Musa Nishat
Data Structures
Professor Bari
19 February 2019

# Course Registration System: Design and Implementation

My course registration system culminates in the CourseManagementApp class, but there are a number of classes that have been created and used that function as a basis for the system. These classes function as abstract data types since the person writing the CourseManagementApp does not need to understand how the methods in these other classes work, but simply needs to implement them. These classes are User, Admin, Student, Course, and allCourses.

The Course class contains all the attributes of a course, including its name, ID, instructer, max students, an array list of students, etc, and contains methods to interact with the individual course including adding and deleting students, printing information, and comparing its attributes to other courses given as parameters. This class implements the comparable interface so the compareTo() method is overridden from that interface so that courses can be compared to each other using the number of students in them. An array list of Course objects is created in the allCourses class. In this class, methods are created that allow courses to interact as a whole, allowing one to input a course ID as a parameter and then being able to interact with that course and its methods and includes methods to add or delete a student to or from a specific course in the list, print the whole list, print only the full course, etc. The CourseManagementApp creates an instance of the allCourses class and imports the list of courses from the csv file of courses. These courses added to the allCourses object and it is serialized alongside a list of Admins and Students that are currently registered in the system, and once deserialized, this allows the current user of the app to log in by having their inputs checked against the list. The methods from allCourses are finally used in the App to complete the required functions.

The Student and the Admin classes both have their own interfaces that define their methods and they both inherit from the abstract User class, which contains a first and last name, a username, and a password. The student also contains an array list of courses. The setPassword() and setUsername() methods are overloaded in the Admin class from their use in the User class to allow the current user to input the new username or password rather than having it as a parameter of the function. Because we don't want a student user to be able to have the same privileges as an admin, the data fields have been encapsulated and the student user is limited to setting their password, viewing classes, and joining or withdrawing from them while an admin will have fewer limitations but still will not have direct access to data types.

## User Guide

1. Run the CourseManagementApp class
2. If this is your first time running the program, the information will automatically be imported from the csv file and serialized
3. Once the data is imported, there is only one user at the start. Username: Admin, Password: Admin001. Use these credentials to log in and use any function the admin is allowed to use
4. If you create a new student, you must exit the program to log in. When the student is created their username is given and you may use this to login as them, create a password, and use any functions a student is allowed to use.
5. IMPORTANT. Whenever you are finished using the program you must manually save and exit using the options menu or your data may be lost.