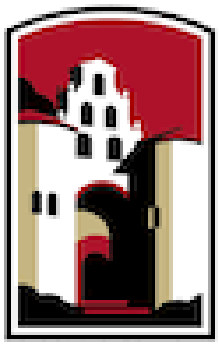
Software Design Specification
Class Roster Software
Version 2.0



SAN DIEGO STATE UNIVERSITY

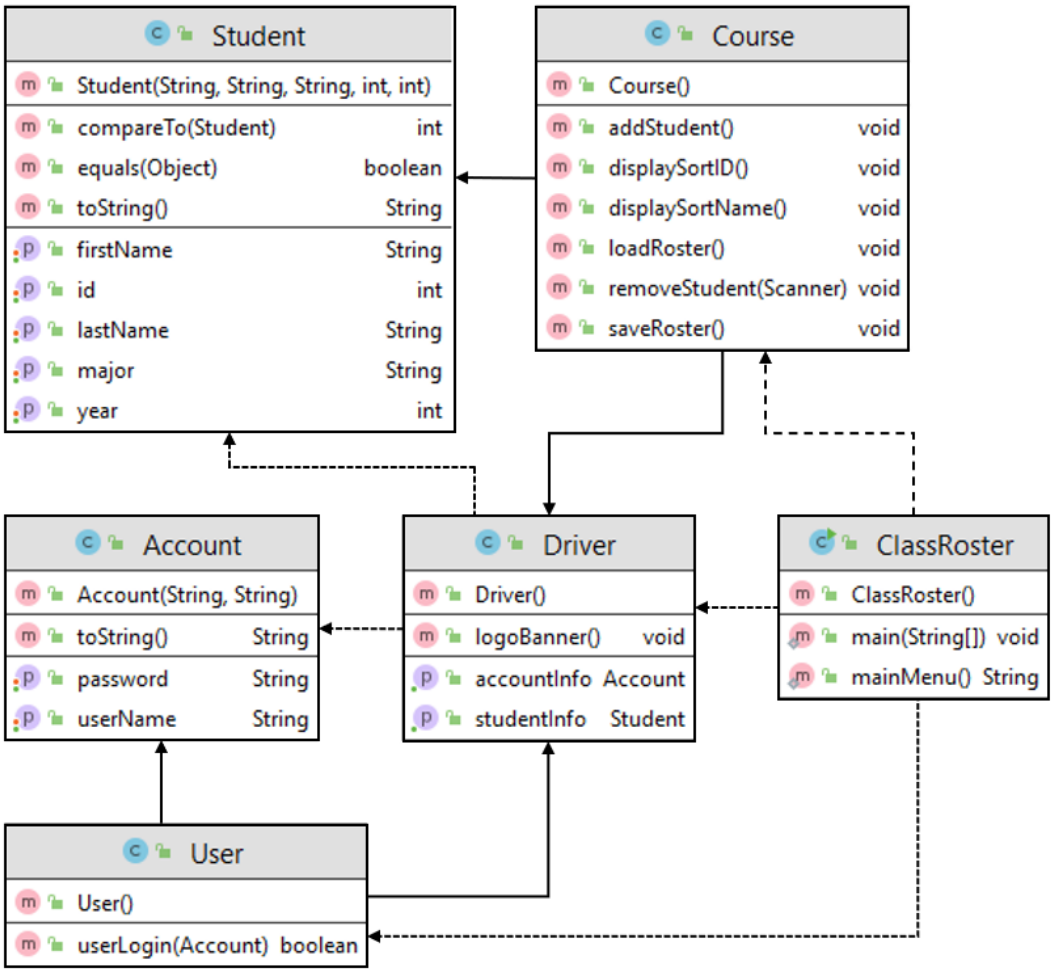
Team
Amiel Nava
Anoodnya Sangam
Shane Moro

