Hello, in this discussion, I’m going to talk about the bank account program.

Firstly, I’m going to talk about the general structure of the bank account program. The program consists of account class, transaction class and three child classes that are inherited from the transaction class, a bank class and the main program class.

In the account class, it contains all the necessary information for account objects, as well as the actions such as deposit, withdraw and printing out the information of the account.

Next, is the transaction class, it is an abstract class, I’ll talk about it in the next part. The transaction class contains all the information and the actions that all types of transactions have in common, and all these are inherited by the deposit transaction class, the withdraw transaction class and the transfer transaction class.

Next, is the bank class, this class knows the account objects and the transaction objects, it contains all the actions that a bank can do such as add an account, look up accounts, execute transactions and list out the transaction history.

Finally, the program class. This class contains a user interface and all the actions a bank account program should do.

In this section I’m going to talk about how each of the key principals of object-oriented programming are applied in this program.

Firstly, I’m going to talk about Abstraction. Take an example of the account class, it knows the information such as name and balance of a bank account, and actions like withdraw, deposit and having the ability to print out account information. All these attributes and actions describe what a bank account should be in real life. This is how abstraction is applied in this program.

Next up, is the Encapsulation. In each class, the fields are made private, so the values are kept secret from the outside, any access or interaction with their private fields from the outside is controlled by the properties. This is how encapsulation is applied in this program.

Next, inheritance is applied in this program by having the transaction class. There are in total three types of transactions, deposit, withdraw and transfer. They have some information and actions in common such as the execution of transaction, the amount in the transaction, the information about whether the transaction is successful and the time of the execution. The common information and actions can be placed in a base class called transaction class to eliminate code duplication. The child classes only need to inherit the common information and actions from the base class instead of having them within themselves. Another thing to consider is that while these three child classes inherit common information from the transaction class, they also need to make changes to the inherited behaviours to suit their needs, for example, when the execution method is called on the transaction, a withdraw transaction object needs to perform a withdraw action, but a deposition transaction object needs to perform a deposit action instead. In the program, it is achieved by having virtual or abstract methods in the base class and override methods in the respective child classes.

Last one is polymorphism. Because the bank object knows about the transaction objects, so when a transaction executes on a bank object, it will know which specific type of transaction should be executed depending on the actual type of the transaction object. This is how polymorphism is applied in the program.