Design of a banking system

Description:
In this problem, you are going to design a banking system. Follow the below-written instructions, scenarios, and constraints,

a) Design a class (**Bank**) representing a bank containing common attributes and methods. You should design such a class that, using this, you can create instances of different banks.
b) A bank has two types of **Clients** to communicate with, **Employee** and **Customer**. There can be three types of employees - **Manager, Officer**, **Trainee**. Also, there can be two types of customers - **SinglePerson** and **Organization**. Each type of customer has at least the following attributes, such as **name**, **email, accountNumber, BankName** and **phoneNumber** along with others. A SinglePerson customer must have a **BIN** number and an organization-type customer must have a **TIN** number. An employee can be a customer and a customer can have accounts in multiple banks.
c) There can be two types of **Account**s for each type of customer, **Savings** and **Salary**. Each type of account should have a different **tax-cut** scheme and **interest** scheme. For saving type and salary type accounts, they get an annual of 2.5% and 2% interest respectively. You should add some logic so that you can calculate the total amount in an account after some time.
d) There are three types of **MoneySendToAccount** options - **Bkash** based **MobileApplication** System to send money from BKashWallet to an account, **EFT** based (Electronic Fund Transfer) **MobileApplication** system to send money from one account to another, **BankReceipt** based manual system.
e) People can withdraw money from an account in the following ways,
    i) **DirectCheque** based manual System
    ii) Save money from an account to a **BkashWallet**
    iii) **CreditCard** Based System.

You should embed logic so that, you never violate the constraint of withdrawing more money than found in an account.

Evaluation Criteria:
This is a design-related problem. So, you will be evaluated on how many object-oriented designs you have applied in your solution. More specifically, we will evaluate the classes and the in-between relationships you have established in your code - their novelty, logical decision-related paradigms, uses, and connections. There can be multiple valid solutions, but some solutions are more flexible, object-oriented, and easily extendable or easily modifiable. The evaluation metric will bear these points.