Inheritance for banking problem

The problem was creating a system that represents a banking system. The system implements the banking system with a set of functions and variables grouped into classes. There are multiple account types that needs to be considered, where they have unique implementations.

This is solved by using inheritance of a standard banking account. All accounts are required to have an account holder and a balance. Every account has a deposit function, a withdrawal function and an account information function. These base requirements is implemented in the parent class StandardBankAccount. This class has functionality and can be initialized as a regular function and only has the basic requirements.

A computer screen shot of a black background

AI-generated content may be incorrect.

Figure

A requested account type was a savings account, which is implemented in the class SavingsAccount. The saving account class inherits all functions and attributes from StandardBankAccount, including the constructor function. The difference is the addition of a global shared fixed interest rate and a functionality to apply the interest. This difference is the only additional code in the class, as shown in figure 2.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure

The other account type requested was the checking account, implemented in the CheckingAccount class. This account requires a transaction fee when withdrawing any amount of money. This is implemented by simply adding a global transaction fee in the class and overriding the withdraw class. The withdraw function now takes in the amount requested to be drawn and adds the transaction fee. Then it calls the parents withdraw function with the new amount, as seen in the figure 3.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure

The inherited classes minimize new code by inheriting as many functionalities as possible. The constructor, deposit, withdrawn and account information functions are reused without any additional lines. This makes function adjustments and general upkeep of the code easier. The new functionalities will only add new functionality, any repeated functionality is retrieved from the parent. Saving account only adds, so the objects will have the structure of the standard account with the addition of additional functionality. The Checking account changes the withdraw function without repeating itself. It does not need to repeat the logic and actions in the parent, as the only difference is the transaction fee on top of the amount. Therefore, a very simple yet elegant solution is to call the super function for the rest of the logic. These design choices write the least amount of repeated code as possible, while implementing new functionalities and changing previous.