Fall 2022– CS 203 Object-Oriented Programming

Homework # 4: Project Report

Name: Haeun Suh, BlazerId: hsuh

Introduction

This project is designed to provide the solutions for the questions regarding a given student data. Main purpose of this project will be to demonstrate the understanding of OOP concepts, an approach to applying OOP to the project, and the ability to implement code in general. The program mainly works like as follow:

1. Main Class will execute 6 functions within its class, first function will read CSV file to load student data, and other 5 functions will be corresponding with solutions for the questions each.
2. Interface defines the solution methods that will returns answers to each question.
3. There exist only one class that actually conducts problem solving and applying solution methods.

This project is solely conducted by me, with no collaborations.

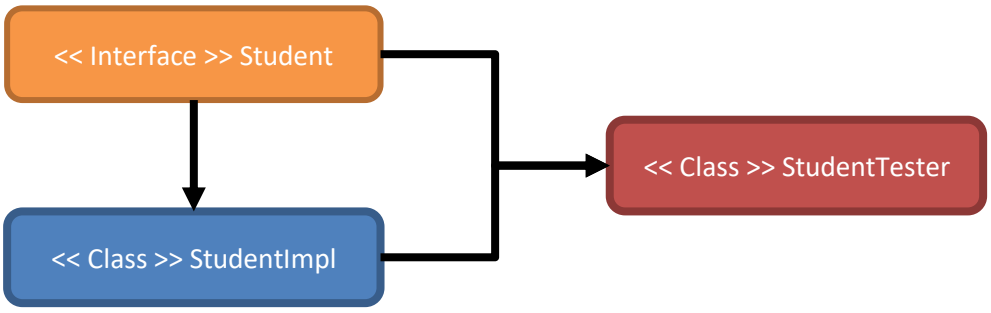
Code Explanation

1) Summary

This project included 1 interface, 1 General class implemented the interface, and Main class that are showing out the solution of given questions. OOP is applied to the project such as using override methods, implementing interface so to inherit by its child class and clarifying that abstract methods to be implemented within its child class, and using polymorphism so to utilize the class methods but to restrain access some of its peculiar methods from first level class (main class or main function).

2) Structure and hierarchy nature of implemented classes

The structure of the project is like as below:



Interface ‘Student’ Class defines the abstract functions to be implemented and ‘StudentImpl’ implements those functions by implementing the ‘Student’ interface. Since no significant hierarchy observed from given data, other inheritance will not apply but ‘StudentImpl’ class utilized Map/HashMap properties to grouping student data by department. The project executed main () function from the class ‘StudentTester’. Even though ‘StudentImpl’ class has its mutators and accessors for its properties (or attributes) within class, the main class uses ‘Student’ interface as its reference variable so to admit the explicit classes declared as abstract method within ‘Student’.

3) The way of working as a whole project.

From the ‘StudentTester’ class, main () function will be executed. From the main class, the methods that are corresponded to each given question will be conducted sequentially. Initially, the ‘StudentImple’ class instance will be declared but referenced by ‘Student’ interface variable so to only make call within the methods that defined in ‘Student’ interface only. Then, for the first time and by an execution of initial method readStudentData(), the generally use data for the executions will be read and loaded. After that, the solution methods will be executed. The methods are ordered as the same as the order of the given questions. Each main methods are encapsulating the methods defined in ‘Student’ interface. Therefore, via ‘Student’ interface, ‘StudentImpl’ methods will be executed. Also, in order to reduce dependency of print method, these methods, which are main class level methods, are designated to include print method instead of ‘Student’ or ‘StudentImpl’. After executing all defined method from main class. the program will be terminated automatically.

4) Properties and methods of each class.

a) Main Class ‘StudentTester’
* Contains 6 designated methods as sequentially within main() function.
* ‘StudentImpl’ instance as ‘Student’ referencing will be initiated from the beg
* Due to the nature of main() function, the methods are static method.
* Each method is independent to each other.
* Exception means the denoted exceptions are thrown by the corresponding method.

