Advanced Programming (OOP) Lab

Assignment–5

Report Instructions

This document specifies the mandatory structure and assignment-specific questionnaires for the handwritten report corresponding to Assignment–5: Packages, Wrapper Classes, Streams, and File Handling. The report will be evaluated for modular design reasoning, correct abstraction, and understanding of Java I/O and stream-based processing.

General Submission Rules

- The report must be handwritten and submitted individually.
- Do not rewrite complete class implementations; include only selected code fragments.
- Explanations must focus on design intent, data flow, and abstraction boundaries.
- All implemented requirements and attempted exercises must be reflected in the report.
- Reports showing mechanical or copied explanations will be penalized.

__Tip__: Use multiple colors for highlighting your explanations, references and annotations.

Section A: Package and Modular Design Rationale

Explain why multiple packages were created in this assignment. Describe the responsibility of each package and justify how package-level separation improves maintainability, reuse, and clarity.

Section B: Utility Class Responsibilities

For any three utility classes you implemented:
1. State the primary responsibility of the class.
2. Explain why its methods are declared static or non-static.
3. Describe how this class interacts with other classes across packages.

Section C: Wrapper Class Usage Analysis

Explain where and why wrapper classes are used in this assignment. Describe the conversion process from file data to numeric values. State one error scenario that can occur during conversion and how it is handled.

Section D: File I/O Flow Explanation

Trace the flow of data from file input to file output:
- How files are opened and read
- How data is processed in memory
- How results are written back to a file

Include one annotated code fragment illustrating safe file handling.

Section E: Stream-Based Processing Reasoning

Explain how Java Streams are used for filtering, sorting, or aggregation. Answer the following:
1. What advantage do streams provide over traditional loops?
2. Identify one stream operation used and explain its role in the pipeline.

Section F: Error Handling and Robustness

Describe how file-related exceptions are handled in your program. Explain what happens when a file is missing, unreadable, or empty. Justify your choice of exception handling strategy.

Section G: Exercise or Extension Explanation

If you implemented any additional exercises (mathutilities, appending files, longest word, etc.):
- Describe the algorithm used
- Explain how correctness is ensured
- Discuss how this feature integrates with existing utilities

Section H: What-If Design Analysis

Answer any two of the following:
1. What changes if buffered streams are replaced with unbuffered streams?
2. How would the design change if files are very large?
3. What problems arise if all utilities are placed in a single package?
4. What breaks if streams are replaced entirely with loops?

Section I: Debugging and Learning Reflection

Describe one non-trivial issue faced while implementing file or stream operations. Explain how the issue was diagnosed and what deeper understanding was gained.

Section J: AI Usage Disclosure

Declare whether AI or online tools were used. Clearly specify:
- Which part of the design or logic was assisted
- How the output was verified, modified, or improved