**Project**

The project has three parts. They are Framework, Bank, and Credit Card applications. Bank and Credit Card applications are using the framework code as a backbone and reuse the framework code in the Bank and Credit card.

The Patterns we used in our project are :

1. **Command Pattern**

Command Pattern is used to do all the operations from UI. When ActionPerformed is executed, it creates the concrete command by using the interface ICommand. With the help of CommandManager it invokes the command and does the operations like AddPerson, AddCompany, AddAccount, AddInterest, Deposit, Withdraw, and report generation. Each command calls FinCo as the receiver. This helps to decouple the UI from the implementation logic. We can achieve some principles like SRP, LSP,

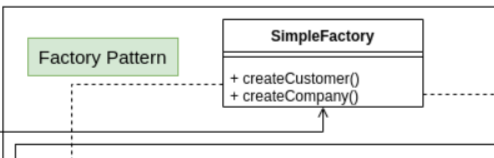
Diagram

Description automatically generated

1. **Factory Pattern**

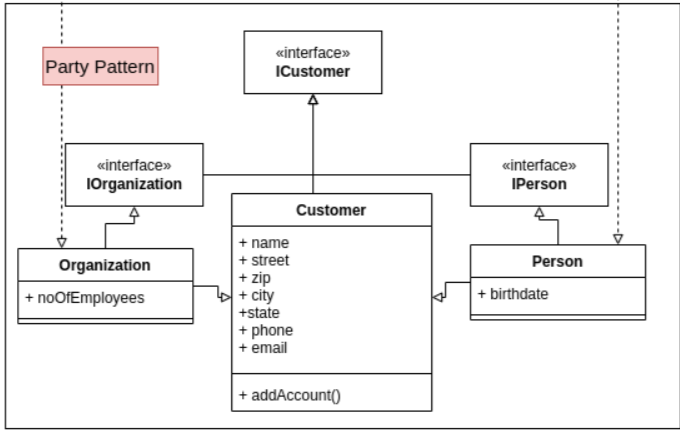
We use factory pattern to wrap the creational login of customers and account in public static methods which reuse in the code easily.

Diagram

Description automatically generated

1. **Party Pattern**

Party pattern used for represent two types of Customer in the bank. Abstract Customer, IPerson, IOrganization implements ICustomer to refer to all customers with a similar type with different behaviors or operations. When application creates a customer, it have to implement the ICustomer.



1. **Façade Pattern**

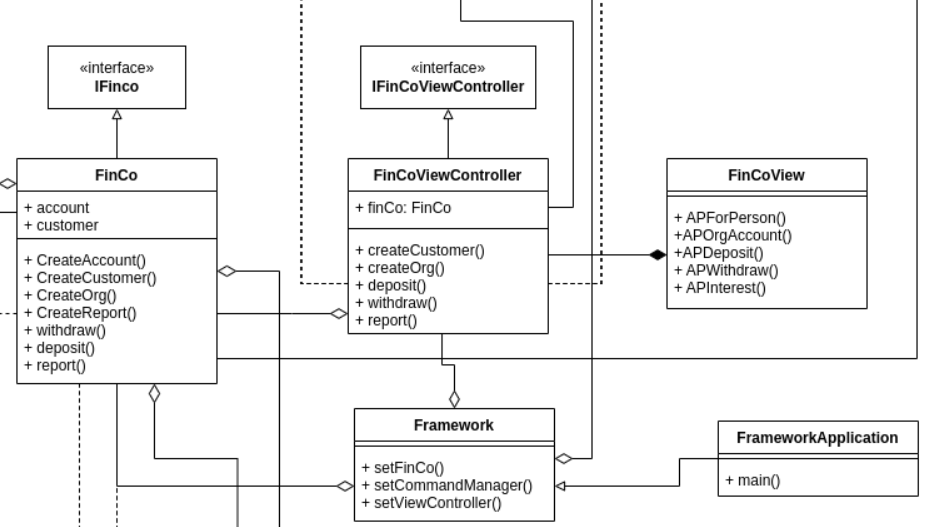
We use façade pattern to hide the details implementation of the persistence layer in database. IRepository provides a simplified interface to a framework which can be used to store the data.

Diagram

Description automatically generated

1. **MVC Pattern**

We use the MVC pattern in all three applications where the Model contains the pure application data, it doesn’t contain logic describing how to present data to a user, the View represents the model’s data to the user, and the Controller exits between view and model, listens to the ActionListener to perform the action. Since the view and model are connected through a notification mechanism, the result of the action is then automatically reflected in the view.



1. **Strategy Pattern**

We used a strategy pattern to generate report according to the different strategies. For Bank and credit card, there is different strategies to generate the reports. The bank uses AllAccountsReport and credit card uses MonthlyBillingReport which are different strategies in the runtime it will select the strategy according to the context.

