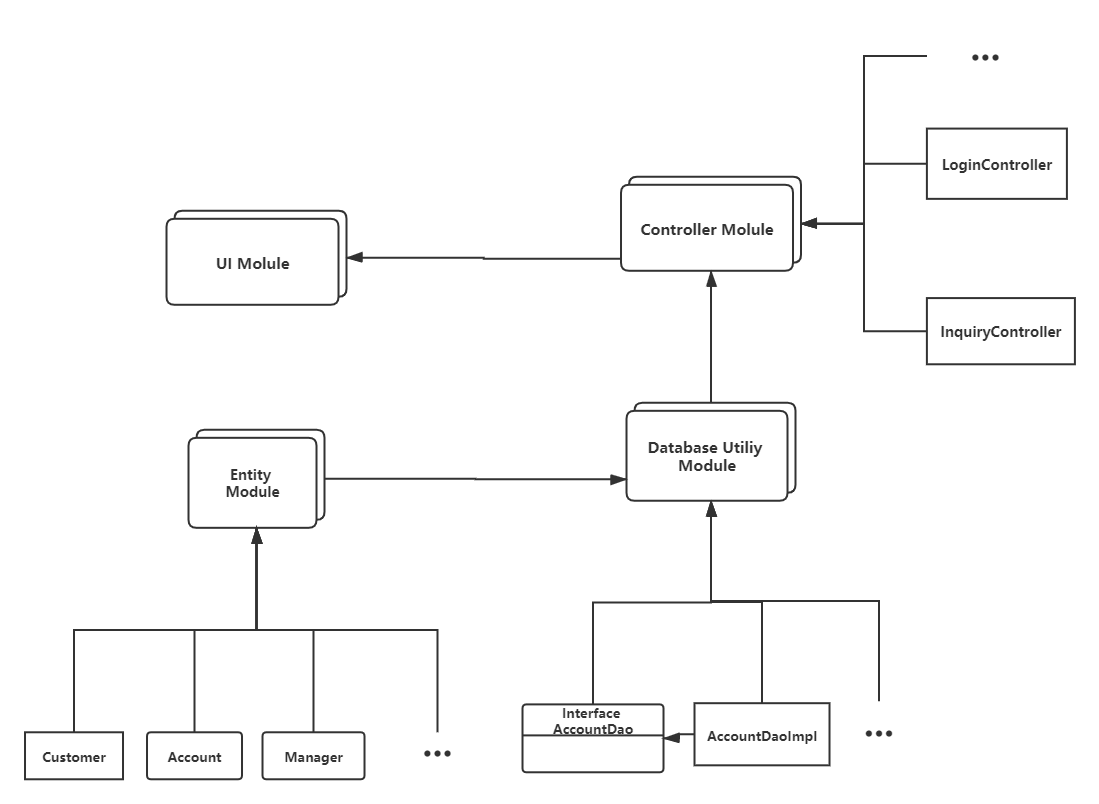
# Design Choice

For this assignment, we decide to create four modules to build the bank system. First of all, we need to create a module that defines all entity classes, and we call it Entity Module, which include classes like Customer, Account, Bank and so on. All these entity classes will be the carriers of the data that flow in our bank system. And considering that we need several currencies, so we define class Currency and three classes that inherits from it. These three classes are implemented in Single Pattern since we shouldn’t create a new Currency instance every time we need it. Then we create a class DigitMoney to encapsulate the Currency and number of money.

Next, we create a module that interacts with the database of the our bank, which we call Database Utility Module. We use sqlite database, which is light-weighted and powerful enough for the project. In this module, we define several interfaces with the tables of database they interact with, and implement them in specific classes. Every time we interact with database, we employ instances of Entity Module as carriers of the data.

Then, we create the Controller Module. This module calls methods defined in Database Utility Module to complete some operations like login, sign up and other operations users may need in our bank system.

