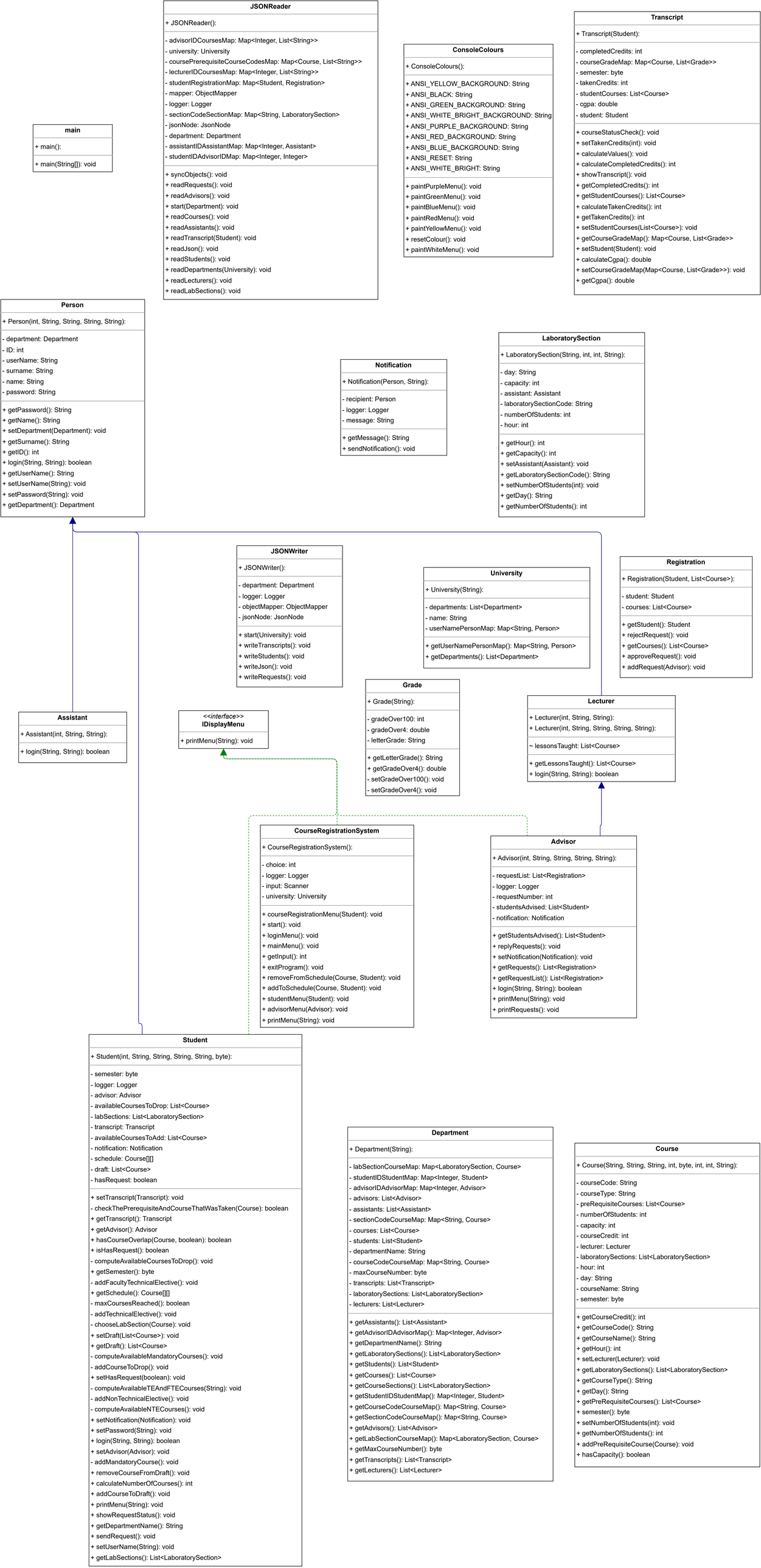
JSONReader
+ JSONReader():
- advisorIDCoursesMap: Map<Integer, List<String>>
- university: University
- coursePrerequisiteCourseCodesMap: Map<Course, List<String>>
- lecturerIDCoursesMap: Map<Integer, List<String>>
- studentRegistrationMap: Map<Student, Registration>
- mapper: ObjectMapper
- logger: Logger
- sectionCodeSectionMap: Map<String, LaboratorySection>
- jsonNode: JsonNode
- department: Department
- assistantIDAssistantMap: Map<Integer, Assistant>
- studentIDAdvisorIDMap: Map<Integer, Integer>
+ syncObjects(): void
+ readRequests(): void
+ readAdvisors(): void
+ start(Department): void
+ readCourses(): void
+ readAssistants(): void
+ readTranscript(Student): void
+ readJson(): void
+ readStudents(): void
+ readDepartments(University): void
+ readLecturers(): void
+ readLabSections(): void
ConsoleColours
+ ConsoleColours():
+ ANSI_YELLOW_BACKGROUND: String
+ ANSI_BLACK: String
+ ANSI_GREEN_BACKGROUND: String
+ ANSI_WHITE_BRIGHT_BACKGROUND: String
+ ANSI_PURPLE_BACKGROUND: String
+ ANSI_RED_BACKGROUND: String
+ ANSI_BLUE_BACKGROUND: String
+ ANSI_RESET: String
+ ANSI_WHITE_BRIGHT: String
+ paintPurpleMenu(): void
+ paintGreenMenu(): void
+ paintBlueMenu(): void
+ paintRedMenu(): void
+ paintYellowMenu(): void
+ resetColour(): void
+ paintWhiteMenu(): void
Transcript
+ Transcript(Student):
- completedCredits: int
- courseGradeMap: Map<Course, List<Grade>>
- semester: byte
- takenCredits: int
- studentCourses: List<Course>
- cgpa: double
- student: Student
+ courseStatusCheck(): void
+ setTakenCredits(int): void
+ calculateValues(): void
+ calculateCompletedCredits(): int
+ showTranscript(): void
+ getCompletedCredits(): int
+ getStudentCourses(): List<Course>
+ calculateTakenCredits(): int
+ getTakenCredits(): int
+ setStudentCourses(List<Course>): void
+ getCourseGradeMap(): Map<Course, List<Grade>>
+ setStudent(Student): void
+ calculateCgpa(): double
+ setCourseGradeMap(Map<Course, List<Grade>>): void
+ getCgpa(): double
main
+ main():
+ main(String[]): void
Person
+ Person(int, String, String, String, String):
- department: Department
- ID: int
- userName: String
- surname: String
- name: String
- password: String
+ getPassword(): String
+ getName(): String
+ setDepartment(Department): void
+ getSurname(): String
+ getID(): int
+ login(String, String): boolean
+ getUserName(): String
+ setUserName(String): void
+ setPassword(String): void
+ getDepartment(): Department
Notification
+ Notification(Person, String):
- recipient: Person
- logger: Logger
- message: String
+ getMessage(): String
+ sendNotification(): void
LaboratorySection
+ LaboratorySection(String, int, int, String):
- day: String
- capacity: int
- assistant: Assistant
- laboratorySectionCode: String
- numberOfStudents: int
- hour: int
