**Course Registration System Design:**

For the Large part the design of my Course Registration works through Scanner prompts and static void methods to use recursion on the menus.

The system deserializes the stored information at the beginning, if there is no stored information it reads the csv file and serializes the information and then deserializes it to be stored into an Object.

I store the lists of students and courses in one object class called StoredLists that is passed over to all the methods so they can be used throughout all the menus.

The Scanner object is also passed over through all the methods.

At all the menus submitting ‘exit’ allows you to return to the prior menu.

When you submit ‘exit’ at the first menu it serializes and effectively saves the work you’ve done.

**I’ve Implemented:**

**Method overloading**

I added 2 different implementations of addCourse() one would just take the course and put it on the top of the list, the other would take the course and an integer to specify where in the ArrayList it would be added to.

**Method overriding**

For Both Student and Admin I overrided the getter and setter classes for username and password. For Admin it was to ensure that the username and password stayed Admin and Admin001 respectively, but for Student it was purely a product of paranoia to ensure that it functioned.

**Abstract Class**

User is an abstract Class

**Inheritance**

Student and Admin inherit the username, password, first name and lastName variables and associated methods from User

**Polymorphism**

AddToCourse() from Admin vs addCourse() in StoredLists would’ve been an example if they shared the same name.

The setPassword() method in Admin vs the setPassword() method in User.

**Encapsulation**

Storage is basically this one massive container for all the relevant information to pull from.

UML Diagram and Flowchart will be shared as images in the submission



