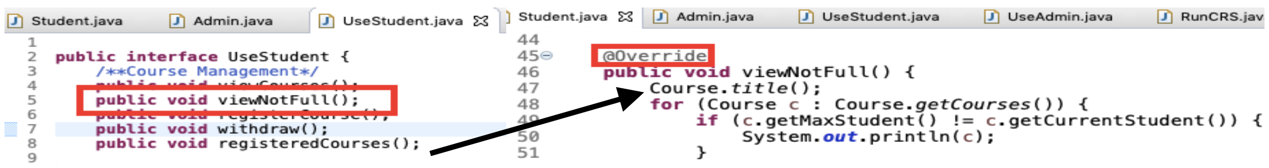
- Method Overloading
  - This happens when there are methods with different argument lists but same name.
- Method Overriding
  - annotation @Overrride
  - Ex: There are abstract methods declared in the abstract superclass/type or interface, so the subclass/type will need to implement those methods with overriding.The viewNotFull( ) method in UseStudent interface was implemented by Student Class via Inheritance. Showing below:

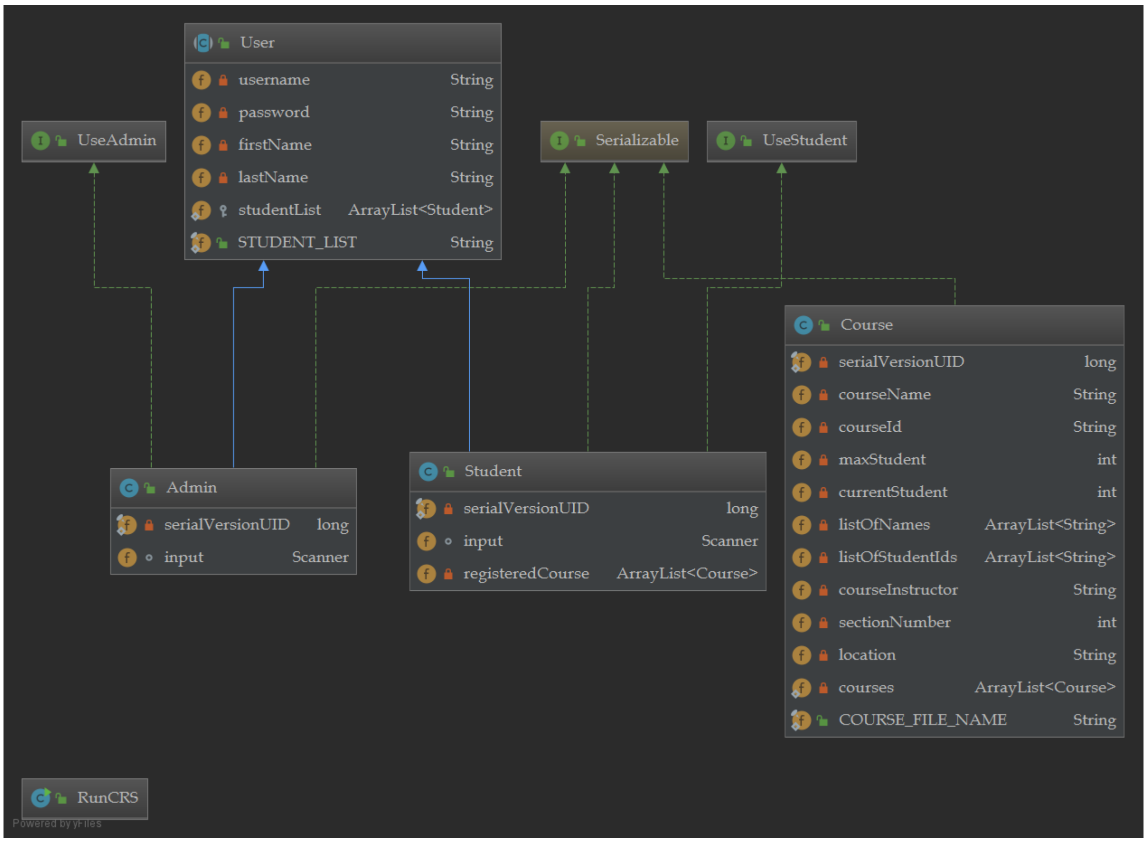


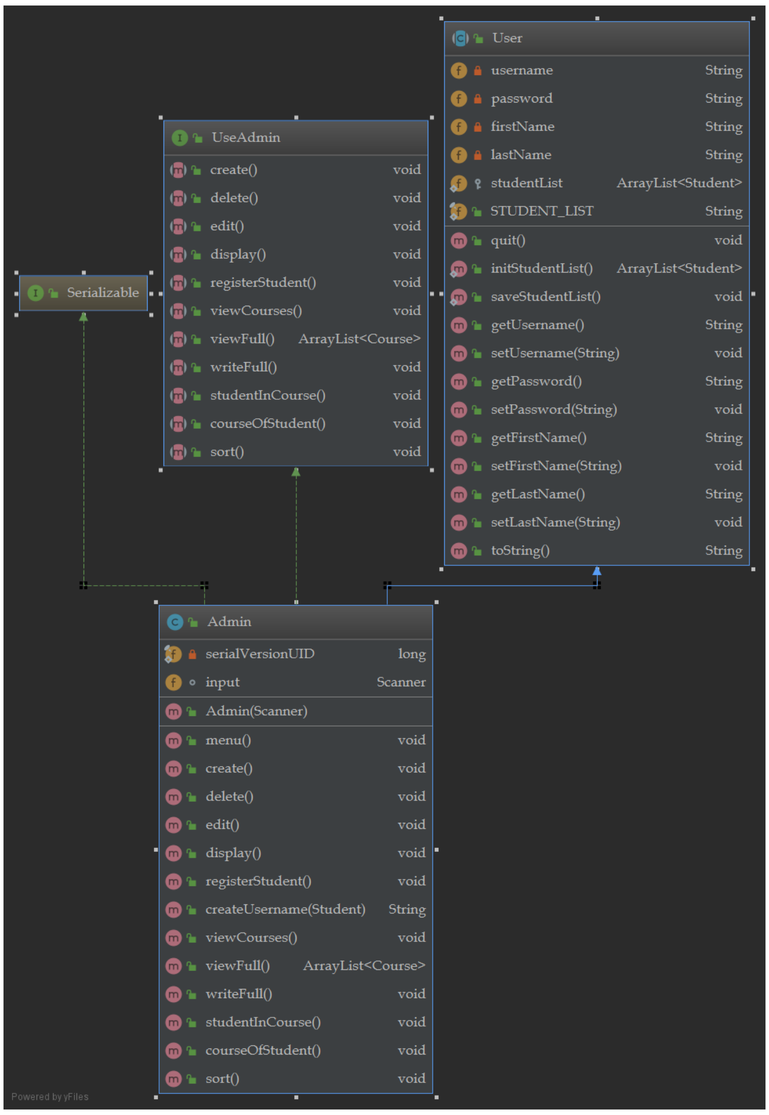
```

public interface UseStudent {
    /**Course Management*/
    public void viewCourses();
    public void viewNotFull();
    public void registerCourse();
    public void withdraw();
    public void registeredCourses();

```

```

    @Override
    public void viewNotFull() {
        Course.title();
        for (Course c : Course.getCourses()) {
            if (c.getMaxStudent() != c.getCurrentStudent()) {
                System.out.println(c);
            }
```

  - Ex:In the Course Class, I rewrite the default toString( ) method of the Course Object in order to adjust the output formats. The overriding method would be called every time I am going to print a Course Object, which gives out the desired spacing and lining of output.
- Abstract Class
  - The class User is an abstract class. It cannot be instantiated but can be subclassed so that the abstract methods can be instanced and non-abstract methods can be used by non-abstract subclasses, like Student and Admin. This restricts the inheritance hierarchy in order to full fill that User must be either Admin or Student.
- Inheritance
  - Ex: the Student Class and Admin Class both inherit from the User Class so that they can access the non-instance fields and methods in the User Class. I put the fields and methods that are needed by both Student and Admin in the User, and for instance, I could get the username of the Student or Admin defined in User Class by calling getUsername().
  - Student Class implements the UseStudent Interface, and Admin Class implements the UseAdmin Interface.
- Polymorphism
