ASSIGNMENT 7 : REFACTORING

EMAIL: GREGOIRE.COUSIN@UCDCONNECT.IE;
GREGOIRE COUSIN STUDENT NUMBER: 18204188; COMP 47480 PRACTICAL

INTRODUCTION :

In this lab we were given the instruction to explore the possibilities of refactoring code and understand all that deals with it. Code smells are the root of defining what a problem can exist in a program at given sections. To identify a code smell within a piece of code you need to first understand the framework in which you are working on and what the regulations are.

CODE SMELLS :

Code smells can come from a verity of categories such as Bloaters which are code smells based on the accumulation of unnecessary code. These code smell names include Long Methods, Large classes, primitive Obsession, Data dumps and so on. Objective Orientation Abusers are another type of code smell that deal with the incomplete and incorrect state of a program. Objective Orientation Abusers include switch statement faults, temporary fields, refused bequests and class and interface differences. Other categories of Code smells are Change preventers, Disposables and couplers. Change preventers deal with refactoring one place of code that eventually will lead you to changing the same principle in other sections of your code. A good example of this would be the Shotgun Surgery where you need to change a whole lot of variables at once. Where as Dispensable are based on pointless code such as over commenting, lazy classes and duplicated code. The Last Category is based on Coupling. This code smell includes extensive coupling between classes. A good example of this category is the feature envy. feature envy is where a smell occurs after fields are moved to a data class. Moving the operation to this class will be necessary in the refactoring period.

REFACTORING :

Refactoring code plays a big role in the creating or a program and should be respected by developers. The means that if another developer has refactored peaces of code but has handed off the code to another, it should not be touched or be limited to change without communication. This is said with the understanding that refactoring happens on a case by case basis to give code clarity. This might mean that a developer would be obliged to extract a local, constant or other field based variables, change and extract a long methods and class function, use the pull up or push down method or even extract interfaces to organize the schematic of the program. Code clarity is vital when it comes to building a good program since the goal is to provide concise and well organized programs. This increases the ability to reuse segments and expand on particular sections of your code with a more explicit manner. Code issues can also happen in unorganized code fragments. This means that is you have lines of code not embedded in a class, for in a method oof some sort you can damage the ability to develop future features to your program. This is very important to avoid when working with OOP. When extracting a method. it is normally because a method can be too long or has unnecessary fragments. this can decrease readability for a program and make functionalities and features confusing. Since we read programs like documents it is important to take in consideration what we are extracting and refactoring. This means that renaming variables is highly recommended to provide sense to the programs functionality. Every IDE has the potential to read segments oof code and analyze repetition by a find function within the IDE and give you the ability to change code via a tool set. Code duplication is one of the most common types of code smells. This can be seen by redeclaring the same variable which can be easy to spot and refactor but in a more difficult setting code duplication can also exist in the same functionality of a classes behavior. Another popular reason developers find themself refactoring is through finding conditional code statements confusing. This means that a while or for loop statement would cost more load on the desired functionality than is the developer had used a switch statement. Therefore, the developer needs to refactor the appropriate segments and match it with the right function.