Programming Laboratory-I & II
Course Policy, Grading & Guide

This courses bridge your foundational courses with hands-on software engineering practices. This guide outlines the course execution structure, grading policies, attendance regulations, technical workflows, and debugging methodologies required throughout the semesters.

1. Course Execution & Weekly Laboratory Workflow

Laboratory sessions are held weekly in computer labs. Each session centers around a practical **Lab Sheet** containing designated implementation steps.

1.1 In-Class Evaluation (90 Points)

- Each lab sheet consists of guided steps totaling **90 points**.
- Toward the end of the laboratory period, **Teaching Assistants (TAs)** will inspect your running project and source code.
- You will receive an in-lab score up to a maximum of 90 points based on the tasks completed during lab hours.

1.2 At-Home Completion & GitHub Classroom (10 Points)

- If you have missing steps or minor flaws from the in-class check, you may complete them at home.
- **Flawless & Complete Submission (+10 Points):** If all missing steps are resolved completely, compile cleanly, and run without errors when pushed to your official GitHub Classroom repository before the deadline, you earn the remaining **10 points**, bringing your lab score up to **100**.
- **Incomplete or Erroneous Submission (+0 Points):** Submissions that still contain missing requirements, syntax errors, or runtime exceptions receive **0 additional points**. Your grade remains strictly your in-lab score (capped at a maximum of 90).

2. Assessment, Examination & Passing Criteria

2.1 Midterm and Final Grade Calculation Formula

Both the Midterm and Final exam scores incorporate your continuous laboratory performance leading up to each exam. The overall grade for each examination period is computed using the following formula:

$$\text{Exam Grade} = \frac{(\text{Lab Sheet Average up to Exam} \times 0.40) + (\text{Exam Score} \times 0.60)}{2}$$

- **Lab Sheet Average:** The arithmetic mean of all evaluated lab sheets assigned prior to that specific exam.
- **Exam Score:** The score earned in the written/practical midterm or final exam.

2.2 Course Passing Thresholds

To pass the course, students must satisfy **both** of the following requirements simultaneously at the end of the semester:

1. **Final Exam:** A minimum score of **50 / 100** on the Final Exam.
2. **Overall Course Average:** A minimum overall weighted average of **50 / 100**.

Failure to meet either threshold results in a failing grade for the course.

3. Attendance & Absence Regulations

Attendance is strictly tracked and mandatory for professional laboratory training.

- **Minimum Attendance Requirement: 80%** of all scheduled laboratory hours.
- **Maximum Absence Allowance: 20%** of scheduled laboratory hours.
- **Unexcused Absence:** If you miss a lab session without an officially documented, approved excuse:
    - Your lab sheet score for that week is recorded as **0**.
- **Excused Absence:** If you provide official medical or institutional documentation approved by the department:
    - That week’s lab sheet is **excluded from your grade calculation** (it will not lower your lab sheet average).

4. Technical Guide: The Three W's of Debugging

An IDE is a diagnostic instrument. When your application fails, avoid random changes. Systematically trace the failure using the Three W's:

1. Identify "WHAT" Happened

Read the error message carefully before looking at your code:

- **Compilation / Syntax Errors:** The code violates syntax or type rules. The program cannot be built.
- **Runtime Exceptions:** The code compiles, but crashes during execution due to illegal operations (e.g., NullPointerException, IndexOutOfBoundsException, ZeroDivisionError).
- **Logical Errors:** The application runs smoothly but produces incorrect output.

2. Locate "WHERE" It Happened

Inspect the console's **Stack Trace** from top to bottom:

- Locate the highest stack trace line pointing to **your own source code** (e.g., at com.lab.StudentService.calculateAverage(StudentService.java:48)).

- In modern IDEs, click the blue hyperlink to jump directly to the exact statement where execution stopped.

3. Diagnose "WHY" It Happened

- **Trace Variable States:** Is an object reference uninitialized (null)? Has a loop index exceeded the bounds of an array or list?
- **Use the Interactive Debugger:**
    1. Set a **Breakpoint** directly before the suspicious line.
    2. Launch the application in **Debug Mode** (bug icon).
    3. Use **Step Over** ($F8$ / $F10$) to advance line by line.
    4. Use **Step Into** ($F7$ / $F11$) to inspect nested method calls.
    5. Check the **Variables / Watch panel** to observe data state changes in real time.

5. Git & GitHub Classroom Submission Workflow

Every weekly sheet must be maintained and finalized via GitHub Classroom.

Repository Setup

1. Open the assignment link shared in class by the instructors.
2. Accept the assignment using your linked GitHub account. GitHub Classroom will automatically generate a private repository under the course organization (e.g., lab-01-oop-basics-username).
3. Finally, the completed worksheet codes are uploaded to the repository via file upload or by using Git commands.

Verification via Browser

Always log into GitHub via your browser after pushing:

1. Confirm your latest commit is visible on the repository dashboard.
2. Ensure no build artifacts, binaries, or IDE-specific directories (.idea/, .vscode/, bin/, obj/, target/) were pushed.