**Centennial College**

**COMP 228: Java Programming**

**LAB #4 – Developing UI using JavaFx**

**Studen**t: **Hisham Abu Sanimeh**

Due Date: **To be announced.**

Purpose: The purpose of this Lab assignment is to:

1. Practice the use of JavaFx GUI event handling

References: Learning materials for weeks 7 and 8, textbook, and other references (if any)

Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students.

YOU NEED TO SUBMIT THE FOLLOWING 2 DOCUMENTS IN THE DROPBOX TITLED LAB4:

1. THE FIRST ONE IS A WORD DOCUMENT. USE THIS DOCUMENT AND ADD SCREEN SHOTS OF THE RUNNING STATE OF EACH EXERCISE (If there are more than 1 exercise). DO NOT DELETE THE QUESTIONS. THE SCREEN SHOTS SHOULD FOLLOW EACH QUESTION AND COVER ALL THE ASPECTS/FUNCTIONALITIES OF EACH EXERCISE. AFTER THE SCREEN SHOTS PLEASE COPY THE CODE FROM THE CODE WINDOW AND PASTE THE COMPLETE CODE. DO NOT GIVE ME SCREEN SHOTS OF THE CODE. DO NOT ZIP THIS FILE AND KEEP IT SEPARATE FROM YOUR ZIPPED PROGAM FILE.

2. SUBMIT ALSO ONE ZIPPED PROJECT FILE THAT CONTAINS ALL THE EXERISES SEPARATELY INTO THE SAME DROP BOX.

This material provides the necessary information you need to complete the exercises.

You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228Labnumber**

Example: **JohSmith\_COMP228Lab4**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228Labnumber.zip**

Example: **JohSmith\_COMP228Lab4.zip**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

## Exercise 1

Write a Java application using JavaFx which allows the user to enter student information (see the sample design below).

#### 

#### The user will enter *full* *name*, *address,* *city, province, postal code, phone number and email* in *text field* controls. The student’s *major* (Computer Science or Business) will be selected from two *radio buttons*.

#### A combo box will display the list of *courses* for each program whenever the user selects the desired program.

#### A course will be added to a *list box w*henever the user selects a course from the corresponding combo box. Make sure that the user cannot add a course several times.

Additional information about the student will be provided from a group of *check boxes* (such as involvement in various activities, etc).

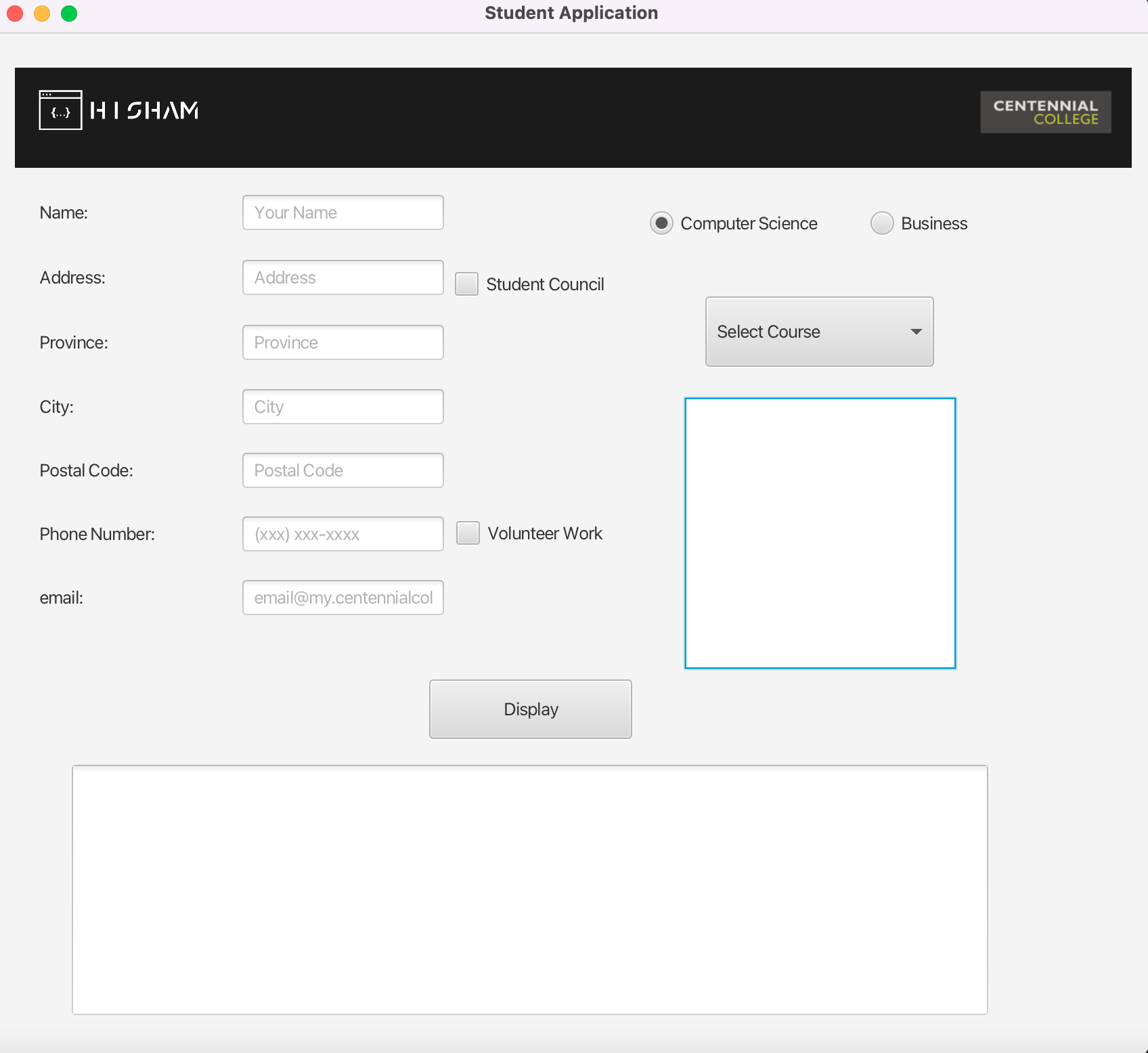
All the information about the student will be displayed in a text *area* component. Use simple JavaFX layout managers, such as FlowPane, BorderPane, and GridPane to create the JavaFX GUI