Name	Type	Parameter	Return	Description	Exception
main	Static Method	String[] args	None	Main function of project	FileNotFoundException, ArrayIndexOutOfBoundsException, NullPointerException, NumberFormatException, IOException
readStudentData	Static Method	Student student, String dataFileName, boolean hasHeader	None	Read and load data from given filePath	None
displayAllAdv	Static Method	Student student	None	Print the number of advisors and their names	None

diaplayUnderGPAList	Static Method	Student student, double gpa	None	Print the student list whose GPA is under the given GPA	None
displayAvgCH	Static Method	Student student	None	Print the average credit hours by college level	None
displayAvgGPAByMajor	Static Method	Student student, String department	None	Print the average GPA by the given department level	None
displayNumOfAdvByDept	Static Method	Student student	None	Print the number of advisors from each department	None

b) Interface 'Student
* Only contains abstract methods. (Skip descriptions)
* Exception means the denoted exceptions are thrown by the corresponding method.

Name	Type	Parameter	Return	Exception
readStudentData	Abstract Method	String filePath, boolean hasHeader	None	FileNotFoundException, ArrayIndexOutOfBoundsException, NullPointerException, NumberFormatException, IOException
getAllAdvisors	Abstract Method	None	Set<String> allAdvisors	None
getAdvisorsByDept	Abstract Method	String department	Set<String> advisorsByDept	None
getGPAUnderThan	Abstract Method	String dataFileName	List<StudentImpl> underPerformed	None
getAvgCH	Abstract Method	None	double avgCreditHours	None
getAvgGPAByDept	Abstract Method	String department	double avgGPAByDept	None
getAllNumOfAdvByMajor	Abstract Method	None	Map<String, Integer> advsByDept	None

c) Class 'StudentImpl'
* Description of Assessor/Mutator is skipped.
* All private properties have their own assessor/mutator.
* This class implements Interface 'Student'

Name	Type	Parameter	Return	Description
name	String	* These are attributes (or Properties)		Student's name
major	String			Student's major
degree	String			Student's degree program
gpa	double			Student's GPA

creditHours	int			Student's credit hours enrolled
ta	String			Whether Student doing TA
advisor	String			Advisor of the student
studentList	Map <String,List<StudentImpl>>			List of the Student grouped by their department
departments	Set<String>			List of the departments
StudentImpl	Constructor Method	None	None	Initiate Map and Set attributes
StudentImpl	Constructor Method	String name, String major, String degree, double gpa, int creditHours, String ta, String advisor	None	Instantiate and set values to corresponding properties
readStudentData (*)	Implemented Method	String filePath, boolean hasHeader	None	File read and set the values to the corresponding Map properties
getAllAdvisors	Implemented Method	None	Set<String> allAdvisors	Return non-duplicated advisor list by college level
getAdvisorsByDept	Implemented Method	String department	Set<String> advisorsByDept	Return non-duplicated advisor list by department level
getGPAUnderThan	Implemented Method	double gpa	List<StudentImpl> underPerformed	Return the list of students whose GPA is lower than given GPA
getAvgCH	Implemented Method	None	double avgCreditHours	Return the average credit hours by college level
getAvgGPAByDept	Implemented Method	String department	double avgGPAByDept	Return the average GPA by department level
getAllNumOfAdvByMajor	Implemented Method	None	Map<String, Integer> advsByDept	Return the map properties that contains key as department and value as the number of advisors of the department
toString	Override Method	None	String toString	Return predesigned string utilizing its attributes

(*) Exceptions have been thrown from this method only. The exceptions are like as below: FileNotFoundException, ArrayIndexOutOfBoundsException, NullPointerException, NumberFormatException, IOException

5) Benefits of the implementation (OOP and certain properties)

* By using interface: This not only can clarify the actual methods which are related to the main purpose but also can hide unused, unrelated, and avoid being revealed from main class level code that might can make confusion. Also, using interface is beneficial as it prevents programming flaws to be occurred from compiling level.

