Software Engineering ACS56000

HW-11 – Refactoring

Summary

In the original HealthCalculation code, there were a lot of code smells. The primary issue was large class where the main method was handling multiple responsibilities including user input, file reading, data processing, and output display. This violated the single Responsibility Principle and made the code difficult to test and maintain. There was also data clumping, where related primitive data types were used together but not properly encapsulated like duration, current pulse, max pulse, and calories. Additionally, the error handling was inconsistent and generic, with some exceptions being caught and others potentially propagating up the call stack.

The refactoring addressed these smells by first introducing proper unit tests to ensure functionality wouldn’t break during modifications. The read-data() method was modified to return a List<HealthCalculation> instead of using it internally, making it more testable and reusable. Error handling was improved to provide more specific error messages and consistent exception handling patters. The changes made the code more maintainable as each component became more focused and testable. I also made the file path handling more robust to prevent potential file-not-found issues during testing.

The impact of these changes is significant in several ways. First the code is now more reliable with proper error handling and validation. Second, the addition of unit test provides confidence when making future modifications. Third, the code is more maintainable as each component has a single, well-defined responsibility. While the original functionality remains unchanged, the code is now structured in a way that makes it easier to modify, test and extend in the future. The separation of concerns also makes it easier for other developers to understand and work with the code, as each method has a clear purpose and the flow of data is more explicit.