Life - Project Report

CAB201 – Semester 2 2020

**Full Name: Emilie Pomeroy**

**Student ID: n10265813**

**Date: 24/10/2020**

# **Encapsulation**

* Provide at least one good example of encapsulation in your code. Discuss how using encapsulation has benefitted your code quality.
* Encapsulation is the process of grouping related properties, methods and other members into a single object (i.e. creating a class).

Within the Options class I utilised encapsulation as the variable are declared as private. To access the private variables (rows, columns, generations, update rate and etc), I am using accessors which implement the get and set method to retrieve and set values of private fields. The accessors are public so I can access them from my program and argument processer classes.

This is beneficial to my code quality as it hides the inner implementation of the class, so it is not obvious to the user how the class is storing values in the private variables. The user only knows the values are being passed to the accessor and the variables are getting initialised to that value. Additionally, encapsulation is easy to change to new requirements as I could include if statements within the set accessor to ensure the value meets certain requirements before being stored in the private variables.

# Inheritance

* Provide at least one good example of inheritance in your code. Discuss how using inheritance has benefitted your code quality.
* Inheritance is the ability to create new classes based on existing classes.

# Polymorphism

* Provide at least one good example of polymorphism in your code. Discuss how using polymorphism has benefitted your code quality.
* Polymorphism is where multiple classes can be used interchangeably, even though each class implements the same properties or methods in different ways.
* Hint: Just overriding an Object method such as ToString is not enough.

# Exception Handling

* Provide at least one good example of exception handling in your code. Discuss how using exception handling has benefitted your code quality. Be sure to contrast it against simpler error handling practices.

The argument processor class utilises exception handling through a try catch finally block to check the command line arguments. This catches any errors from encountered when processing the arguments and reverts to the values and prints a warning exception message. The finally block ensures that regardless of whether an exception is caught or not, a message displaying all the options and their values is printed out.

This use of exception handling has benefited my code quality as it makes the program better equipped to continue operating at a reduced level is some part of the system fails. Additionally, it is more robust as it is more resilient to stress and maintain correct functioning. If I used the simpler exception handling practices, I had in Part A which were a series if statements and decisions that might detect errors, the program would be more complicated and difficult to understand.