Transcript

-gpa : Double
-passedCredit :Integer
-failedCredit:Integer
-failedCourses : List<Course>
-notTakenCourses : List <Course>
-currentlyTakenCourses : List<Course>
- passedCourses : HashMap<Course,Double>

+addCurrentlyTakenCourse(course : Course):void
+addFailedCourse (course : Course) : void
+addNotTakenCourse (course : Course) : void
+ removeNotTakenCourse:(course:Course):void
+ removeFailedCourse(course:Course):void
+ calculateGPA : Double

Course

- courseCode : String
- courseTitle : String
- givenSemester : Integer
- enrolledStudents :List<Student>
- courseCredit : Integer
- prerequisites : List<Course>
- instructor : Instructor
- dates : List<WeeklyDate>
- maxSeats : Integer
- takenSeats : Integer
- minCreditReq : Integer

+ addWeeklyDate(weeklyDate : WeeklyDate) : void
+ addPrerequisite(course : Course) : void
+ increaseTakenSeat() : void
+ decreaseTakenSeat() : void

<<Enum>>
AddCourseReturnType

SlotNotempty
NoAvailableSEats
NotExistOnAvailableCourses
Success
WaitingScheduleReview
Locked

<<Enum>>
RemoveCourseReturnType

Success
WaitingScheduleReview
Locked
Notexist

<<*Enum*>>
Grade

- grade : Double

+valueOfGrade(point:Double):String

<<*Abstract*>>
Person

- name : String
- uuid : UUID
- surname : String
- email : String
- birthDate : Date

+ addCourseToSchedule(course:Course):void

Schedule

- courses : List<Course>
- approved:Boolean
- senToReview:Boolean

+addCourse(course : Course):void
+ removeCourse(course:Course):void

School

- schoolInstance : School
- students : List<Students>
- instructors : List<Instructor>
- courses : List<Course>
- config : Config

+ addCourse(course : Course) : void
+ addInstructor(instructor : Instructor) : void
+ addStudent(student : Student) : void

Student

-studentId :Long
-yearEnrolled: Integer
-weeklySchedule:Schedule
-transcript: Transcript
-advisor : Advisor
-semester : Integer

+addCurrentlyTakenCourseToTranscript(course:Course):void
+ addNotTakenCourseToTranscript(course:Course):void
+ addFailedCourseToTranscript(course:Course):void
+ removeFromTranscriptNotTakenCourse(course:Course):void
+ removeFromTranscriptFailedCourse(course:Course):void
+ removeFromSchedule(course:Course):void
+ calculateGPA : Double

Instructor

- presentedCourses : List<Course>
- weeklySchedule : Schedule

+ addPresentedCourse(course : Course) : void

WeeklyDate

- dayName : DayName
- hours : Integer

<<Enum>>
DayName

MON
TUE
WED
THU
FRI
SAT
SUN

Advisor

-students : List<Student>

+addStudent(student : Student) : void

Config

-pasprobabilty : Double
-gradeLuck : Double
-gradeVariance : Double
-minimumCreditReq : Integer
-numberOfStudents: Integer

View

start() : void
printStudentMenu() : void
printInstructorMenu() : void
printTranscript(student : Student) : void
printSchedule(schedule : Schedule) : void
printScheduleDays(courses : List<Course>, dayName :DayName) : void
printAvailableCoursesStudent(student : Student) : void
printStudentInfo (student : Student) : void
printInstructorInfo(instructor : Instructor) : void
configureLogger():void
printAddDropMenu(student : Student):void
removeCourseMenu(student : Student):void
addCourseMenu(student : Student) :void

StudentGetDTO

- uuid: UUID
- name: String
- surname: String
- email: String
- birthDate: Date
- studentId: Long
- yearEnrolled: Integer
- passedCourses: List<String>
- failedCourses: List<String>
- advisor: String
- semester: Integer

InstructorGetDTO

- uuid: UUID
- name: String
- surname: String
- email: String
- birthDate: Date
- presentedCourses: List<String>
- isAdvisor: Boolean

CourseGetDTO

- courseCode: String
- courseTitle: String
- givenSemester: Integer
- courseCredit: Integer
- prerequisites: List<String>
- weeklyDate : List<String>
- maxSeats : Integer
- takenSeats : Integer
- minCreditReq : Integer

TranscriptGetDTO

- gpa : double
- passedCredit : Integer
- failedCredit : Integer
- passedCourses : List<String>
- failedCourses : List<String>
- notTakenCourses : <String>
- currentlyTakenCourses : <String>

+addCurrentlyTakenCourse (course : String) : void
+addFailedCourse (course : String) : void
+addNotTakenCourse (course : String) : void
+addPassedCourse (course : String) : void

ScheduleGetDTO

- courses : List<string>
- approved : Boolean
- sendToReview : Boolean</string>

+addCourse(course : String) : void

Extends

1

*

1

1

1

*

*

*

1

*

*

1

*

1

dayName

service package

<<Interface>>
AdvisorService

+ approveSchedule(student: Student): void
+ denySchedule(student : Student):void

<<Interface>>
CourseService

+ addPrerequisites(courseGetDTOS: List<CourseGetDTO>): void

<<Interface>>
Json Service

+ readStudentFromJson(jsonFormattedStudent:String): Student
+ readInstructorsFromJson(jsonFormattedInstructor:String): List<Instructor>
+ readCoursesFromJson(jsonFormattedCourse:String): List<Course>
+ readScheduleFromJson(jsonFormattedSchedule:String): Schedule
+ readConfigFromJson(jsonFormattedConfig :String) :Config
+ readTranscriptFromJson(jsonFormattedTranscript) : Transcript
+ end():void
+ saveTranscripts():void
+ saveSchedules():void

Implements

AdvisorServiceImpl

Implements

CourseServiceImpl

Implements

JsonServiceImpl

<<Interface>>
StudentService

+ addCourseToSchedule(student: Student, courseCode: String, availableCourses: List<Course>): AddCourseReturnType
+ getAvailableCourses(student:Student,isRandom:Boolean):List<Course>
+ removeCourseFromSchedule(student:Student,courseCode:String):RemoveCourseReturnType
+assignedRandomCourses(student :Student):void
+enrollRandomCourses:void
+sendToReview(schedule :Schedule):Boolean

<<Interface>>
SchoolService

+ uploadJsons(): void

Implements

SchoolServiceImpl

Implements

StudentServiceImpl

repository package

<<Interface>>
StudentRepository

+ save(student: Student): void
+ findByStudentId(studentID: Long): Student

<<Interface>>
CourseRepository

+ save(course: Course): void
+ findByCourseCode(courseCode: String): Course

<<Interface>>
InstructorRepository

+ save(instructor: Instructor): void
+ findByEmail(email: String): Instructor
+ findAdvisorByEmail(advisor: String): Advisor

Implements

StudentRepositoryImpl

Implements

CourseRepositoryImpl

Implements

InstructorRepositoryImpl

Mapper package

<<Interface>>
CourseMapper

+ mapTo(courseGetDTO: CourseGetDTO): Course

<<Interface>>
InstructorMapper

+ mapTo(instructorGetDTO: InstructorGetDTO): Instructor

<<Interface>>
StudentMapper

+ mapTo(studentGetDTO: StudentGetDTO): Student

<<Interface>>
ScheduleMapper

+ mapTo(scheduleGetDTO: ScheduleGetDTO): Schedule
+ mapTo(schedule: Schedule): ScheduleGetDTO

Implements

CourseMapperImpl

Implements

InstructorMapperImpl

Implements

StudentMapperImpl

Implements

ScheduleMapperImpl