Comp 47480 Learning Journal – Assignment 5

1. Reflection on my Learning on Refactoring.

Refactoring is an important part of the software design process which can assist in the improvement of your software design. This makes code easier and clearer to read and will assist in making the system easy to extend over time. Software that is poorly designed can often be confusing and lead to systems that take excessive amounts of time to extend or understand what the code is doing if meaningful names aren't given to methods in advance. This also allows other programmers to adjust the code without much previous knowledge of the code base.

This practical I found difficult just to follow the code through the refactoring process. It shows the importance of using meaningful names as the method and variable names and drives home that refactoring should be done as you go, for instance when you see something that doesn't tell you what it does or if an bug is introduced to the system or if you have a code smell then it would be beneficial to refactor.

This system suffers from giving a class and method too many responsibilities so the first thing I tried to do was to make classes of Car, Motorbike and All Terrain and have them extend the Vehicle class. The statement method in the Customer class was also doing too many things. This was broken down into a method that updates the frequent points and a method that gets the rental cost for the vehicle. This is a better as it allows us to erase the switch statement from the method. The ability to refactor the code allows us to come up with a solution to a problem and refactor it as we go creating a flexible and improved solution.

In this step we would also observe a code smells. These relate to parts of the code that aren't just in need of simple refactoring but redesigning to get rid of duplicate code and eliminate the issues related to the code. Common code smells like duplicated code we can extract the code and make it a method which can be called this allows us to use the code in many places with the implementation only be written once this is good practice as if we need to fix something in the duplicate code we now only have to change it in the method instead of finding every instance of the code. Feature envy is also common this is when a class is given a function that shouldn't be in the class. This leads to classes being coupled and being dependant on one another. Like duplicate code long methods can also be extracted to a separate method which gives a better understanding of the method. Divergent change can occur where a class violates the single responsibility principle which means when we change one class it influences another class and again this can occur as what is known as shotgun surgery when you change something in one class that requires small changes in many classes. When dealing with currency or special strings these could be modelled as a class this allows us to control our use of primitives which could prove detrimental to the design if overly utilized.

Refactoring is an essential part of the software development process where it is not used regularly systems often contain code smells and naming issues that can make your system difficult to alter and expand. It is imperative to refactor code regularly, so your system can be kept simple, flexible and easy to understand.

I didn’t follow the steps as I should have so I didn’t get as far I could have with the refactoring had I explicitly followed the steps. I got rid of the switch statement and extended the vehicle class first, so it was a bit unclear then what to do next, so I don’t think I have refactored the practical as I should.