GRADING SYSTEM IN JAVA

1. Describe the overall design of your grading system in Java.

Answer:
I can design a flexible system accepting various assessments (exams, assignments, etc.) with configurable weights and grading scales (percentage, letter grades, etc.).

2. How would you handle different types of assessments with varying score ranges?

Answer:
I can use separate score normalization methods for each assessment type while maintaining an overall weighted average considering individual weights.

3. How would you ensure the grading system is fair and consistent across multiple students?

Answer:
The system can apply the same grading logic and criteria to all students, avoiding manual overrides unless justifiable and documented.
Data Structures and Algorithms:

4. What data structures would you use to store student information, grades, and weights?

Answer:
I can use Maps (e.g., HashMap) to store student data with names and IDs as keys. Nested Maps or Lists can store assessment scores and weights.

5. How would you calculate the weighted average for a student's final grade?

Answer:
I can iterate through each assessment, multiply the score by its weight, and sum the weighted scores. The final grade is the sum divided by the total weight.

6. How would you implement different grading scales (e.g., percentage to letter grade conversion)?

Answer:
I can use configurable ranges and corresponding letter grades in a Map or switch statement. The final grade is converted based on its position within the defined ranges.
User Interface and Input/Output:

7. How would you allow users to input student information, assessment scores, and weights?

Answer:
I can provide a command-line interface or a GUI using Java Swing or JavaFX for data entry. User-friendly validation can ensure input accuracy.

8. How would you display the calculated grades and relevant information for each student?

Answer:
I can present a formatted table or report showing student names, assessment scores, weights, final grades, and optionally letter grades.
Extensibility and Error Handling:

9. How would you design your system to be extensible for future modifications?

Answer:
I can use interfaces and abstract classes for assessments, allowing easy addition of new assessment types without code changes in the core logic.

10. How would you handle potential errors and exceptions (e.g., invalid input, data inconsistencies)?

Answer:
I can implement input validation, exception handling with informative messages, and logging mechanisms to identify and troubleshoot issues.