**IT5016D: Software Development Fundamentals**

**Assessment #2: Reflective Journal**

|  |  |
| --- | --- |
| **Student Name:** | **David Jacobs** |
| **Student ID:** | **20230671** |
| **Due Date:** | **21 / 4 / 2023** |

**Exploration of design patterns.**

Design patterns are proven development standards that can accelerate the development process without having to reinvent patterns every time a problem arises. Design patterns are proven solutions to solve commonly reoccurring coding issues when attempting to meet a program’s requirements.

**Analysis of design principles.**

Design principles are general guidelines that can enhance your class structure and relationships when coding. Some basic software design principals that I have used are as follows.

* **KISS (Keep it simple stupid) Principle**

Keep code simple to minimize bugs and syntax errors and make it easier to modify later on in the development process if necessary. When it comes to coding KISS basically means less is more so keep it simple.

* **DRY (Don’t repeat yourself) Principal**

This principal is crucial for clean and easy to modify code. When writing code, you want to avoid duplication of data and duplication of logic.

* **Open / Closed Principal**

The aim of this principal is to make your code open to extension but closed to modification. This means releasing code that prevents direct modification but encourages extension. This separates core behaviour from modified behaviour.

* **Clean code over clever code Principal**

This principal encourages writing code cleanly so your code does not appear over complex. Clever code may work logically but if it is not easily understood then it is wasting valuable time in the development process.

* **Refactor, refactor, refactor Principal**

As an inexperienced programmer code rarely comes out right the first time. It may feel right when you implement a new feature, but as your program grows in complexity future features may be hindered by how you wrote that early ones. Therefore it is completely normal, and healthy, to revisit, rewrite, or even redesign entire chunks of code

**Integration of research and practice.**

* In writing the code for my assignment I have used a main class which acts as the point of execution for my program. I also used a sub class that is derived from the main class with most of my functions.
* I have applied the necessary indentation to nest my code.
* I have added comments throughout my code to explain how the classes and functions operate.
* In accordance with the KISS principle I have used global variables sparingly and rely of my classes to iterate the processes within my program.
* In accordance with the DRY principle I have created functions to be reusable
* In accordance with the Open / Closed principal my code can be extended with no need for modification.
* In accordance with the ‘clean code over clever code’ principle I have kept my program simple and avoided using to many variables that clutter my code.

* In accordance with the ‘refactor’ principal I have adapted and evolved my code to work optimally through continuous redesign.

**Summary**

I have enjoyed learning to code in Python. Using classes was difficult at first but when I wrote the code for classes and saw the program functioning properly I felt I had a good understanding of how classes work. I feel confident I could write code in python again based on the software project I completed and the short programs I wrote leading up to my assignment.