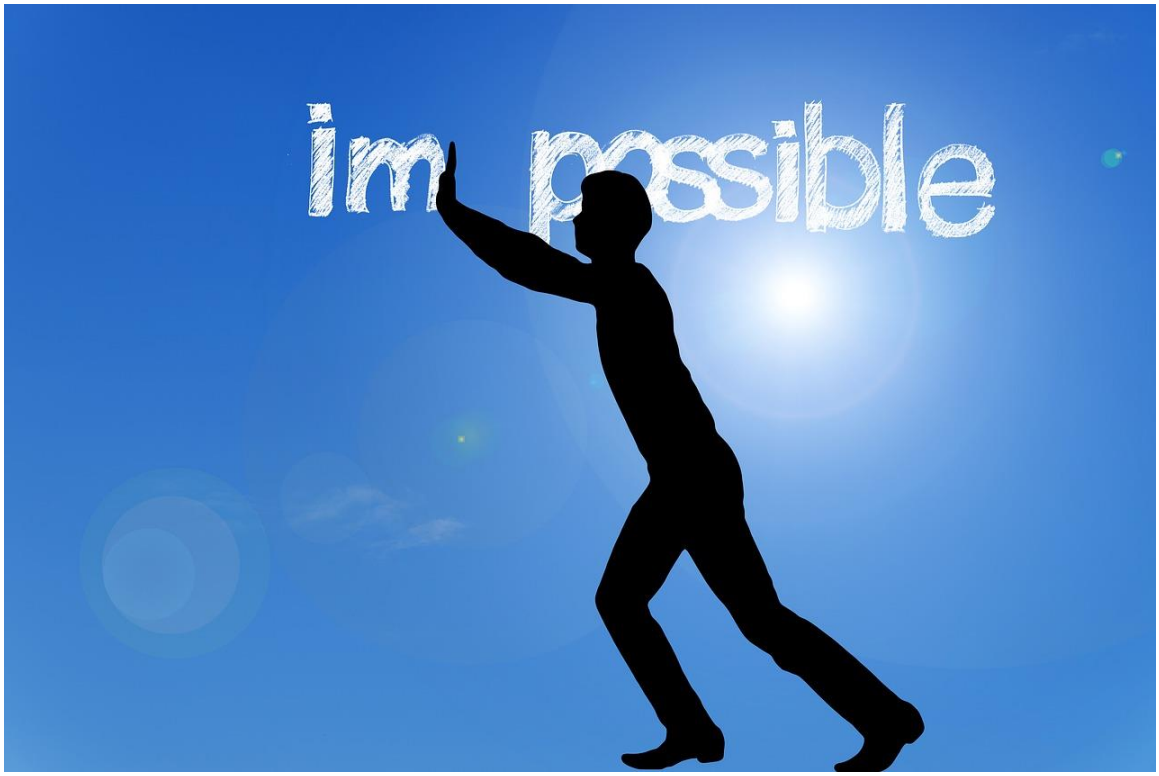

REFACTORING AND CODE OPTIMIZATION TECHNIQUES

OBJECT ORIENTED PROGRAMMING I

Sercan Külcü | Object Oriented Programming I | 10.01.2023

Contents

Introduction

As software applications grow and evolve, it is common for the code to become complex and difficult to maintain. Refactoring is the process of improving the design and structure of existing code without changing its behavior. Code optimization, on the other hand, is the process of improving the performance of existing code. In this chapter, we will explore some of the most effective refactoring and code optimization techniques for advanced Java programming.

Refactoring Techniques

Refactoring can improve the quality of code and make it easier to maintain. The following are some of the most effective refactoring techniques for Java:

Extract Method: Extracting a method from a larger method can improve the readability and maintainability of the code. It can also make the code more reusable by creating a new method that can be called from multiple places in the code.

Extract Class: Extracting a class from an existing class can help to organize the code better and make it more maintainable. It can also help to reduce the size and complexity of the existing class.

Replace Conditional with Polymorphism: Replacing a long and complex conditional statement with a polymorphic solution can improve the readability and maintainability of the code. It can also make it easier to add new functionality to the code in the future.

Move Method: Moving a method from one class to another can help to organize the code better and make it more maintainable. It can also help to reduce the complexity of the existing class.

Code Optimization Techniques

Code optimization can improve the performance of code and make it more efficient. The following are some of the most effective code optimization techniques for Java:

Use Final Keyword: Using the final keyword for variables and methods can improve the performance of the code by allowing the compiler to make certain optimizations.

Use StringBuilder Instead of String Concatenation: String concatenation can be slow, especially when used in a loop. Using a StringBuilder can improve the performance of the code.

Use ArrayList Instead of Arrays: ArrayList can be faster than arrays for certain operations, such as adding or removing elements.

Use Enhanced For Loop: Using the enhanced for loop can improve the readability and performance of code that iterates over collections.

Conclusion

Refactoring and code optimization are essential skills for advanced Java programming. Refactoring can improve the quality and maintainability of code, while code optimization can improve its performance and efficiency. By applying these techniques in your code, you can make your software applications more robust, efficient, and scalable. Remember to always test your code after making changes to ensure that it still works as expected. With practice and experience, you can become a proficient Java developer who can write high-quality, maintainable, and performant code.