General Observations

- **Administrator Class**: Matches the class diagram.
- **Professor Class**: Matches the class diagram.
- **Course Class**: Matches the class diagram.
- **Parents Class**: Missing in the code but present in the class diagram.
- **Task Class**: Empty in the class diagram but contains a method in the code.
- **Menu Class**: Contains a method not present in the class diagram.
- **Student Class**: Contains a method (grade method) not in the class diagram.
- **AcademyGradeRegister Class**: Matches the class diagram but includes a method with "Conditional Complexity" due to a switch statement and multiple if-else statements, making it a "Long Method".
- **Menu Class**: Lacks the USER_FILE = "users.json" attribute in the class diagram, but other variables and methods are well used.
- **User Class**: Matches the class diagram, well-structured, follows good coding practices.

Detailed Observations

Grade Class

- **Attributes**:
    - studentId: Matches between diagram (studentId : String) and code (private String studentId).
    - subjectCode: Matches between diagram (subjectCode : String) and code (private String subjectCode).
    - grade: Matches between diagram (grade : double) and code (private double grade).
- **Code Quality**:
    - **Uncommunicative Names**: Names are clear and communicative.
    - **Inconsistent Names**: Consistent throughout the code.
    - **Long Methods**: Methods are short and specific.
    - **Duplicate Code**: No duplicate code present.
    - **Class Explosion**: No signs of class explosion; class is concrete and specific.
    - **Large Classes**: Small and focused on a single responsibility.
    - **Conditional Complexity**: No complex conditional logic.
    - **Redundant or Meaningless Comments**: Remove initial comments generated by NetBeans and add meaningful comments.

    - **Dead Code**: No dead code present.
    - **Speculative Generality**: No signs of speculative generality.
    - **Additional Recommendations**:
        - **Use of Gson Annotations**: Verify correct use of @Expose annotations.
        - **Data Validation**: Implement validation for grade values to prevent invalid data (e.g., negative grades).

Subject Class

- **Attributes**:
    - code: Matches between diagram (code : String) and code (private String code).
    - name: Matches between diagram (name : String) and code (private String name).
    - credits: Matches between diagram (credits : int) and code (private int credits).
- **Code Quality**:
    - **Uncommunicative Names**: Names are clear and communicative.
    - **Inconsistent Names**: Consistent throughout the code.
    - **Long Methods**: Methods are short and specific.
    - **Duplicate Code**: No duplicate code present.
    - **Class Explosion**: No signs of class explosion; class is concrete and specific.
    - **Large Classes**: Small and focused on a single responsibility.
    - **Conditional Complexity**: No complex conditional logic.
    - **Redundant or Meaningless Comments**: Remove initial comments generated by NetBeans and add meaningful comments.
    - **Dead Code**: No dead code present.
    - **Speculative Generality**: No signs of speculative generality.
    - **Additional Recommendations**:
        - **Use of Gson Annotations**: Verify correct use of @Expose annotations.
        - **Data Validation**: Implement validation for credits to prevent invalid data (e.g., negative credits).

IndividualActivity Class

- **Attributes**:
    - grade: Matches between the implicit diagram expectation (grade : double) and code (private double grade).

- **Code Quality**:
    - **Uncommunicative Names**: Names are clear and communicative.
    - **Inconsistent Names**: Consistent throughout the code.
    - **Long Methods**: Methods are short and specific.
    - **Duplicate Code**: No duplicate code present.
    - **Class Explosion**: No signs of class explosion; class is concrete and specific.
    - **Large Classes**: Small and focused on a single responsibility.
    - **Conditional Complexity**: No complex conditional logic.
    - **Redundant or Meaningless Comments**: Remove initial comments generated by NetBeans and add meaningful comments.
    - **Dead Code**: No dead code present.
    - **Speculative Generality**: No signs of speculative generality.
    - **Additional Recommendations**:
        - **Data Validation**: Implement validation for grade values to prevent invalid data (e.g., negative grades).

QuimestralExam Class

- **Attributes**:
    - grade: Matches between the implicit diagram expectation (grade : double) and code (private double grade).
- **Code Quality**:
    - **Uncommunicative Names**: Names are clear and communicative.
    - **Inconsistent Names**: Consistent throughout the code.
    - **Long Methods**: Methods are short and specific.
    - **Duplicate Code**: No duplicate code present.
    - **Class Explosion**: No signs of class explosion; class is concrete and specific.
    - **Large Classes**: Small and focused on a single responsibility.
    - **Conditional Complexity**: No complex conditional logic.
    - **Redundant or Meaningless Comments**: Remove initial comments generated by NetBeans and add meaningful comments.
    - **Dead Code**: No dead code present.
    - **Speculative Generality**: No signs of speculative generality.

- **Additional Recommendations**:
  - **Data Validation**: Implement validation for grade values to prevent invalid data (e.g., negative grades).

This inspection reveals a generally well-structured codebase with good alignment between the code and the class diagrams. However, some areas require attention, particularly in ensuring data validation and removing unnecessary comments.