**Student Class inherits User**

Attributes

* Username: Student
* Password: Student001
* List of Course objects

Methods

* View all courses that are not full
  + If course.isNotFull() is true
* Register in a class
  + Add student to course
  + Add course to student
* Withdraw from a class
  + Remove student from course
  + Remove course from student
* View registered class 🡪 print list of courses

**User Class**

Attributes

* Username
* Password
* First name
* Last name
* keepGoing (to continue the loop)

Methods (that both Student and Admin share)

* View all courses
* exit(): return false to end loop

**Course Class**

Attributes

* Course name
* ID
* Maximum number of students
* Current number of students
* Instructor name
* Section number
* Location
* List of students registered

Methods

* isNotFull(): return true if current < max
* compare to method 🡪 so that arraylist of courses can be sorted.

**CRS (Main Method) Class**

1. De-serialize list of students and courses
2. log-in
   1. Student 🡪 check if student is a registered student 🡪 student methods
   2. Admin 🡪 Admin methods
   3. Loop until user chooses to exit
3. Serialize list of students and courses

CRS Assumptions and Instructions

**Admin Class inherits User**

Attributes

* Username: Admin
* Password: Admin001

Methods

* View all courses 🡪 override parent method
* View course that are full
  + If course.isNotFull() is false
* Write a file
  + For each line write name of course for which isNotFul() is false
* View all students in a course
  + Get list of students from course
* View courses for a student
  + Get list of courses from student
* Sort courses
  + Get number of current students for each course and compare
* Create course
  + Instantiate new Course object and add to system
* Delete course
  + Remove course from system
  + Remove course from all student’s courses
* Edit course
  + Set new values to instance variables of a Course object
* Display information for a course
  + Print out all values of course’s attributes
* Register a student
  + Instantiate a new student and add to system
* A student must be registered by an admin first - otherwise the system will assume that he/she is an unauthorized user and the system will be exited
* All admins have the same username and password
* All students log-in using the same username and password; once they’ve successfully logged in, they must specify their first and last names to continue.
* Students only have first and last names
* A student may register into as many classes as he/she wants
* If an admin deletes a course from the system, students (who were registered in that course) are automatically withdrawn from the course.
* User may choose actions to take from the menu by entering the appropriate option number – I’m assuming the user always enters acceptable values as inputs.
* Student username: Student, password: Student001
* Admin username: Admin, password: Admin001

Definitions

* **Method overloading**: there are overloaded constructors in the Admin, Student, and Course classes
* **Method overriding**: the Admin class overrides viewAllCourses method from the User class, the methods declared in the interfaces are also overridden in Student and Admin classes.
* **Abstract class**: contain abstract methods and can also have non-abstract methods. It cannot implement methods and can only declare them. Subclasses inherit from the abstract class.
* **Inheritance**: Admin and Student inherit User class, inheriting User’s methods, and attributes.
* **Polymorphism**: when a variable reference of a parent class can be used to reference a child class instance.
* **Encapsulation**: when attributes of an object of a class is private so that it cannot be manipulated outside of the class; the attributes are only accessible through getter and setter methods.
* **ADT**: the idea that methods defined within a class performs complex functions and can be called outside the class if an object of the class should want to perform those actions – like a black box. The code of the method being executed is ‘hidden.’