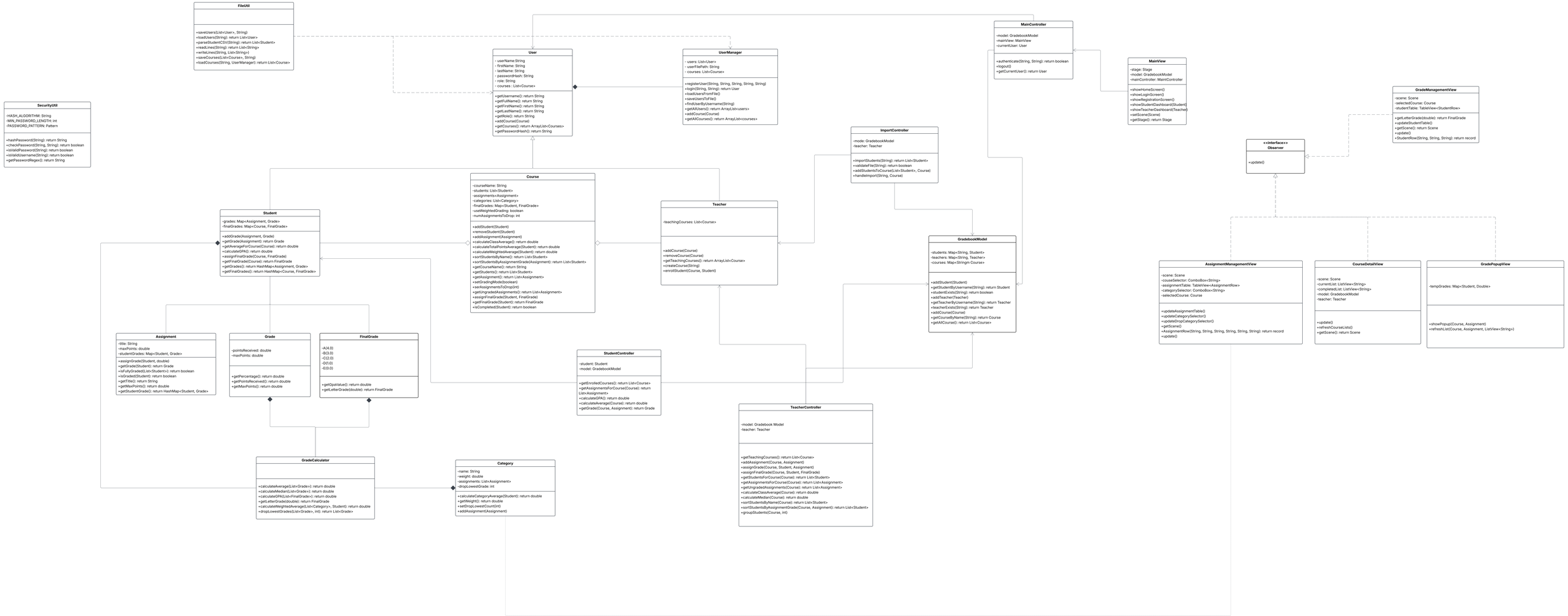
FileUtil
+saveUsers(List<User>, String)
+loadUsers(String): return List<User>
+parseStudentCSV(String): return List<Student>
+readLines(String): return List<String>
+writeLines(String, List<String>)
+saveCourses(List<Course>, String)
+loadCourses(String, UserManager): return List<Course>
User
- userName:String
- firstName: String
- lastName: String
- password#Hash: String
- role: String
- courses : List<Course>
+getUsername(): return String
+getFullName(): return String
+getFirstName(): return String
+getLastName(): return String
+getRole(): return String
+addCourse(Course)
+getCourses(): return ArrayList<Courses>
+getPasswordHash(): return String
UserManager
- users: List<User>
- userFilePath: String
- courses: List<Course>
+registerUser(String, String, String, String, String)
+login(String, String): return User
+loadUsersFromFile()
+saveUsersToFile()
+findUserByUsername(String)
+getAllUsers(): return ArrayList<users>
+addCourse(Course)
+getAllCourses(): return ArrayList<courses>
MainController
-model: GradebookModel
-mainView: MainView
-currentUser: User
+authenticate(String, String): return boolean
+logout()
+getCurrentUser(): return User
MainView
-stage: Stage
-model: GradebookModel
-mainController: MaintController
+showHomeScreen()
+showLoginScreen()
+showRegistrationScreen()
+showStudentDashboard(Student)
+showTeacherDashboard(Teacher)
+setScene(Scene)
+getStage(): return Stage
GradeManagementView
-scene: Scene
-selectedCourse: Course
-studentTable: TableView<StudentRow>
+getLetterGrade(double): return FinalGrade
+updateStudentTable()
+getScene(): return Scene
+update()
+StudentRow(String, String, String): return record
SecurityUtil
-HASH_ALGORITHM: String
-MIN_PASSWORD_LENGTH: int
-PASSWORD_PATTERN: Pattern
+hashPassword(String): return String
+checkPassword(String, String): return boolean
+isValidPassword(String): return boolean
+isValidUsername(String): return boolean
+getPasswordRegex(): return String
Course
-courseName: String
-students: List<Student>
-assignments<Assignment>
-categories: List<Category>
-finalGrades: Map<Student, FinalGrade>
-useWeightedGrading: boolean
-numAssignmentsToDrop: int
+addStudent(Student)
+removeStudent(Student)
+addAssignment(Assignment)
+calculateClassAverage(): return double
+calculateTotalPointsAverage(Student): return double
+calculateWeightedAverage(Student): return double
+sortStudentsByName(): return List<Student>
+sortStudentsByAssignmentGrade(Assignment): return List<Student>
+getCourseName(): return String
+getStudents(): return List<Student>
+getAssignment(): return List<Assignment>
+setGradingMode(boolean)
+setAssignmentsToDrop(int)
+getUngradedAssignments(): return List<Assignment>
+assignFinalGrade(Student, FinalGrade)
+getFinalGrade(Student): return FinalGrade
+isCompleted(Student): return boolean
Teacher
-teachingCourses: List<Course>
+addCourse(Course)
+removeCourse(Course)
+getTeachingCourses(): return ArrayList<Course>
+createCourse(String)
+enrollStudent(Course, Student)
ImportController
-mode: GradebookModel
-teacher: Teacher
+importStudents(String): return List<Student>
+validateFile(String): return boolean
+addStudentsToCourse(List<Student>, Course)
+handleImport(String, Course)
<<interface>>
Observer
+update()
Student
-grades: Map<Assignment, Grade>
-finalGrades: Map<Course, FinalGrade>
+addGrade(Assignment, Grade)
+getGrade(Assignment): return Grade
