

Assignment 3

Array Lists

Goals:

- Learn to work with Java ArrayLists
- Continue learning to design and implement classes
- Continue learning how to test

Program:

Develop a program that tests an object of the class College that has Student objects. The program should support the following operations:

- Create a College object.
- Adding a new student to the college.
- Looking up an existing student based on the student number.
- Deleting a student from the college using the student number.
- Adding the grade point value and credit units earned for a course taken by an existing student.
- Retrieving the login id for an existing student (using the student number to look up the student).
- Finding a student with the college's highest GPA.

About your implementation:

- Once the user quits the program, all the data on the College is lost.
- You should create the following three classes:
 - The Student class from assignment # 2 (with the specifications given there): each Student object stores the name, student number (given by the system at creation), address, and the total grade points for a student plus the number of credits taken so far. You may need to modify your Student public interface from assignment #2 (or fix the class if it was not correct).
 - The College class with the list of all students (use an ArrayList of Student objects).
 - The CollegeTester class with a main method which creates a College object (which manipulates Student objects) based on user input.
 - Modularize the main program.
 - In this CollegeTester you are to ask the user for input using the Scanner class.
 - Print the possible commands in a 'menu' for the user to know what the possible commands are.
 - Get different commands from the user to manipulate the College.
- Your program is to handle erroneous data. Display appropriate error messages. For example,
 - Check whether the user enters a valid command.

- Determine if the student (based on the student number) is indeed in the college for operations such adding a course for the student.
- Check whether the student number is valid input (in the sense of it being numeric and greater than or equal to the smallest possible student number).
- A student name must have exactly two names (words) separated by at least one blank: check that the user enters such a student name.
- The address is “free formatted” so be careful when using the methods `nextInt/nextDouble` and `nextLine` from the Scanner class.
- Comment your classes and methods appropriately using the Javadoc notation. A document that explains how to write the comments is under <http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html>

Submit a single ZIP (Compressed) file to D2L containing:

The source code for all three classes (in three different files): Student.java College.java and an application class like CollegeTester.java.

Marking Scheme

[30] College Class design and implementation

- [10] Constructor And Private instance variable
- [20] Public Methods
 - [5] Adding a Student, parameter is Student object, and Deleting a Student given a student number.
 - [5] Finding a student given a student number.
 - [5] Finding a student with highest GPA.
 - [5] toString method.

[50] CollegeTester

- [35] functionality
 - Adding a new Student to the College and deleting a student from the college.
 - Looking up an existing student
 - Adding a course to a specific student
 - Retrieving the login ID for a specific student
 - Finding a student with the highest GPA
- [10] Handle Errors
 - Check that user entered a valid command
 - Valid student number consisting of 8 digits
 - Student name must have two names separated by space
- [5] Readability of output produced by CollegeTester

[10] Documentation: Javadoc for all the methods headers, purpose of the program, inline comments

[10] Programming Style: Meaningful variable names and constants following the conventions, consistent indentation.