Finally, we need the User Interface Module to provide our users with clear and easy-to-use graphic interfaces with Java Swing. In this module, we employ the Controller Module to take responsibility of handling the users’ input and other interactions with the GUI, which including all kinds of events that are required to offer feedback to the user.



# Benefit of Design

We separate the project into 4 modules, the details of logic operations in each module are kept hidden to each other. So basically we don’t need to worry about the extensibility. For example, if we need to provide customer with a new type of account, then we can add a class extending Account and override some methods. What we do in Entity Module will not affect other modules. Also, we use single pattern to define the currency classes, so that we don’t need to create the currency objects every time we need them. If we need to add security account, we may need two new classes: one represents the security account extending from class Account, the other define the stocks which may change every day.

# Object Design

Class Account:

This class represents the accounts which allow customers to deposit, withdraw and transfer money. Customers need to open an account before they make use of the functions of our bank.

Class CheckingAccount:

This class extends the class Account which allow customers to deposit, withdraw and transfer money. It has a new attribute service\_fee and overrides the method withdraw()

Class SavingAccount:

This class extends the class Account which allow customers to deposit, withdraw and transfer money. It has a new attribute interest and the balance of this account will increase by the interest.

Interface AccountDao:

This interface defines methods that interact with the database of the bank. These methods are used to deal with table Account in the database.

Class AccountDaoImpl:

This class is responsible to connect to the database of the bank and do some operations with table Account, which implements the Interface AccountDao.

Class Bank:

This class represents the bank which store some basic information like the profit and total balance.

Interface BankDao:

This interface defines methods that interact with the database of the bank. These methods are used to deal with table Bank in the database.

Class BankDaoImpl:

This class is responsible to connect to the database of the bank and do some operations with table Bank, which implements the Interface BankDao.

Class User:

This class represents the user of the bank, which can be a customer or a manager. In this class some basic methods are defined to maintain users’ info.

Class Customer:

This class represents the customer of the bank, which stores information like name, id, password and address.

Interface CustomerDao:

This interface defines methods that interact with the database of the bank. These methods are used to deal with table Customer in the database.

Class CustomerDaoImpl:

This class is responsible to connect to the database of the bank and do some operations with table Customer, which implements the Interface CustomerDao.

Class Manager:

This class represents the manager of the bank, which provides basic information of a manager.

Interface ManagerDao:

This interface defines methods that interact with the database of the bank. These methods are used to deal with table Manager in the database.

Class ManagerDaoImpl:

This class is responsible to connect to the database of the bank and do some operations with table Manager, which implements the Interface ManagerDao.

Class Transaction:

This class represents a transaction record of the bank. In a transaction, the account that starts the transaction and the account that receive the transaction is recorded, which is represented by an object of Class Transaction and stored in database.

Interface TransactionDao:

This interface defines methods that interact with the database of the bank. These methods are used to deal with table Transaction in the database.

Class TransactionDaoImpl:

This class is responsible to connect to the database of the bank and do some operations with table Transaction, which implements the Interface TransactionDao.

Class Currency:

This class represents the currency of the money that is used for the bank. It has the attribute exchanging rate, which can be used to exchange the type of currency.

Class USDollar:

This class extends the class Currency and represents the currency, USDollar. It is implemented in single pattern, which is not allowed to be initialized outside the class.

Class EuroDollar:

This class extends the class Currency and represents the currency, EuroDollar. It is implemented in single pattern, which is not allowed to be initialized outside the class.

Class CHYen:

This class extends the class Currency and represents the currency, CHYen. It is implemented in single pattern, which is not allowed to be initialized outside the class.

Class DigitMoney:

This class represents the money that is used in the bank system, which have two attributes a number and a Currency object.

Class LoginController:

This class is used to implement methods related to login and sign up interface of the bank system.

Class InquiryController:

This class is used to implement methods for manager to check customers’ information.

Class TransactionController:

This class is used to implement methods for manager to check all transaction records or for customers to check their own transactions records.

Class OpenAccountController:

This class is used to implement methods for customers to open an account and do some transactions with the accounts they have already own.

.