****

TECHNICAL REPORT:

JAVA ENROLLMENT MANAGEMENT CONSOLE APP



, 2022

PROGRAM: BACHELOR OF INFORMATION TECHNOLOGY (BP162)

COURSE NAME: INTRODUCTION TO INFORMATION TECHNOLOGY

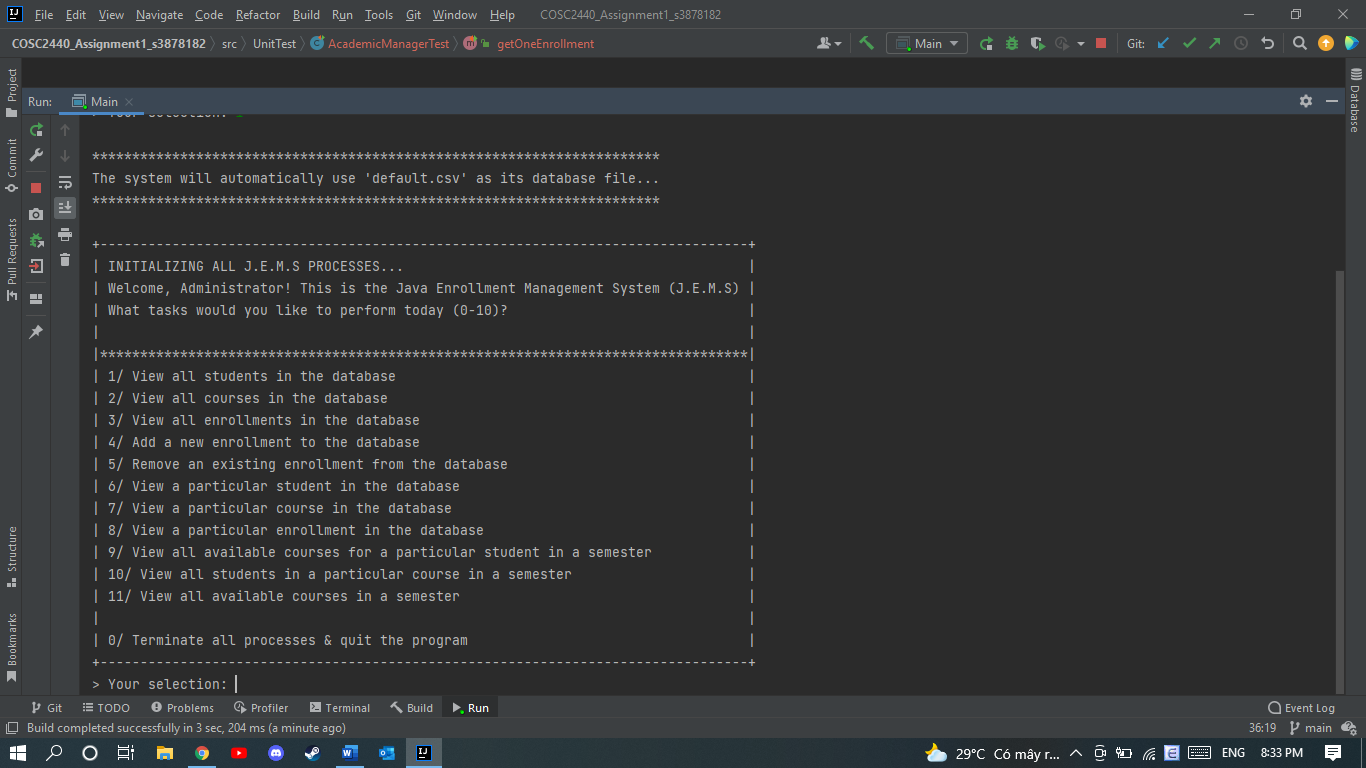
COURSE CODE: COSC2440

LECTURER: MR. MINH VU THANH

**1/ Introduction:**

As a concise description, this project revolves around the core concepts of a typical Enrollment Management Application commonly found in most educational institutions nowadays. Some of the primary functions include viewing all students/courses/enrollments status, viewing all information from a particular student/course/enrollment, adding/deleting an enrollment status from the database, filtering data based on user queries to display only the selected students/courses within a specific semester code. The entirety of this intricate system is encapsulated by a neatly designed User Interface (UI) with a captivating menu display yet remains minimalistic according to User-Centered Design (UCD) practices, and users can easily explore their options, perform operations as well as exiting a program without any trouble.

