# BankProject

December 27, 2023

## 0.1 Bank Domain

### 0.1.1 Problem Statement: (Use parameterized constructors in all classes to initialize default values)

**Create a bank class with the following attributes:**

```
    o IFSC_Code
    o bankname
    o branchname
    o loc
```

**Create a customer class with the following attributes:**

```
o CustomerID
o custname
o address
o contactdetails
```

**Create an account class that inherits from bank class with the following attributes (Use Super () to pass** value to the base class): AccountID Cust Object of Customer balance #### Add the following methods to get account information, withdraw, and deposit: getAccountInfo() deposit(2000,'true') widthraw(500) getBalance()

**Create a SavigsAccount class that inherits from the account with the following attributes (Use Super () to** pass valued to the base class): SMinBalance #### Add the following methods to get account information, withdraw, and deposit: getSavingAccountInfo() deposit(2000,'true') widthraw(500) getBalance() ### Validate MinBalance before allowing withdrawals. #### Create a class that runs the program and accepts input from the end user to create respective class #### objects and print details. Add a method to perform deposit and withdrawal transaction based on the end user input.

[ ]:
```python
class Bank:
    def __init__(self, ifsc_code, bank_name, branch_name, location):
        self.IFSC_Code = ifsc_code
        self.bankname = bank_name
        self.branchname = branch_name
        self.loc = location
```

```python
class Customer:
    def __init__(self, customer_id, customer_name, address, contact_details):
        self.CustomerID = customer_id
        self.custname = customer_name
        self.address = address
        self.contactdetails = contact_details


class Account(Bank):
    def __init__(self, ifsc_code, bank_name, branch_name, location, account_id,␣
 ↪customer, balance=0.0):
        super().__init__(ifsc_code, bank_name, branch_name, location)
        self.AccountID = account_id
        self.Cust = customer
        self.balance = balance

    def getAccountInfo(self):
        return f"Account ID: {self.AccountID}\nCustomer ID: {self.Cust.
 ↪CustomerID}\nCustomer Name: {self.Cust.custname}\nBalance: {self.balance}"

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            return True
        else:
            return False

    def withdraw(self, amount):
        if amount > 0 and self.balance - amount >= 0:
            self.balance -= amount
            return True
        else:
            return False

    def getBalance(self):
        return self.balance


class SavingsAccount(Account):
    def __init__(self, ifsc_code, bank_name, branch_name, location, account_id,␣
 ↪customer, s_min_balance, balance=0.0):
        super().__init__(ifsc_code, bank_name, branch_name, location,␣
 ↪account_id, customer, balance)
        self.SMinBalance = s_min_balance

    def getSavingAccountInfo(self):
        return f"{super().getAccountInfo()}\nMinimum Balance: {self.
 ↪SMinBalance}"

    def withdraw(self, amount):
        if super().withdraw(amount) and self.balance >= self.SMinBalance:
            return True
        else:
            return False


class BankProgram:
    def __init__(self):
        self.accounts = []

    def createAccount(self):
        ifsc_code = input("Enter IFSC Code: ")
        bank_name = input("Enter Bank Name: ")
        branch_name = input("Enter Branch Name: ")
        location = input("Enter Location: ")

        customer_id = int(input("Enter Customer ID: "))
        customer_name = input("Enter Customer Name: ")
        address = input("Enter Address: ")
        contact_details = input("Enter Contact Details: ")

        account_id = int(input("Enter Account ID: "))
        balance = float(input("Enter Initial Balance: "))

        customer = Customer(customer_id, customer_name, address,␣
 ↪contact_details)
        account = Account(ifsc_code, bank_name, branch_name, location,␣
 ↪account_id, customer, balance)

        self.accounts.append(account)
        print("Account created successfully!")

    def performTransaction(self):
        account_id = int(input("Enter Account ID for transaction: "))
        amount = float(input("Enter transaction amount: "))
        action = input("Enter 'D' for deposit or 'W' for withdrawal: ")

        for account in self.accounts:
            if account.AccountID == account_id:
                if action.upper() == 'D':
                    if account.deposit(amount):
                        print("Deposit successful.")
                    else:
                        print("Invalid deposit amount.")
                elif action.upper() == 'W':
                    if account.withdraw(amount):
                        print("Withdrawal successful.")
                        print("Available Balance:",account.getBalance())
                    else:
                        print("Invalid withdrawal amount or insufficient␣
 ↪balance.")
                else:
                    print("Invalid action. Please enter 'D' for deposit or 'W'␣
 ↪for withdrawal.")
                break
        else:
            print("Account not found.")

    def printAccountInfo(self):
        account_id = int(input("Enter Account ID to view information: "))
        for account in self.accounts:
            if account.AccountID == account_id:
                print(account.getAccountInfo())
                break
        else:
            print("Account not found.")


# Example Usage:

bank_program = BankProgram()

while True:
    print("\n1. Create Account\n2. Perform Transaction\n3. Print Account␣
 ↪Info\n4. Exit")
    choice = input("Enter your choice (1/2/3/4): ")

    if choice == '1':
        bank_program.createAccount()
    elif choice == '2':
        bank_program.performTransaction()
    elif choice == '3':
        bank_program.printAccountInfo()
    elif choice == '4':
        print("Exiting program.")
        break
    else:
        print("Invalid choice. Please enter a valid option.")
```

```
1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  1
Enter IFSC Code:  yes002
Enter Bank Name:  yes bank
Enter Branch Name:  madhapur
Enter Location:  hyderabad
Enter Customer ID:  1001
Enter Customer Name:  vamshimarikanti
Enter Address:  kothapet,hyderabad
Enter Contact Details:  vamshimarikanti7@gmail.com
Enter Account ID:  0001
Enter Initial Balance:  5000

Account created successfully!

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  2
Enter Account ID for transaction:  0001
Enter transaction amount:  500
Enter 'D' for deposit or 'W' for withdrawal:  W

Withdrawal successful.
Available Balance: 4500.0

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  3
Enter Account ID to view information:  0001

Account ID: 1
Customer ID: 1001
Customer Name: vamshimarikanti
Balance: 4500.0

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit
```

```
Enter your choice (1/2/3/4):  3
Enter Account ID to view information:  00001

Account ID: 1
Customer ID: 1001
Customer Name: vamshimarikanti
Balance: 4500.0

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  3
Enter Account ID to view information:  20001

Account not found.

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  2
Enter Account ID for transaction:  0001
Enter transaction amount:  5000
Enter 'D' for deposit or 'W' for withdrawal:  D

Deposit successful.

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  3
Enter Account ID to view information:  0001

Account ID: 1
Customer ID: 1001
Customer Name: vamshimarikanti
Balance: 9500.0

1. Create Account
2. Perform Transaction
3. Print Account Info
4. Exit

Enter your choice (1/2/3/4):  4
```