Assignment 2

By

Vyshnavi Duppekar

pranvith elukurthi

sushrith manchala

Arun Kumar Nyalakonda

Object Oriented Design Development

Objectives, Questions, and Metrics

Goal; This research study is meant to look at how problems with code affect the organization of software focusing on C&K metrics which are concerned with component connection and integration. It is important to know what these problems do to these measures so that we can be able to enhance the quality and maintainability for software systems.

Question; The research has two main questions that it aims at answering; First, what impact does code bad smells have on organizational metrics in terms of connectivity and unity? Second, do classes with bad coding practices exhibit any signs of being poorly organized compared to clean well-organized ones?

Metrics; coupling (C) and cohesion (K)were the measures selected for this particular study as they provide indicators for assessing code arrangement. C determines how much classes depend on each other while K examines coherence and strength within a class. Looking at these will help us understand relationship between code quality, code issues and software design.

Subject Programs (Data Set)

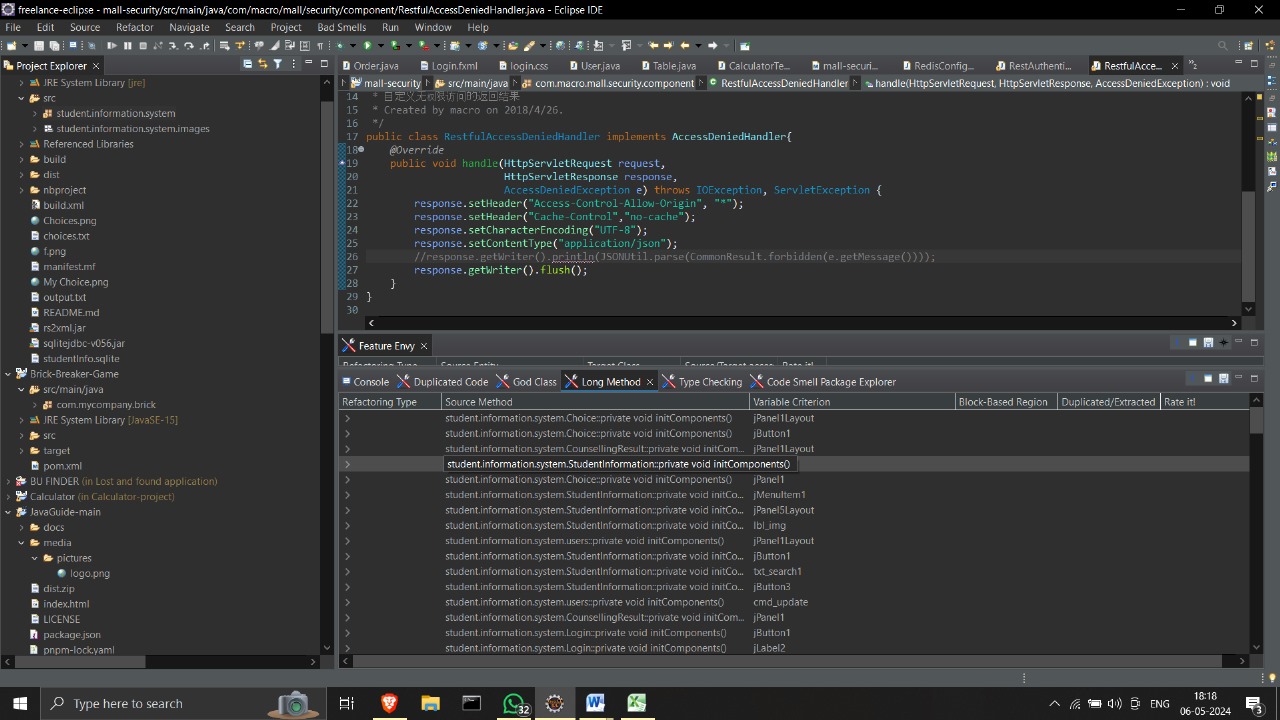
| **Project Name** | **Size (LOC)** | **Age (Years)** | **Developers** | **Description** |
| --- | --- | --- | --- | --- |
| Online Admission Counseling System | 25000 | 3 | 5 | Admissions, Verification, Counseling |
| Banking Application | 18000 | 4 | 3 | Financial Operations, Services, Management |
| ATM Interface | 12000 | 2 | 2 | Banking Transactions, Inquiries, Interface |
| Calculator Project | 5000 | 1 | 1 | Mathematical Computations, Basic Operations |
| Basic Bank System | 20000 | 3 | 4 | Core Banking Functions, Account Management |
| Calorie Calculator | 8000 | 2 | 2 | Calorie Intake Calculation |
| Digital Clock | 3000 | 1 | 1 | Time Display, Customizable Features |
| Home Services | 15000 | 2 | 3 | Maintenance, Appointments, Repairs |
| Dynamic Project Manager | 22000 | 4 | 5 | Task Organization, Collaboration, Timelines |
| Vehicle Loan | 10000 | 3 | 2 | Loan Management, Application, Approval |

We have been very cautious in picking out different Java projects on GitHub for this research which should meet our standards in terms of size, age and developer participation. We ensured that each software application from our group is unique in terms of its domain and complexity so as to enable us carry out an exhaustive assessment about how bad smells affect modularity within different software contexts.

