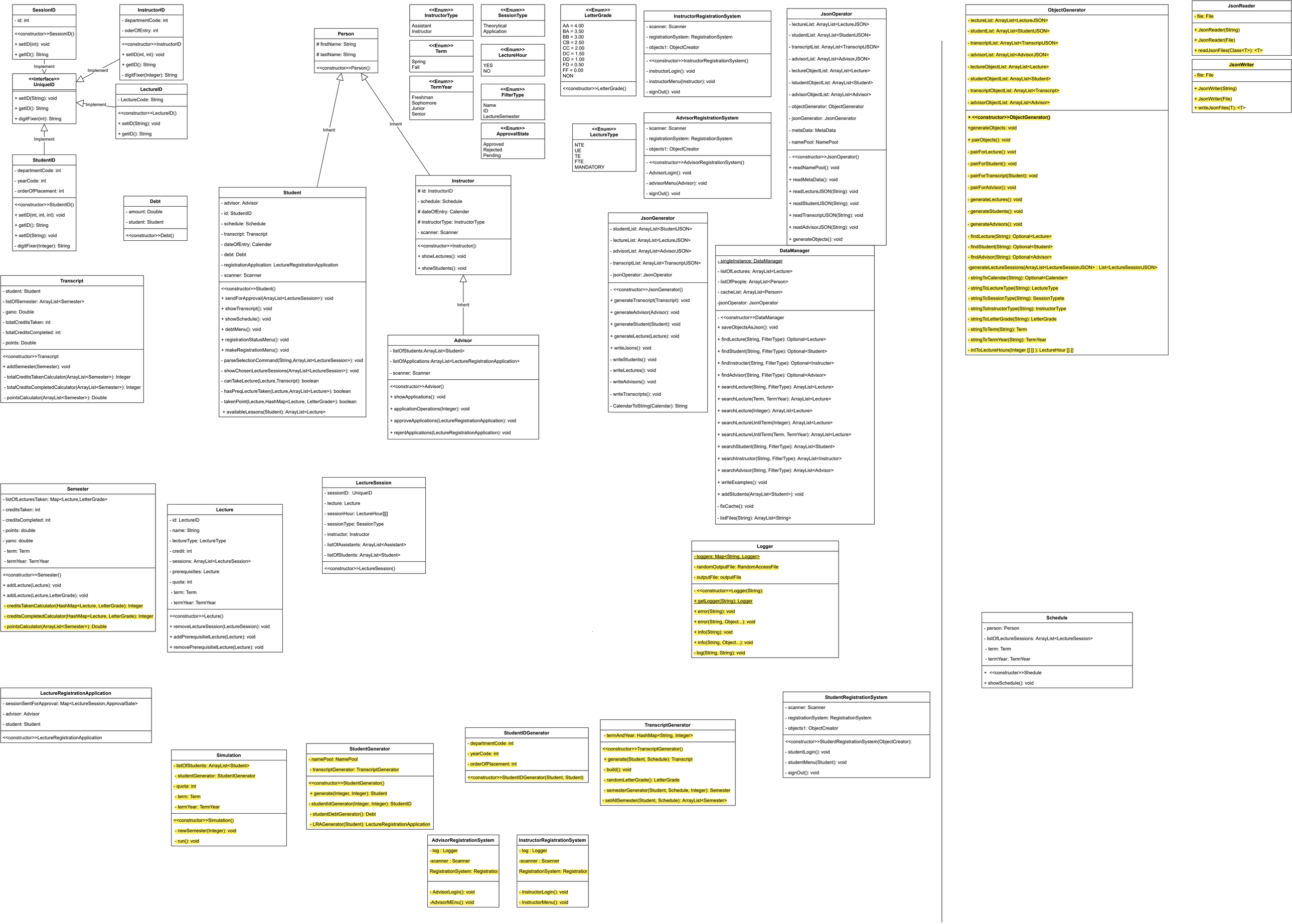
SessionID
- id: int
<<constructor>>SessionID()
+ setID(int): void
+ getID(): String
InstructorID
- departmentCode: int
- oderOfEntry: int
<<constructor>>InstructorID
+ setID(int, int): void
+ getID(): String
- digitFixer(Integer): String
Implement
Implement
<<interface>>
UniqueID
+ setID(String): void
+ getID(): String
+ digitFixer(int): String
Implement
Implement
LectureID
- LectureCode: String
<<constructor>>LectureID()
+ setID(String): void
+ getID(): String
Implement
StudentID
- departmentCode: int
- yearCode: int
- orderOfPlacement: int
<<constructor>>StudentID()
+ setID(int, int, int): void
+ getID(): String
+ setID(String): void
- digitFixer(Integer): String
Debt
- amount: Double
- student: Student
<<constructor>>Debt()
Person
firstName: String
lastName: String
<<constructor>>Person():
Inherit
Inherit
<<Enum>>
InstructorType
Assistant
Instructor
<<Enum>>
Term
Spring
Fall
<<Enum>>
TermYear
Freshman
Sophomore
Junior
Senior
<<Enum>>
SessionType
Theorytical
Application
<<Enum>>
LectureHour
YES
NO
<<Enum>>
FilterType
Name
ID
LectureSemester
<<Enum>>
ApprovalState
Approved
Rejected
Pending
Student
- advisor: Advisor
- id: StudentID
- schedule: Schedule
- transcript: Transcript
- dateOfEntry: Calender
- debt: Debt
- registrationApplication: LectureRegistrationApplication
- scanner: Scanner
<<constructor>>Student()
+ sendForApproval(ArrayList<LectureSession>): void
+ showTranscript(): void
+ showSchedule(): void
+ debtMenu(): void
+ registrationStatusMenu(): void
+ makeRegistrationMenu(): void
- parseSelectionCommand(String,ArrayList<LectureSession>): void
- showChosenLectureSessions(ArrayList<LectureSession>): void
- canTakeLecture(Lecture,Transcript): boolean
- hasPreqLectureTaken(Lecture,ArrayList<Lecture>): boolean
- takenPoint(Lecture,HashMap<Lecture, LetterGrade>): boolean
+ availableLessons(Student): ArrayList<Lecture>
Instructor
id: InstructorID
- schedule: Schedule
dateOfEntry: Calender
instructorType: InstructorType
- scanner: Scanner
<<constructor>>Instructor():
+ showLectures(): void
+ showStudents(): void
Inherit
Advisor
- listOfStudents:ArrayList<Student>
- listOfApplications:ArrayList<LectureRegistrationApplication>
- scanner: Scanner
<<constructor>>Advisor()
+ showApplications(): void
+ applicationOperations(Integer): void
+ approveApplications(LectureRegistrationApplication): void
+ rejentApplications(LectureRegistrationApplication): void
<<Enum>>
LetterGrade
AA = 4.00
BA = 3.50
BB = 3.00
CB = 2.50
CC = 2.00
DC = 1.50
DD = 1.00