+getAverageForCourse(Course): return double
+calculateGPA(): return double
+assignFinalGrade(Course, FinalGrade)
+getFinalGrade(Course): return FinalGrade
+getGrades(): return HashMap<Assignment, Grade>
+getFinalGrades(): return HashMap<Course, FinalGrade>
GradebookModel
-students: Map<String, Student>
-teachers: Map<String, Teacher>
-courses: Map<String<m Course>
+addStudent(Student)
+getStudentByUsername(String): return Student
+studentExists(String): return boolean
+addTeacher(Teacher)
+getTeacherByUsername(String): return Teacher
+teacherExists(String): return Teacher
+addCourse(Course)
+getCourseByName(String): return Course
+getAllCourses(): return List<Course>
AssignmentManagementView
-scene: Scene
-couseSelector: ComboBox<String>
-assignmentTable: TableView<AssignmentRow>
-categorySelector: ComboBox<String>
-selectedCourse: Course
+updateAssignmentTable()
+updateCategorySelector()
+updateDropCategorySelector()
+getScene()
+AssignmentRow(String, String, String, String, String, String): return record
+update()
CourseDetailView
-scene: Scene
-currentList: ListView<String>
-completedList: ListView<String>
-model: GradebookModel
-teacher: Teacher
+update()
+refreshCourseLists()
+getScene(): return Scene
GradePopupView
-tempGrades: Map<Student, Double>
+showPopup(Course, Assignment)
+refreshList(Course, Assignment, ListView<String>)
Assignment
-title: String
-maxPoints: double
-studentGrades: Map<Student, Grade>
+assignGrade(Student, double)
+getGrade(Student): return Grade
+isFullyGraded(List<Student>): return boolean
+isGraded(Student): return boolean
+getTitle(): return String
+getMaxPoints(): return double
+getStudentGrade(): return HashMap<Student, Grade>
Grade
-pointsReceived: double
-maxPoints: double
+getPercentage(): return double
+getPointsReceived(): return double
+getMaxPoints(): return double
FinalGrade
-A(4.0)
-B(3.0)
-C(2.0)
-D(1.0)
-E(0.0)
+getGpaValue(): return double
+getLetterGrade(double): return FinalGrade
StudentController
-student: Student
-model: GradebookModel
+getEnrolledCourses(): return List<Course>
+getAssignmentsForCourse(Course): return List<Assignment>
+calculateGPA(): return double
+calculateAverage(Course): return double
+getGrade(Course, Assignment): return Grade
TeacherController
-model: Gradebook Model
-teacher: Teacher
+getTeachingCourses(): return List<Course>
+addAssignment(Course, Assignment)
+assignGrade(Course, Student, Assignment)
+assignFinalGrade(Course, Student, FinalGrade)
+getStudentsForCourse(Course): return List<Student>
+getAssignmentsForCourse(Course): return List<Assignment>
+getUngradedAssignments(Course): return List<Assignment>
+calculateClassAverage(Course): return double
+calculateMedian(Course): return double
+sortStudentsByName(Course): return List<Student>
+sortStudentsByAssignmentGrade(Course, Assignment): return List<Student>
+groupStudents(Course, int)
GradeCalculator
+calculateAverage(List<Grade>): return double
+calculateMedian(List<Grade>): return double
+calculateGPA(List<FinalGrade>): return double
+getLetterGrade(double): return FinalGrade
+calculateWeightedAverage(List<Category>, Student): return double
+dropLowestGrades(List<Grade>, int): return List<Grade>
Category
-name: String
-weight: double
-assignments: List<Assignment>
-dropLowestGrade: int
+calculateCategoryAverage(Student): return double
+getWeight(): return double
+setDropLowestCount(int)
+addAssignment(Assignment)