Project Name: “Prime Academy Simulator”

Concept: You’re building a mini simulation system for a fictional academy for “prime minds” . The system will manage students, classes, exams, and performance stats. It’s modular, mathematical, and object-oriented — exactly the kind of playground to apply the first 9 chapters.

How Each Chapter Fits In

Chapter	How it applies in the project
1. Introduction to Computers, Programs, Java	Create the main Java program structure, console interaction, input/output handling.
2. Elementary Programming	Variables for student info, basic calculations (grades, averages).
3. Selections	Conditional logic: e.g., pass/fail students, assign honors based on grades.
4. Math Functions, Characters, Strings	Compute GPAs, string manipulations for names, ID formatting, sorting students alphabetically.
5. Loops	Iterating over lists of students, classes, exams; printing reports.
6. Methods	Reusable methods: calculate average, print report, assign grades, etc.
7. Single-Dimensional Arrays	Store student scores in 1D arrays for individual exams.
8. Multidimensional Arrays	Store all students’ scores across multiple exams in a 2D array.
9. Objects and Classes	Classes like `Student`, `Exam`, `Course`, with instance/static variables, constructors, getters/setters, immutable fields, and methods.

Core Classes & Objects

1. **Student**
    - Fields: `name`, `id`, `birthDate` (object), `examScores` (1D array), etc.
    - Methods: calculate GPA, print info, etc.
2. **BirthDate**
    - Immutable date object (practice immutability).
3. **Course / Exam**
    - Fields: course name, max score, student scores (2D array).
    - Methods: add scores, calculate averages, assign grades.
4. **Academy**

    - Manages all students, courses, exams.
    - Static fields for total students, total courses.

Features / Mini Tasks

- Add students with personal info.
- Assign courses and exams.
- Input exam scores.
- Print student report cards (using loops + string formatting).
- Compute course averages and top performers (use math functions + loops + selections).
- Practice **passing objects to methods**: e.g., a method that updates exam scores by passing `Student` objects.
- Experiment with **immutable objects**: BirthDate objects cannot be changed once created.
- Use **arrays** for exam scores (1D) and multiple exams (2D).
- Implement a **menu-driven console system** using selections and loops.

Stretch Goals (Prime-Level Challenges)

- Sort students by GPA or alphabetical order (strings + arrays).
- Track history of score changes using stack-like logic.
- Add static methods to compute overall academy statistics.
- Optionally, serialize data to save/load students.

Why this is perfect for you:

- Covers **all 9 chapters**.
- Mixes **mathematical reasoning** (grades, GPAs, averages).
- Lets you apply **OOP principles** fully.
- Gives room to **experiment creatively** without being mundane.