Luu Tuan Hoang ID: 104180391

Semester Test

1. Describe the principle of polymorphism and how it was used in Task 1.

- After encapsulation and inheritance, polymorphism is frequently referred to as the third pillar of the object-oriented programming paradigm (Savinov 2014). Virtual methods can be defined and implemented by base classes, and they may be overridden by derived classes, providing their own definition and implementation.
- In task 1, we create a new base class (abstract class) **SummaryStrategy** with derived class **MinMaxSummary** and **AverageSummary**. In **SummaryStrategy**, we create an abstract method **PrintSummary()** and override it in each derived class to output different types of summary in the program.

2. Using an example, explain the principle of abstraction.

- The object-oriented programming concept of abstraction is to "shows" only necessary attributes and "hides" extraneous data (Raut 2020). Abstraction's main goal is to create an extra layer that's simpler to use, to hide the complicated stuff inside.. It is the process of choosing and filtering information from a larger pool so that the user only sees pertinent details of the object. In other words, users only need to know what can be done (what), and how to do it does not matter (how). As such, abstraction should be the process of hiding information and implementing functionality, not choosing what is needed to form the object.
- Example: Give a person the remote without the case (above) and ask that person to turn it on through channel 1. Of course he/she will eventually find out which button is channel 1, but that takes quite a while. Next, reinstall the cover for the remote, and ask him/her to switch to channel 5. That should be easy without thinking. Thus, the remote without the shell is still a remote, still usable normally. However, if there is an extra outer shell, it will be easier to use.

3. What was the issue with the original design in Task 1? Consider what would happen if we had 50 different summary approaches to choose from instead of just 2.

- With the original design, each type of summary requires a new variable to be created, leading to code duplication and potentially making the code harder to maintain and scale up.
- If we had 50 different summary approaches to choose from instead of just 2, it would result in 50 different variables for each summary approach, the DataAnalyser class will be tightly coupled to specific summary approaches, which is not good for modularity.

Reference list

Raut, R 2020, *Research Paper on Object-Oriented Programming (OOP)*, IRJET, International Research Journal of Engineering and Technology (IRJET).

Savinov, A 2014, *Concept-Oriented Programming: References, Classes and Inheritance Revisited*, viewed 25 June 2023, <https://arxiv.org/pdf/1409.3947.pdf>.