**Class 1: BankAccount (Base Class)**

This class will represent a general bank account. It will act as the **parent class** for specific types of accounts (like Savings and Checking accounts).

**Attributes (Variables) for BankAccount:**

1. **account\_holder**: Name of the account holder (String).
2. **account\_number**: Unique identifier for the account (Integer).
3. **balance**: Current balance in the account (Float).
4. **password**: Password for account authentication (String).
5. **is\_active**: Indicates if the account is active or closed (Boolean).

**Methods for BankAccount:**

1. **\_\_init\_\_**: Constructor to initialize the account (sets initial values).
2. **deposit(amount)**: Adds the given amount to the balance.
3. **withdraw(amount)**: Deducts the given amount if balance allows (with validation).
4. **check\_balance()**: Returns the current balance.
5. **authenticate(password)**: Checks if the provided password matches the account's password.
6. **close\_account()**: Marks the account as inactive.

**Class 2: SavingsAccount (Derived Class)**

This class will represent a savings account, inheriting from BankAccount. It will add specific attributes and methods related to savings accounts.

**Additional Attributes for SavingsAccount:**

1. **interest\_rate**: Annual interest rate for the savings account (Float).
2. **min\_balance**: Minimum required balance to avoid penalties (Float).

**Additional Methods for SavingsAccount:**

1. **apply\_interest()**: Calculates and adds interest to the balance.
2. **withdraw(amount)**: Overridden method to ensure the minimum balance is maintained.
3. **check\_min\_balance()**: Checks if the current balance meets the minimum balance requirement.

**Class 3: CheckingAccount (Derived Class)**

This class will represent a checking account, inheriting from BankAccount. It will have features specific to checking accounts, such as overdraft limits.

**Additional Attributes for CheckingAccount:**

1. **overdraft\_limit**: Maximum allowable overdraft (Float).
2. **overdraft\_fee**: Fee charged when the account enters overdraft (Float).

**Additional Methods for CheckingAccount:**

1. **withdraw(amount)**: Overridden method to allow withdrawals within the overdraft limit.
2. **apply\_overdraft\_fee()**: Deducts the overdraft fee when overdraft is used.

**Class 4: Bank (Aggregation Class)**

This class will represent the **bank system** as a whole. It will manage multiple accounts and provide services like account creation, deletion, and transaction history tracking.

**Attributes for Bank:**

1. **accounts**: A list of all the bank accounts (List of BankAccount objects).
2. **next\_account\_number**: Tracks the next available account number (Integer).
3. **transaction\_logs**: A log of all transactions performed (List of Strings).

**Methods for Bank:**

1. **create\_account(account\_type, holder\_name, initial\_deposit, password)**: Creates and adds a new account (Savings or Checking) to the accounts list.
2. **close\_account(account\_number)**: Marks the specified account as inactive.
3. **find\_account(account\_number)**: Searches and returns the account with the given account number.
4. **log\_transaction(details)**: Records a transaction into the logs.
5. **generate\_report()**: Prints a summary report of all accounts and their statuses.

**Relationships Between Classes:**

* **Inheritance**:
  + SavingsAccount and CheckingAccount inherit from BankAccount.
* **Aggregation**:
  + Bank contains a collection of BankAccount objects, including both savings and checking accounts.

**Summary**

**Classes:**

1. **BankAccount** (Base Class)
2. **SavingsAccount** (Inherits from BankAccount)
3. **CheckingAccount** (Inherits from BankAccount)
4. **Bank** (Manages multiple accounts)

**Attributes/Variables:**

* **BankAccount**: account\_holder, account\_number, balance, password, is\_active
* **SavingsAccount**: interest\_rate, min\_balance
* **CheckingAccount**: overdraft\_limit, overdraft\_fee
* **Bank**: accounts, next\_account\_number, transaction\_logs

**Methods:**

* **BankAccount**: deposit(), withdraw(), check\_balance(), authenticate(), close\_account()
* **SavingsAccount**: apply\_interest(), withdraw(), check\_min\_balance()
* **CheckingAccount**: withdraw(), apply\_overdraft\_fee()
* **Bank**: create\_account(), close\_account(), find\_account(), log\_transaction(), generate\_report()

This breakdown provides a clear structure for how the classes, variables, and methods fit together in an OOP-based bank account system.