Steps of Design Class Daigram

Transcript

+getSemesterPeroid(): void

First design: 3/11/2022

StudentCreator

							RegularCourse		I I-CourseCode	
-advisor: Advisor -semester: Semester -CoursesTook: List <course></course>	-FullName: String -StudentList: <list> +ApproveOrDisapprove(selectedCourses:List<course>,</course></list>			-CompletedCourses: List <couse> -FailedCourses: <list> -GPA: Double -CompletedCredits:int</list></couse>			TE NTE -Type: String		-Quota:Intege -Prerequisite -Credits: Inte -CourseSem	
					_				-StudentsEnro -NumberOfSt	
+CreateStudentID: int				+getCompletedCourses(): List < +getFailedCourses(): List <cour +getGPA(): Double +getCompletedCredits():int</cour 			+getCourseType(type: String): CourseType		-CourseType: -CourseSeme	
+CreateStudent(index: int , fullname string):Student				+getCompletedCredits().int			L		+getCourseC	
		TranscriptCreator]		Course	Salastar	1	+getQuota():I +getPrerequi	
		-				Course Selector		1	+getCredits() +getCourseS	
Student		+createTranscript(): Transcript +takeCompletedCourses(): List <completedcourses> +takeFailedCourses(): List<completedcourses> +calculateGPA(): double +calculateTotalCredits(): int</completedcourses></completedcourses>							+getStudents +getNumber	
-FullName: String -StudentID: int -Advisor: Advisor						selectCourses(student Student, courses: List <course>, teCourses: List<course>,nteCourses: List<course):List<course></course></course </course></course>			+getCourseT +getCourseS	
-Transcript: Trancript -takenCourses: List <takencourse></takencourse>		+calculate rotal credits(). Int						, —		
								Coi	mpletedCourses	
+getFullName(): String				Semester					-CourseCode: String -LetterGrade: String	
+getStudentID(): int +getAdvisor(): Advisor +getTranscript(): Trancript +getTakenCourses: List <takencourse> +SendToApprove(selectedCourses: List<course>):List<course< td=""><td colspan="2" rowspan="2"></td><td colspan="2" rowspan="2">-SemesterNumber: int -semesterPeriod: SemesterPeroid +getSemesterNumber(): int</td><td></td><td>all Spring</td><td></td><td></td></course<></course></takencourse>				-SemesterNumber: int -semesterPeriod: SemesterPeroid +getSemesterNumber(): int			all Spring			
									ourseCode: String etterGrade: String	

Advisor

-CourseCode: String -Quota:Integer -PrerequisiteCourse: Course -PrerequisiteCourse: Course -Credits: Integer -CourseSemester: Semester -StudentsEnrolled: <List> -NumberOfStudents: int -CourseType: CourseType -CourseSemester: Semester +getCourseCode(): String +getQuota():Integer +getPrerequisiteCourse(): Course +getCredits(): Integer +getCourseSemester(): Semester +getStudentsEnrolled(): <List> +getNumberOfStudents(): int

+getCourseType(): CourseType +getCourseSemester(): Semester

FailedCourses

+getCourseCode: String

-CourseCode: String

Course

CourseType

PogularCourse

Second design: 10/11/2022

Course

-courseCode: String -quota:Integer -prerequisiteCourse: Course -credits: Integer

-studentsEnrolled : <List> -numberOfStudents: int

-courseType: CourseType -theoreticalCourseHour: int -practicalLessonHour: int

+getCourseCode(): String

+getQuota():Integer

+getPrerequisiteCourse(): Course

+getCredits(): Integer

+getStudentsEnrolled(): <List>

+getNumberOfStudents(): int

+getCourseType(): CourseType

CourseSection

-studentList: List<Student> -lecturer: Lecturer -sectionNumber: int

+setCourseNumber(): int +getLecturer():Lecturer

+getStudentList: List<Student>

CourseType

RegularCourse NTE -Type: String

+getType(type: String): CourseType

Advisor

-studentList: List<Student>

+ApproveOrDisapprove(selectedCourses:List<Course>, student: Student): List<Course> +assignStudentList(): List<Student>

Student

-studentID: int

-advisor: Advisor

-transcript: Trancript

-takenCourses: List<TakenCourse>

+getFullName(): String +getStudentID(): int

+getAdvisor(): Advisor

+getTranscript(): Trancript

+getTakenCourses: List<TakenCourse>

CourseRegistrationSystem

numberOfStudents: int

studentList: List<Student>

-numberOfAdvisors: int -advisorsList: List<Advisor>

-numberOfLecturers: int

-lecturersList: List<Lecturer>

-coursesList: List<Course>

-teCourses: List<Course>

-nteCourses:List<Course>

-user: Student

-selectedCourses: List<Course>

+readInputData()

+Login()

+getAdvisor(): Advisor

+getTrascript(): Transcript +getUser(): Student

+getCourses(): List<Courses>

+getTeCourses(): List<Course>

+getNteCourses():List<Course>

+selectCourses(student StudentID, courses: List<Course>,

teCourses: List<Course>,nteCourses: List<Course>):List<Course>

+SendToApprove(selectedCourses: List<Course>):List<Course>

+LogOut()