In addition, the menu is illustrated sequentially with small increments in between to the purpose of allowing users the time to adjust and not overwhelm them with all the elements at once, and this console application will continue to function perpetually (unless the user selects the exit prompt). This essentially means users are not required to manually restart the program after a session is finished, as it is setup to automatically regenerate itself once an operation is executed.

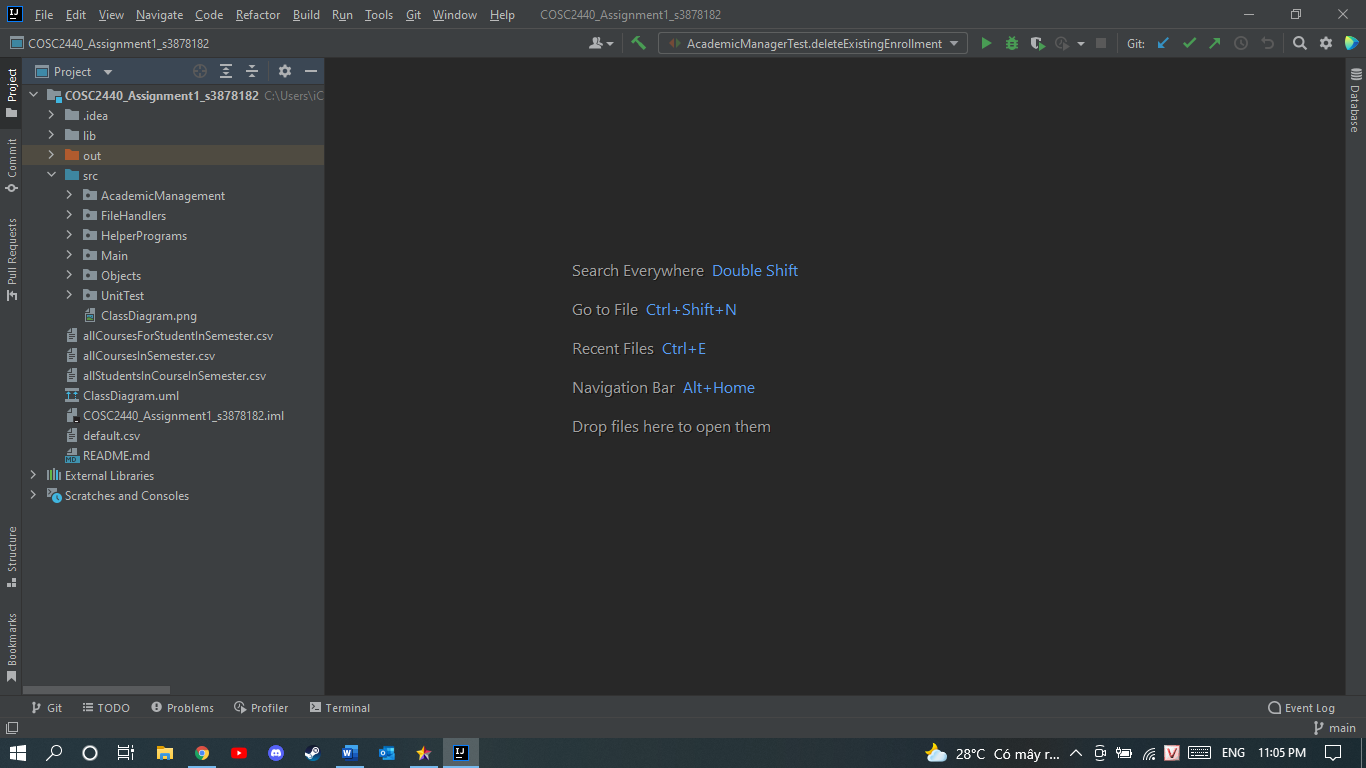


As illustrated in the screenshot above, the menu is quite well-organized via a table format (manually printed using regular symbols), separating the welcome messages and the choice prompts into 2 distinct sections. The ensuing paragraphs will provide a more in-depth explanation of the working mechanism behind this system as well as decomposing the architecture of this program to understand the packages, associating classes and their relationships with one another.

**2/Code analysis & explanation:**

1. **Naming the system:**

Despite the fact that this assignment does not compel students to come up with an official name for their system, I accidentally came across this fantastic idea for a project name, since its fits the assessment details on Canvas while having a memorable acronym. Having said that, I have decided to call my Java Console App as “Java Enrollment Management System” or simply abbreviated as “J.E.M.S”. The welcome menu also shows this name to present itself to the administrator (or whoever may be authorized to use the system).

