🧠 Review & Coding Practice

✅ Review Yourself:

• Do you understand how classes and objects work?  
yes, a class is a blueprint (like student) and an object is an instance and instance

• Can you explain why private is used in class attributes?

It protects data by restricting direct access; other classes must use getters/setters, ensuring encapsulation.

• What’s the purpose of a constructor in Java?  
A constructor initializes an object’s data when it's created new Student(name, id, grade) sets up the object.

• Why use ArrayList instead of a fixed-size array?  
ArrayList can **grow/shrink** dynamically, whereas arrays have a **fixed size** once created.

• When should you use scanner.next() vs scanner.nextLine()?

scanner.next() reads **one word** (stops at space), scanner.nextLine() reads the **entire line** including spaces.

• How is a Lambda Expression used in this project?

for sorting

• How does Comparator help sort the list?

It defines custom sorting logic, e.g., by grade or name

• What’s the difference between == and .equals() in Java?

= = compares memory locations (reference), .equals() compares **actual content** (values).

• How would you add, update, and delete students in this system?

Add: students.add(new Student(...))

**update:** Loop, find by ID, call setName() or setGrade()

**Delete:** Loop, find by ID, then students.remove(student)

• What does each menu option represent logically?  
Each option maps to a **specific function** like adding, viewing, updating, or exiting — it models **user actions** on the system.

🌟 Bonus Missions (Challenge Yourself):

• Show students who failed (grade < 60)

for (Student s : students) {

if (s.getGrade() < 60) s.displayInfo();

}

• Add a feature to update a student’s name using ID  
for (Student s : students) {

if (s.getId() == targetId) s.setName(newName);

}

• Create a confirmation prompt before deleting a student  
System.out.print("Are you sure? (yes/no): ");

if (scanner.nextLine().equalsIgnoreCase("yes")) students.remove(student);

• Count how many students have grades between 80 and 90  
long count = students.stream().filter(s -> s.getGrade() >= 80 && s.getGrade() <= 90).count();

• Return the student with the second highest grade

students.sort(Comparator.comparingDouble(Student::getGrade).reversed());

Student secondTop = students.size() > 1 ? students.get(1) : null;

• Add a login system before showing the menu (admin only)

System.out.print("Enter admin password: ");

if (!scanner.nextLine().equals("admin123")) System.exit(0);

• Create an export option to save student data in a file

FileWriter writer = new FileWriter("students.txt");

for (Student s : students) writer.write(s.getName() + "," + s.getId() + "," + s.getGrade() + "\n");

writer.close();

• Group students by performance level (Excellent, Good, Fail)

for (Student s : students) {

String level = s.getGrade() >= 90 ? "Excellent" : s.getGrade() >= 60 ? "Good" : "Fail";

System.out.println(s.getName() + ": " + level);

}