CS3330 Final Project Document
Requirements

Devin Cargill
Dc23b
18132100

1. Object oriented elements that you write the code for:
   a. Classes: You can find an example of a class in the HomeController.java file on 38.

```
38  public class HomeController extends Switchable implements Initializable {
39      private zipModel zipper;
40      private Stage stage;
41      private File source;
42      private File destination;
43
44      @FXML
45      private Label sourceLabel;
46      @FXML
47      private Label destinationLabel;
48      @FXML
49      private TextArea message;
50      @FXML
51      private ProgressBar progressBar;
```

   b. Subclasses: You can see an example of a subclass in the same screenshot above. HomeController extends the Switchable class thereby making it a subclass.
   c. Abstract Class: You can see an example of an abstract class in the Switchable.java file on 18.

```
17  */
    public abstract class Switchable
19  {
20      public static Scene scene;
21      public Parent root;
```

   d. Interface: There are 2 examples of interfaces used in this program. The first one is with Notification which is a functional interface in the Notification.java file on line 13. The second interface I used can be found in userAuth.java on line 11.

```
11  */
    public interface userAuth {
        String appUser = "devin";
        String appPass = "1234";
        String userError = "Username incorrect";
        String passError = "Password incorrect";
        String hint = "User: devin || Pass: 1234";
        int hintAttempts = 3;
19
        public int compareUser(String input);
        public int comparePass(String input);
        public void messageHint();
23  }
24
```

2. Code elements that utilize:
   a. One or more collection classes: You can see an example of a collection class in the Switchable.java file on line 22.

```
21      public Parent root;
        public static final HashMap<String, Switchable> controllers = new HashMap<>();
23
```

b. Exception Handling: You can see examples of exception handling throughout my entire program. A specific example is in the HomeController.java file on line 69. It takes the exception and sends it to the displayExceptionAlert function on line 191 to display.

```
65            if(file != null){
66                try {
67                    writer = new FileWriter(file);
68                    writer.write(message.getText());
69                } catch (IOException ex) {
70                    displayExceptionAlert(ex);
```

**** Same File ****

```
191     private void displayExceptionAlert(Exception ex){
192         Alert alert = new Alert(Alert.AlertType.ERROR);
193         alert.setTitle("Exception Dialog");
194         //alert.setHeaderText("Exception!");
195         alert.setHeaderText(ex.getClass().getCanonicalName());
196         alert.setContentText(ex.getMessage());
197
198         StringWriter sw = new StringWriter();
```

3. The application must have a clearly defined model (as in the M in MVC): You can see an example of a model in my zipModel.java file. It takes in variables from the controller and does the legwork in zipping your directory.

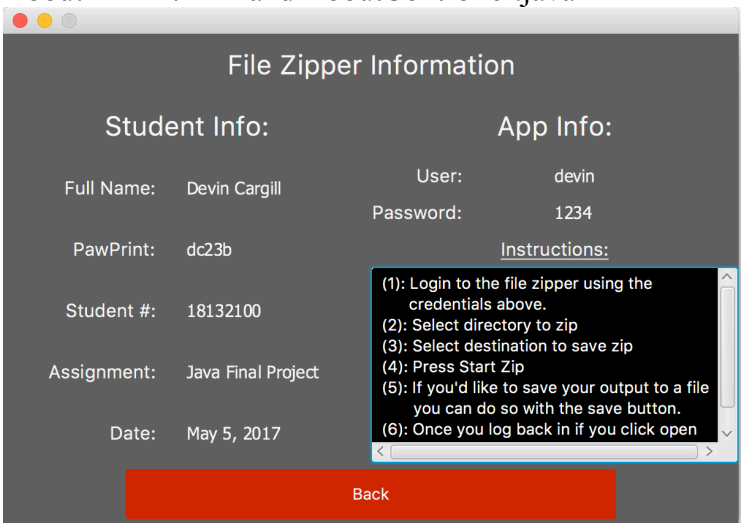
```
26  public class zipModel extends Thread {
27      private File sourceDirectory;
28      private File destinationDirectory;
        private File[] files;
30      public Boolean stop = false;
31      private Notification notification;
32      static final int BUFFER = 2048;
33
34      public zipModel(File sourceDirectory, File destinationDirectory) {
35          this.sourceDirectory = sourceDirectory;
36          this.destinationDirectory = destinationDirectory;
            files = sourceDirectory.listFiles(new FilenameFilter(){
38              @Override
                public boolean accept(File directory, String filename){
40                  return true;
41              }
42          });
43      }
44
45      public int getNumFiles() {
46          if (files == null) {
```

4. The UI must utilize multiple scenes and at least one of the scenes will have the contents of the scene graph changed based on the application state: You can find an example of switching through multiple scenes when you login to my app or click on the View App Info button. When you try to login it will display different content depending on if you have the username or password wrong. You can find this example in the HubController.java file starting on line 73 and ending on 95.

```
73      @FXML
74      private void login_validate(ActionEvent event) {
75          String inputUser = user.getText();
76          String inputPass = pass.getText();
77
78          //bug testing
79          System.out.println(logonAttempts);
80          // bug testing
81
82          if(compareUser(inputUser) == 1){
83              if(comparePass(inputPass) == 1){
84                  message.setText("logged in");
85                  Switchable.switchTo("homeFXML");
86              } else {
87                  message.setText(userAuth.passError);
88              }
89          } else {
90              message.setText(userAuth.userError);
91          }
92
93          messageHint();
94
95      }// end login_validate
```