FD = 0.50
FF = 0.00
NON
<<constructor>>LetterGrade()
<<Enum>>
LectureType
NTE
UE
TE
FTE
MANDATORY
InstructorRegistrationSystem
- scanner: Scanner
- registrationSystem: RegistrationSystem
- objects1: ObjectCreator
- <<constructor>>InstructorRegistrationSystem()
- instructorLogin(): void
- instructorMenu(Instructor): void
- signOut(): void
AdvisorRegistrationSystem
- scanner: Scanner
- registrationSystem: RegistrationSystem
- objects1: ObjectCreator
- <<constructor>>AdvisorRegistrationSystem()
- AdvisorLogin(): void
- advisorMenu(Advisor): void
- signOut(): void
JsonOperator
- lectureList: ArrayList<LectureJSON>
- studentList: ArrayList<StudentJSON>
- transcriptList: ArrayList<TranscriptJSON>
- advisorList: ArrayList<AdvisorJSON>
- lectureObjectList: ArrayList<Lecture>
- lstudentObjectList: ArrayList<Student>
- advisorObjectList: ArrayList<Advisor>
- objectGenerator: ObjectGenerator
- jsonGenerator: JsonGenerator
- metaData: MetaData
- namePool: NamePool
- <<constructor>>JsonOperator()
+ readNamePool(): void
+ readMetaData(): void
+ readLectureJSON(String): void
+ readStudentJSON(String): void
+ readTranscriptJSON(String): void
+ readAdvisorJSON(String): void
+ generateObjects(): void
JsonGenerator
- studentList: ArrayList<StudentJSON>
- lectureList: ArrayList<LectureJSON>
- advisorList: ArrayList<AdvisorJSON>
- transcriptList: ArrayList<TranscriptJSON>
- jsonOperator: JsonOperator
- <<constructor>>JsonGenerator()
+ generateTranscript(Transcript): void
+ generateAdvisor(Advisor): void
+ generateStudent(Student): void
+ generateLecture(Lecture): void
+ writeJsons(): void
- writeStudents(): void
- writeLectures(): void
- writeAdvisors(): void
- writeTranscripts(): void
- CalendarToString(Calendar): String
DataManager
- singleInstance: DataManager
- listOfLectures: ArrayList<Lecture>
- listOfPeople: ArrayList<Person>
- cacheList: ArrayList<Person>
-jsonOperator: JsonOperator
- <<constructor>>DataManager
+ saveObjectsAsJson(): void
+ findLecture(String, FilterType): Optional<Lecture>
+ findStudent(String, FilterType): Optional<Student>
+ findInstructer(String, FilterType): Optional<Instructer>
+ findAdvisor(String, FilterType): Optional<Advisor>
+ searchLecture(String, FilterType): ArrayList<Lecture>
+ searchLecture(Term, TermYear): ArrayList<Lecture>
+ searchLecture(Integer): ArrayList<Lecture>
+ searchLectureUntilTerm(Integer): ArrayList<Lecture>
+ searchLectureUntilTerm(Term, TermYear): ArrayList<Lecture>
+ searchStudent(String, FilterType): ArrayList<Student>