of this application.

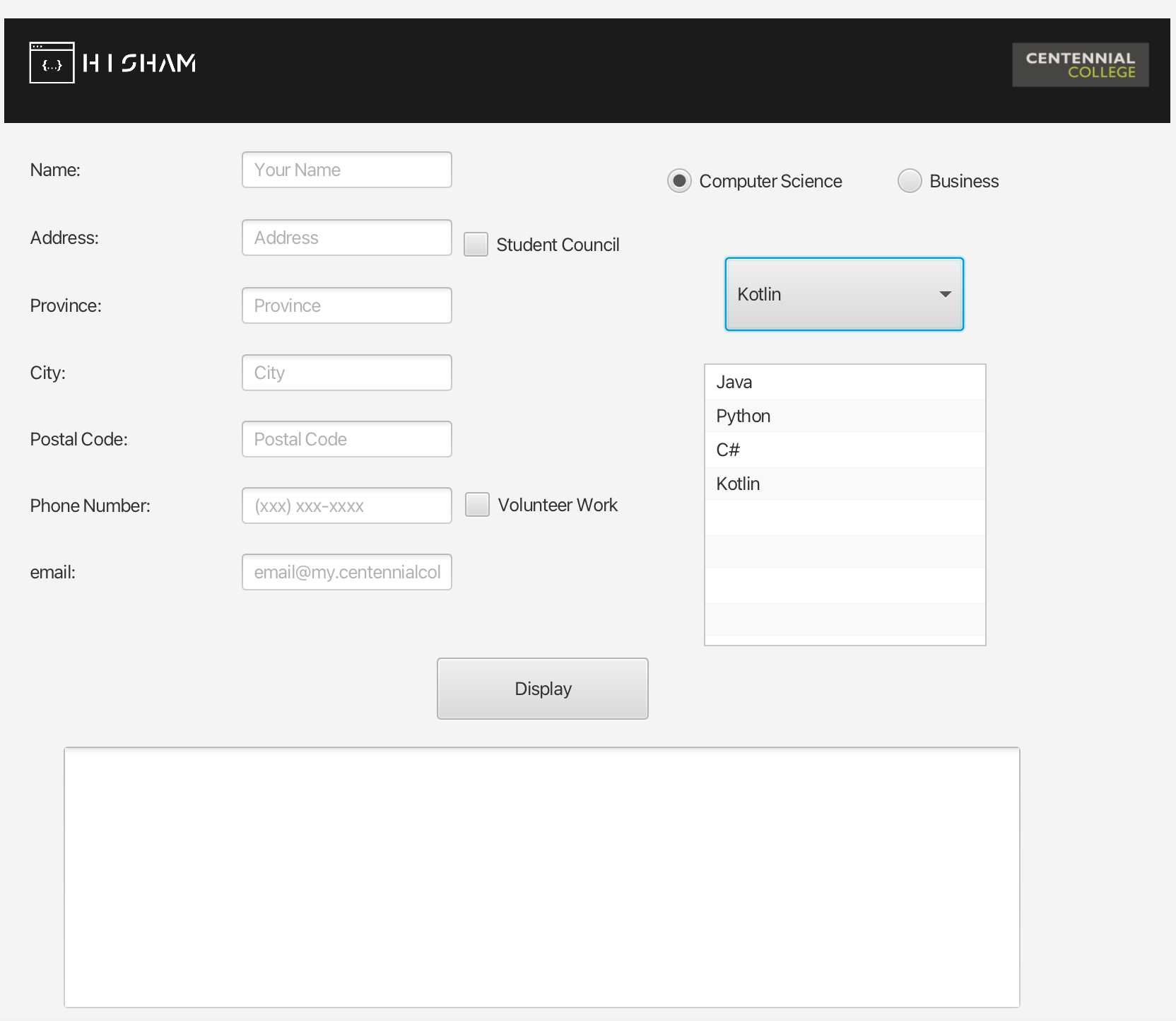
(10 marks)

