**HashMaps:**

Text

Description automatically generated

Figure 1: getter and setter methods used in ID and Passwords to obtain the different HashMap containing logins for teachers and students

Using HashMaps to store ID and Passwords (both Strings) or a Key and a Value. Using .put() to put elements into the HashMap. Can easily match the username and passwords in a log in page by calling out using the .get() or .containsKey to check for correct password to username or if a username/password even exists at all.

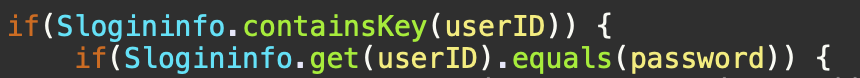


Figure 2: LoginPage using .containsKey to see if username exists and then checks to see if the password is correct

**Arrays:**

Stored the different characteristics of contacts and rooms with different arrays, each of different type accordingly. All set to have size 20 as the maximum number of rooms and contacts is 20 according to Client. Also passed in different default values. (See Appendix).

These are constantly being changed/monitored no matter which page we are moving to but initially set they are written in the Main File as below.

Text

Description automatically generated

Figure 3: Default Arrays for each characteristic of contacts and rooms

When moved to each page, these will get copied to another instance of the same Arrays (with different names) so the entire class can access the different arrays.

Text

Description automatically generated

Figure 4: example of how instances of each array are created every time when passing between different Pages/Java Files

**GUI and ActionListeners:**

The GUI class produced all of the GUIs and the events are allowed to occur due to implementing ActionListener to the student and teacher pages different components.

Creating all the GUI Components including JFrame, JPanel, JLabel, JButton, JTextArea, JComboBox, JButtonGroup, etc.

Text

Description automatically generated

Used

.setBounds(): set up the boundaries for placing components on a GUI.

.setColor() to set the color.

.setText() to set the text

.setFont() to set font of the text

.setVisible() to set so components can be seen or not

.setLocationRelativeTo(null) so then it does not have to go according to the relative location on

one’s screen

Text

Description automatically generated with medium confidence

Figure 5: Default lines of code I make to create the frames/each Page

I would set the basic components of everything and use .addActionListener() after each Button or component that would start it’s “own method” or job. For example, using it after a submit Button so that we can use the actionPerformed method from the ActionListener() class to define what happens after that button is clicked.

Text

Description automatically generated

Figure 6: example of setting up the basic boundaries and defaults of each component on frame

**Images and Files:**

Used to upload images and files as well as edit them. In this example, after clicking the upload Button, teachers would be able to select a file from their desktop to use as the room’s image.

Text

Description automatically generated

Figure 7: allowing users to choose and upload an image from their own computer to use as this room's image

If(e.getSource() == \_\_\_ ) takes the action/event that occurred and checks if the action was on the button, Using the fileChooser().showOpenDialog to select file from the computer or desktop to open and then .APPROVE\_OPTION. Then I would get the absolute path, or the directory of the image and sing the .getselectedFile() and .getAbsolutePath(). .getselectedFile() checks for the file and .getAbsolutePath() will return the path of the file. We would change this to .getImage() of the Image Class to get the image and then resize the image uusing .getScaledInstance() and finally set the image on top of a label.

.getSelectedIndex() was used a lot of the ComboBoxes to get the selected item on the list to apply different methods on the contacts or rooms. To refresh the ComboBoxes’s display I would use .setModel(new DefaultComboBoxModel(array)) to set to the newest most updated array. Basically using a array of strings as a model to display choices on the combo box.

**Bubble Sort:**

I used bubble sort in multiple areas of the text as the parameters/size of the arrays were all really short as the maximum is 20 items. It was also because it is easy to use and there were other characteristics that has to be moved other than just one characteristics. For example, when trying to sort the contacts by grades, you cannot just change/sort the grades of the contact you have to change/sort the contact’s name, availability, etc. With bubble sort I can sort them all at once.

Graphical user interface, text, chat or text message

Description automatically generated

**Deleting Items in an Array:**

I would use a while loop to copy the next item in the array to the current item until there are no more items in the array. If the array is not full, there would be items that are null or also 0 which would let the loop stop.

Text

Description automatically generated

Figure 8:Deleting an item from multiple arrays. This example is of deleting a entire contact

**Changing what texts show and radio buttons are select based on the selected item on a list**

since I structed all the characteristics of contacts and rooms in a way so that the index of the selected item corresponds with the index of all the different characteristics of the room (ie the room name, room images, room descriptions) I would just use .getSelectedIndex() to get the selected item’s index from the combo box and use the index to display texts, images, and buttons from the corresponding arrays. Using setText() or setIcon() and such as shown below

A screenshot of a computer

Description automatically generated with medium confidence

Figure 9: setting the preview scene to show the correect descriptions, room name, image, numbers, etc

**Changing Route Instructions based off Next Button:**

I first created a method to be called on every time I needed to change the image and description to something else. It would pass in the index of the next image or description that needed to be displayed and this index (int) will be used to display the images on the corresponding panels and text fields as shown.

Text

Description automatically generated

Text

Description automatically generated

When the go button is clicked, it will generate a maximum value of “slides” that the next button can be clicked until the images are supposed to rotate back to the beginning again. Based on the size of the start and end room number, the displays will move in different orders.

Text

Description automatically generated

Figure : if the starting room is a smaller room then the next image will have to be in the larger index of the array that stores the next room image. vice versa which is why we have to see if we are starting from a larger room or a smaller room.