Homework 6: Reflection

1. **In retrospect, what could you have done better to reduce the time you spent solving this assignment?**

   I could have approached the assignment with a better understanding of Java's package structure and compilation process. A significant amount of time was spent fixing package dependencies and serialization issues that could have been avoided with proper planning. Specifically, I should have:

   - Conducted a comprehensive initial code review to identify all dependencies between packages (particularly between hw4 and hw6)
   - Understood Submitty's strict compilation requirements (-Werror flag) from the beginning
   - Established a proper test environment that mimicked Submitty's compilation settings
   - Addressed serialization properly by adding serialVersionUID to exception classes and using transient on non-serializable fields in the SharedSetCountsTask
   - Extracted the BenchmarkRunner and BenchmarkResult classes from BenchmarkLegoPaths class earlier in the process

   For the implementation of the benchmark system, I should have first created clear class diagrams to visualize the relationships between classes before coding. This would have helped me identify the potential issues with inner classes being inaccessible to tests. By extracting the benchmark functionality into separate public classes from the start, I could have avoided the refactoring work later.

   Additionally, I should have created more focused unit tests for each component early in development. The fact that tests ran locally but failed on Submitty indicates a gap in test coverage and environment configurations that could have been identified sooner.

2. **What could the Principles of Software staff have done better to improve your learning experience in this assignment?**

   The assignment would have benefited from more explicit guidance on:

   - The strict compilation requirements in the grading environment (such as -Werror treating warnings as errors)
   - Best practices for structuring code with multiple dependent packages
   - Proper handling of serialization in Java, particularly in the context of exception classes and inner classes
   - How to design test cases that properly verify benchmark functionality
   - Guidelines on package visibility and access modifiers for test accessibility

   It would have been helpful to have a dedicated section on common compilation issues and their resolutions, especially regarding inner classes, serialization warnings, and cross-package dependencies. A sample project structure demonstrating proper organization of components that depend on previous assignments would have provided a valuable reference.

   The assignment could have included a more comprehensive troubleshooting guide for common Submitty errors. For instance, knowing that serialization warnings are treated as errors would

have significantly reduced debugging time. Additionally, clearer examples of how test discovery works in the grading environment would have helped ensure that tests were structured properly.

3. **What do you know now that you wish you had known before beginning the assignment?**

   I now have a much deeper understanding of several key aspects that would have been valuable from the start:

   - The importance of properly handling serialization in Java classes, including adding serialVersionUID and making non-serializable fields transient
   - How package-private inner classes cannot be properly accessed by test classes, necessitating extraction to separate public classes
   - The significance of proper package structure and import management across multiple homework assignments
   - How strict compilation settings (-Werror) can turn warnings into errors during grading
   - The interplay between test discovery mechanisms and class visibility
   - Techniques for creating local build tasks that mimic the grading environment's strict compilation settings

   I've gained valuable insights into designing benchmarking systems, particularly how to structure them for testability. The experience of fixing serialization issues and package dependencies has taught me to be more deliberate about class design from the beginning.

   The most important lesson was understanding how to debug compilation issues when the local environment behaves differently from the grading environment. Creating custom Gradle tasks to replicate strict compilation settings has given me a powerful tool for future assignments.

   Most significantly, I've learned to more carefully consider the visibility and accessibility of classes in relation to their intended use in tests. Designing with testability in mind from the start would have prevented many of the issues encountered.