Semester

-SemesterNumber: int

-semesterPeriod: SemesterPeroid

+getSemesterNumber(): int +getSemesterPeroid(): void

SemesterPeroid

Fall Spring

Transcript TranscriptCreator -completedCourses: List <CompletedCourse> -failedCourses: List<FailedCourse> -GPA: Double -completedCredits:int +createTranscript(): Transcript +takeCompletedCourses(): List<CompletedCourses> +getCompletedCourses(): List <Couse> +takeFailedCourses(): List<CompletedCourses> +getFailedCourses(): List <Course> +calculateGPA(): double +getGPA(): Double +calculateTotalCredits(): int +getCompletedCredits():int

CompletedCourse

-CourseCode: String -CourseGrade: String

+getCourseCode(): String +getCourseGrade(): String

FailedCourse

-CourseCode: String

+aetCourseCode: Strina

Final design:17/11/2022

Main	
-	
+main(): void	

Advisor					
-advisorID: int -fName: String -IName: String					
+getters(): type +setters(type: Type): void +advisorControl(chosenClasses: List <strings>, student: Student): void</strings>					

Courses
-name: String -courseCode: String -prerequisite: List <string> -credit: int -courseType: int -semester: int -courseYear: int -theoreticalCourseHour: int -practicalLessonHour: int</string>
+getters(): type

+setters(type: Type): void +checkIfPrerequisite(courses: Courses):Boolean +checkIfTwoPrerequisite(courses: Courses):Boolean +getPreRequisiteName(): String +getTwoPreRequisiteName(courses: Courses):ArrayList<String>

Student -studentId: int -fName: String -IName: String -totalCredits: int -advisorId: int -gpa: double -currentYear: int -currentSemester: int -currentSelectedCourses: List<String> -completedCourses: List<CompletedCourses> -availableCourses: List<String> -failedCourses: List<FailedCourses> +getters(): type +setters(type: Type): void +selectFromAvailableCourses(): void +checklfCourseFailed(courseCode: String): boolean

+sendToAdvisorSelectedClasses(advisors: Advisor[]): void +changeSelectedCourses(advisorApprovedCourses: ArrayList<String>, advisorRejectedCoursesAndReasons:ArrayList<String>): void

+gpaCalculator(courses: Courses[]): void

Generate Student					
-student: List <student> -courses: List<courses> -firstSemesterCourses: List<string> -secondSemesterCoursesHash: HashMap<string, list<string="">> -thirdSemesterCoursesHash: HashMap<string, list<string="">> -fourthSemesterCoursesHash: HashMap<string, list<string="">> -fifthSemesterCourses: HashMap<string, list<string="">> -sixthSemesterCoursesHash: HashMap<string, list<string="">> -seventhSemesterCoursesHash: HashMap<string, list<string="">> -seventhSemesterCoursesHash: HashMap<string, list<string="">> -prerequisiteList: HashMap<string, list<string="">> -prerequisiteList: HashMap<string, list<string="">></string,></string,></string,></string,></string,></string,></string,></string,></string,></string></courses></student>					
<cconstructer>> GenerateStudents(student: Student[],courses: Courses[]) +addCourseNames(): void +generateYear(student: Student): void +semesterSetter(s: Student, semester: String): void +setCoursesList(s: Student): void +assignFailedCourses(currentSemesterFailed: List<failedcourses>, courseCode: String): void +prerequisiteControlAndLock(courseCode: String, lockedCourses: HashMap<string, list<string="">>): void +addCompletedCourses(currentSemesterCompleted: List<completedcourses>, courseCode: String, grade: String, finishedSemester: int): void +simulateFailedCourses(s: Student, currentSemesterCompleted: List<completedcourses>, currentSemester: int): void +unlockLockedCoursesAndSetAvailable(s: Student, completedCourses: List<completedcourses>, lockedCourses: HashMap<string, list<string="">>): void +checkAvailableCourse(s: Student, currentSemesterCompleted: List<completedcourses>, currentSemesterFailed: List<failedcourses>, lockedCourses: HashMap<string, list<string="">>, currentSemesterFailed: List<failedcourses>, lockedCourses: HashMap<string, list<string="">>, currentSemesterCourses: HashMap<string, list<string="">>, currentSemesterFailed: List<failedcourses>, lockedCourses: HashMap<string, list<string="">>, currentSemesterCourses: HashMap<string, list<string="">>); void</string,></string,></failedcourses></string,></string,></failedcourses></string,></failedcourses></completedcourses></string,></completedcourses></completedcourses></completedcourses></string,></failedcourses></cconstructer>					

Randomizer					
-					
+setAvailableCoursesForEachStudent(students: Student[], courses: Courses[], advisors: Advisor[]): void +addCourseNames(students: Student[], courses: Courses[]): void +generate(students: Student[]): ArrayList <integer> +assignRandomGrades(): String</integer>	t				

+checkPrerequisiteCourselsGiven(s: Student, courseCode: String, semester: int): boolean

+checkCourseGiven(s: Student): void

+removeDuplicates(s: Student): void

+setStudentAdvisor(s: Student): void

+assignRandomGrades(): String

+simulate(): void

+checkCourseHasPrerequisite(courseCode: String): void

+simulateSemester(s: Student, semester: String): void

+courselsGivenAlready(s: Student, courseCode: String): boolean

Transcript

-completedCourses: List <CompletedCourses> -failedCourses: List<FailedCourses> -gpa: Double

-completedCredits: int

+getters(): type

+setters(type: Type): void +printTranscript(students2: Student[])

CompletedCourses

-courseName: String -courseGrade: String -givenSemester: int

+getters(): type +setters(type): void +toString(): String

FailedCourses

-courseName: String -courseGrade: String

+getters(): type +setters(type: Type): void +toString(): String