SOEN 6431 - Deliverable 3

SUMMER SEMESTER 2024

INSTRUCTOR: DR. Pankaj Kamthan

By: Darsh Patel, Salma Taha, Namratta Brahmbhatt, Piyush Singla

https://users.encs.concordia.ca/~kamthan/courses/soen-6431/

This document serves to provide additional details related to the execution of Project DÉJÀ VU, which targets a software re-engineering of the KBC Game.

- Tools used:

Github Copilot IntelliJ SonarQube

- Frameworks used:

Complete Undesirables List

#	Undesirables	Type	Location	Re-engineering Method	Definition
1	**Duplicate Code**	Code Smell	Lifeline methods, user interaction blocks	Extract method	Identify duplicated code blocks and extract them into a separate method that can be reused
2	**Primitive Obsession**	Code Smell	Method parameters & variables	Replace primitives with objects	Replace groups of primitive values with small classes representing a concept
3	**Empty String Arguments**	Anti-pattern	Method calls	Remove unnecessary arguments	Eliminate unnecessary arguments to clean up the method signature
4	**Implementation Smells - Comments**	Code Smell	Throughout the code	Remove redundant comments	Eliminate redundant comments when code is self-explanatory
5	**Vague Variable Names**	Code Smell	Method & user prompt variables	Rename variable	Rename variables with descriptive names that indicate its purpose
6	**Redundant case statements**	Anti-pattern	Switch statements	Remove redundant cases	Simplify switch-case statements by removing cases that perform the same action
7	**Improper indentation and spacing**	Code Smell	Throughout the code	Format code	Improve code readability by applying consistent formatting and standard spacing
8	**God Class**	Anti-pattern	Simple.java	Extract Class	Divide the large class into smaller, more focused classes, each handling a specific responsibility.
9	**Magic Numbers**	Code Smell	Timing & loop conditions	Replace with constants	Replace hard-coded numbers with named constants

10	**Shotgun Surgery**	Code Smell	Methods making multiple changes	Move methods to appropriate classes	Localize changes by moving related methods to the appropriate classes
11	**Lack of Error Handling**	Anti-pattern	Input handling sections	Introduce error handling.	Add proper error handling mechanisms, such as try-catch blocks, to manage exceptions gracefully
12	**Refused Bequest**	Code Smell	Inheritance hierarchies	Refactor inheritance	Reconsider the inheritance hierarchy if a subclass does not use or need inherited behavior
13	**Hardcoded Credentials**	Anti-pattern	Print statements and messages	Externalize strings	Move hardcoded strings to external resources or constants for easier management and internationalization
14	**Speculative Generality**	Code Smell	Unused methods or variables	Remove unused code	Eliminate unnecessary code or that is used to reduce clutter & potential maintenance overhead
15	**System.out for User Interaction**	Anti-pattern	Print statements	Use logging framework	Replace direct calls to System.out with a logging framework for more flexibility and control
16	**Data Clump**	Code Smell	Method parameters	Introduce parameter object	Encapsulate groups of parameters frequently passed together into a single object
17	**Inconsistent formatting**	Anti-pattern	Throughout the code	Apply consistent formatting	Adopt a consistent coding style and format the code accordingly to improve readability
18	**Feature Envy**	Code Smell	Methods accessing data from other classes	Move Method	Move methods accessing data from other classes to the class where the data is located

References

1. Joshi, R. (2024). GitHub Profile. https://github.com/Rohitjoshi9023
2. IEEE. (2024). IEEE Xplore Digital Library. https://ieeexplore.ieee.org/abstract/document/8486173
3. IEEE. (2024). IEEE Xplore Digital Library. https://ieeexplore.ieee.org/abstract/document/242539
4. Springer. (2024). Arabian Journal for Science and Engineering. https://link.springer.com/article/10.1007/s13369-011-0117-x
5. SonarSource. (2024). SonarQube. https://www.sonarsource.com/products/sonarqube/
6. GitHub, Inc. (2024). GitHub. https://github.com/
7. Software Freedom Conservancy. (2024). Git. https://git-scm.com/
8. JetBrains s.r.o. (2024). IntelliJ IDEA. https://www.jetbrains.com/idea/
9. Anthropic. (2024). Claude AI. https://claude.ai/
10. OpenAI. (2024). ChatGPT. https://chatgpt.com/
11. Fowler M. Refactoring: Improving the Design of Existing Code. Addison-Wesley; 1999.
12. Sharma T, Spinellis D. A survey on software smells. Journal of Systems and Software. 2018;138:158-173.
13. Lanza M, Marinescu R. Object-Oriented Metrics in Practice. Springer; 2006.
14. Martin RC. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall; 2009.
15. Suryanarayana G, Samarthyam G, Sharma T. Refactoring for Software Design Smells: Managing Technical Debt. Morgan Kaufmann; 2014.
16. Brown WJ, Malveau RC, McCormick HW, Mowbray TJ. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons; 1998.
17. Telles M, Hsieh Y. The Science of Debugging. Coriolis Group Books; 2001.
18. Howard M, LeBlanc D. Writing Secure Code. Microsoft Press; 2003.
19. Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley; 1994.
20. Martin RC. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall; 2008.
21. Bloch J. Effective Java. Addison-Wesley Professional; 2018.