Code Refactoring: Code Smells

Part 1: Code Smells and Refactoring Patterns

Code Smells

1. Long Method - Bloaters
2. Dead Code - Dispensable
3. Switch Statements - Object-Orientation Abusers

Refactoring Patterns – methodology (idea)

Detection of Long Method code smells found through method dependencies and the code size. Extract Method refactoring splits a long method into smaller ones resulting in easier readability, manageability, and maintenance [2]. Dependencies between statements can cause a challenge when breaking the body of a long method. Understanding overall statement dependence before refactoring results in fewer bugs and improves code efficiency. Implementing move methods improves or preserves class cohesion after refactoring [2]. Dead Code smell is a dispensable unneeded field, parameter, variable, or methods not used within a program [1]. Changing or correcting code may result in unused or unreachable code. Unneeded subclasses or superclasses can be deleted with an Inline Class or Collapse Hierarchy, while parameters use Remove Parameter [3]. Complex Switch Statements can later cause difficulty when adding additional conditions. A single switch may include code throughout an entire program, causing complications in modifying it [4]. Use Extract Method then Move Method for isolating switch statements and placing them in suitable classes. When dealing with hierarchy subclasses, redefine methods containing conditionals using Replace Conditional with Polymorphism [3].

Code Quality Improvement

Methods or functions longer in length are difficult to understand, maintain, and find duplicate code within. Increasing the number of functions permits readable code and allows for the exploration of effective code restructuring [3]. Removing obsolete code allows the computer's operating memory to become available for other functions reducing the overall size of the program. Resulting in a faster runtime and future support becoming easier [3]. Extracting, moving, and replacing switch conditionals improved organization within the project’s code [3].

Part 2: Refactoring Tools and Refactoring Patterns

Refactoring Tool – program that helps identify what part of the code should be refactored

1. Lint : Android Studio - integrated development environment (IDE)

Refactoring Tool Capabilities

Android studio is software used by programmers for developing Android applications. Offering many features such as metrics reporting, smell detection, and refactoring using code inspection tools such as Lint [6]. Specializing in structural issues, this tool assists in detecting minor code issues, potential bugs, and improvements in optimization. Some examples of these capabilities are accessibility, internationalization, overall code syntax correctness, performance, security, and usage [6]. Poorly structured code can impact the ability to maintain the reliability and efficiency of your project. Lint detects resource files that contain unused namespaces taking up unnecessary space or processing power while also providing a simple way to remove them. You can hover your mouse over any variable or method name to see all code usage and dependencies, then effortlessly delete or move them in one click without corrupting the functionality of the project within Android Studio. These tools and functionalities are easily accessible and to use.

Addressing Refactoring Patterns

Lint supports issues on different security levels such as Globally (entire project), Project module, Production module, Test module, Open files, Class hierarchy, Version Control System (VCS) scopes. Errors are displayed through pop-up text in the Code Editor or in the Lint Inspection Results window [6]. Programmers can utilize many included features by using the Lint tool. A favorable feature of Android Studio using the Lint tool is unused or dead code detection [6]. If a function or method is not utilized somewhere within the project, the color of the function name is grey. Once in use, the function name lights up accordingly. For example, variable names are purple while functions names are yellow. Another convenient feature is variable and function usage detection. This feature becomes quite helpful when editing variable or method names, moving or deleting methods, or adding hierarchical values [6]. These tool features make reorganization is very simple in removing any Long Method code smells.

References

1. Satwinder, S. and Sharanpreet, K. ScienceDirect.com | Science, health and medical journals, full text articles and books. *Pdf.sciencedirectassets.com*, 2017. https://www.sciencedirect.com/science/article/pii/S2090447917300412.
2. Mahnoosh, S., Mehrdad, A. and Morteza, Z. Journal of Systems and Software | Vol 187, May 2022 | ScienceDirect.com by Elsevier. *Sciencedirect.com*, 2022. https://www.sciencedirect.com/journal/journal-of-systems-and-software/vol/187.
3. Shvets, A. Refactoring: clean your code. *Refactoring.guru*, 2014. https://refactoring.guru/refactoring.
4. Tushar, S., Vasiliki, E., Panos, L., Diomidis, S. Code smell detection by deep direct-learning and transfer-learning. *Sciencedirect.com*, 2021. https://www.sciencedirect.com/science/article/abs/pii/S0164121221000339.
5. E. Murphy-Hill and A. P. Black, "Programmer-Friendly Refactoring Errors," in IEEE Transactions on Software Engineering, vol. 38, no. 6, pp. 1417-1431, Nov.-Dec. 2012, doi: 10.1109/TSE.2011.110.
6. Meet Android Studio | Android Developers. Android Developers, 2022. https://developer.android.com/studio/intro/accessibility.