Assignment 2: Algorithmic Analysis and Peer Code Review — Code Evaluation Report

1. General Evaluation

Overall, the submitted Selection Sort implementation is functionally correct and adheres to the main assignment requirements. However, several issues were found that may reduce the overall grade during evaluation. These include algorithmic inefficiencies, minor code style violations, and lack of detailed analysis reporting.

2. Detected Errors and Issues

- ■ **Lack of time precision consistency:** Recorded times use commas instead of dots for decimal separation (e.g., 6,024 instead of 6.024).
- ■ **No input validation:** The algorithm assumes non-empty arrays. Edge cases like empty arrays or null references are not explicitly handled.
- ■ **Performance tracking precision:** Time measurements below 1 ms appear as 0,000 or 0,001, which makes micro-benchmarks inaccurate.
- ■ **Early termination logic missing full validation:** Though mentioned, the early-termination condition is not clearly visible in the code snippet and may not properly detect already sorted arrays.
- ■■ **Inconsistent CSV output formatting:** The exported results file mixes comma and dot decimal separators; this may cause parsing issues when importing into analysis tools.
- ■■ **Missing memory usage metrics:** The assignment requires memory profiling, but current implementation tracks only comparisons, moves, and time.
- ■■ **Lack of command-line interface (CLI) options:** CLI input patterns and sizes appear hard-coded instead of being user-configurable.
- ■■ **Limited empirical validation:** The report does not include performance plots or trend verification for theoretical complexity alignment.

3. Recommendations for Improvement

- Implement strict decimal formatting (use `String.format(Locale.US, ...)` for time precision).
- Add input validation and edge case handling for empty or null arrays.
- Use `System.nanoTime()` for more accurate time measurements in microseconds.
- Ensure early termination logic properly checks for already sorted arrays before performing unnecessary iterations.
- Normalize CSV export to use `.` as the decimal separator and verify consistency.
- Add optional CLI parameters for array size, input pattern, and number of repetitions.
- Include memory profiling metrics using `Runtime.getRuntime().totalMemory()` and `freeMemory()`.

- Generate time-complexity validation plots (n vs time) to confirm $O(n^2)$ and $\Omega(n)$ behavior.

4. Compliance Summary

- Implementation Quality: 8/10 - Analysis Depth: 7/10 - Empirical Validation: 6/10 - Communication & Documentation: 8/10 **Overall Grade Estimate:** B+

5. Conclusion

The Selection Sort implementation demonstrates solid technical understanding and proper integration with Maven and testing tools. However, the lack of detailed empirical validation, missing CLI flexibility, and inconsistent data formatting reduce its research-grade quality. Fixing the issues listed above will likely elevate this project to an A-grade level.