Report

on

Refactoring project

Chemistry Calculator

Submitted to

Dipok Chandra Das

Assistant Professor, IIT, NSTU

Submitted by

Md. Mahabub Alam (ASH1825003M)

Abrar Hossain (ASH1825005M)

Nishat Tasnim Tamanna (BKH1825006F)

Md. Faisal Ahmed (ASH1825015M)

Moon Moon Das (BKH1825027F)

Introduction

Refactoring is the process of restructuring and improving the design of an existing codebase without changing its external behavior. The goal of the refactoring process is to improve the readability, maintainability, and extensibility of the codebase, making it easier for developers to understand and modify the code.

In this report, we will discuss the refactoring of a project named **Chemistry Calculator**, which provides many functionalities like-

- Equation balancing
- Electron configuration
- Molar msss
- Calculations of concentration and titration

In this document the refactoring process are performed by following these steps-

- Identification
- Selection
- Application
- Evaluation

Identification

The **Chemistry Calculator** project was identified as a potential candidate for refactoring due to several factors, including the size of the codebase, the complexity of the calculations, and the lack of clear structure in the code. The code was difficult to maintain, with many redundant and unnecessary statements that made it challenging to understand and modify the code. The code has several issues, such as poor naming conventions, lack of modularity, and duplicate code. The project also lacked proper documentation and testing, making it challenging to verify the accuracy of the calculations.

Selection

To begin the refactoring process, a thorough analysis of the existing codebase was conducted, and a list of potential areas for improvement was identified. The focus was on removing redundant code, simplifying complex calculations, and improving the structure and organization of the code. The selected approach for refactoring was a gradual refactoring process, where changes were made incrementally, ensuring that the code remained functional and accurate throughout the process.

Chemistry Calculator Refactoring Report

Below are the code smells that are found in the project codebase-

- Remove Unused Imports
- Primitive Obsession
- Long Method
- Long Parameter List
- Large Class
- Comments
- Duplicated Code
- Inappropriate Naming
- Divergent Change
- Speculative Generality
- Dead Code
- Shotgun Surgery
- Inappropriate Intimacy

Class Name	Code Smell
Atom	Primitive Obsession, Inappropriate Naming, Shotgun Surgery
Compound	Long Method
CompoundManager	Long Method
Concentration	
Converter	
Database	Divergent Change, Speculative Generality, Shotgun Surgery
DatabaseSerializer	Long Method, Divergent Change, Speculative Generality
EquationBalancer	Long Method, Comments
Fraction	
History	Inappropriate Naming, Speculative Generality, Inappropriate Intimacy
InsufficientDataException	
InvalidAtomException	

InvalidEquationException	
Matrix	Inappropriate Naming
Titration	Long Parameter List, Inappropriate Naming
ConcentrationPanel	Long Method, Large Class, Inappropriate Naming
ElectronConfigPanel	Long Method, Inappropriate Naming, Large Class
EquationBalancePanel	Long Method, Inappropriate Naming, Large Class
Formater	Long Method, Inappropriate Naming
FxPieChart	
HistoryFrame	Long Method, Comments
Home	Long Method, Inappropriate Naming, Shotgun Surgery
MolarMassPanel	Primitive Obsession, Long Method, Inappropriate Naming, Large Class, Comments
NeedHelpPanel	Long Method, Inappropriate Naming
PercentOfCompletionPanel	Primitive Obsession, Long Method, Inappropriate Naming, Large Class, Comments
Sidebar	Long Method, Comments, Shotgun Surgery
TitrationPanel	Long Method, Large Class, Comments

Table 1: Code smell found for each class

Application

The refactoring process involved several steps, starting with

- Restructuring each and every file with proper indentation and removing blank lines to make it more readable
- Removing redundant code (code that were commented to debug certain portion of the code)
- Performing remedy for each and every code smell that are found in the **Selection** process