March 12, 2021
CS-496 Intro Software Systems

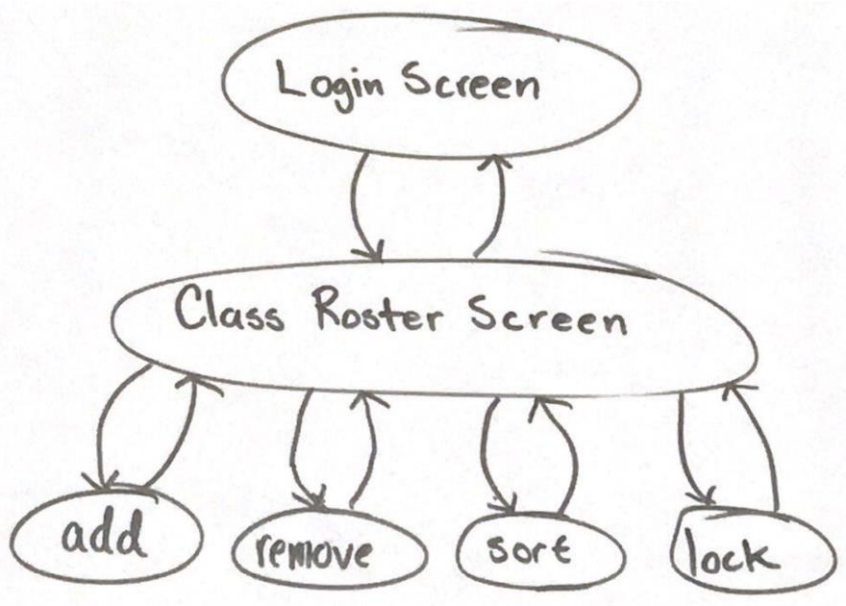
Table of Contents

1.0 Software Architecture Overview

1.1 Architectural Diagram



1.2 Top Level Description



The flow of the program is described on the above state diagram. First, the user will be prompted with a login screen when they start up the application. The login screen will prompt for a username and password. After a successful login, the application will go directly to the class roster screen which will show the roster with the default sort view along with a menu on the bottom or on the top listing the available functions to the user. Those functions are adding a new student, removing an existing student, sorting the class roster by name or id, and locking the class roster. Each of these functions will display instructions and will have the option to cancel the function and go back to viewing the class roster. Also, each of these functions would display an error if the user entered an incorrect input. Lastly the class roster screen or in our case the display of the class roster, will have the option to exit the program and go to the login screen.

The driver class will be in charge for the flow of the program using the other classes such as user student, course, and account. These classes will be used to use to let the user edit their class roster and display the class roster.

1.3 Description of Classes

1.3.1 Driver

- Attributes:
    - N/A.
- Operations:
    - Login: This method takes in a User object, and calls the verify method in the Account class, and returns a boolean true if the login was successful, and false otherwise.
    - addStudent: This method takes in identifying information of the student, and calls the addStudent method in the Course class, to add the student to the class roster. It returns true if successful, false if not.
    - removeStudent: This method takes in the ID of an existing student, and calls the removeStudent method from the Course class to remove the student from the class roster.
    - listStudent: This method has no parameters, but will prompt user to choose if they want to sort by name or by ID, and then return a List of Student objects.
    - setRosterStatus: This method calls the setLocked method in the Course class, which allows the user to lock and unlock the roster. Returns true if successful, false otherwise.
    - fileManagement: This method will save the current class roster into a file, so that it will be accessible in between sessions.
- Interactions:

- Driver class is dependent on User class and Course class. User class contains user info and passwords that Driver uses during login. Course class contains the list of students that Driver class interacts with by add/remove students, list students, or save to a file.

1.3.2. User

- Attributes:
  - userName: This string type is used to identify a user who has an account to use the system.
  - password: This string type is used to validate a user into a session in the system to view and edit a class roster.
- Operations:
  - User: This is a constructor for the User class, that creates an account with a user and password that allows the user to login in to the system.
  - changePSWD: This method takes in a string, their new password, and changes their current password to the new string provided.
- Interactions:
  - User class interacts with Account class and Driver class. User class is dependent on Account class. Driver class is dependent on User class. Driver class uses User class to store usernames and passwords.

1.3.3. Account

- Attributes:

- user: A User class type that is used to specify a particular user to an account.

- Operations:
  - verify: This method takes in strings, the username and password that the user entered, and compares it with the current password on their Account. If it matches, it returns a true, and returns false otherwise.
- Interactions:
  - This class interacts with the User class. The User class is dependent on the Account class to verify username and password.

1.3.4. Course

- Attributes:
  - students: A list of student class types. This list is used to specify the list of students in a particular course.
  - className: A string type containing the name of the current course or class name.
  - teacher: A string type containing the name of the professor or teacher of the course.
  - MAX_STUDENTS: A constant integer type that will determine the maximum number of students allowed in the current course. This constant will be useful to automatically lock the class roster.
  - lock: A boolean type used to determine if the class roster is lock (true) or not (false).
- Operations:

- addStudent: This method takes in a Student object, and adds them to the course roster.
- removeStudent: This method takes in the ID of the student and removes them from the course roster.
- sortName: This method sorts the students in the class roster (which is a List<Student>), and returns a List<Student> that is sorted alphabetically by last name.
- sortID: This method sorts the students in the class roster (which is a List<Student>), and returns a List<Student> that is sorted numerically by ID number.
- setLock: This method locks the roster so that no more students can be added to it. Returns true if successful, false if not.

