1. In this solution, TransferTransaction, DepositTransaction and WithdrawTransaction are inherited from the abstract class Transaction, they pass up the amount values to the transaction constructor, all these demonstrate the concept of inheritance. In terms of polymorphism, when the virtual and abstract methods in the base class are called, due to the different implementations in the child classes, even with the same method called, the execution will be different according to the specific “kind” of transaction. In addition to that, is that the transaction list can contain elements of all kinds of transactions and each of them are able to perform the right type of Execute() and Print() of their own kind.
2. If the method is virtual or abstract, then each child class is able to perform their own version of that method by using override.
3. No changes are required for the Bank class in order to include a new transaction type. As long as the new type is a class that is inherited from Transaction, then the transaction list in the Bank class is able to contain that new type and perform any methods specifically for that particular type.
4. We can get rid of a lot of the duplication in our code, because all the transaction types share a lot of the attributes, with inheritance, we can put these attributes in one class for every type to share.
5. Polymorphism on the other hand allows all child classes to still have their uniqueness when required while sharing the attributes and actions that are in common.