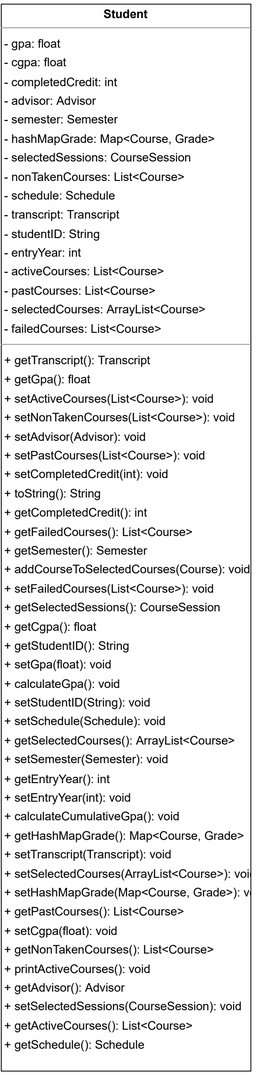
Student
- gpa: float
- cgpa: float
- completedCredit: int
- advisor: Advisor
- semester: Semester
- hashMapGrade: Map<Course, Grade>
- selectedSessions: CourseSession
- nonTakenCourses: List<Course>
- schedule: Schedule
- transcript: Transcript
- studentID: String
- entryYear: int
- activeCourses: List<Course>
- pastCourses: List<Course>
- selectedCourses: ArrayList<Course>
- failedCourses: List<Course>
+ getTranscript(): Transcript
+ getGpa(): float
+ setActiveCourses(List<Course>): void
+ setNonTakenCourses(List<Course>): void
+ setAdvisor(Advisor): void
+ setPastCourses(List<Course>): void
+ setCompletedCredit(int): void
+ toString(): String
+ getCompletedCredit(): int
+ getFailedCourses(): List<Course>
+ getSemester(): Semester
+ addCourseToSelectedCourses(Course): void
+ setFailedCourses(List<Course>): void
+ getSelectedSessions(): CourseSession
+ getCgpa(): float
+ getStudentID(): String
+ setGpa(float): void
+ calculateGpa(): void
+ setStudentID(String): void
+ setSchedule(Schedule): void
+ getSelectedCourses(): ArrayList<Course>
+ setSemester(Semester): void
+ getEntryYear(): int
+ setEntryYear(int): void
+ calculateCumulativeGpa(): void
+ getHashMapGrade(): Map<Course, Grade>
+ setTranscript(Transcript): void
+ setSelectedCourses(ArrayList<Course>): voi
+ setHashMapGrade(Map<Course, Grade>): v
+ getPastCourses(): List<Course>
+ setCgpa(float): void
+ getNonTakenCourses(): List<Course>
+ printActiveCourses(): void
+ getAdvisor(): Advisor
+ setSelectedSessions(CourseSession): void
+ getActiveCourses(): List<Course>
+ getSchedule(): Schedule
RegisterSystem
- advisor: Advisor
- studentObject: Student
- courseController: CourseController
- randomNumberGenerator: Random
+ selectCourse(Course): void
+ getRandomGenerator(): Random
+ startRegistration(): void
+ setRandomGenerator(Random): void
+ selectCourseToActiveCourse(): void
+ setStudent(Student): void
+ getCourseController(): CourseController
+ selectRandomElective(Course): Course
+ assignNextSemester(Student, Semester): v
+ getStudent(): Student
+ getAdvisor(): Advisor
+ setCourseController(CourseController): voi
+ setAdvisor(Advisor): void
Transcript
- hashMapTranscript: Map<String, Object>
- cumulativeGrade: int
- grade: int
- credit: int
- cumulativeCredit: int
- student: Student
+ getGrade(): int
+ addGPA(): void
+ getCumulativeGrade(): int
+ setCumulativeGrade(int): void
+ setGrade(int): void
+ getStudent(): Student
+ addCourse(Course): void
+ getCredit(): int
+ getCumulativeCredit(): int
+ setStudent(Student): void
+ createSemester(): Map<String, Object>
+ getHashMapTranscript(): Map<String, Object>
+ setHashMapTranscript(Map<String, Object>): vo
+ AppendSemester(Semester): void
+ setCredit(int): void
+ setCumulativeCredit(int): void
inputCreator
Person
- surname: String
- id: int
- name: String
- emails: List<String>
+ setId(int): void
+ setEmails(List<String>): void
+ toString(): String
+ getSurname(): String
+ setName(String): void
+ getId(): int
+ setSurname(String): void
+ getName(): String
+ getEmails(): List<String>
CourseSession
- scheduleList: List<Schedule>
- course: Course
- sessionId: int
+ getSessionId(): int
+ getScheduleList(): List<Schedule>
+ setCourse(Course): void
+ toString(): String
+ setSessionId(int): void
+ getCourse(): Course
+ setScheduleList(List<Schedule>): CourseSes
Semester
+ semesterNo: int
- semesterName: String
+ defineSemesterName(int): String
+ semesterNoGenerate(): int
+ getSemesterNo(): int
+ getSemesterName(): String
+ compareTo(Semester): int
- setSemesterName(): void
+ toString(): String
Schedule
- status: boolean
- semesterName: String
+ isStatus(): boolean
+ createSchedule(List<Course>): voi
+ setStatus(boolean): void
Advisor
- Name : String
- student : Student
- course : Course
- client : List<Student>
- givenCourses : List<Courses>
- hmAdvisor : Map <String,Advisor>
~ Approve(Course): Boolean
+ FTECheck(Student) : void
+ MinCreditCheck(Student) : void
~ name() : String
~ TECheck(Student) :void
+addClient(Student) : void
+ addGivenCourse(Course) : void
+getClient() : List<Student>
+getCourse() :Course
+getElectiveCourseCheck() : boolean
+getGivenCourses() : List<Course>