1. **Package structure:**

My approach towards this assignment was to quite literally “divide and conquer”, which led to this specific style of package organization. Each of the packages here serves its own unique purpose, and those functionalities are as follows:

+ Academic Management: this package contains the main interface (Student Enrollment Manager) along with its implementing class (Academic Manager). Together, these 2 files work in conjunction to perform all the methods that are displayed on the main menu, by importing objects, file handlers and manipulating them in such a way that satisfies user requirements. To further elaborate, methods like add, delete, get one, get all, and all data filtering/retrieving methods are all located in this package for the main actor (an Academic Manager) to supervise his or her students/courses/enrollment status.

+ File Handlers: this package works most directly to the aforementioned one, as it contains an indispensable procedure of the entire working process: populating data. After gathering all necessary objects from the “Objects” package, this package will begin redistributing these data into its respective array lists for storage (that is the reason why there are 3 different file handlers to cater for each type of object) and provide implementations for the important methods in this assignment. As a result, the “Academic Management” package will only need to call these completed methods for everything to run seamlessly.

+ Objects: as the name implies, all attributes and conversion methods for objects are encompassed in this convenient file package. The 3 primary object types for this project are students, courses and enrollments, so it is common OOP technique to delegate them into separate files within the same package to avoid code duplication and clashing. Additionally, the “File Handlers” package needs to import this package into theirs before commencing any tasks, since “Objects” is basically the blueprint of what that object will appear as.

+ Main, Helper Programs & Unit Test: the 2 remaining packages may be less prioritized in importance, but absolutely necessary to the success of this project. “Main” is just the place to initialize a new “Academic Manager” and “Menu Display” object to start the whole program from the beginning, while Helper Program contains the CSV Reader class used for interpreting the default.csv file (the provided database).

1. **Procedure of functionality:**

The previous section of this document has briefly explained the relationships and inner mechanism of my system, and now I will provide detailed phases of which this console app will work upon.

Phase 1: Reading CSV file and populating all data

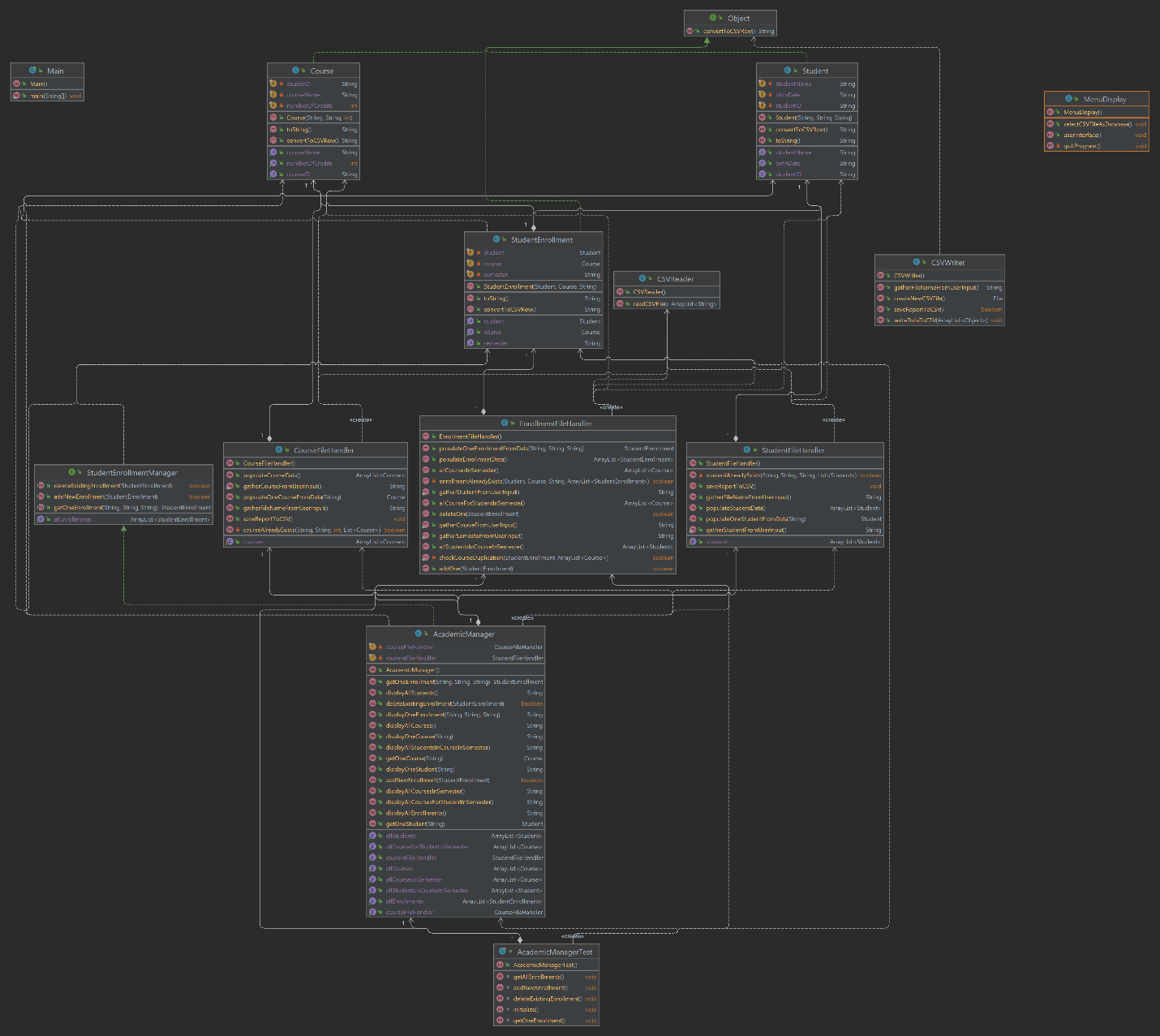
The moment a user chooses to run this program on an IDE, the “Academic Manager” class will be the first to instantiate, calling all necessary methods all across the architecture of the system, one of which will be the “CSV Reader” class responsible for reading the database file (default.csv) line by line before the menu even appears. After that, the populate methods will be triggered to take the stored array list of csv strings from the “CSV Reader” to populate data in the three established object types, ready for queries from the user.