**Answer**

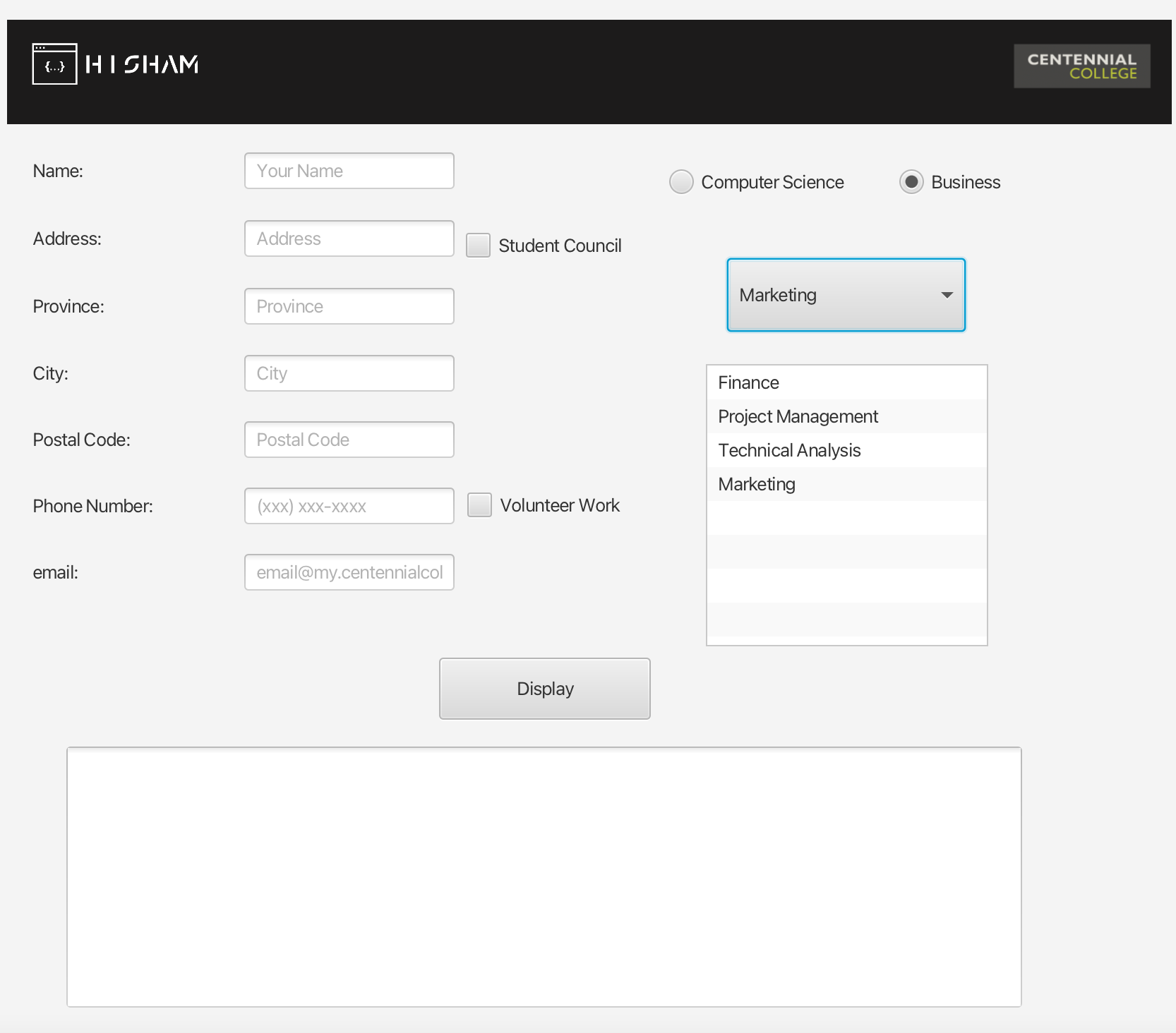
Before Fill Data



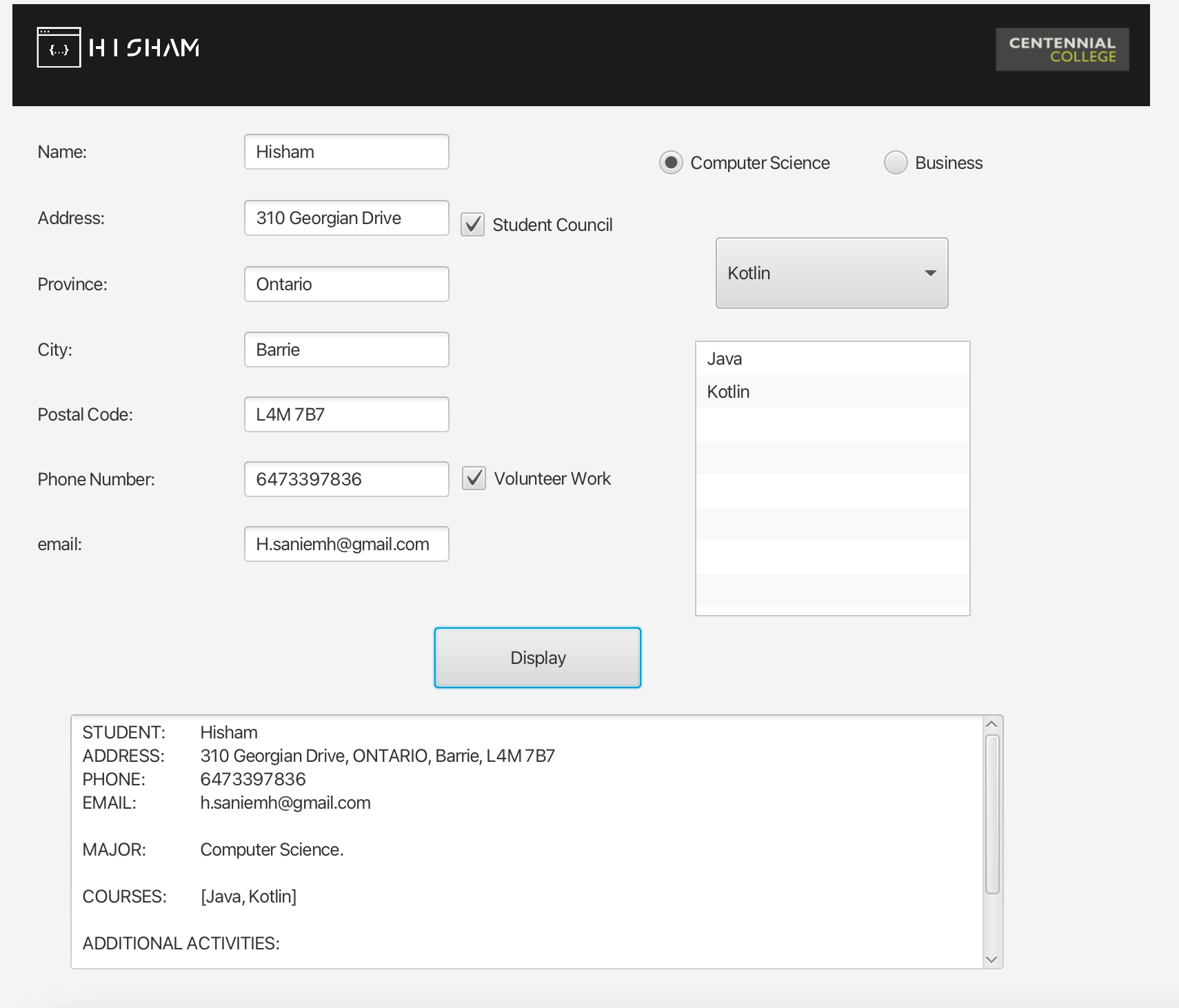
Select Computer Science courses

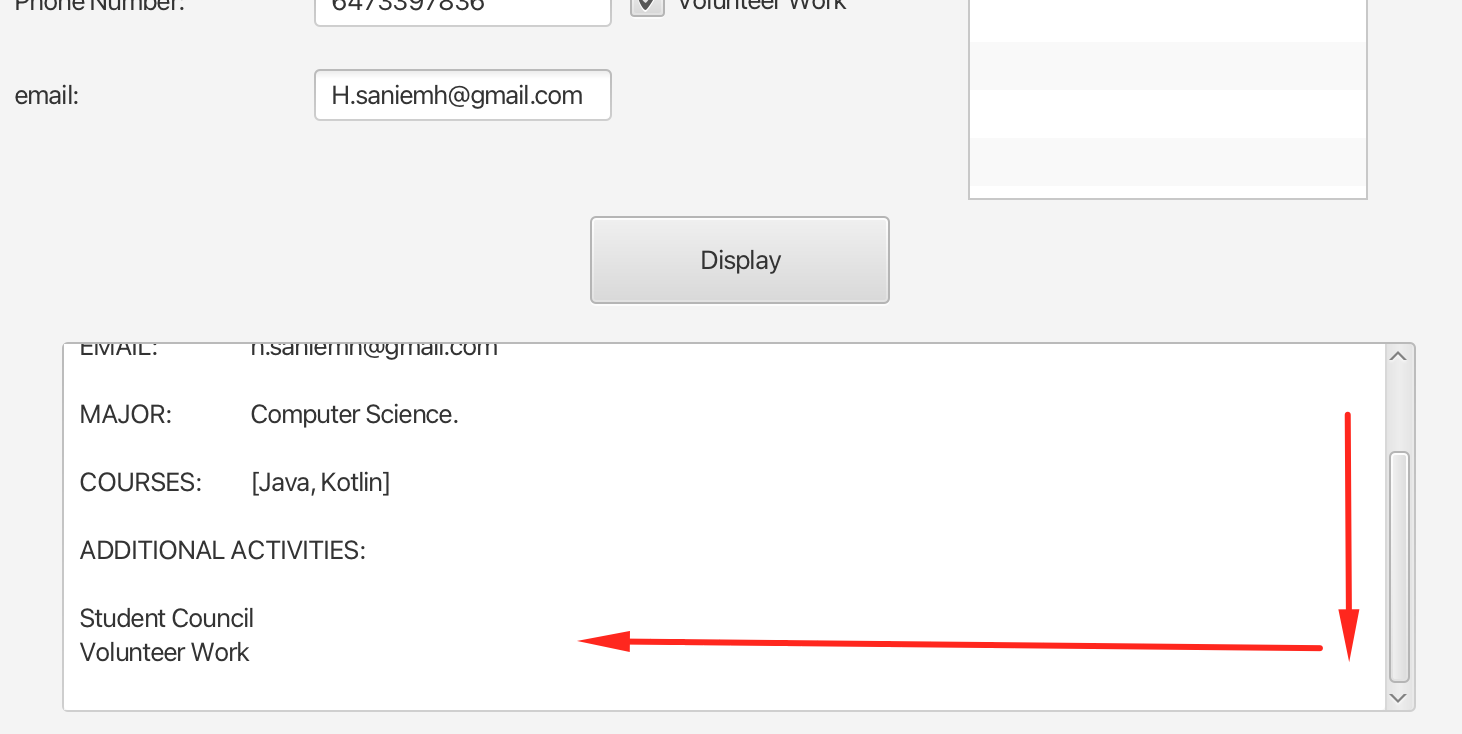


Select Business courses



Final Result

****

****

**Source Code**

1. **package** com.centennial.comp228.lab4;
3. **import** javafx.application.Application;
4. **import** javafx.fxml.FXMLLoader;
5. **import** javafx.scene.Scene;
6. **import** javafx.stage.Stage;
8. **import** java.io.IOException;
10. **public** **class** StudentApplication **extends** Application {
11. @Override
12. **public** void start(Stage stage) throws IOException {
13. FXMLLoader fxmlLoader = **new** FXMLLoader(StudentApplication.**class**.getResource("student-view.fxml"));
14. Scene scene = **new** Scene(fxmlLoader.load(), 852, 757);
15. stage.setTitle("Student Application");
16. stage.setScene(scene);
17. stage.show();
18. }
20. **public** **static** void main(String[] args) {
21. launch();
22. }
23. }
24. <?**xml** version="1.0" encoding="UTF-8"?>
26. <?**import** javafx.scene.control.\*?>
27. <?**import** javafx.scene.image.Image?>
28. <?**import** javafx.scene.image.ImageView?>
29. <?**import** javafx.scene.layout.\*?>
30. <**Pane** maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="757.0"
31. prefWidth="852.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1"
32. fx:controller="com.centennial.comp228.lab4.StudentController">
33. <**children**>
34. <**AnchorPane** layoutX="14.0" layoutY="77.0" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
35. minWidth="-Infinity" prefHeight="657.0" prefWidth="825.0">
36. <**children**>
37. <**GridPane** layoutX="18.0" layoutY="32.0" prefHeight="332.0" prefWidth="299.0">