  - This happens due to the inheritance. For example, in the edit() in Admin I used Set<Integer> set = new HashSet<Integer>(), where Set is the declared type and HashSet is the actual type. Methods called by the object set must the signature in actual type Set so that there will be no compilation error.
- Encapsulation
  - The modifier "private" is a kind of encapsulation. Private fields can be only obtained by public setter and getter.
- The concept of ADT (Abstract Data Types)
  - The ArrayList is an implementation of the ADT List.
- To avoid reconstructing new scanners in each class, I passed in the Scanner defined in the main of RunCRS by reference into the constructors of Admin and Student Class. As Scanner is not serializable, I added a "transient" modifier to the Scanner variable in Student Class so that it wouldn't be considered during serialization and cause trouble.

HW1 Non-functional Requirements


User
username String
password String
firstName String
lastName String
studentList ArrayList<Student>
STUDENT_LIST String
UseAdmin
Serializable
UseStudent
Course
serialVersionUID long
courseName String
courseId String
maxStudent int
currentStudent int
listOfNames ArrayList<String>
listOfStudentIds ArrayList<String>
courseInstructor String
sectionNumber int
location String
courses ArrayList<Course>
COURSE_FILE_NAME String
Admin
serialVersionUID long
input Scanner
Student
serialVersionUID long
input Scanner
registeredCourse ArrayList<Course>
RunCRS
Powered by yFiles


User
username String
password String
firstName String
lastName String
studentList ArrayList<Student>
STUDENT_LIST String
quit() void
initStudentList() ArrayList<Student>
saveStudentList() void
getUsername() String
setUsername(String) void
getPassword() String
setPassword(String) void
getFirstName() String
setFirstName(String) void
getLastName() String
setLastName(String) void
toString() String
UseAdmin
create() void
delete() void
edit() void
display() void
registerStudent() void
viewCourses() void
viewFull() ArrayList<Course>
writeFull() void
studentInCourse() void
courseOfStudent() void
sort() void
Serializable
Admin
serialVersionUID long
input Scanner
Admin(Scanner)
menu() void
create() void
delete() void
edit() void
display() void
registerStudent() void
createUsername(Student) String
viewCourses() void
viewFull() ArrayList<Course>
writeFull() void
studentInCourse() void
courseOfStudent() void
sort() void
Powered by yFiles