- Interactions:
  - Course class interacts with Driver class and Student class. Driver class is dependent on Course class for course information and student list. Course class is associated with Student class. Course class contains list of Students.

1.3.5 Student

- Attributes
  - firstName: A string type that has the first name of a student.
  - lastName: A string type that has the last name of a student.

    - ID: An integer type that identifies a unique student. It is used to identify a student even if the student has the same name as another student. The unique id is assigned by the application.
    - major: A string type that will contain the student's major.
    - year: An integer type that will contain the student's expected graduation year.
- Operations:
    - Student: This is the constructor for the Student class. It takes in a string first name, string last name, int ID, string major, and int year, and creates a Student object that represents a physical student.
- Interactions:
    - Student class is associated with the Course class. The Course class contains list of students.

2.0 Development Plan and Timeline

2.1 Partitioning of Tasks

- Anoodnya: Will oversee displaying the class roster along with the available functions to the user.
- Amiel: Will oversee implementing the functions that will edit a class roster and read and write the state of the class roster in a csv file. Amiel will also make sure the information is persisted between sessions.
- Shane: In charge of the user and account classes that will validate a user. Those classes will let a user access the class roster. Shane will be setting up the other classes.

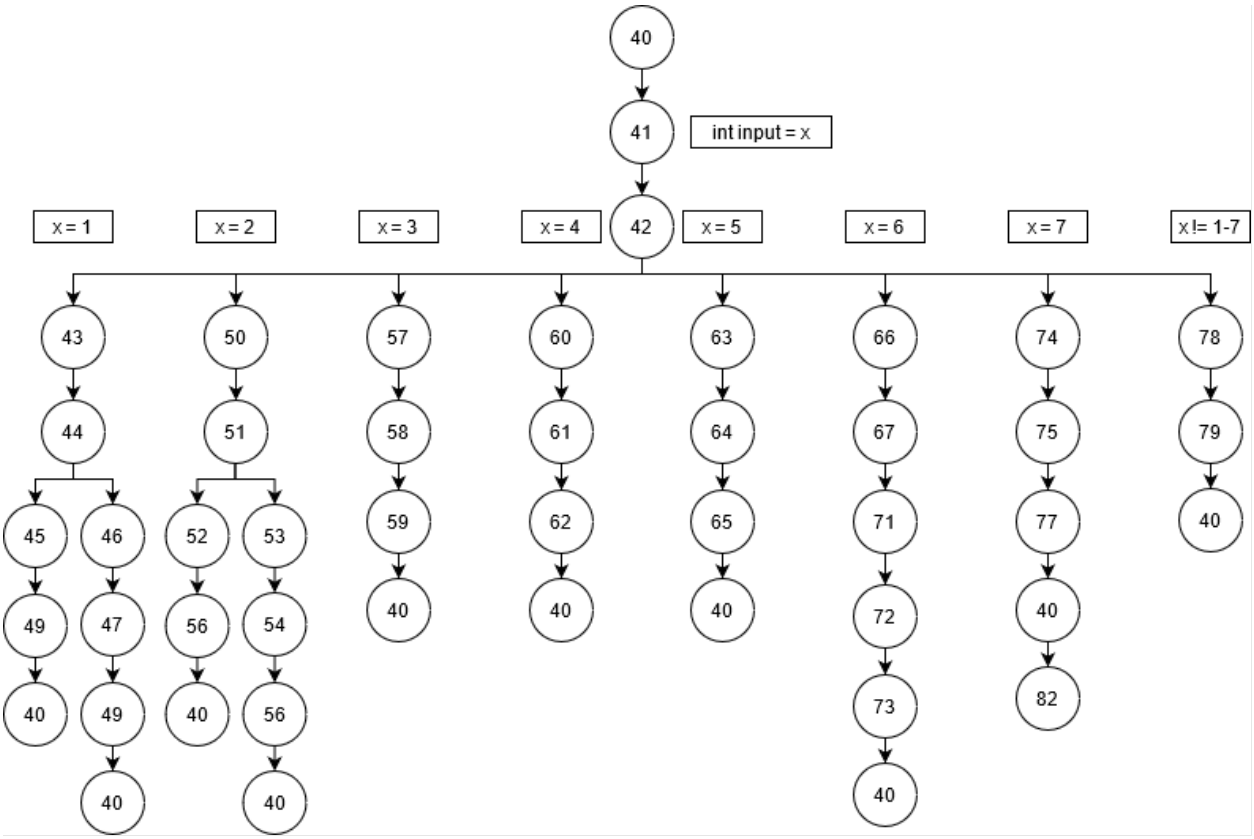
2.2 Team Member Responsibilities

- Shane will be responsible for setting up group meetings and getting feedback from the stakeholder (office hours).
- Amiel will be responsible for debugging the system and reviewing the software or documentation.
- Anoodnya will be responsible for integrating each developers' assigned class into project.

