**Name: Lenell Davis**

**Class: CMIS 242**

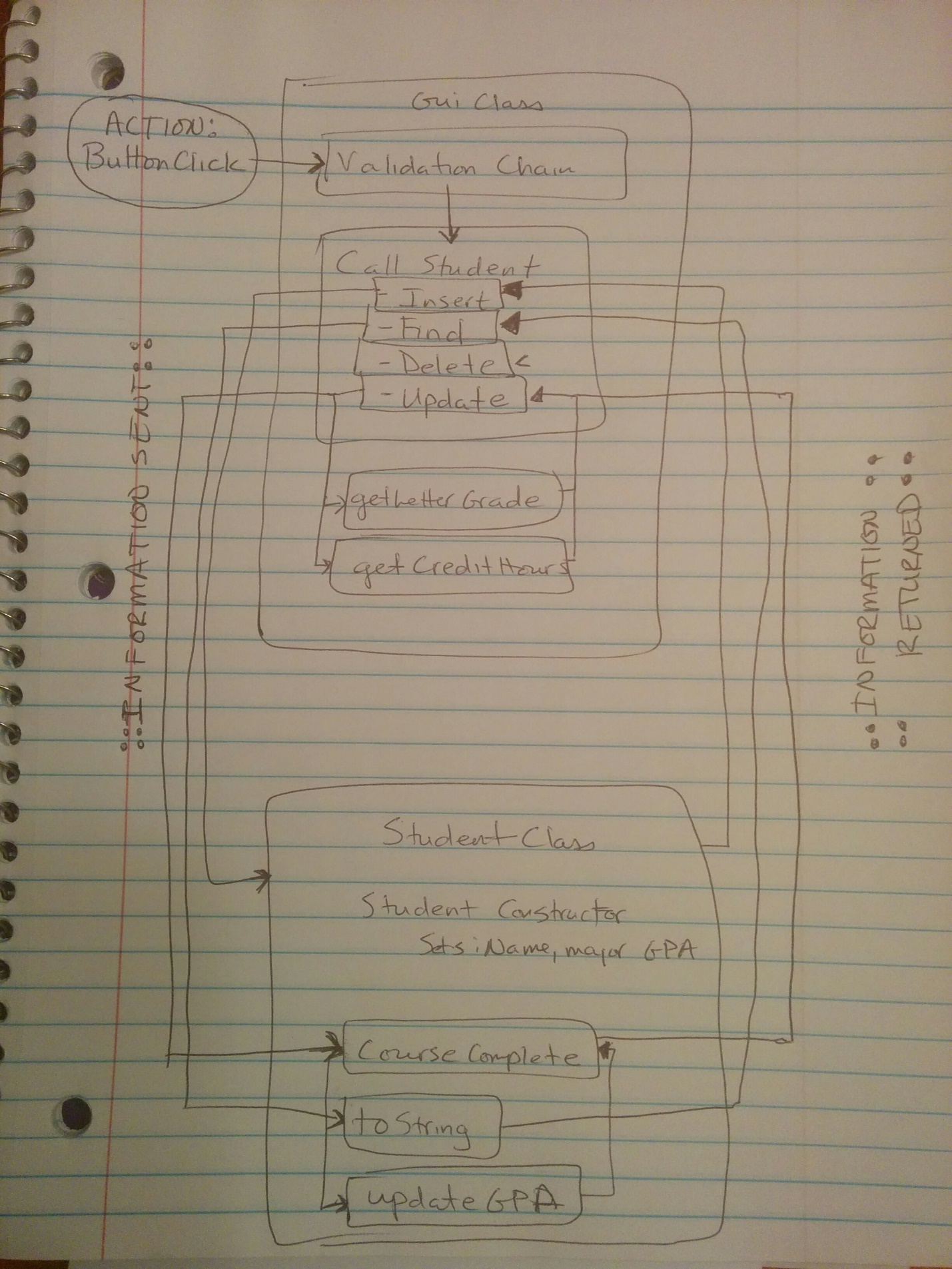
**Professor: Goode**

**Assignment: Project Four**

**Date: 5/8/16**

**Problem Analysis**

Create an interface that will allow a user to create a student database, via a hashmap. The user should be able to add a new student to database, delete a student from the database, find a student within the database and update the GPA of the student.



**Design**

The program requires two classes. My program will have 3 classes. One will hold the main method which will run the program. Another class will contain the GUI components and will house the hashmap for the data base. The final class will do contain the Student class. The Student class will be responsible for creating a new student, updating a student’s GPA, as well as returning a properly formatted String for display to the user.

**Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | Expected Output | Actual Output | Passed |
| Validated that all TextFields have proper Values (used regex patterns to validate Strings) | Used a “validation chain” to go through each text Field and validate the entries. Should display an option pane to the user and illustrate the proper input format | Shows the user the proper dialog and indicates the correct entry format. See Validation Docx. | Y |
| Check that proper grades and credit hours are being selected | Should display println for all letter grades in the switch case with differing credit hours | Println displays correct values for inputs. See Validation docx | Y |
| Check that the find option works and displays the correct value depending on the hash ID | Should display println of each Student, including ID, Name, Major, GPA | Println Displays the correct values for each object. See Validaton docx | Y |
| Check that any null objects from the Student Hashmap are handled with and the user is alerted. | Should display OptionPane with dialog that the Student the user is looking for can not be found. | Option Dialog shows up. See Validation docx | Y |
| Test that new Student Object can be inserted to the hashmap, that the object can be deleted, that the object can be found and that the object’s GPA can be updated. | Two students Lenell Davis, Barrington Davis will be inserted. Both will be found. Barrington will be deleted. Barrington will be found again and display “Not Found” dialog. Update Lenell’s class to an A. Add another class with a grade of B. GPA should be 3.50 | See Validation docx | Y |

**Code**

**Main Program**

1. /\*
2. \* Lenell Davis
3. \* CMIS 242
4. \* Project 4
5. \* 5/8/16
6. \* TestDriver.java
7. \*/
9. **public** **class** TestDriver {
11. **public** **static** **void** main(String[] args) {
12. GpaGui myView = **new** GpaGui();
13. myView.setVisible(**true**);
15. }
16. }

**GpaGui.java**

1. /\*\*
2. \* Lenell Davis
3. \* CMIS 242
4. \* Project 4
5. \* 5/8/16
6. \* GpaGui.java
7. \*/
9. **import** javax.swing.\*;
10. **import** java.awt.\*;
11. **import** java.awt.event.\*;
12. **import** java.util.\*;

15. **public** **class** GpaGui **extends** JFrame{
16. **private** **final** Font font = **new** Font ("Arial", Font.BOLD, 18);
18. //Takes care of the Frames for the Error corrections, Successful input and the Update Grade Frames
19. **private** **final** JFrame frame = **new** JFrame();
20. **private** **final** JFrame gradeFrame = **new** JFrame("Grade information");
21. **private** **final** String[] boxOption = {"Insert", "Delete", "Find", "Update"};
22. **private** **final** String[] gradeOption = {"A", "B", "C", "D", "F"};
23. **private** **final** String[] creditsOption = {"3 credits", "6 credits"};
25. //Creates the Labels for the Gui
26. **private** **final** JLabel idLabel = **new** JLabel("ID: (Numbers only)");
27. **private** **final** JLabel nameLabel = **new** JLabel("First and Last Name:");
28. **private** **final** JLabel majorLabel = **new** JLabel("Major:");
29. **private** **final** JLabel selectLabel = **new** JLabel("Choice Selection:");
31. //Creates the TextFields for the Gui
32. **private** **final** JTextField idField = **new** JTextField(25);
33. **private** **final** JTextField nameField = **new** JTextField(25);
34. **private** **final** JTextField majorField = **new** JTextField();
36. //Creates the ComboBox and Button for the Gui
37. **private** **final** JComboBox selectBox = **new** JComboBox(boxOption);
38. **private** **final** JButton processBtn = **new** JButton("Process Request");
40. //Creates the Panels for the Gui
41. **private** **final** JPanel mainPanel = **new** JPanel();
42. **private** **final** JPanel optionPanel = **new** JPanel();
44. //Creates the Hashmap and Student Object for the database
45. **private** **final** HashMap<Integer, Student> studentHash = **new** HashMap<>();
46. **private** Student newStudent;

49. /\*\*
50. \* Gui Constructor
51. \*/
52. **public** GpaGui(){
53. setFrame();
54. setAttributes();
55. setPanels();
56. gradeFrame.addWindowListener(**new** CloseGradeFrame());
57. }
59. /\*
60. \* Sets the attributes for the frame
61. \*/
62. **private** **void** setFrame() {
63. setSize(600, 300);
64. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
65. setTitle("Student Database");
66. add(mainPanel);
67. addWindowListener(**new** CloseGradeFrame());
68. }
70. /\*\*
71. \* Sets Attributes for Components
72. \*/
73. **private** **void** setAttributes(){
74. optionPanel.setLayout(**new** GridLayout(4,2));
75. idLabel.setFont(font);
76. nameLabel.setFont(font);
77. majorLabel.setFont(font);
78. selectLabel.setFont(font);
79. processBtn.setFont(font);
80. selectBox.setFont(font);
81. processBtn.addActionListener(**new** ButtonClick());
82. }
84. /\*\*
85. \* Adds components to the panels
86. \* Adds the panels to the main panel
87. \*/
88. **private** **void** setPanels(){
89. optionPanel.add(idLabel);
90. optionPanel.add(idField);
91. optionPanel.add(nameLabel);
92. optionPanel.add(nameField);
93. optionPanel.add(majorLabel);
94. optionPanel.add(majorField);
95. optionPanel.add(selectLabel);
96. optionPanel.add(selectBox);
98. mainPanel.add(optionPanel);
99. mainPanel.add(processBtn);
100. }

103. /\*\*
104. \* CallStudent method
105. \* Called after entries are validated.
106. \* Performs an action based on which element of the ComboBox is selected
107. \*/
108. **private** **void** callStudent(){
109. String selectedOption = (String)selectBox.getSelectedItem();
110. **int** currentStudentID = Integer.parseInt(idField.getText());
112. **switch**(selectedOption){
113. **case** "Insert":
114. newStudent = **new** Student(nameField.getText(), majorField.getText());
115. studentHash.put(currentStudentID, newStudent);
116. success("Insertion");
117. emptyTextFields();
118. **break**;
119. **case** "Delete":
120. studentHash.remove(currentStudentID);
121. success("Deletion");
122. emptyTextFields();
123. **break**;
124. **case** "Find":
125. **try**{
126. Student currentStudent = studentHash.get(currentStudentID);
127. String db\_StudentName = currentStudent.name;
128. String db\_StudentMajor = currentStudent.major;
129. Double db\_StudentGPA = currentStudent.gpa;
131. String showUser = newStudent.toString(currentStudentID, db\_StudentName, db\_StudentMajor, db\_StudentGPA);
133. JOptionPane.showMessageDialog(frame, showUser);
134. emptyTextFields();
135. }
136. **catch**(NullPointerException n){
137. JOptionPane.showMessageDialog(frame, "The Student you requested was not found. Verify ID Number and try again.");
138. emptyTextFields();
139. }
140. **break**;
141. **case** "Update":
142. **double** letterValue = getLetterGrade();
143. **int** creditValue = getCredits();
144. Student currentStudent = studentHash.get(currentStudentID);
145. **double** db\_StudentGPA = currentStudent.gpa;
146. **double** updatedGPA = newStudent.courseCompleted(letterValue, creditValue, db\_StudentGPA);
147. currentStudent.gpa = updatedGPA;
148. emptyTextFields();
149. **break**;
150. **default**:
152. }
153. }
155. /\*\*
156. \* Clears the textFields for new inputs
157. \*/
158. **public** **void** emptyTextFields(){
159. idField.setText("");
160. nameField.setText("");
161. majorField.setText("");
162. }
164. /\*\*
165. \* Part 1 of the Validation chain.
166. \* Validates that the id is an integer
167. \*/
168. **private** **void** guiIdValidation(){
169. String id = idField.getText();
171. **try**{
172. **int** idNumber = Integer.parseInt(id);
173. guiNameValidation();
174. }
175. **catch**(NumberFormatException e){
176. JOptionPane.showMessageDialog(frame, "Please enter an integer.");
177. }
178. }
180. /\*\*
181. \* Part 2 of the Validation chain.
182. \* Validates that the name is a string of the correct pattern using regex
183. \*/
184. **private** **void** guiNameValidation(){
185. //.\* tests the entire string and not one character. The \\s test for white space
186. String regexName = "[A-Za-z][^0-9].\*"+" "+"[A-Za-z][^0-9].\*";
187. String name = nameField.getText();
189. **if**(name.matches(regexName)){
190. guiMajorValidation();
191. }
192. **else**{
193. JOptionPane.showMessageDialog(frame, "Please enter a name. Example: Susan Rogers");
194. }
196. }
198. /\*\*
199. \* Part 3 of the Validation chain.
200. \* Validates that the major is a string of the correct pattern using regex
201. \*/
202. **private** **void** guiMajorValidation(){
203. //.\* tests the entire string and not one character. The \\s test for white space
204. String regexName = "[A-Za-z][^0-9].\*"+" "+"[A-Za-z][^0-9].\*";
205. String regexMajor = "[A-Za-z][^0-9].\*";
206. String major = majorField.getText();
208. **if**(major.matches(regexName)){
209. callStudent();
210. }
211. **else** **if** (major.matches(regexMajor)){
212. callStudent();
213. }
214. **else**{
215. JOptionPane.showMessageDialog(frame, "Please enter a major. Example: Anthropology or Computer Science");
216. }
217. }
218. /\*\*
219. \* Shows that the operation was a success
220. \*/
221. **private** **void** success(String a){
222. JOptionPane.showMessageDialog(frame, a + " Operation Successful.");
223. }
225. /\*\*
226. \* Shows inputDialog for Letter Grade
227. \*/
228. **private** **double** getLetterGrade(){
229. String letter = (String)JOptionPane.showInputDialog(gradeFrame, "Please Choose a Letter Grade", "Letter Grade Option", JOptionPane.QUESTION\_MESSAGE, **null**, gradeOption, gradeOption[0]);
230. **double** returnLetterValue = 5.0;
232. **try**{
233. **switch**(letter){
234. **case** "A":
235. returnLetterValue = 4.0;
236. **break**;
237. **case** "B":
238. returnLetterValue = 3.0;
239. **break**;
240. **case** "C":
241. returnLetterValue = 2.0;
242. **break**;
243. **case** "D":
244. returnLetterValue = 1.0;
245. **break**;
246. **case** "F":
247. returnLetterValue = 0.0;
248. **break**;
249. }
250. }
251. **catch**(NullPointerException n){
252. System.out.println("Null Object Created. returnLetterValue not being set. Will manually set returnLetterValue to 0.0");
253. returnLetterValue = 0.0;
254. }
255. **return** returnLetterValue;
256. }
258. /\*\*
259. \* Shows inputDialog for Credit Hours
260. \*/
261. **private** **int** getCredits(){
262. String credit = (String)JOptionPane.showInputDialog(gradeFrame, "Please Choose the Number of Credits", "Credit Option", JOptionPane.QUESTION\_MESSAGE, **null**, creditsOption, creditsOption[0]);
263. **int** returnCreditValue = 1;
265. **try**{
266. **switch**(credit){
267. **case** "3 credits":
268. returnCreditValue = 3;
269. **break**;
270. **case** "6 credits":
271. returnCreditValue = 6;
273. **break**;
274. **default**:
275. returnCreditValue = 1;
276. }
277. }
278. **catch**(NullPointerException n){
279. System.out.println("Null Object Created. returnCreditValue not being set. Will manually set returnCreditValue to 1");
280. }
282. **return** returnCreditValue;
283. }
285. /\*\*
286. \* Inner class that controls the Button
287. \* Begins the Validation process before calling the callStudent Method
288. \* Ensures that all fields are valid before the operation can begin
289. \*/
290. **private** **class** ButtonClick **implements** ActionListener{
292. @Override
293. **public** **void** actionPerformed(ActionEvent e) {
294. guiIdValidation();
295. }
296. }
298. /\*\*
299. \*Inner class that handles the WindowClose Event
300. \*/
301. **private** **class** CloseGradeFrame **extends** WindowAdapter{
303. @Override
304. **public** **void** windowClosed(WindowEvent e){
305. System.exit(0);
306. }
307. }
309. }

**Student.java**

1. /\*\*
2. \* Lenell Davis
3. \* CMIS 242
4. \* Project 4
5. \* 5/8/16
6. \* Student.java
7. \*/
9. **public** **class** Student {
10. **public** **final** String name;
11. **public** **final** String major;
12. **public** **double** gpa;
13. **public** **int** creditHours;
15. /\*\*
16. \* Constructor for the Student Class
17. \* Sets the name, major and GPA for each student depending on input
18. \*/
19. **public** Student(String guiName, String guiMajor){
20. **this**.name = guiName;
21. **this**.major = guiMajor;
22. gpa = 0.00;
23. creditHours = 0;
24. }
26. /\*\*
27. \* Called when Update is Selected
28. \*/
29. **public** **double** courseCompleted(**double** letterGrade, **int** credits, **double** currentGPA){
30. **double** newGPA = (letterGrade \* credits) / credits;
31. **double** returnedGPA = updateGPA(newGPA, currentGPA);
33. **return** returnedGPA;
34. }
36. /\*\*
37. \* Called when Find is Selected. Returns string about student.
38. \*/
39. **public** String toString(**int** id, String name, String major, **double** gpa){
40. String returnTo = String.format("Student ID: %d\nStudent Name: %s\nMajor: %s\nGPA: %.2f", id, name, major, gpa);
41. **return** returnTo;
42. }
44. /\*\*
45. \* Updates the GPA when multiple Grades are entered
46. \*/
47. **public** **double** updateGPA(**double** newGPA, **double** currentGPA){
48. **double** updatedGPA = 0.0;
49. **if**(currentGPA > 0 ){
50. updatedGPA = (currentGPA + newGPA) / 2;
51. }
52. **else**{
53. updatedGPA = newGPA;
54. }
55. **return** updatedGPA;
56. }
58. }

**Lessons Learned**

\*\*Issues: NullObject created when returning letter grade value from the switch case. Attempts to set a default value did not get rid of the error. Solution: surrounded with try-catch and set the value manually. Explanation also inside code.

\*\*Issues: GPA improperly calculated due to the aforementioned issue. Problem only arrises when trying to add multiple classes. Example: Exiting each OptionPane will yield a GPA of 0 the first time. Updating the GPA to 4.0 for one class, will also yield the correct GPA. However, when the user chooses the “Update” option and exits each OptionPane again, the GPA for the user will be 2.0

Improvements: Allow users to find student based off of any value. Would have to rework the validation process. However, the validation process can be used to further strengthen the program by checking multiple values to make sure the program has the correct student object.