38. <**columnConstraints**>
39. <**ColumnConstraints** hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"/>
40. <**ColumnConstraints** hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"/>
41. </**columnConstraints**>
42. <**rowConstraints**>
43. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
44. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
45. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
46. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
47. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
48. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
49. <**RowConstraints** minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"/>
50. </**rowConstraints**>
51. <**children**>
52. <**Label** text="Name:"/>
53. <**Label** text="Address:" GridPane.rowIndex="1"/>
54. <**Label** text="Province:" GridPane.rowIndex="2"/>
55. <**Label** text="City:" GridPane.rowIndex="3"/>
56. <**Label** text="Postal Code:" GridPane.rowIndex="4"/>
57. <**Label** text="Phone Number:" GridPane.rowIndex="5"/>
58. <**Label** text="email:" GridPane.rowIndex="6"/>
59. <**TextField** fx:id="txtName" promptText="Your Name" GridPane.columnIndex="1"/>
60. <**TextField** fx:id="txtAddress" promptText="Address" GridPane.columnIndex="1"
61. GridPane.rowIndex="1"/>
62. <**TextField** fx:id="txtProvince" promptText="Province" GridPane.columnIndex="1"
63. GridPane.rowIndex="2"/>
64. <**TextField** fx:id="txtCity" promptText="City" GridPane.columnIndex="1" GridPane.rowIndex="3"/>
65. <**TextField** fx:id="txtPostalCode" promptText="Postal Code" GridPane.columnIndex="1"
66. GridPane.rowIndex="4"/>
