```
StudentId
- depCode: String = "1501" {readOnly}
- student: Student
- studentId: String
+ StudentId(registrationOrder: int, currentYear: int)
+ setStudentId()
- getYearString(): String
- getRegistrationString(): String
```

```
Advisor
- name: String
+ Advisor(name: String)
+ approveCourse(courseSection:CourseSection,
student: Student)
```

```
Schedule
+ DAYS: int = 5 {readOnly}
+ HOURS: int = 8 {readOnly}
- program: CourseSection[5][8]
- student: Student
+ Schedule (student: Student)
+ addToProgram(courseSection: CourseSection)
+ collidedSections(courseSection: CourseSection): List<CourseSection>)
+ isCollision(courseSection: CourseSection): boolean
```

```
Course {abstract}
- courseCode: String
- quota: int
- credits: int
- theoretical: int
- practical: int
- courseSection: CourseSection
- registrationSystem: RegistrationSystem
- nonRegisteredCollision: Set<Student>
- nonRegisteredQuota: Set<Student>
# Course(courseCode: String, quota: int, credits: int, theoretical: int,
practical: int)
+ isEligiblePastCourse(student: Student): boolean
+ whenRequested(student: Student): booelan
+ totalHours(): int
```

```
CourseSection
- course: Course
- students: List<Student>
- courseProgram:boolean[5][8]
+ CourseSection(course:Course)
+ setCourseProgram()
+ collidesWithSameSemester(randomHour: int, randomDay: int): boolean
+ addStudent(student:Student)
+ isFull(): boolean
```

```
RegistrationSystem
- semester: Semester
- data: dict
- totalStudents: int[4]
- students: List<Student>
- advisors: List<Advisor>
- mandatoryCourses: List<MandatoryCourses>
- nonTechElectiveCourses: List<NonTechnicalUniversityElectiveCourse>
- techElectiveCourses: List<TechnicalElectiveCourse>
- facultyElectiveCourses: List<FacultyTechnicalElectiveCourse>
- statisticsBuffer: String
+ RegistrationSystem()
+ startTheSimulation()
- readData()
- initCourses()
- initMandatoryCourses()
- initFinalCourses()
- initNTECourses()
- initTECourses()
- initFTECourses()
- findCourse(courseCode: String):Course
- printStatistics()
- initializeAdvisors()
- initializeStudents()
- initStudentsByCount(currentYear: int, numOfStudents: int)
- appointAdvisors()
- addPastCourses()
- addPastMandatories(student: Student)
- addPastElectives(student: Student)
- printMandatoryStatistics()
- printFinalProjectStatistics()
- printElectiveStatistics()
- requestCourses()
+ offeredTECount(student: Student): int
+ offeredFTECount(student: Student): int
+ offeredNTECount(student: Student): int
+ getOfferedMandatories(student: Student): List<CourseSection>
+ getOfferedElectives(student: Student): List<CourseSection>
+ isThereEmptyNonTechSection(): boolean
+ isThereEmptyTechSection(): boolean
+ isThereEmptyFacTechSection(): boolean
+ setSemester(semester: String)
```

```
Student
- name: String
- studentId: StudentId
- registrationOrder: int
- currentYear: int
- semesterNumber: int
- advisor: Advisor
- schedule: Schedule
- transcript: Transcript
- registrationSystem: RegistrationSystem
+ Student:(name: String, currentYear: int,
           registrationOrder: int, registrationSystem: RegistrationSystem)
+ addToCurrentCourses(courseSection: CourseSection)
+ requestCourseSection(courseSection: CourseSection)
+ requestCourses()
+ requestMandatoryCourses()
+ requestElectiveCourses()
+ getNumOfPastElectives(semesterNums: List<Integer>): int
+ setSemesterNum()
+asDict(): dict
```

```
Transcript
- currentCourses: List<Course>
- grades: List<Grade>
+ Transcript()
+ getCompletedCredits(): int
+ getPassedCourses(): List<Course>
+ addPassedCourse(course: Course)
+ hasPassedCourse(course: Course): booelan
+ hasPassedCourses(course: List<Course>): booelan
+ addPassedCourse(course: Course)
+ addFailedCourse(course: Course)
+ asDict(): dict
```

```
Grade
- intGrade: int
- course: Course
- letterGrade: String
+ Grade(intGrade: int,
        course: Course)
+ isPassed(): booelan
+ setLetterGrade
```

```
ElectiveCourse {abstract}
- semesters: List<Integer>
+ ElectiveCourse(courseCode: String, quota: int, credits : int,
        theoretical: int, practical: int, semesters: List<Integer>
+ whenRequested(student: Student): boolean
+ whenRejected(student: Student): {abstract}
+ getRandomElective(): Course {abstract}
```

```
MandatoryCourse
- semesterNumber: float
- semester: Semester
- preRequisties: List<Course>
- nonRegisteredPrereq: Set<Student>
+ MandatoryCourse(courseCode: String, quota: int, credits : int,
        theretical: int, practical: int, preRequesities: List<Course>)
+ isEligiblePastCourse(student: Student): boolean
+ isOfferableForStudent(student: Student): booelan
+ whenRequested(student: Student): boolean
+ setSemester()
```

```
FacultyTechnicalElecetiveCourse
+ FacultyTechnicalElectiveCourse(courseCode: String, quota: int, credits : int,
        theretical: int, practical: int, semesters: List<Integer>)
+ whenRejected(student: Student)
+ getRandomElective(): Course
+ toString(): String
```

```
NonTechnicalUniversityElectiveCourse
+ NonTechnicalUnversityElectiveCourse(courseCode: String, quota: int, credits : int,
        theretical: int, practical: int, semesters: List<Integer>)
+ whenRejected(student: Student)
+ getRandomElective(): Course
+ toString(): String
```

```
TechnicalElectiveCourse
- requiredCredits: int
- preRequisites: List<Course>
- nonRegisteredCredit: Set<Student>
+ TechnicalElectiveCourse(courseCode: String, quota: int, credits : int,
        theretical: int, practical: int, semesters: List<Integer>, requiredCredits:
        preRequisites: List<Course>)
+ isEligiblePastCourse(student: Student): boolean
+ whenRequested(student: Student): boolean
+ whenRejected(student: Student)
+ getRandomElective(): Course
+ checkCreditCondition(student: Student): boolean
```

```
FinalProjectMandatoryCourse
- requiredCredits: int
- nonRegisteredCredit: Set<Student>
+ FinalProjectMandatoryCourseCourse(courseCode: String, semester: float, quota: int, credits : int,
        theretical: int, practical: int, preRequesities: List<Course>, requiredCredits: int)
+ isEligiblePastCourse(student: Student): boolean
+ whenRequested(student: Student): boolean
+ checkReqCredits(student: Student): boolean
+ getRequiredCredits(): int
```