**Banking System Documentation**

**1. Introduction**

This program implements a banking management system to handle operations such as employee management, account handling, and branch operations. It uses object-oriented principles like inheritance, polymorphism, and aggregation to provide a flexible, scalable, and modular design.

**Purpose**

The main objective is to simulate a banking environment where branch and head office employees interact with accounts and customers under a centralized management system.

**Scope**

The system supports:

* Employee management (branch and head office employees).
* Account management (savings and current accounts).
* Branch-level and head-office-level operations.

**2. System Architecture**

**Class Overview**

The system is designed with the following primary classes:

* **HeadOffice**: Manages multiple branches and head office employees.
* **Employee**: Base class for all employees.
* **BranchEmployee** and **HeadEmployee**: Derived classes to manage branch-specific and head office employees, respectively.
* **Account**: Base class for all bank accounts.
* **SavingsAccount** and **CurrentAccount**: Derived classes to manage specialized account types.
* **Bank**: Represents a branch, managing its employees and accounts.

**Relationships**

* **Inheritance**:
  + Employee → BranchEmployee, HeadEmployee.
  + Account → SavingsAccount, CurrentAccount.
* **Aggregation**:
  + Bank manages a collection of employees and accounts.
  + HeadOffice aggregates multiple Bank instances and head office employees.

**3. Classes**

**3.1 Employee (Base Class)**

* **Purpose**: Represents a generic employee with common attributes and methods.
* **Attributes**:
  + name (string): Name of the employee.
  + dob (int): Date of birth in ddmmyy format.
  + post (string): Job title.
  + salary (double): Monthly salary.
* **Methods**:
  + display(): Pure virtual function to be implemented by derived classes.

**3.2 BranchEmployee (Derived Class)**

* **Purpose**: Represents an employee assigned to a specific branch.
* **Attributes**:
  + branch (string): Name of the branch.
  + doj (int): Date of joining.
* **Methods**:
  + display(): Outputs the employee's details, including branch information.
* **Friend Function**:
  + loadbanksemp(): To load the data of all branch employee from database
* **Friend Class:**
  + Bank: Bank class needs to use its private data members.

**3.3 HeadEmployee (Derived Class)**

* **Purpose**: Represents an employee at the head office.
* **Attributes**:
  + doj (int): Date of joining.
* **Methods**:
  + display(): Outputs the head office employee's details.
* **Friend Function**:
  + loadheadofficeemp(): To load the data of all head employee from database
* **Friend Class:**
  + HeadOffice : HeadOffice class needs to use its private data members.

**3.4 Account (Base Class)**

* **Purpose**: Represents a generic bank account.
* **Attributes**:
  + accountnumber (int): Unique identifier for the account.
  + pin (string): Password for account access.
  + holdername (string): Name of the account holder.
  + balance (double): Current account balance.
* **Methods**:
  + deposit(): Deposits a specified amount.
  + withdraw(): Virtual function to handle withdrawals.
  + displayAccountDetails(): Displays account details.
  + Getter functions for name, account number.

**3.5 SavingsAccount (Derived Class)**

* **Purpose**: Specialized account with savings features.
* **Attributes**:
  + minBal (int): Minimum required balance.
  + overdraft (bool): Indicates overdraft capability.
  + rateOfInterest (int): Interest rate for the account.
* **Friend Function**:
  + pointerToSavingsAccount(): Return the pointer of desired saving account
* **Friend Class:**
  + Bank: Bank class needs to use its private data members.

**3.6 CurrentAccount (Derived Class)**

* **Purpose**: Specialized account for current (business) users.
* **Attributes**:
  + minBal (int): Minimum required balance.
  + overdraft (bool): Indicates overdraft capability.
* **Friend Function**:
  + pointerToCurrentAccount(): Return the pointer of desired saving account.
  + Withdraw() : overridden withdraw function , in current account balance can be negative.
* **Friend Class:**
  + Bank: Bank class needs to use its private data members.

**3.7 Bank**

* **Purpose**: Represents a branch of the bank.
* **Attributes**:
  + branch (string): Branch name.
  + IFSC (string): Unique branch code.
  + allBranchEmployees (vector): List of branch employees.
  + allSavingsAccounts (vector): List of savings accounts.
  + allCurrentAccounts (vector): List of current accounts.
  + Public variable: minBalCurrent, minBalSavings, rateOfInterest
* **Methods**:
  + addBranchEmployee(): Adds a new branch employee.
  + openSavingsAccount(), openCurrentAccount(): Handles account creation.
  + displayAllBranchEmployees(): Lists all branch employees.
  + loadSavingsAccount() and loadCurrentAccount() : to load data from database.
  + Get function for ifsc code.
  + Display function for number of accounts, all savings account, all current accounts and all branch employees.
* **Friend Function**:
  + loadbankemp(): To load the data of all banks employee from database
  + loadheadofficeBanks(): To load the data of all Bank in vector of HeadOffice from database
  + pointer to savings account and current account as stated above.
* **Friend Class:**
  + HeadOffice : HeadOffice class needs to use its private data members.