67. <**TextField** fx:id="txtPhoneNumber" promptText="(xxx) xxx-xxxx" GridPane.columnIndex="1"
68. GridPane.rowIndex="5"/>
69. <**TextField** fx:id="txtEmail" promptText="email@my.centennialcollege.com" GridPane.columnIndex="1"
70. GridPane.rowIndex="6"/>
71. </**children**>
72. </**GridPane**>
73. <**Button** fx:id="btnDisplay" layoutX="306.0" layoutY="401.0" mnemonicParsing="false"
74. onAction="#displayStudent" onKeyPressed="#displayStudent" prefHeight="44.0" prefWidth="150.0"
75. text="Display"/>
76. <**TextArea** fx:id="taStudentSummary" layoutX="42.0" layoutY="464.0" prefHeight="185.0" prefWidth="677.0"
77. text="
78. "/>
79. <**RadioButton** fx:id="rbComputerScience" layoutX="469.0" layoutY="55.0" mnemonicParsing="false"
80. selected="true" text="Computer Science">
81. <**toggleGroup**>
82. <**ToggleGroup** fx:id="Program"/>
83. </**toggleGroup**>
84. </**RadioButton**>
85. <**RadioButton** fx:id="rbBusiness" layoutX="632.0" layoutY="55.0" mnemonicParsing="false" text="Business"
86. toggleGroup="$Program"/>
87. <**CheckBox** fx:id="studentCouncil" layoutX="325.0" layoutY="100.0" mnemonicParsing="false"
88. text="Student Council"/>
89. <**CheckBox** fx:id="volunteerWork" layoutX="326.0" layoutY="284.0" mnemonicParsing="false"
90. text="Volunteer Work"/>
91. <**ComboBox** fx:id="cbCourses" layoutX="510.0" layoutY="118.0" prefHeight="52.0" prefWidth="169.0"
92. promptText="Select Course"/>
93. <**ListView** fx:id="listView" layoutX="495.0" layoutY="193.0" prefHeight="200.0" prefWidth="200.0"/>
94. </**children**>
95. </**AnchorPane**>
96. <**AnchorPane** layoutX="14.0" layoutY="26.0" prefHeight="35.0" prefWidth="825.0"
97. style="-fx-background-color: #1c1b1b;">
98. <**children**>
