**Brendon Samson**

**REFACTORING**

**Summary**

# Chapter 3: Bad Smells in Code

In this chapter the concept of *Code Smells* is introduced. It is about the symptoms of potential problems in code that may require the need for refactoring. There are several code smells that were discussed such as:

* *Duplicated Code* which occurs when the same pieces of code appear multiple places in the system although it does the same thing exactly. Having duplicated pieces of code increases your number of lines of codes which decreases the readability of your code making it harder to understand.
* *Long Methods* which is when methods or functions contain long number of lines of code. Methods should be small and should only perform one specific task instead of multiple to maintain readability and maintainability of your code.
* *Large Classes* occurs when a class contains too many responsibilities and too many methods. The scope of the responsibilities of a class should only be related to it.
* *Long Parameter Lists* occurs when a function or method requires multiple number of arguments for it to work. Long parameter list means longer number of characters in your line of code and more variables that you have to check.
* *Divergent Change* occurs when one class is commonly changed in different ways for different reasons. Software is meant to be soft so you should be able to make changes easily.
* *Switch Statements* occurs when you use the switch statement. It is used to perform different actions based on the value of a variable so there is possibility that you will have to add another line of code each time your variable will have a new value. Switch statements also indicate the violation of object-oriented design principles.

Detecting Code Smells in your code does not automatically make your code a bad code. It is just a warning sign that suggests the need for refactoring as it could create error in the future if left unchecked.

# Chapter 4: Building Tests

This chapter emphasizes on the importance of testing when performing refactoring. There are several frameworks that could provide guidance on how to create effective tests to support your refactoring process.

When testing your refactored code, you should set goals on what should be the appropriate result after the test. There are different types of tests that you can use in a software development project. An *Automated Test* is the most desirable type of test because you can run it immediately and easily ensure that the changes done to your code do not trigger errors or create unintended results.

In this chapter two testing framework were introduced *JUnit* and *Unit and Functional Tests.* JUnit is an open-source unit testing framework for Java programming language developed by Erich Gamma and Kent Beck. It is used to write and execute automated tests. JUnit is used designed for writing unit tests. Functional tests are often performed with other tools like GUI-based tools.