**3.8 HeadOffice**

* **Purpose**: Represents the head office managing branches and employees.
* **Attributes**:
  + allBanks (vector): List of associated branches.
  + allHeadEmployees (vector): List of head office employees.
* **Methods**:
  + addHeadEmployee(): Adds new entities.
  + addBank(Bank b) and addBank() : function overloading for directly adding data from database or by entering new data manually.
  + displayAllBanks(): Displays details of all banks.
  + displayAllHeadEmployees(): Displays details of all Head Employees.
  + getBankByIFSC(): Retrieves a branch by its IFSC code.
* **Friend Function**:
  + loadbankemp(): To load the data of all banks employee from database

**System Workflow**

This section describes how the different components of the banking system interact during typical operations. Here are the key workflows:

**1. Adding a New Branch Employee**

1. **Input**:
   * Employee details such as name, date of birth, post, salary, branch name, and date of joining.
2. **Process**:
   * Create a BranchEmployee object using the provided details.
   * Add the object to the allBranchEmployees vector of the corresponding Bank.
3. **Output**:
   * Confirmation message of the employee's addition.
   * Optionally display the employee's details.

**2. Adding a New Head Office Employee**

1. **Input**:
   * Head office employee details such as name, date of birth, post, salary, and date of joining.
2. **Process**:
   * Create a HeadEmployee object.
   * Add the object to the allHeadEmployees vector of the HeadOffice.
3. **Output**:
   * Confirmation of the head office employee's addition.

**3. Opening a New Savings Account**

1. **Input**:
   * Account holder details such as name, initial deposit, account number, and password.
2. **Process**:
   * Create a SavingsAccount object with attributes such as minBal, rateOfInterest, and overdraft capability.
   * Add the account to the allSavingsAccounts vector in the corresponding Bank.
3. **Output**:
   * Account creation success message.
   * Display of account details (e.g., balance, account number).

**4. Opening a New Current Account**

1. **Input**:
   * Similar to the savings account process but specific for business-related attributes.
2. **Process**:
   * Create a CurrentAccount object.
   * Add the object to the allCurrentAccounts vector in the Bank.
3. **Output**:
   * Account creation confirmation.

**5. Displaying All Employees in a Branch**

1. **Input**:
   * Call the displayAllBranchEmployees() method for a specific Bank object.
2. **Process**:
   * Iterate over the allBranchEmployees vector.
   * Display the details of each BranchEmployee object.
3. **Output**:
   * List of all branch employees, including their name, post, salary, and date of joining.

**6. Managing Multiple Banks in the Head Office**

1. **Adding a Bank**:
   * Input: Branch name and IFSC code.
   * Process: Create a Bank object and add it to the allBanks vector in HeadOffice.
   * Output: Confirmation of bank addition.
2. **Retrieving a Bank by IFSC**:
   * Input: IFSC code.
   * Process: Search the allBanks vector for a matching IFSC.
   * Output: Return the matching Bank object.
3. **Displaying All Banks**:
   * Input: Call displayAllBanks() from HeadOffice.
   * Process: Iterate over the allBanks vector and display each bank's details.
   * Output: List of all banks and their associated IFSC codes.

**7. Performing Account Operations**

1. **Depositing Money**:
   * Input: Account number, deposit amount.
   * Process: Locate the account, add the amount to balance, and update the total.
   * Output: Updated balance confirmation.
2. **Withdrawing Money**:
   * Input: Account number, withdrawal amount.
   * Process:
     + Check account type (SavingsAccount or CurrentAccount).
     + Verify sufficient balance and overdraft limits.
     + Deduct the amount from balance.
   * Output: Withdrawal confirmation or error message for insufficient balance.

**8. Displaying All Accounts in a Branch**

1. **Input**:
   * Call displayAllAccounts() from a Bank object.
2. **Process**:
   * Combine allSavingsAccounts and allCurrentAccounts vectors.
   * Iterate and display details for each account.
3. **Output**:
   * List of all accounts in the branch, including holder names and balances.

List of Options Provided:

1. Add bank
2. Add employee in head office
3. Login as admin of a bank
4. Display the details of this Bank
5. Display employees of this branch
6. Display all branches of banks
7. Display all employees in head office
8. Add employee in this branch
9. Open an account in this branch
10. Become admin of an account
11. Display all savings accounts
12. Display all current accounts
13. Change pin
14. Withdraw
15. Deposit