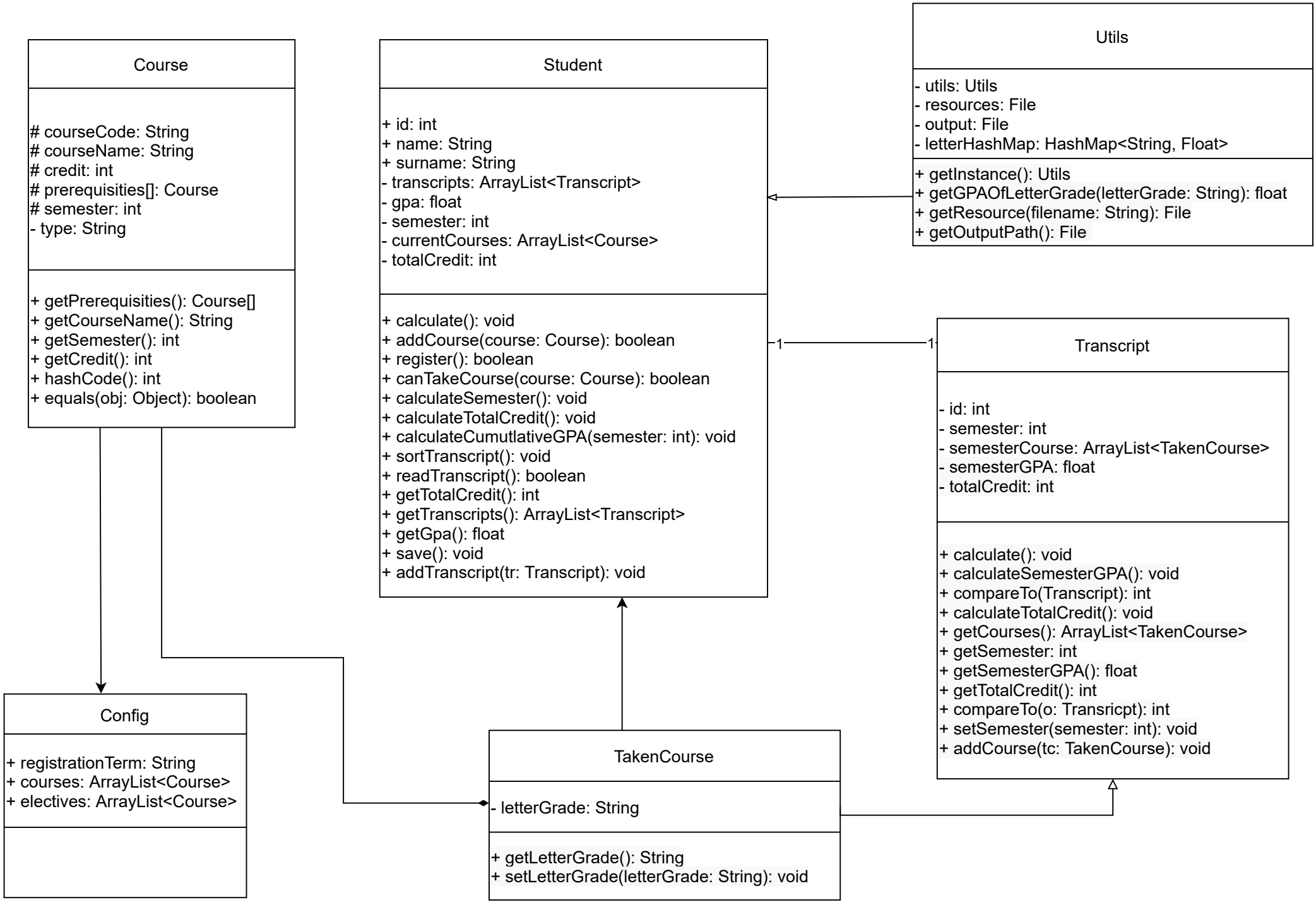
Course
courseCode: String
courseName: String
credit: int
prerequisities[]: Course
semester: int
- type: String
+ getPrerequisities(): Course[]
+ getCourseName(): String
+ getSemester(): int
+ getCredit(): int
+ hashCode(): int
+ equals(obj: Object): boolean
Student
+ id: int
+ name: String
+ surname: String
- transcripts: ArrayList<Transcript>
- gpa: float
- semester: int
- currentCourses: ArrayList<Course>
- totalCredit: int
+ calculate(): void
+ addCourse(course: Course): boolean
+ register(): boolean
+ canTakeCourse(course: Course): boolean
+ calculateSemester(): void
+ calculateTotalCredit(): void
+ calculateCumutlativeGPA(semester: int): void
+ sortTranscript(): void
+ readTranscript(): boolean
+ getTotalCredit(): int
+ getTranscripts(): ArrayList<Transcript>
+ getGpa(): float
+ save(): void
+ addTranscript(tr: Transcript): void
Utils
- utils: Utils
- resources: File
- output: File
- letterHashMap: HashMap<String, Float>
+ getInstance(): Utils
+ getGPAOfLetterGrade(letterGrade: String): float
+ getResource(filename: String): File
+ getOutputPath(): File
1
1
Transcript
- id: int
- semester: int
- semesterCourse: ArrayList<TakenCourse>
- semesterGPA: float
- totalCredit: int
+ calculate(): void
+ calculateSemesterGPA(): void
+ compareTo(Transcript): int
+ calculateTotalCredit(): void
+ getCourses(): ArrayList<TakenCourse>
+ getSemester: int
+ getSemesterGPA(): float
+ getTotalCredit(): int
+ compareTo(o: Transricpt): int
+ setSemester(semester: int): void
+ addCourse(tc: TakenCourse): void
Config
+ registrationTerm: String
+ courses: ArrayList<Course>
+ electives: ArrayList<Course>
TakenCourse
- letterGrade: String
+ getLetterGrade(): String
+ setLetterGrade(letterGrade: String): void