100. <**ImageView** fitHeight="50.0" fitWidth="149.0" layoutY="10.0" pickOnBounds="true" preserveRatio="true">
101. <**image**>
102. <**Image** url="@images/my-logo.png"/>
103. </**image**>
104. </**ImageView**>
106. <**ImageView** fitHeight="70.0" fitWidth="149.0" layoutX="700.0" layoutY="4.0" pickOnBounds="true"
107. preserveRatio="true">
108. <**image**>
109. <**Image** url="@images/college-logo.png"/>
110. </**image**>
111. </**ImageView**>
113. </**children**>
114. </**AnchorPane**>
115. </**children**>
116. </**Pane**>
117. **package** com.centennial.comp228.lab4;
119. **import** javafx.fxml.FXML;
120. **import** javafx.fxml.Initializable;
121. **import** javafx.scene.control.\*;
123. **import** java.net.URL;
124. **import** java.util.ResourceBundle;
125. **import** java.util.regex.Matcher;
126. **import** java.util.regex.Pattern;
128. **public** **class** StudentController implements Initializable {
130. //Scene Components
131. @FXML
132. **private** TextArea taStudentSummary;
133. @FXML
134. **private** TextField txtName;
135. @FXML
136. **private** TextField txtAddress;
137. @FXML
138. **private** TextField txtProvince;
139. @FXML
140. **private** TextField txtCity;
141. @FXML
142. **private** TextField txtPostalCode;
143. @FXML
144. **private** TextField txtPhoneNumber;
145. @FXML
146. **private** TextField txtEmail;
147. @FXML
148. **private** ComboBox cbCourses;
149. @FXML
150. **private** ToggleGroup Program;
151. @FXML
152. **private** RadioButton rbComputerScience;
153. @FXML
154. **private** RadioButton rbBusiness;
155. @FXML
156. **private** ListView listView;
157. @FXML
158. **private** CheckBox studentCouncil;
159. @FXML
160. **private** CheckBox volunteerWork;