+getStudent() : Student
+graduationProjectCheck(Student) : void
+prerequsiteCheck():boolean
+quotaCheck() : boolean
+selectNonTakenCourse(Course): Course
+setClient(List<Student>) : void
+setCourse(Course) : void
+setGivenCourses(List<Course>) : void
+setName(String) : void
+setStudent(Student) : void
StudentController
- advisors: List<Advisor>
- HMstudent: Map<Integer, Student>
+ setHMstudent(Map<Integer, Student>): void
+ createStudent(int, String, String, List<String>, Semester):
- getStudent(int, String, String, List<String>, Semester): Stu
+ getAdvisors(): List<Advisor>
+ getHMstudent(): Map<Integer, Student>
+ printStudents(): void
+ setAdvisors(List<Advisor>): void
CourseController
- NT_UCourses: List<ElectiveCourse>
- technicalCourses: List<ElectiveCourse>
- mandatoryCourses: List<Course>
- HMsemester: Map<Integer, Semester>
- facultyTechnicalCourses: List<ElectiveCourse>
- courses: List<Course>
+ addTechnicalCourse(ElectiveCourse): void
+ addPrerequisiteCourse(Course, Course): void
+ getNT_UCourses(): List<ElectiveCourse>
+ setHMsemester(Map<Integer, Semester>): void
+ searchCourse(String): Course
+ getHMsemester(): Map<Integer, Semester>
+ setFacultyTechnicalCourses(List<ElectiveCourse>): void
+ printMandatoryCourses(): void
+ createSession(int, Course, List<Schedule>): Session
+ setTechnicalCourses(List<ElectiveCourse>): void
+ createMandatoryCourse(String, String, int, int, String, int): Co
+ getCourses(): List<Course>
+ setCourses(List<Course>): void
+ createElectiveCourse(String, String, String, int, int): ElectiveC
+ printTechnicalElectiveCourses(): void
+ removeStudentsFromCourses(): void
+ printFacultyTechnicalCourses(): void
+ addFacultyTechnicalCourse(ElectiveCourse): void
+ addSession(Session): void
+ printNT_UCourses(): void
+ getMandatoryCourses(): List<Course>
+ addNTUCourse(ElectiveCourse): void
+ createSemester(): void
+ getFacultyTechnicalCourses(): List<ElectiveCourse>
+ setMandatoryCourses(List<Course>): void
+ setNT_UCourses(List<ElectiveCourse>): void
+ getTechnicalCourses(): List<ElectiveCourse>
+ addMandatoryCourse(Course): void
Course
- prerequisites: List<Course>
- courseId: String
- name: String
- semester: Semester
- quota: int
- credit: int
- session: CourseSession
- students: List<Student>
+ setQuota(int): void
+ setSession(CourseSession): void
+ getCredit(): int
+ setCourseId(String): void
+ setStudents(List<Student>): void
+ setCredit(int): void
+ addStudentToArraylist(Student): voi
+ getName(): String
+ setName(String): void
+ getPrerequisites(): List<Course>
+ setPrerequisites(List<Course>): void
+ toString(): String
+ getSession(): CourseSession
+ getQuota(): int
+ getStudents(): List<Student>
+ setSemester(Semester): void
+ getCourseId(): String
+ getSemester(): Semester
ElectiveCourse
MandatoryCourse
+ toString(): String
TechnicalCourse
- minimumCredit: int
+ getMinimumCredit(): int
+ toString(): String
FacultyTechnicalCourse
+ toString(): String
NT_UCourse
+ toString(): String
Checker
- student: Student
+ setStudent(Student): void
+ getStudent(): Student
+ checkerMessage(): String
+ raiseCourse(): Course
GraduateChecker
- course: Course
+ checkerMessage(): String
+ getCourse(): Course
+ setCourse(Course): void
+ raiseCourse(): Course
QuotaChecker
- course: Course
+ setCourse(Course): void
+ checkerMessage(): String
+ raiseCourse(): Course
+ getCourse(): Course
CreditChecker
- course: Course
+ raiseCourse(): Course
+ getCourse(): Course
+ setCourse(Course): void
+ raiseError(): String
TeChecker
- course: Course
+ setCourse(Course): void
+ getCourse(): Course
+ raiseError(): String
+ raiseCourse(): Course
PrerequisiteChecker
- course: Course
- prereq: Course
+ setCourse(Course): void
+ getCourse(): Course
+ raiseCourse(): Course
+ setPrereq(Course): void
+ getPrereq(): Course
+ checkerMessage(): String
ProjectChecker
- course: Course
+ getCourse(): Course
+ setCourse(Course): void
+ raiseCourse(): Course
+ checkerMessage(): String
CollisionChecker
- secondCourse: Course
- firstCourse: Course
+ setFirstCourse(Course): void
+ checkerMessage(): String
+ getSecondCourse(): Course
+ setSecondCourse(Course): void
+ raiseCourse(): Course
+ getFirstCourse(): Course