Phase 2: Showing menu interface and gathering user choices

The “Menu Display” class will instantiate. This class will function both simultaneously and independently, dealing with the front-end aspects (displaying menu in table format, showing prompts, taking inputs) and processing the inputs that user make and perform the appropriate methods (1 of the 11 options in the main menu).

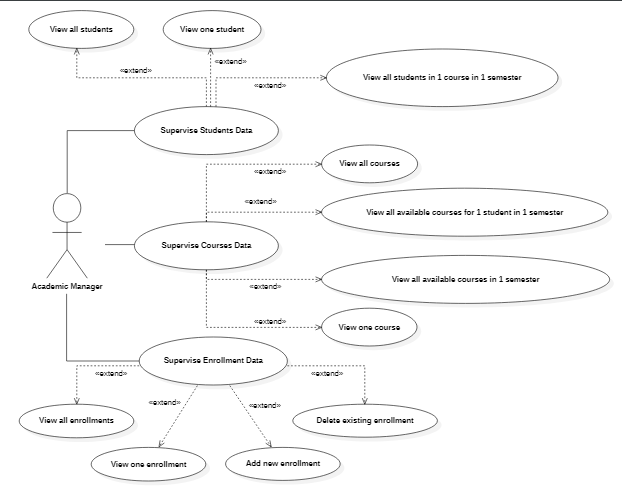
Phase 3: The actual methods will run according to the submitted option

Once the user has made a decision on the task which they would like to perform, the “Menu Display” class will simply call the preset method in that case (in the switch case explained above) and if the entire process in finished (the user has received the outcome that they want), the system will automatically restart itself for another session, so users can remove the inefficiency of having to click “run” after a session has concluded. The only exceptions lie in cases 9-11, because once users have viewed the data that the console app returns, then they will be asked whether or not to save the report (if they select “yes”, the system will immediately write the recently queried data into a new csv file named by the users themselves). This ensures personalization and after this extra step is followed through, the system will obviously start a new session.

**3/ Class diagram:**

This image illustrates my class diagram sketch for this complicated console application, showing all connections, inheritance, implementations and attributes pertaining to all respective classes created in each package. Should you have any issue viewing this screenshot due to blurriness, I will also include a link to my GitHub, where you can find all folders and files of my project.

**4/Use Case Diagram:**

****

**5/Instructions (for instructor):**

To see all files of this program properly, click the following link to gain access to my GitHub repository for this assignment (all files can be seen there, including actual diagram files and this technical report).

To run this program, kindly go to the src directory and run on the Main.java file, as it is where all Java programs are supposed to run on.

Link to GitHub repo: <<https://github.com/KaydenGiang2512/COSC2440_Assignment1_s3878182.git> >