3.0 Verification and Validation Test Plan

3.1 Unit Testing

- Unit testing will be performed on each function and control flow statement. Functions will be tested to ensure output matches the expected output. Control flow statements will be tested to ensure proper execution order of statements.
- Control flow statement test example shown below details the proper statement path for the statements in the while loop located on line 40 of ClassRoster.java. Tester will input the following integers [1, 2, 3, 4, 5, 6, 7, 8] when prompted by the main menu and ensure that the following statement paths are executed. Statements are designated with their corresponding line number. Lines containing only brackets are excluded.



3.2 Integration Testing

- Integration testing will be performed between the java program and the database to ensure that database modification is executing correctly. All functions that modify the database will be called followed by display database function in order to make sure modifications were made correctly.
- Integration testing will be performed as shown in the example below.
    - Call display database function.
        - Item three from Main Menu.
    - Modify database record.
        - Item one from Main Menu.
    - Call display database function.

        - Item three from Main Menu.
    - Note changes between first display call and second.

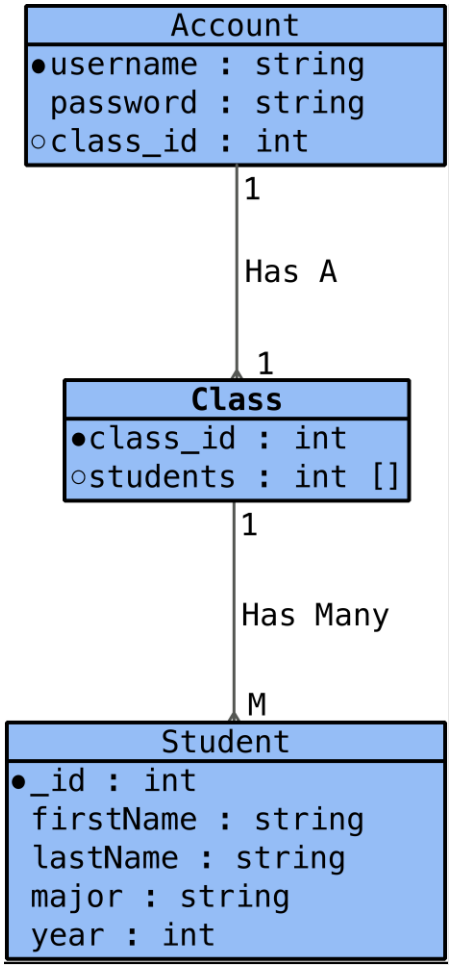
4.0 Data Management Strategy

4.1 Database Design

Due to the fast implementation of our system and the simple data the system will be storing, the Class Roster system will be using the NoSQL database system. Another reason for using this database design is that the data the system will interact with will primarily be for reading the data, since most users will only need it to look who is in their class and will only update the class roster the first two weeks of class or whenever the timespan is before the registration deadline.

The NoSQL database system that the Class Roster system will use will be a Document Database. The Class Roster system will be using this type of NoSQL database due to MongoDB being the most popular NoSQL database and its simple database interaction in Java. This type of database stores documents in BSON format which is storing data in JSON format but in binary. This format is widely available and is simple for a system administrator to edit the database if needed.

4.2 Database Implementation



Our database implementation will consist of three collections: accounts, classes, and students.

4.2.1 Account Collection

The collection of accounts will be used to store authorized users(professors) for the Class Roster system. These accounts will also be used to connect to the database and retrieve their class information. The account unique identifier will be the username; hence every username will be

unique. The account has a password that will authorize the use of the database. The last attribute of this collection is the class_id for which the user will have to access their class roster.

4.2.2 Class Collection

The collection of classes is used to collect a list of classes along with a collection of students. Each document will have a class_id, which will be unique to that class. The second and last attribute has students which will be an array of student's ids that belong to that class.

4.2.3 Student Collection

The collection of students holds a list of all students in the school. Each student has a unique school id number(_id). The second attribute is the student's first name. The third attribute is the student's last name. The fourth attribute will be the student's major. Finally, the student's last attribute will be student's expected graduation year.

4.3 Data Components

The data for this system will keep a professor's class roster which consists of a list of students. Each student will have a unique id, first name, last name, major, and year of expected graduation.