163. @Override
164. **public** void initialize(URL url, ResourceBundle resourceBundle) {
166. populateComboBox();
168. initRadioButton();
170. initComboBox();
172. }
174. **private** void initComboBox() {
175. //Event Handler for Courses ComboBox
176. cbCourses.setOnAction((event) -> {
178. **if** (cbCourses.getValue() != **null**) {
179. String value = cbCourses.getValue().toString();
180. populateTextArea(value);
181. }
183. });
184. }
186. **private** void initRadioButton() {
187. //Event Handler for ComputerScience Radio Button
188. rbComputerScience.setOnAction((event) -> {
189. populateComboBox();
190. listView.getItems().clear();
191. taStudentSummary.clear();
192. });
193. //Event Handler for Business Radio Button
194. rbBusiness.setOnAction((event) -> {
195. populateComboBox();
196. listView.getItems().clear();
197. taStudentSummary.clear();
198. });
199. }
201. //Method to populate the ComboBox according to Radio Button selection
202. **public** void populateComboBox() {
204. **if** (rbComputerScience.isSelected()) {
205. cbCourses.getItems().clear();
206. cbCourses.getItems().addAll("Java", "Python", "C#", "Kotlin");
208. } **else** {
209. cbCourses.getItems().clear();
210. cbCourses.getItems().addAll("Finance", "Project Management", "Technical Analysis", "Marketing");
212. }
214. }
216. //Method to populate the LIstView
217. **public** void populateTextArea(String course) {
219. **if** (listView.getItems().contains(course)) {
220. Alert fail = **new** Alert(Alert.AlertType.INFORMATION);
221. fail.setHeaderText("failure");
222. fail.setContentText("This course has already been selected");
223. fail.showAndWait();
224. } **else** {
225. listView.getItems().add(course);
226. }
228. }
230. **private** boolean validateFields() {
232. **if** (txtName.getText() == **null** || txtName.getText().trim().isEmpty() ||
233. txtAddress.getText() == **null** || txtAddress.getText().trim().isEmpty() ||
234. txtProvince.getText() == **null** || txtProvince.getText().trim().isEmpty() ||
235. txtCity.getText() == **null** || txtCity.getText().trim().isEmpty() ||
236. txtPostalCode.getText() == **null** || txtPostalCode.getText().trim().isEmpty() ||
237. txtPhoneNumber.getText() == **null** || txtPhoneNumber.getText().trim().isEmpty() ||
238. txtEmail.getText() == **null** || txtEmail.getText().trim().isEmpty()) {
240. Alert fail = **new** Alert(Alert.AlertType.WARNING);
241. fail.setHeaderText("failure");
242. fail.setContentText("Please fill in all the fields");
243. fail.showAndWait();
245. **return** **false**;
246. } **else** **if** (listView.getItems() == **null** || listView.getItems().isEmpty()) {
247. Alert fail = **new** Alert(Alert.AlertType.WARNING);
248. fail.setHeaderText("failure");
249. fail.setContentText("Please Select your courses");
250. fail.showAndWait();
252. **return** **false**;
253. }
255. **return** **true**;
256. }
258. // Email validation
259. **public** boolean validateEmail() {
260. Pattern p = Pattern.compile("[a-zA-Z0-9][a-zA-Z0-9.\_]\*@[a-zA-Z0-9]+([.][a-zA-Z]+)+");
261. Matcher m = p.matcher(txtEmail.getText());
263. **if** (m.find() && m.group().equals(txtEmail.getText())) {
264. **return** **true**;
265. } **else** {
