***Course Registration and Scheduling System***  
Karina Cuadrado  
SWENG 311  
April 27, 2018

**Introduction**

This application manages a course registration and scheduling system. It keeps track of the rooms, courses and students added as well as what courses are available in a day and the students assigned to them. It also gives the user the option to delete a room, course or student given that they have not being assigned to anything else. The courses are an hour long and fall in traditional 9-5 hours. The system is represented through a GUI and gives the user the ability to save and load the information entered/stored.

**Requirements**

There are three major components to this application:  
1- Adding objects  
2- Removing objects  
3- Displaying objects

The first step is to enter the maximum number of rooms for the system. Next, a room must be added by entering its number ID. At this time the user may either add more rooms, add students or add courses. To add courses they must enter its name, select the room and time from the respective combo box. To add students they must enter their first name, last name, phone area code and the following 7 digits of the phone number. As these objects are added, they are displayed in JLists. Now, the user can add students to courses. The application will display conflicts through dialogs. Some examples are: invalid phone numbers, repeated objects, unavailability of room at time specified, etc. This concludes the adding portion of this application.

Removing objects is as simple as selecting the item on the list and clicking the “remove” button. However, conflicts must be avoided in order to successfully complete the deletion. For example, assuming a student is assigned to a course, in order to remove the room this course is assigned to you must first remove the student from the course, delete the course and then remove the room. Otherwise, the application will present a dialog frame with the appropriate error message.

Displaying functionality depends on the object. To display the courses attended by a student, you select the student from the combo box. To display the courses in a room, you select it from the list. To display the courses at a given time, the time is selected through the combo box.

The way all these functionalities are executed are by keeping track of the selected indices in combo boxes and lists. The action listeners of all these components call the functions from the control class, which has ArrayLists of each object.

**Design**

**a) Description of classes**

1. Student – contains all the information of a student. This includes their first name, last name and phone number.

2. Course – contains all the information of a course. This includes the course name, room assigned to, time and an ArrayList of students assigned to the course.

3. Room – contains the information of a room. This includes its number ID, an ArrayList of courses assigned to it and an array that keeps track of the times the room is busy.

4. CRSS – provides all the functionality of the system. It contains methods to add the objects, delete the objects, assign objects to others and display objects. Instances of the Student, Course and Room classes are created and added to the appropriate ArrayLists. It contains variables to have the max number of rooms and ArrayLists of objects to know what’s on the system.

5. GUI – contains the necessary components to construct a graphical user interface. This includes instances of CRSS, JLists, JButtons, JComboBoxes, JTabbedPane, etc. Within the action listeners of the components, the necessary methods of the CRSS are called to complete the functionality. For example, to delete a student the method DeleteStudent is called on the instance of CRSS in the Delete Student button’s action listener. In this listener the JLists are also updated.

All the classes provide the necessary setting, getting and printing methods.

**b) Class diagram**

*See attached CRSSUML.pdf*

**Challenges**

Most of the issues I ran into were similar to those run into in lab. For example, I often had the issue that if I added a course to one room at a certain time I could not add another course at the same time in another room. It would tell me it was occupied at the time. It turns out this was because I created the objects using the same ArrayList. All the objects referenced the same object. Fortunately, I was able to fix it by creating a new object of the specific ArrayList every time a new room, course or student was added.  
 Another issue I ran into was coming up with a friendly User Interface. It was a very new experience to me. I had to see examples to familiarize myself with a good system.

**Shortcomings**

I believe I was able to fix all the bugs of my program. There might be some issues with exceeding array indices for some test cases. This is because I had to add an if statement to check if the selection was empty or not. I might’ve missed specific cases. One thing I wish I would have been able to implement better is the updating of the lists and combo boxes. I had to copy and paste a lot of this code and could not implement a reusable way. This is something I hope to fix with practice.

**Conclusion and Future work**

Overall, this application is a fully functioning course registration and scheduling system. There could be some improvements like the ability to modify the entered information without having to delete the objects and a less confusing way of adding students to courses and removing them. At the moment, if nothing is selected and the buttons are pressed nothing happens and it be a little confusing for first time users. Other than that, it is a well-rounded fault-tolerant system with complete functionality.