Life - Project Report

CAB201 – Semester 2 2020

**Full Name: Chiran Walisundara**

**Student ID: n10454012**

**Date: 23/10/2020**

# **Encapsulation**

Encapsulation has been used for the interface implementation in my code as the ‘INeighbors’ interface maps on to the classes which calls from interface as public methods. The mapped classes include ‘MooreNeighborhood’, ‘VonNeumannNeighborhood’ and ‘OldNeighbors’. Although I could have typed up the methods and their respective classes separately, through encapsulation I am able to reduce the number of errors I make, simplify the maintenance of the program and make the application easier to understand. This in turn increases the overall quality of the code as well. I have also created a private method named ‘WriteToFile’ which is the implementation of the output file specification. As the access level is set to ‘private’ I am able to protect the method content from unwanted access and allows access to the level without revealing the complex details below that level.

# Inheritance

Inheritance has been used in the ‘GetNeighbors’ method in the abstract ‘INeighbors’ interface implementation. New classes namely ‘MooreNeighborhood’, ‘VonNeumannNeighborhood’ and ‘OldNeighbors’ have been created through the ‘INeighbors’ interface. This way I am able to reduce code redundancy and simultaneously reuse code. Moreover, users can clearly see which clearly identify the specific neighbourhood through the created classes for each instance provided.

# Polymorphism

Polymorphism has been implemented using the ‘GetNeighbors’ method in ‘INeighbors’ interface as the 3 classes namely ‘MooreNeighborhood’, ‘VonNeumannNeighborhood’ and ‘OldNeighbors’ mapped on by the interface implements the ‘GetNeighbors’ method in 3 different ways according to the requested neighbourhood in runtime. This is more specifically called ‘Runtime Polymorphism’. By doing so I am able to reuse, test and implement the same code as much as I required which saves time while it was also useful in terms of debugging my code overall which then increased code quality.

# Exception Handling

Exception handling has been used overall in many places such as checking for command line arguments in the ‘PerformingChecks’ method to detect if the correct form of input and existence is checked given by the user. If exceptions are detected, they are reported to the user in the command window just before runtime. For an example, I have used exception handling in the ‘FileContents’ method to display the handled exception if the provided file path is invalid or if the provided file path is missing. By doing so I am able to separate error handling code from regular error codes and group and differentiate various error types. This calls for a cleaner, efficient and overall well code quality compared to simpler error handling techniques which aren’t able to identify the specific error so gives out the whole error message and may even halt the running program.