+ getHour(): int
+ getCapacity(): int
+ setAssistant(Assistant): void
+ getLaboratorySectionCode(): String
+ setNumberOfStudents(int): void
+ getDay(): String
+ getNumberOfStudents(): int
JSONWriter
+ JSONWriter():
- department: Department
- logger: Logger
- objectMapper: ObjectMapper
- jsonNode: JsonNode
+ start(University): void
+ writeTranscripts(): void
+ writeStudents(): void
+ writeJson(): void
+ writeRequests(): void
University
+ University(String):
- departments: List<Department>
- name: String
- userNamePersonMap: Map<String, Person>
+ getUserNamePersonMap(): Map<String, Person>
+ getDepartments(): List<Department>
Registration
+ Registration(Student, List<Course>):
- student: Student
- courses: List<Course>
+ getStudent(): Student
+ rejectRequest(): void
+ getCourses(): List<Course>
+ approveRequest(): void
+ addRequest(Advisor): void
Grade
+ Grade(String):
- gradeOver100: int
- gradeOver4: double
- letterGrade: String
+ getLetterGrade(): String
+ getGradeOver4(): double
- setGradeOver100(): void
- setGradeOver4(): void
Assistant
+ Assistant(int, String, String):
+ login(String, String): boolean
<<interface>>
IDisplayMenu
+ printMenu(String): void
Lecturer
+ Lecturer(int, String, String):
+ Lecturer(int, String, String, String, String):
~ lessonsTaught: List<Course>
+ getLessonsTaught(): List<Course>
+ login(String, String): boolean
CourseRegistrationSystem
+ CourseRegistrationSystem():
- choice: int
- logger: Logger
- input: Scanner
- university: University
+ courseRegistrationMenu(Student): void
+ start(): void
+ loginMenu(): void
+ mainMenu(): void
+ getInput(): int
+ exitProgram(): void
+ removeFromSchedule(Course, Student): void
+ addToSchedule(Course, Student): void
+ studentMenu(Student): void
+ advisorMenu(Advisor): void
+ printMenu(String): void
Advisor
+ Advisor(int, String, String, String, String):
- requestList: List<Registration>
- logger: Logger
- requestNumber: int
- studentsAdvised: List<Student>
- notification: Notification
+ getStudentsAdvised(): List<Student>
+ replyRequests(): void
+ setNotification(Notification): void
+ getRequests(): List<Registration>
+ getRequestList(): List<Registration>
+ login(String, String): boolean
+ printMenu(String): void
+ printRequests(): void
Student
+ Student(int, String, String, String, String, byte):
- semester: byte
- logger: Logger
- advisor: Advisor
- availableCoursesToDrop: List<Course>
- labSections: List<LaboratorySection>
- transcript: Transcript
- availableCoursesToAdd: List<Course>
- notification: Notification
- schedule: Course[][]
- draft: List<Course>
- hasRequest: boolean
+ setTranscript(Transcript): void
- checkThePrerequisiteAndCourseThatWasTaken(Course): boolean
+ getTranscript(): Transcript
+ getAdvisor(): Advisor
+ hasCourseOverlap(Course, boolean): boolean
+ isHasRequest(): boolean