+ searchInstructor(String, FilterType): ArrayList<Instructor>
+ searchAdvisor(String, FilterType): ArrayList<Advisor>
+ writeExamples(): void
+ addStudents(ArrayList<Student>): void
- fixCache(): void
- listFiles(String): ArrayList<String>
ObjectGenerator
- lectureList: ArrayList<LectureJSON>
- studentList: ArrayList<StudentJSON>
- transcriptList: ArrayList<TranscriptJSON>
- advisorList: ArrayList<AdvisorJSON>
- lectureObjectList: ArrayList<Lecture>
- studentObjectList: ArrayList<Student>
- transcriptObjectList: ArrayList<Transcript>
- advisorObjectList: ArrayList<Advisor>
+ <<constructor>>ObjectGenerator()
+generateObjects: void
+ pairObjects(): void
- pairForLecture(): void
- pairForStudent(): void
- pairForTranscript(Student): void
- pairForAdvisor(): void
- generateLectures(): void
- generateStudents(): void
- generateAdvisors(): void
- findLecture(String): Optional<Lecture>
- findStudent(String): Optional<Student>
- findAdvisor(String): Optional<Advisor>
-generateLectureSessions(ArrayList<LectureSessionJSON> : List<LectureSessionJSON>
- stringToCalendar(String): Optional<Calendar>
- stringToLectureType(String): LectureType
- stringToSessionType(String): SessionTypete
+ stringToInstructorType(String): InstructorType
- stringToLetterGrade(String): LetterGrade
- stringToTerm(String): Term
- stringToTermYear(String): TermYear
- intToLectureHours(Integer [] []): LectureHour [] []
JsonReader
- file: File
+ JsonReader(String)
+ JsonReader(File)
+ readJsonFiles(Class<T>): <T>
JsonWriter
- file: File
+ JsonWriter(String)
+ JsonWriter(File)
+ writeJsonFiles(T): <T>
Transcript
- student: Student
- listOfSemester: ArrayList<Semester>
- gano: Double
- totatCreditsTaken: int
- totalCreditsCompleted: int
- points: Double
<<constructor>>Transcript
+ addSemester(Semester): void
- totalCreditsTakenCalculator(ArrayList<Semester>): Integer
- totalCreditsCompletedCalculator(ArrayList<Semester>): Integer
- pointsCalculator(ArrayList<Semester>): Double
Semester
- listOfLecturesTaken: Map<Lecture,LetterGrade>
- creditsTaken: int
- creditsCompleted: int
- points: double
- yano: double
- term: Term
- termYear: TermYear
<<constructor>>Semester()
+ addLecture(Lecture): void
+ addLecture(Lecture,LetterGrade): void
- creditsTakenCalculator(HashMap<Lecture, LetterGrade): Integer
- creditsCompletedCalculator(HashMap<Lecture, LetterGrade): Integer
- pointsCalculator(ArrayList<Semester>): Double
Lecture
- id: LectureID
- name: String
- lectureType: LectureType
- credit: int
- sessions: ArrayList<LectureSession>
- prerequisities: Lecture