Class Name	Remedy Applied
Atom	Replace array with object, Extract method, Move Method, Move Field
Compound	Extract Method
CompoundManager	Extract Method
Database → AtomDatabase	Extract Class, Extract Method, Inline method, Move Method, Move

	Field
EquationBalancer	Extract Method
History → HistoryManager	Inline method
Matrix	Extract Method
Titration	Extract Method
ConcentrationPanel	Extract Method, Extract Class
ElectronConfigPanel	Extract Method, Extract Class
EquationBalancePanel	Extract Method, Extract Class
Formater	Extract Method
HistoryFrame	Extract Method, Move method
Home	Extract Method, Move Method, Move Field
MolarMassPanel	Replace array with object, Extract Method, Extract Class
NeedHelpPanel	Extract Method
PercentOfCompletionPanel	Replace array with object, Extract Method, Extract Class
Sidebar	Extract Method, Move Method, Move Field
TitrationPanel	Extract Method, Extract Class

Table 2: Code smell remedy for each class

Evaluation

To evaluate whether refactored code improves maintainability, readability and modularity some of the techniques are-

1. Code Quality Metrics
2. Peer Review
3. Unit Testing
4. User Feedback

For **Chemistry Calculator** project evaluation we performed code quality metrics.

Code Quality Metrics

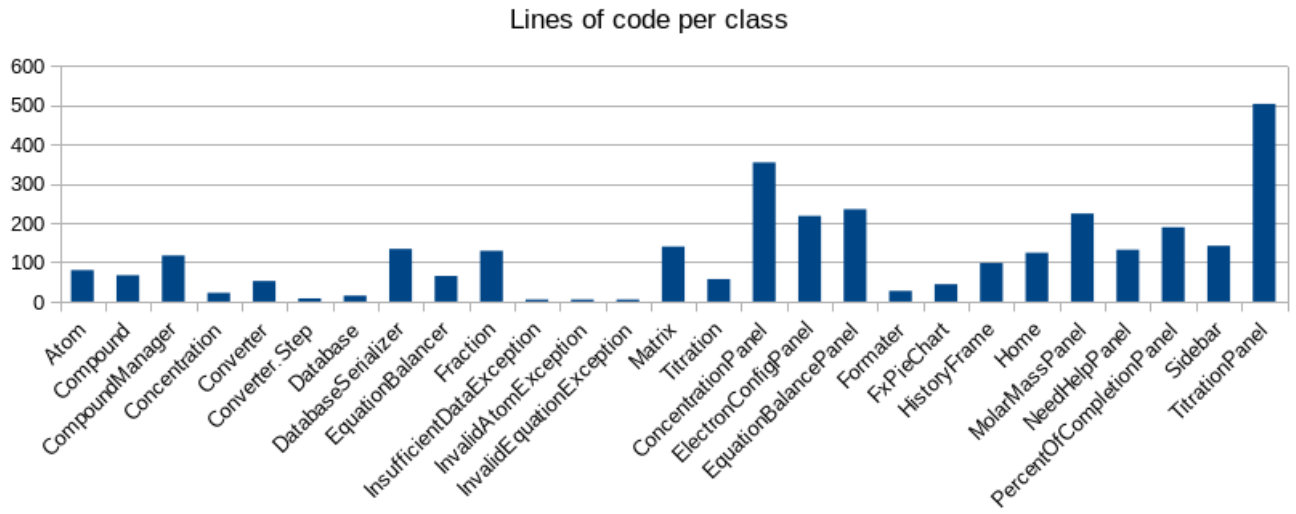
Code quality metrics are used to measure the quality of software code. Some commonly used code quality metrics that we used for **Chemistry Calculator** project:

- **Weighted Methods per Class (WMC):** WMC is a software metric that measures the complexity of a class in object-oriented programming. WMC is calculated by counting the number of methods in a class and assigning a weight to each method based on its complexity.
- **Cyclomatic complexity:** Measures the complexity of a program by counting the number of decision points in the code.
- **Lines of code (LOC):** Measures the number of lines of code in a program.