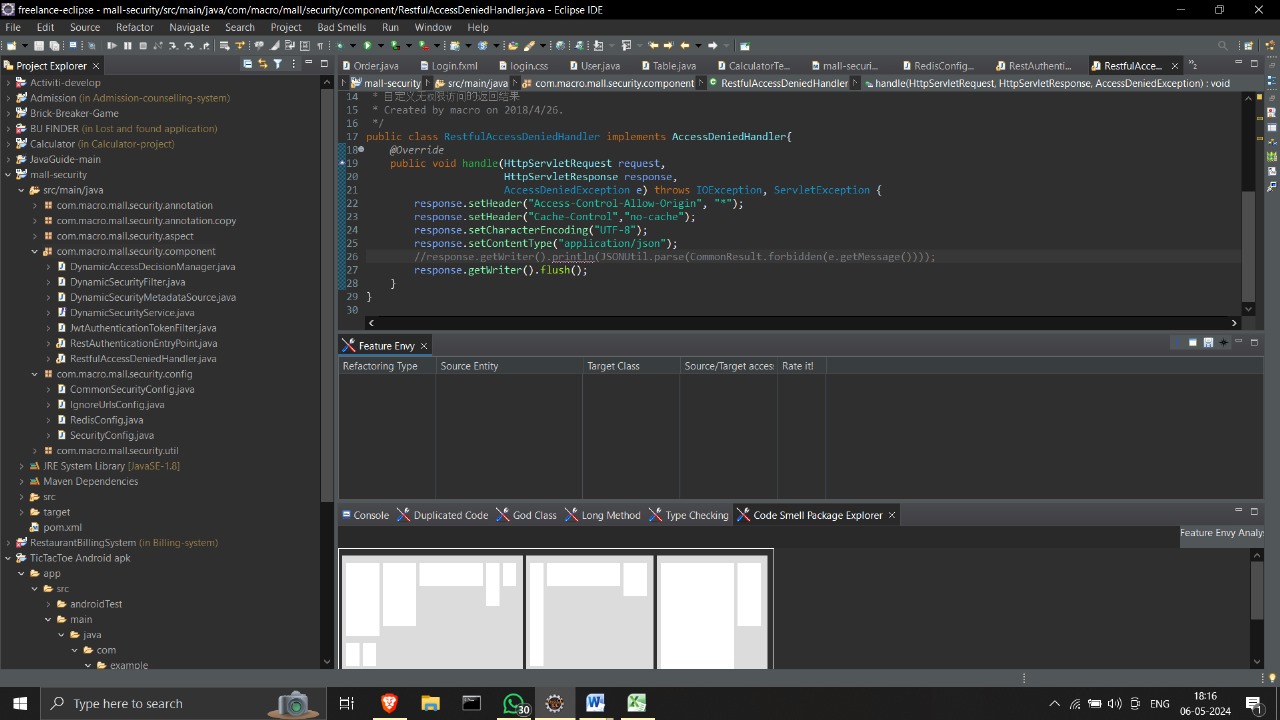
The table below gives a summary view of all the programs under analysis with their key attributes such as project name, lines of code (LOC), years since creation, number of contributors involved and a short description about its functionality. Through this organized method we can be able to compare modularization trends across different projects thus getting insights that may be useful beyond this study’s scope.

**Tool Description**

To investigate the code bad smells in the Java projects, we have used JDeodorant, which is an Eclipse plug-in. It has a strong detection method and easy-to-use interface for detecting bad smells in codes. We can detect many kinds of bad smells like duplication of code, long methods or improper calls to method by using JDeodorant.



We chose JDeodorant because it is widely adopted and reliable among software engineering communities. Additionally, it can be integrated into Eclipse IDE so that our work becomes more convenient; hence we just interpreted modularity metrics affected by detected bad smells



Results:

The results from the analysis of C & K metrics tools have significantly enhanced the major goals and differences considering the classes with and without any bad smells. The classes that are having bad smells shown a high range of coupling and also shown the lower cohesion metrics. This results in the weaker structure when classes must be cleaned. The bar chart gives the good understanding of all these findings and thereby showcasing the different sets of modularity that is trending with in different classes with in the given projects.

The results clearly depicts the detrimental effect of code bad smells on software considering the modularity and thereby clearly emphasized the important code practices in the development of any software.

Conclusion:

In conclusion, our empirical study shows a clear relationship between code malfunctions and reduced software modularity, as measured by the C&K metric for coupling and coordination. Bad classes exhibit flexible characteristics of modularity, reducing maintenance and software quality challenges. Addressing code bad smells early in the development lifecycle is critical to achieving a modular, maintainable codebase. Future research could explore automated code smells detection and mitigation methods, further enhancing software modularity and overall system quality.

References:

https://github.com/kishanrajput23/Java-Projects-Collections?search=1