**Brendon Samson**

**REFACTORING**

**Summary**

# Chapter 1: Refactoring, A First Example

Refactoring is the process of updating or changing a piece of a software code in a way that does not affect its external behavior but improves its internal structure and design. Refactoring is an important factor for maintaining and improving software quality over time.

There are multiple techniques that you could use to refactor your code. *Extract Method* is one of the techniques. It is a type of technique that breaks the original code into smaller, more readable chunks. There are also others techniques such as the *Move Method* wherein you copy the code over to another function and adjust it to fit in the new function.

By applying one or more of the refactoring methods you will be able to maintain the readability and extensibility of your code. You will also be able to notice significant effects of having your code refactored.

# Chapter 2: Principles in refactoring

In this chapter several key principles that will guide developers in successfully refactoring their code were introduced.

The first principle is about making sure that the refactored code remains functional. This means that whatever the behavior of the piece of code you refactored before is being done exactly after you refactored it. Although its appearance has changed its behavior should not.

The second principle is to implement small and incremental changes when refactoring. This means that you should not be refactoring a large chunk of your code and should only do it little by little. It is better to refactor small targeted pieces of code to make it easier for you to test and validate that its behavior is still the same.

The third principle is to use version control system. Version control system is used to track changes so that you will be able know where to rollback your code if you encountered an error. This is important especially when working on large and complex codebases where it is easy to introduce unintended consequences.

The fourth principle is eliminating duplication. Having duplicated pieces of your code means more number of lines and more lines that you have to read and understand. You should refactor duplicated pieces of code to a single function that you could reuse to remove duplication.

The fifth principle is ensuring that the code is readable and is easy to be understood. Always use descriptive naming sense for your variable, functions, and classes. You should also organize your code in a logical consistent manner for easy maintainability.

It is emphasize that when you refactor something you should always test it. Automated tests should be used to make sure that you do not introduce any new bugs that could potentially break the flow or the whole system. By following the five principles, developers will be able to ensure that their refactoring efforts results a more maintainable and extensible code.