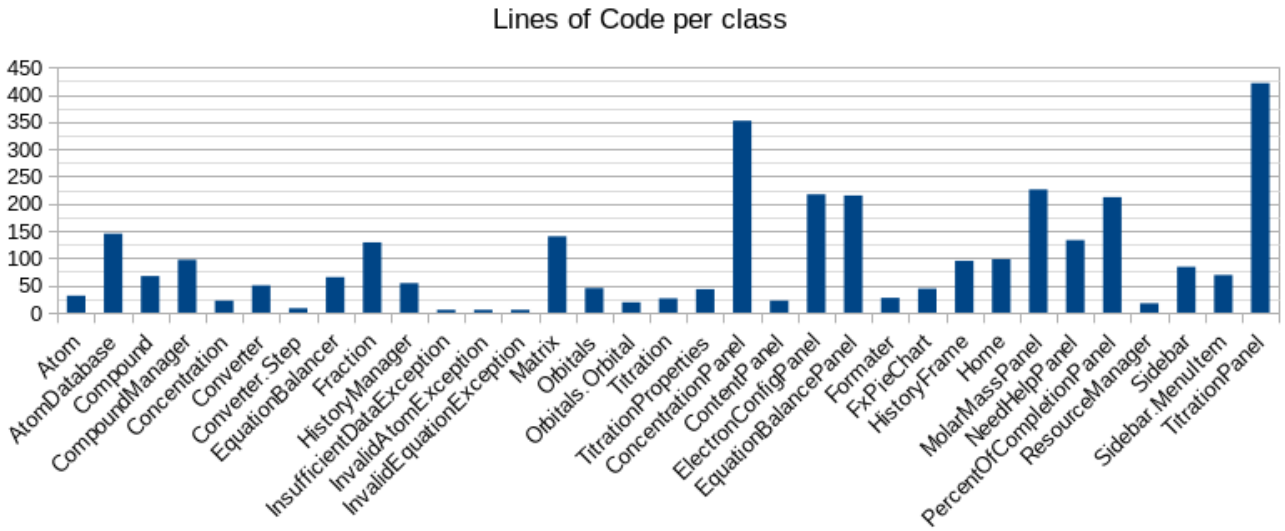
These quality metrics performed with **MetricesReloaded** plugin from **Intellij Idea (Community Edition)** software.

