**Refactoring Summary**

This document summarizes the refactoring process undertaken for the project. The objective was to address code smells, improve readability, enhance maintainability, and ensure robust test coverage. Below are the details of the issues identified, changes made, and their impact.

**1. Identified Code Smells**

- Duplicate Code: In `DataAnalyzer`, multiple methods for calculating statistical metrics like mean, median, and mode were duplicated for both `Double` and `Long` data types.  
- Single Responsibility Principle (SRP) Violations: The `Main` class handled user input, business logic, and report generation, violating separation of concerns.  
- Lack of Reusability: Validation logic for file types and data processing was hardcoded, limiting flexibility.  
- Poor Separation of Concerns: Core application logic was mixed with user interaction code in `Main`.  
- Limited Extensibility: The `ReportGenerator` was restricted to generating reports in `.txt` format only.

**2. Key Refactoring Changes**

1. DataAnalyzer Improvements:  
 - Unified methods for `Double` and `Long` using generics to eliminate duplication.  
 - Extracted common logic, such as sorting and frequency map creation, into reusable helper methods.  
 - Added validation for empty datasets with meaningful error messages.

2. CSVReader Enhancements:  
 - Moved file validation into a dedicated helper method for better reusability.  
 - Used custom exceptions to provide clear feedback when invalid files are encountered.

3. Main Class Redesign:  
 - Delegated data processing tasks to a new `DataProcessingService` class.  
 - Focused `Main` solely on user interaction, simplifying its responsibilities.  
 - Encapsulated analysis results in a new `AnalysisResults` class to better structure data.

4. ReportGenerator Improvements:  
 - Extracted file extension validation into a reusable method.  
 - Enhanced flexibility for supporting future file formats.

5. Unit Tests:  
 - Added comprehensive tests for `DataAnalyzer`, covering both standard and edge cases.  
 - Incorporated logging into tests to aid debugging and traceability.

**3. Impact of Refactoring**

- Improved Code Readability: Consolidating duplicate logic and introducing helper methods made the code cleaner and easier to understand.  
- Enhanced Maintainability: Decoupling functionality into smaller, focused classes allows for easier updates and feature additions.  
- Increased Testability: The separation of concerns facilitated the creation of granular unit tests for individual components.  
- Better Error Handling: Custom exceptions and detailed error messages provide better feedback and robustness.  
- Flexibility for Future Enhancements: The `ReportGenerator` is now prepared for additional report formats like `.json` or `.xml`.

**4. Next Steps**

1. Adopt Advanced Logging Frameworks:  
 Instead of relying on basic logging mechanisms like `System.out.println` or `java.util.logging`, integrate advanced logging frameworks such as SLF4J or Log4j. These frameworks provide richer functionality, such as log levels, message formatting, and integration with external tools. This will enhance debugging and make log management more efficient for larger projects.

2. Expand Report Format Support:  
 Extend the `ReportGenerator` to support additional formats such as JSON or XML. This can be achieved by implementing a strategy pattern, allowing the generator to dynamically switch between different output formats. This will make the application more versatile and adaptable to varied use cases.

3. Optimize Performance:  
 For projects dealing with large datasets, consider optimizing data processing by leveraging parallel streams in Java or batch processing techniques. This will improve performance and scalability when handling high volumes of data.

The refactoring process has significantly improved the codebase by addressing identified issues, making the project more modular, testable, and scalable. These changes not only align with software engineering principles but also ensure the project is prepared for future enhancements.