Design a Course Registration Platform

Students can

1. Login using Student Id and Password
2. Select semester and program
3. See list of courses
4. Register for courses or add to waitlist
5. Drop courses

```
Class Student
State:  name, studentId, password, email, phoneNumber, studyPlan
Behavior: login () {
                CourseRegistrationWebsite.login(studentId, password);
        }
Behavior: createStudyPlan() {
                SemesterStudyPlan studyPlan =  new SemesterStudyPlan;
        }
Behavior: addCourse() {
                CourseRegistrationWebsite.registerCourse(this.studyPlan, course);
        }
Behavior: dropCourse() {
                CourseRegistrationWebsite.dropCourse(this.studyPlan, course);
        }
Behavior: viewCourses() {
                CourseRegistrationWebsite.listCourses(semester, program);
        }
```

```
Class Course
Data: courseId, professor, courseDuration , prerequisites, courseDescription, creditHours,
numofStudents, maxNumberofStudents, List<Student> waitList

Behavior: isCourseAvailable(){
                If (this.numOfStudents < maxNumberOfStudents) {
                        return TRUE;
                } else {

                        Return FALSE;
                }
        }
```

```
Class CourseRegistrationWebsite
Data: List<StudentId> studentIds, List<Course> courses, List<Semester> semesters, List<Program>
programs
Behavior: login (StudentId, Password) {
        If (Router.IsConnectedToInternet() == FALSE) {
            // Return Login failure
        }
        If (studentId in List of studentIds) {
            If (password match) {
                // Login Successful
            }
        }
        // Login failed
    }
Behavior: registerCourse(studyPlan, course) {
        If (course.isCourseAvailable()) {
            Studyplan.AddCourseToStudyPlan(course);
        } else {
            course.waitList.add(student);
            Studyplan.AddCourseToWaitList(course);
        }
    }
Behavior: dropCourse(studyPlan, courseId){
        Studyplan.DeleteCourseFromStudyPlan(course);
    }
Behavior: listCourses(semester, program) {
        Foreach (sem in semesters) {
            If (sem == semester) {
                Foreach (prog in programs) {
                    If (prog == program ) {
                        //Print prog.courses
                    }
                }
            }
        }
    }
```

```
Class Router
Data : iSPname, model,
Behavior : isConnectedToInternet() {
```

```
            // return TRUE if connected to Internet
        }
```

```
Class SemesterStudyPlan
Data: List<Course>Courses, totalCredits, isSufficientCredits, studentId, List<Course> waitListCourses
Behavior: addCourseToStudyPlan(course) {
            If (course not in this.courses)
                this.courses.add(course);
            }
        }
Behavior: deleteCourseFromStudyPlan(course) {
            If (course in this.courses) {
                Courses.delete(course);
            }
        }
Behavior: addCourseToWaitList(course) {
            waitListCourses.add(course);
        }
Behavior: updateCredits(increment) {
            If (increment) {
                this.totalCredits += 4;
                if (this.totalCredits >= 8) {
                    isSufficientCredits = TRUE;
                }
            } else {
                this.totalCredits -= 4;
            }
        }
```