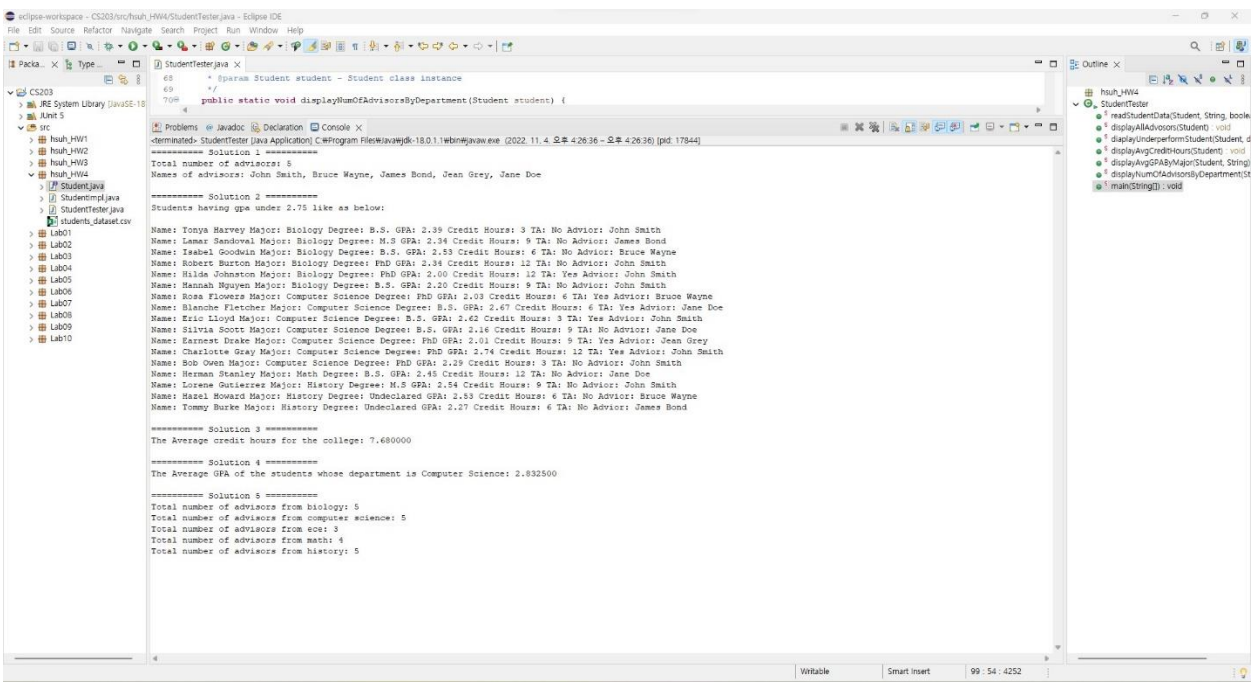
* By utilizing polymorphism: By referencing interface rather than directly initiate as its class instance variables, unused methods such as getters or setters will be hidden and not permit to be added on main class until interface create the link for those methods. This is beneficial for developers who want to avoid uncontrolled access of private variables even if its accessors are declared as public.

* Utilizing map property: Rather than create all corresponding class and get inherited from ‘Student’ class, it is considered more reasonable to use map or hash map properties to map distinguishable attributes such as ‘major’ attribute. Since the difference of department which student is in does not differs attributes regarding the data and questions that are given, creating all class file would be superfluous and only has few benefits. Compared to this, a map property provides more efficient and simple ways to grouping the data by its key properties.

* Utilizing set property: Set property minimizes duplicity and easily concatenate the multiple set to the non-duplicated set.

* By using override methods: This can minimize duplicity and improve reusability.

Result



[Screenshot of Executed Result]

References

1. Class materials including lab slides.
2. Previous code materials, such as HW3.