Note: (1), (2) and (3) at bottom show how the same scene changes based on the state

5. There must be a way to access "About" information that includes information about you and the application: There is an about page included in my app. It's file structure is AboutFXML.fxml and AboutController.java



6. The application must save data and load data. The target for saving/loading data can be files, a network service, and/or a database: You can see an example of this in the HomeController.java file from line 58 to 132. This functionality exists when you login.

```java
58      @FXML
59      private void handleSave(Event event){
60          FileChooser fileChooser = new FileChooser();
61          Stage stage = (Stage) root.getScene().getWindow();
62          File file = fileChooser.showSaveDialog(stage);
63          FileWriter writer = null;
64
65          if(file != null){
66              try {
67                  writer = new FileWriter(file);
68                  writer.write(message.getText());
69              } catch (IOException ex) {
70                  displayExceptionAlert(ex);
71              } catch (Exception ex){
72                  displayExceptionAlert(ex);
73              } finally {
74                  if(writer != null){
75                      try {
76                          writer.close();
77                      } catch (IOException ex) {
78                          displayExceptionAlert(ex);
79                      } catch (Exception ex){
80                          displayExceptionAlert(ex);
81                      }
82                  }
83              }
84          }
85      }
```

```java
87      @FXML
88      private void handleOpen(Event event){
89          FileChooser fileChooser = new FileChooser();
91          Stage stage = (Stage) root.getScene().getWindow();
92          fileChooser.getExtensionFilters().add(
93                  new FileChooser.ExtensionFilter("Text files", "*.txt")
94          );
95          fileChooser.showOpenDialog(stage);
96          File file = fileChooser.showOpenDialog(stage);
97          if(file != null){
98              BufferedReader bufferedReader = null;
99              try {
100                 bufferedReader = new BufferedReader(new FileReader(file));
102                 String document = "";
103                 String line = "";
104                 while((line = bufferedReader.readLine()) != null){
105                     document += line + "\n";
106                 }
107                 message.setText(document);
108             } catch (FileNotFoundException ex) {
109                 displayExceptionAlert(ex);
110             } catch (IOException ex) {
111                 displayExceptionAlert(ex);
112             } finally {
113                 if(bufferedReader != null){
114                     try {
115                         bufferedReader.close();
116                     } catch (IOException ex) {
117                         displayExceptionAlert(ex);
118                     }
119                 }
120             }
        }
    }
```

Additional Notes: To login and test use the following:

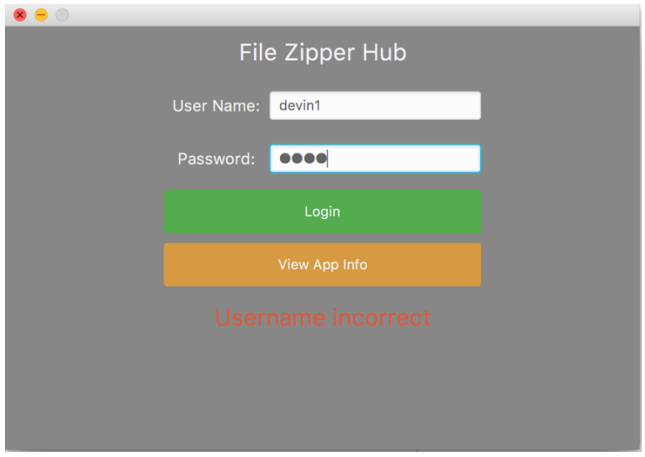
User Name: devin

Password: 1234

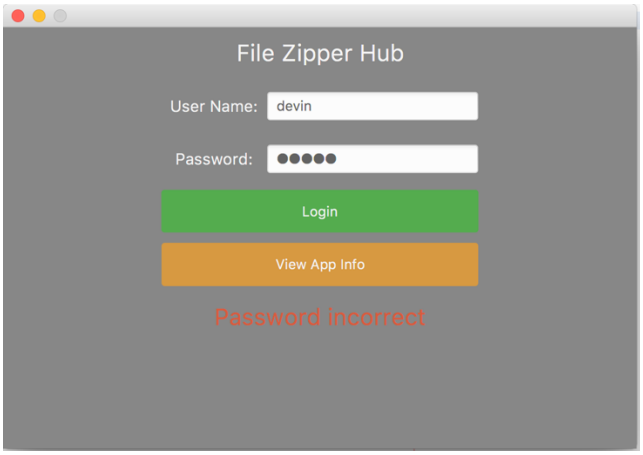
(this information exists on the about page for your reference)

Additional Screenshots:

(1) Bad Username State:



(2) Bad Password State:



(3) Got them wrong more than 3 times state: