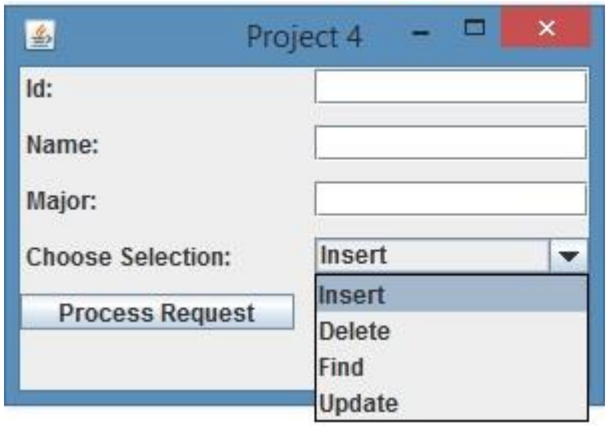
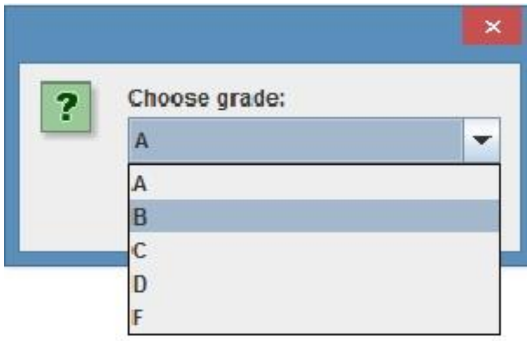
Project 4

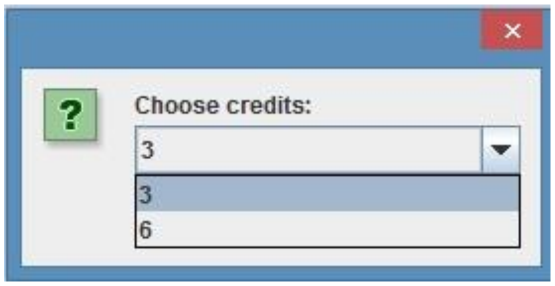
1. Specification

This programming project involves writing a program to manage a student database. The interface to the program should be a Swing based GUI that looks similar to the following:



A combo box should allow the user to select one of the four database actions shown. The database should be implemented as a **HashMap**, with the Id field as the key and a Student record as the value. The selected operation should be performed when the user clicks the **Process Request** button. If the user attempts to insert a key that is already in the database an error message should be displayed using a JOptionPane message dialog box. If the user attempts to delete, find or update a record that is not in the database, a message should also be displayed in a JOptionPane. After each successful operation is completed a JOptionPane window should be displayed confirming the success. In the case of a successful **Find** request, a window should pop up containing the student's Id, name, major and current GPA. When the user selects the **Update** request, the following JOptionPane windows should be displayed to gather information about a course that has just been completed. As a result, the **Student** record in the database will be updated accordingly. User input values should be checked and a warning message should be displayed in a JOptionPane for inappropriate values (such as empty textfields for Id, name or major).





When the main application window is closed, the student records from the database will be written into the text file **outData.txt**, each student data (including his/her GPA) on a separate line. The last line of the file will contain the total number of student records in the database and their average GPA value.

The program should consist of two classes.

1. The first class **P4GUI** should define the GUI and handle the database interactions. It should be hand-coded and not generated by a GUI generator.

2. The second class named **Student**, should define the student record. The class defines the instance variables id, name, major, totalCredits (as total number of credits completed) and totalQP (as total quality points). The quality points for a course is calculated as the numeric value of the grade received in the course (A = 4; B = 3; C = 2; D = 1; F = 0) times the number of credit hours for that course. The values of totalCredits and totalQP will be used to calculate the GPA by dividing totalQP by totalCredits. The class Student should have the following three methods:

a. A constructor with arguments that is used when new student records are created It takes Id, name and major as parameters and will initialize totalCredits and totalQP to zero;
b. A method **courseCompleted** that should accept the course grade and credit numbers and will update the variables used to compute the GPA. It will be called when an Update request is made.
c. A method overriding **toString** that returns a labeled string containing the student id, name, major and GPA.

Finally, when a student has not yet completed any course, the GPA should be displayed as 4.0.

Your program should compile without errors.

The Google recommended Java style guide (https://google.github.io/styleguide/javaguide.html) should be used to format and document your code. Specifically, the following style guide attributes should be addressed:

- Header comments include filename, author, date and brief purpose of the program.
- In-line comments used to describe major functionality of the code.
- Meaningful variable names and prompts applied.
- Class names are written in UpperCamelCase.
- Variable names are written in lowerCamelCase.
- Constant names are in written in All Capitals.
- Braces use K&R style**.**

In addition the following design constraints should be followed:

- Declare all instance variables private
- Avoid the duplication of code

Test cases should be supplied in the form of a table with columns indicating what aspect is tested, the input values, expected output, actual output and if the test case passed or failed. This table should contain 5 columns with appropriate labels and a row for each test case. Note that the actual output should be the actual results you receive when running your program and applying the input for the test record. Be sure to select enough different kinds of employees and situations to completely test the program.

2. Submission Requirements

Submit the following to the Project 2 assignment area no later than the due date listed in your LEO classroom.

1. The source files **P4GUI.java** and **Student.java** and the program generated output file **outData.txt**. The source code should use Java code conventions and appropriate code layout (white space management and indents) and comments. All submitted files may be included in a .zip file.

2. The solution description document **P4SolutionDescription** (.pdf or .doc / .docx) containing the following:

(1) Assumptions, main design decisions, error handling;
(2) Test cases table
(3) Screen captures showing successful program compilation and test cases execution. Each screen capture should be properly labeled, clearly indicated what the screen capture represents.
(4) Lessons learned from the project;

3. Grading Rubric

The following grading rubric will be used to determine your grade:

Attribute	**Meets**	**Does not meet**
P4GUI class	**35 points** a) Defines the GUI. b) Provides a combo box to allow the user to select one of the four database actions including insert, update, delete and find. c) The database is implemented as a HashMap, with the ID field as the key and a student record as the value. d) The operation is performed when the user clicks the Process Request button. e) User input values are checked and warning message is displayed for inappropriate values entered in the Id, name or major textfields. f) If the user attempts to insert a key that is already in the database an error message is displayed using a JOptionPane message dialog box. g) If the user attempts to delete, find or update a record that is not in the database, a message is displayed. h) After each successful operation is completed a JOptionPane is displayed confirming the success. i) In the case of a successful Find request, a window pops-up containing the student's ID, name, major and current GPA.	**0 points** a) Does not defines the GUI. b) Does not provide a combo box to allow the user to select one of the four database actions including insert, update, delete and find. c) The database is not implemented as a HashMap, with the ID field as the key and a student record as the value. d) The operation is not performed when the user clicks the Process Request button. e) User input values are not checked and warning message is not displayed for inappropriate values entered in the Id, name or major textfields. f) If the user attempts to insert a key that is already in the database an error message is not displayed using a JOptionPane message dialog box. g) If the user attempts to delete, find or update a record that is not in the database, a message is not displayed. h) After each successful operation is completed a JOptionPane is not displayed confirming the success. i) In the case of a successful Find request, a window does not pop-up containing the student's ID, name, major and current GPA.

	j) When the user selects the Update request, a JOptionPane is displayed to gather information about a course that has just been completed including the grade and number of credits. k) When the window is closed, the student records from the database are written in a file, each student data (including their GPA) on a separate line. l) The last line of the file contain the total number of student records in the database and their average GPA value.	j) When the user selects the Update request, a JOptionPane is not be displayed to gather information about a course that has just been completed including the grade and number of credits. k) When the window is closed, the student records from the database are not written in a file, each student data (including their GPA) on a separate line. l) The last line of the file does not contain the total number of student records in the database and their average GPA value
Student Class	**40 points** a) Defines the student record. b) Contains instance variables for the student id, name, major and two variables that are used to compute the GPA. c) Contains a variable representing the total number of credits completed d) Contains a variable representing the total quality points, which are the numeric value of the grade received in a course times the number of credit hours. e) Contains a constructor that is used when new student records are created. It should accept Id, name and major as parameters and initialize the fields that are used to compute the GPA to zero. f) Contains a method courseCompleted that accepts the course grade and credit numbers and update the variables used to compute the GPA. g) courseComplete is called when an Update request is made. h) Contains an overridden toString method that returns a labeled string containing the student name, major and GPA.	**0 points** a) Does not define the student record. b) Does not contains instance variables for the student id, name, major and two variables that are used to compute the GPA. c) Does not contain a variable representing the total number of credits completed d) Does not contain a variable representing the total quality points, which are the numeric value of the grade received in a course times the number of credit hours. e) Does not contains a constructor that is used when new student records are created. It should accept Id, name and major as parameters and initialize the fields that are used to compute the GPA to zero. f) Does not contains a method courseCompleted that accepts the course grade and credit hours and update the variables used to compute the GPA. g) courseComplete is not called when an Update request is made. h) Does not contains an overridden toString method that returns a labeled string containing the student name, major

	i) Calculates and displays a GPA of 4.0 for students who have not yet completed any course.	and GPA. i) Does not calculate or display a GPA of 4.0 for students who have not yet completed any course.
Test Cases	**10 points** a) Test cases are supplied in the form of table with columns indicating test case objective, the input values, expected output, actual output and if the test case passed or failed. b) Enough scenarios selected to completely test the program. c) Test cases were included in the supporting word or PDF documentation.	**0 points** a) No test cases were provided.
Documentation and Style guide	**10 points** a) Solution description document **P4SolutionDescription** includes all the required sections appropriate titled. **Source code** criteria b) Header comments include filename, author, date and brief purpose of the program. c) In-line comments used to describe major functionality of the code. d) Meaningful variable names and prompts applied. e) Class names are written in UpperCamelCase. f) Variable names are written in lowerCamelCase. g) Constant names are in written in All Capitals. h) Braces use K&R style. i) Declare all instance variables private. j) Avoids the duplication of code.	**0 points** a) No **solution description document** is included. **Source code** criteria b) Java style guide was not used to prepare the Java code. c) All instance variables not declared private. d) Duplication of code was not avoided.