```
Problem : Design a course registration platform
Thing(Object)/Entity:
        Information(Data)/(State):
        Services(Behaviors)/Actions
Thing:
        Student:
                Data:name, id, registerredCourse(collection of course)
                Behaviors: addToRegisterredCourse,
removeFromRegisteredCourse
        Course:
                Data:name, courseid, prerequisite, belongTerms,
availableSpace, courseInfo
                Behaviors: isOpen()
        CourseSearchService:
                Data:
                Behaviors: search{searchByCourseId(), searchByName()}
        CourseRegisterService:
                Data:
                Behaviors:registerCourse()
        LoginToPlatformAuthService
Sequence of invoking behaviors on objects:
        Student student
        Course course
        CourseSearchService searchService
        CourseRegisterService registerService

        student.LoginToPlatformAuthService -> plaform: authorize

        if (authorize is true):
                searchService.searchByCourseId() -> courseId: course
                if (course.isOpen() && course.availableSpace() >= 1)
                        if (student.finishedCourse()->
course.prerequisite)
                                registerService.registerCourse() ->
student.id, course.id
                                course.availableSpace -= 1
                                student.addToRegisterredCourse() ->
course

                else
                        student.canNotRegister()-> course


                if (student.wantCancelClass)
                        student.findRegisteredClass -> courseId:
wantToCancelledCourse
                        student.removeFromRegisteredCourse ->
wantToCancelledCourse
                        wantToCancelledCourse.availableSpace++
```

```
else
        student.canNotRegisterAllCourses()
```