Chemistry Calculator Refactoring Report

Before refactoring the lines of code per class is shown below

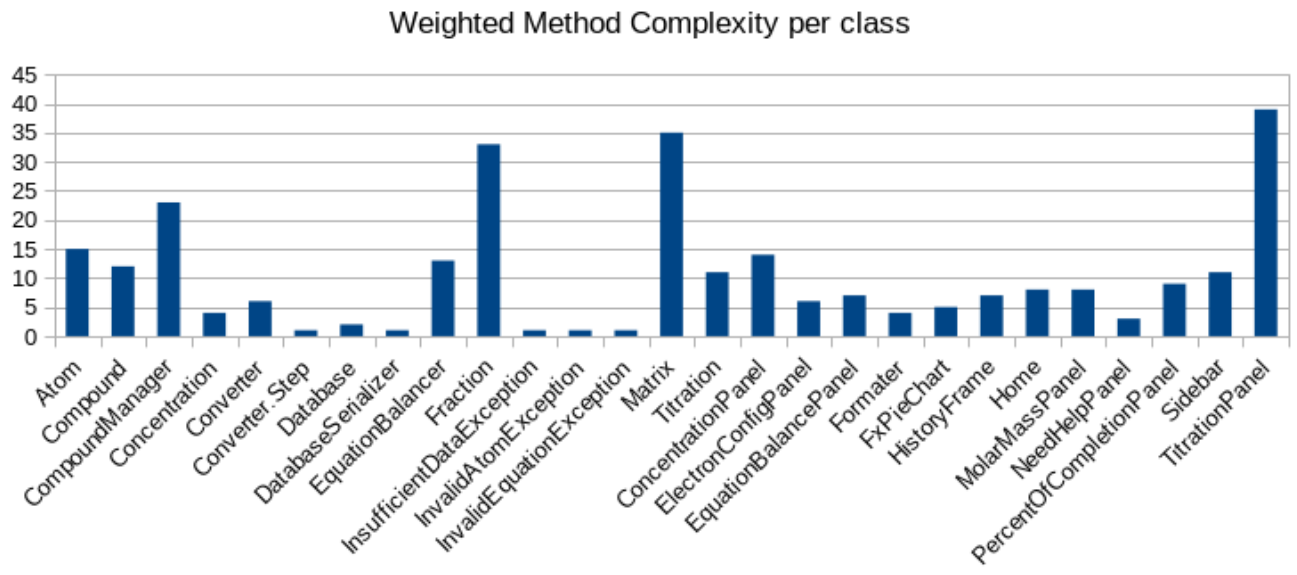


After refactoring the lines of code per class is

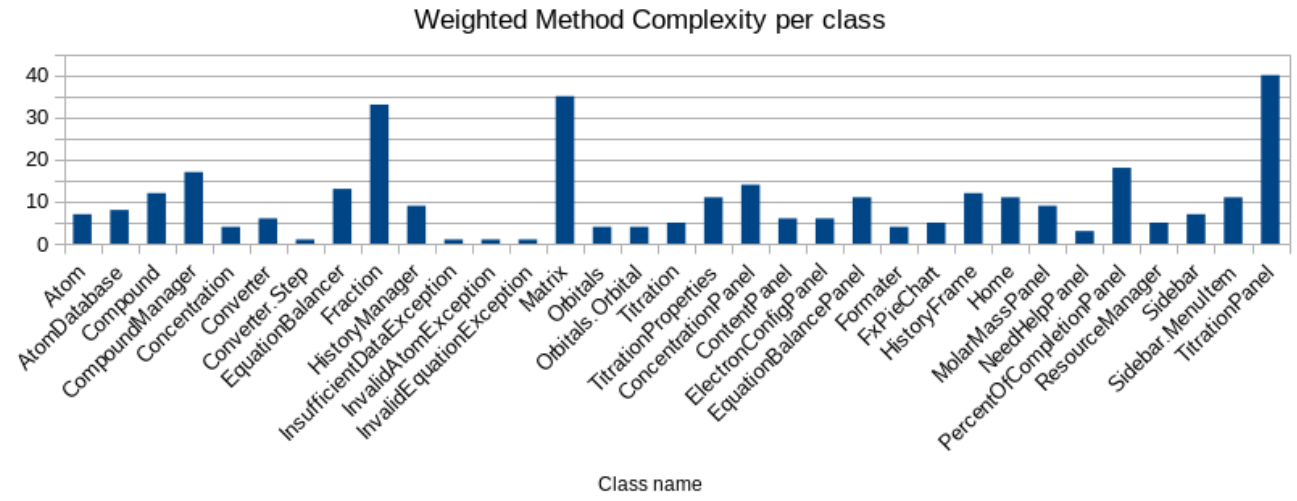


Chemistry Calculator Refactoring Report

Before refactoring the weighted method per class is



After refactoring the weighted method per class is



Conclusion

In conclusion, the refactoring process of the **Chemistry Calculator** project was successful in improving the readability, maintainability, and extensibility of the codebase. The approach used was a gradual refactoring process, ensuring that the code remained functional and accurate throughout the process. The refactored code was evaluated and found to be more readable, maintainable, and extensible than the original code. Overall, the refactoring process resulted in a more reliable and maintainable project that will be easier to modify and extend in the future.