266. Alert alert = **new** Alert(Alert.AlertType.WARNING);
267. alert.setHeaderText("Invalid email");
268. alert.setContentText("Please enter a valid email");
269. alert.showAndWait();
271. **return** **false**;
272. }
273. }
275. //Name Validation
276. **public** boolean validateName() {
277. Pattern p = Pattern.compile("[a-zA-Z]+");
278. Matcher m = p.matcher(txtName.getText());
280. **if** (m.find() && m.group().equals(txtName.getText())) {
281. **return** **true**;
282. } **else** {
283. Alert alert = **new** Alert(Alert.AlertType.WARNING);
284. alert.setHeaderText("Invalid Name");
285. alert.setContentText("Please enter a Name");
286. alert.showAndWait();
288. **return** **false**;
289. }
290. }
292. //Phone Validation
293. **public** boolean validatePhone() {
294. Pattern p = Pattern.compile("[4-9][0-9]{9}");
295. Matcher m = p.matcher(txtPhoneNumber.getText());
297. **if** (m.find() && m.group().equals(txtPhoneNumber.getText())) {
298. **return** **true**;
299. } **else** {
300. Alert alert = **new** Alert(Alert.AlertType.WARNING);
301. alert.setHeaderText("Invalid Phone Format");
302. alert.setContentText("Please enter a valid Phone Number");
303. alert.showAndWait();
305. **return** **false**;
306. }
307. }
309. //Method to retrieve the value of the selected Radio Button
310. **public** String getRadioButton() {
311. RadioButton selectedRadioButton = (RadioButton) Program.getSelectedToggle();
312. String toggleGroupValue = selectedRadioButton.getText();
314. **return** toggleGroupValue;
315. }
317. // Retrieve the values from the CheckBoxes
318. **public** String getAdditionalActivities() {
320. String additionalActivities = "";
322. **if** (studentCouncil.isSelected()) {
323. additionalActivities += studentCouncil.getText() + "\n";
324. }
325. **if** (volunteerWork.isSelected()) {
326. additionalActivities += volunteerWork.getText() + "\n";
327. }
329. **return** additionalActivities;
331. }
333. //to display a summary of all filled in / selected fields
334. **public** void displayStudent() {
336. **if** (validateFields() && validateEmail() && validateName() && validatePhone()) {
338. taStudentSummary.setText("STUDENT:\t\t" + txtName.getText() + "\n" + "ADDRESS:\t\t" + txtAddress.getText() + ", "
339. + txtProvince.getText().toUpperCase() + ", " + txtCity.getText() + ", " + txtPostalCode.getText() + "\n"
340. + "PHONE:\t\t" + txtPhoneNumber.getText() + "\n" + "EMAIL:\t\t\t" + txtEmail.getText().toLowerCase() + "\n" +
341. "\nMAJOR:\t\t" + getRadioButton() + "." + "\n" +
342. "\nCOURSES: " + "\t\t" + listView.getItems() +
343. "\n\nADDITIONAL ACTIVITIES: " + "\n\n" + getAdditionalActivities());
345. }
346. }
347. }
348. module com.centennial.comp228.lab4 {
349. requires javafx.controls;
350. requires javafx.fxml;
352. requires org.controlsfx.controls;
353. requires com.dlsc.formsfx;
354. requires org.kordamp.bootstrapfx.core;

357. exports com.centennial.comp228.lab4;
358. opens com.centennial.comp228.lab4 to javafx.fxml;
359. }

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of UI | 40% |
| Correct implementation of event handling and display of results | 40% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 15% |
| **Total** | 100% |