- quota: int
- term: Term
- termYear: TermYear
<<constructor>>Lecture()
+ removeLectureSession(LectureSession): void
+ addPrerequisitielLecture(Lecture): void
+ removePrerequisitielLecture(Lecture): void
LectureSession
- sessionID: UniqueID
- lecture: Lecture
- sessionHour: LectureHour[][]
- sessionType: SessionType
- instructor: Instructor
- listOfAssistants: ArrayList<Assistant>
- listOfStudents: ArrayList<Student>
<<constructor>>LectureSession()
Logger
- loggers: Map<String, Logger>
- randomOutputFile: RandomAccessFile
- outputFile: outputFile
- <<constructor>>Loggger(String):
+ getLogger(String): Logger
+ error(String): void
+ error(String, Object...): void
+ info(String): void
+ info(String, Object...): void
- log(String, String): void
Schedule
- person: Person
- listOfLectureSessions: ArrayList<LectureSession>
- term: Term
- termYear: TermYear
+ <<constructer>>Shedule
+ showSchedule(): void
LectureRegistrationApplication
- sessionSentForApproval: Map<LectureSession,ApprovalSate>
- advisor: Advisor
- student: Student
<<constructor>>LectureRegistrationApplication
Simulation
- listOfStudents: ArrayList<Student>
- studentGenerator: StudentGenerator
- quota: int
- term: Term
- termYear: TermYear
<<constructor>>Simulation()
- newSemester(Integer): void
- run(): void
StudentGenerator
- namePool: NamePool
- transcriptGenerator: TranscriptGenerator
<<constructor>>StudentGenerator()
+ generate(Integer, Integer): Student
- studentIdGenerator(Integer, Integer): StudentID
- studentDebtGenerator(): Debt
- LRAGenerator(Student): LectureRegistrationApplication
StudentIDGenerator
- departmentCode: int
- yearCode: int
- orderOfPlacement: int
<<constructor>>StudentIDGenerator(Student, Student)
AdvisorRegistrationSystem
- log : Logger
-scanner : Scanner
RegistrationSystem: Registratio
- AdvisorLogin(): void
-AdvisorMEnu(): void
InstructorRegistrationSystem
- log : Logger
-scanner : Scanner
RegistrationSystem: Registratio
- InstructorLogin(): void
- InstructorMenu(): void
TranscriptGenerator
- termAndYear: HashMap<String, Integer>
<<constructor>>TranscriptGenerator()
+ generate(Student, Schedule): Transcript
- build(): void
- randomLetterGrade(): LetterGrade
- semesterGenerator(Student, Schedule, Integer): Semester
- setAllSemester(Student, Schedule): ArrayList<Semester>
StudentRegistrationSystem
- scanner: Scanner
- registrationSystem: RegistrationSystem
- objects1: ObjectCreator
<<constructor>>StudentRegistrationSystem(ObjectCreator):
- studentLogin(): void
- studentMenu(Student): void
- signOut(): void