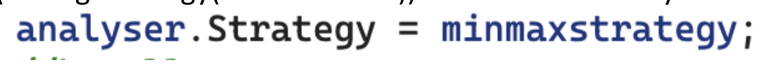
103812143_ Luan Nguyen

Task 2

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

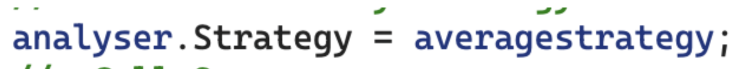
Polymorphism is one of the most important concepts of Object-Oriented Programming that allows objects of different types to be treated as they have the same category that make the code more **flexible** and **extensible.**

In Task 1, MinMaxSummary and the AverageSummary are two different strategies of the SummaryStrategy class or, in other words, Summary Strategies have 2 different forms that are the MinMaxSummary (minmax strategy (local variable)) and the AverageSummary (averagestrategy(local variable)). And the DataAnalyser class can call either



```
analyser.Strategy = minmaxstrategy;
```

or



```
analyser.Strategy = averagestrategy;
```

And the PrintSummary are executed differently based on the given strategy

2. Using an example, explain the principle of **abstraction**. In your answer, refer to how classes in OO programs are designed.

Taking designing a Clock as an example. A "Clock " can **encapsulate** properties such as "seconds"," minutes", and" hours " that are for storing and tracking time and functions such as "set alarm", "start", "stop",… Through **abstraction**, the Clock hides all the complicated properties and functions that we don't need to know but understand the interface and how to interact with it. Not only hiding stuff but abstraction also allows **reusability** and **modularity**. That we can create more clocks such as "digital clock","analog clock" that from the base clock.

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.

The main issue of the First design is that it is not applied Object Oriented Program structure which makes the code insufficient. Imagine if there are not 2 but 50 different summary approach in the code, we need to create 50 different subjects that make the code be much complex and the more important thing is that we can't apply the **reusability** and the **modularity** that are offered by OOP concepts. And this may lead to unreadable due to the complexity