- computeAvailableCoursesToDrop(): void
+ getSemester(): byte
- addFacultyTechnicalElective(): void
+ getSchedule(): Course[][]
- maxCoursesReached(): boolean
- addTechnicalElective(): void
- chooseLabSection(Course): void
+ setDraft(List<Course>): void
+ getDraft(): List<Course>
- computeAvailableMandatoryCourses(): void
- addCourseToDrop(): void
+ setHasRequest(boolean): void
- computeAvailableTEAndFTECourses(String): void
- addNonTechnicalElective(): void
- computeAvailableNTECourses(): void
+ setNotification(Notification): void
+ setPassword(String): void
+ login(String, String): boolean
+ setAdvisor(Advisor): void
- addMandatoryCourse(): void
+ removeCourseFromDraft(): void
+ calculateNumberOfCourses(): int
+ addCourseToDraft(): void
+ printMenu(String): void
+ showRequestStatus(): void
+ getDepartmentName(): String
+ sendRequest(): void
+ setUserName(String): void
+ getLabSections(): List<LaboratorySection>
Department
+ Department(String):
- labSectionCourseMap: Map<LaboratorySection, Course>
- studentIDStudentMap: Map<Integer, Student>
- advisorIDAdvisorMap: Map<Integer, Advisor>
- advisors: List<Advisor>
- assistants: List<Assistant>
- sectionCodeCourseMap: Map<String, Course>
- courses: List<Course>
- students: List<Student>
- departmentName: String
- courseCodeCourseMap: Map<String, Course>
- maxCourseNumber: byte
- transcripts: List<Transcript>
- laboratorySections: List<LaboratorySection>
- lecturers: List<Lecturer>
+ getAssistants(): List<Assistant>
+ getAdvisorIDAdvisorMap(): Map<Integer, Advisor>
+ getDepartmentName(): String
+ getLaboratorySections(): List<LaboratorySection>
+ getStudents(): List<Student>
+ getCourses(): List<Course>
+ getCourseSections(): List<LaboratorySection>
+ getStudentIDStudentMap(): Map<Integer, Student>
+ getCourseCodeCourseMap(): Map<String, Course>
+ getSectionCodeCourseMap(): Map<String, Course>
+ getAdvisors(): List<Advisor>
+ getLabSectionCourseMap(): Map<LaboratorySection, Course>
+ getMaxCourseNumber(): byte
+ getTranscripts(): List<Transcript>
+ getLecturers(): List<Lecturer>
Course
+ Course(String, String, String, int, byte, int, int, String):
- courseCode: String
- courseType: String
- preRequisiteCourses: List<Course>
- numberOfStudents: int
- capacity: int
- courseCredit: int
- lecturer: Lecturer
- laboratorySections: List<LaboratorySection>
- hour: int
- day: String
- courseName: String
- semester: byte
+ getCourseCredit(): int
+ getCourseCode(): String
+ getCourseName(): String
+ getHour(): int
+ setLecturer(Lecturer): void
+ getLaboratorySections(): List<LaboratorySection>
+ getCourseType(): String
+ getDay(): String
+ getPreRequisiteCourses(): List<Course>
+ semester(): byte
+ setNumberOfStudents(int): void
+ getNumberOfStudents(): int
+ addPreRequisiteCourse(Course): void
+ hasCapacity(): boolean