. At the beginning this class, it lists its contents by accessing the following list and contents of its owner through loops. Later it implements the necessary buttons to be placed at the right-hand side of the page. Among this buttons, “Create New Content” creates a *filechooser* object when clicked, which allows user to share content. Other groups are designed for different purposes such as logging out, searching groups-users, accessing owner’s profile and creating a group for Premium members. At the end there are two methods which functions conversely; one of which is hiding the contents by making panels invisible whereas the other does the vice versa.

**prf.java**

This is basically the profile of the users, subclass of JFrame and works with double variable as Visitor and User. At the very beginning, this class implements a method called *panelReset* which is a shortcut for updating the panels when the components are changed. Later, it implements its visual components such as the JLabel for a profile photo, textAreas for listing the Groups and Friends, textArea for displaying the personal information and buttons for following, unfollowing and returning back. One of these buttons has multiple functions, it can be either “Edit Profile” button or “Follow” button according to the visitor’s status. Buttons’ actions are implemented accordingly and previously implemented methods were used for following-unfollowing operations. At the end, Groups and Friends lists are filled with respect to the Groups and Friends that the profile owner has and contents of her/him are displayed in the Panel below.

**profileEdit.java**

This class is also a subclass of JFrame and it has numerous fields, most of which being JTextField. When created, it basically requests User to either fill the personal information parts accordingly or Go Back without saving by clicking the “Go Back” button. When there is a blank line left after clicking on the Save button, an exception message is displayed. If all the processes were successful, provided information are passed to the methods and user’s personal information is updated.

**Group.java**

*Group* class has only one constructor which takes GroupData and visitor as parameters and creates a JFrame object of Group. This object can only be created by the Premium users through clicking the button “Create a Group” in their main page and entering the necessary questions (Group Name, Country Name, Hobbies). JFrame implementation is considerably similar to the *boş* class, only with the differences in the buttons’ names and functions. These buttons’ purposes are to show the group members, remove them if visitor is the creator of the group and go back to the main page, as stated in the instructions. After modifying the buttons according to the membership status of the visitor, there are several basic methods that aim to reset panels, add posts, make them visible and adjust the membership status of the visitor.

**searchTool.java**

*searchTool* is used for searching groups and users. It is a subclass of JFrame and created through the main page (*boş*), by clicking the buttons starting with “Search” keyword at the right-hand side. Just as in the *filechooser*, searchTool has also the method actionPerformed as it implements the interface ActionListener, and this method is shared across the two different constructors. In general, it has the same implementation with *filechooser.*

**Login.java**

This page is the first JFrame that the user sees when she/he starts the application. It has several textFields mostly used for receiving the input and passing them to the methods. Since this class does not contain numerous panels or other fields, it was appropriate to set BorderLayout for this class, and proceed accordingly. JTextFields placed within this class takes the input from the user and if they match with their counterparts in the database, a new *boş* page is created for the user that has matching information and displayed after clicking “Sign in”. If the user does not have an account or wants to have another one, after she/he clicks “Sign up”, a new *signUp* page would be created.

**signUp.java**

Having almost the same implementation with *profileEdit.java*, *signUp* is used to create new users instead of editing the information of existing ones. Through the plenty of JTextFields that it contains, it takes all the necessary input from the user including the account type by a comboBox and processes them to create new account if the given information is valid. Having plenty of fields and the hardships of placing them one by one, FlowLayout was suitable for this class, therefore was set.

**Test.java**

This is the test class in which there are several users created manually and a login page, start point of application.

References

[java - Add scroll into text area - Stack Overflow](https://stackoverflow.com/questions/10177183/add-scroll-into-text-area)

[ScrollPaneLayout - Java Swing - Example - StackHowTo](https://stackhowto.com/scrollpanelayout-java-swing-example/)

[Java AWT | GridBagLayout Class - GeeksforGeeks](https://www.geeksforgeeks.org/java-awt-gridbaglayout-class/)

[Format Specifiers in Java - GeeksforGeeks](https://www.geeksforgeeks.org/format-specifiers-in-java/)

[A Visual Guide to Layout Managers (The Java™ Tutorials > Creating a GUI With Swing > Laying Out Components Within a Container) (oracle.com)](https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html)

[Trail: Creating a GUI With Swing (The Java™ Tutorials) (oracle.com)](https://docs.oracle.com/javase/tutorial/uiswing/index.html)

[Java JScrollPane - javatpoint](https://www.javatpoint.com/java-jscrollpane)

[JScrollPane (Java Platform SE 7 ) (oracle.com)](https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html)

[JFileChooser (Java Platform SE 7 ) (oracle.com)](https://docs.oracle.com/javase/7/docs/api/javax/swing/JFileChooser.html)

[BufferedImage (Java Platform SE 7 ) (oracle.com)](https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html)

[Java BufferedImage Class (tutorialspoint.com)](https://www.tutorialspoint.com/java_dip/java_buffered_image.htm)

[How to Use Combo Boxes (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components) (oracle.com)](https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html)

[How to Write an Action Listener (The Java™ Tutorials > Creating a GUI With Swing > Writing Event Listeners) (oracle.com)](https://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html)

[Introduction to Event Listeners (The Java™ Tutorials > Creating a GUI With Swing > Writing Event Listeners) (oracle.com)](https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html)

[java - How to add an image to a JPanel? - Stack Overflow](https://stackoverflow.com/questions/299495/how-to-add-an-image-to-a-jpanel)

[Java Layout Manager - javatpoint](https://www.javatpoint.com/java-layout-manager)

[Java AWT | BorderLayout Class - GeeksforGeeks](https://www.geeksforgeeks.org/java-awt-borderlayout-class/)

[How to Make Dialogs (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components) (oracle.com)](https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html)

[Arrays in Java - GeeksforGeeks](https://www.geeksforgeeks.org/arrays-in-java/)

[Java - Exceptions (tutorialspoint.com)](https://www.tutorialspoint.com/java/java_exceptions.htm)

[Exception (Java Platform SE 7 ) (oracle.com)](https://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html)