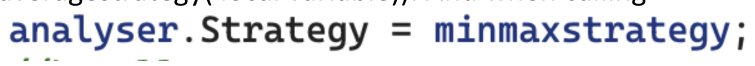
103812143_ Luan Nguyen

Task 2

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

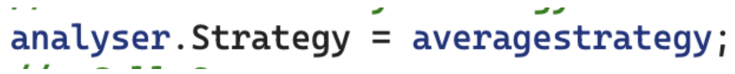
Polymorphism is one of the most important concepts of Object-Oriented Programming that allows objects of different types to be treated as they have the same category that make the code more **flexible** and **extensible.**

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 when calling



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

or



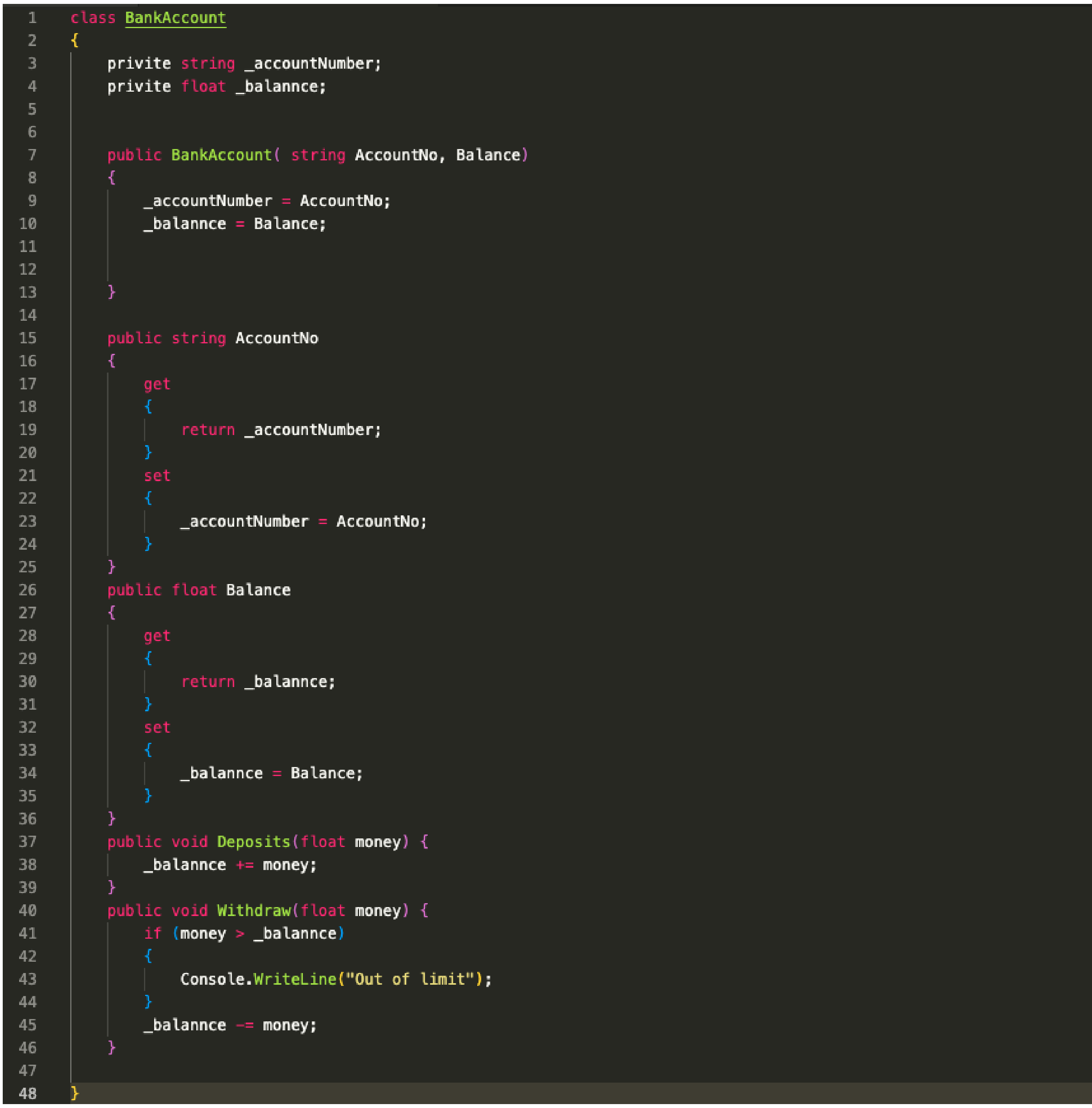
```
analyser.Strategy = averagestrategy;
```

The polymorphism is implemented.

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

Abstraction is one of the 4 key concepts of object-oriented programming that help us simplify the system's complexity by hiding features that are doesn't need to know and only showing what is relevant and making the code more reusable. Moreover, it also helps programmers to create generic reusable classes and interfaces that can be implemented in different ways by different clients.

Consider the BankAccount Class



```
class BankAccount
{
    privite string _accountNumber;
    privite float _balannce;


    public BankAccount( string AccountNo, Balance)
    {
        _accountNumber = AccountNo;
        _balannce = Balance;


    }

    public string AccountNo
    {
        get
        {
            return _accountNumber;
        }
        set
        {
            _accountNumber = AccountNo;
        }
    }
    public float Balance
    {
        get
        {
            return _balannce;
        }
        set
        {
            _balannce = Balance;
        }
    }
    public void Deposits(float money) {
        _balannce += money;
    }
    public void Withdraw(float money) {
        if (money > _balannce)
        {
            Console.WriteLine("Out of limit");
        }
        _balannce -= money;
    }

}
```

People won't go to need to know the private fields like the _accountNumber, _ balance how it is created and how it works, they just need to know their AccountNo and the Balance with the Deposit and the Withdraw without knowing what the fields are created and used for. That is the implementation of abstraction that simplifies how people use bank Accounts.

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 take time and even may be unreadable to fix the bugs (if exist as they seem to be unreadable) or has updates for the code in the future.   While we could use the polymorphism to reduce a lot of work that hiding the implementation details