Bank Account Management System Documentation

Overview

This is a simple bank account management system implemented in Java. It allows users to create and manage two types of bank accounts: **Savings** and **Current**. The system supports basic functionalities like depositing, withdrawing, checking balance, displaying account details, and handling specific features for each account type, such as interest computation for savings accounts and minimum balance checking for current accounts. Additionally, account authentication is implemented using passwords.

Key Features:

1. **Account Creation**: Users can create either a savings or a current account.
2. **Deposit and Withdrawal**: Allows users to deposit and withdraw money from their accounts.
3. **Account Details**: Users can view the details of their accounts.
4. **Interest Computation**: Savings accounts can compute and deposit annual interest.
5. **Minimum Balance Check**: Current accounts ensure a minimum balance is maintained, otherwise, a penalty is imposed.
6. **Account Authentication**: Users authenticate using a password to access their accounts.

Components

1. Interfaces and Abstract Classes

`Authenticatable` Interface

- **Purpose**: Defines the method `authenticate(String password)` to authenticate an account based on the provided password.
- **Method**:
    - `boolean authenticate(String password)`: Returns `true` if the provided password matches the account password.

`Account` Abstract Class

- **Purpose**: Serves as the base class for both `SavingsAccount` and `CurrentAccount`. It contains the common properties and methods related to account management.
- **Attributes**:
  - `String name`: Name of the account holder.
  - `String accType`: Type of account (`Savings` or `Current`).
  - `int accNum`: Unique account number.
  - `int accBalance`: Current balance in the account.
  - `static Set<Integer> usedAccountNumbers`: Ensures unique account numbers.
- **Methods**:
  - `initialize(String name, String accType, int accBalance)`: Initializes account details and generates a unique account number.
  - `protected static int generateAccountNumber()`: Generates a unique 4-digit account number.
  - `abstract void displayDetails()`: Displays account details.
  - `abstract void deposit(int amount)`: Deposits an amount into the account.
  - `abstract void withdraw(int amount)`: Withdraws an amount from the account.

---

2. Concrete Account Classes

`SavingsAccount` Class

- **Purpose**: Represents a savings account, which is a type of `Account`. It implements the `Authenticatable` interface and supports interest computation.
- **Attributes**:
  - `String password`: The account password.
  - `static final double INTEREST_RATE`: The annual interest rate for savings accounts (3%).
- **Methods**:
  - `initializeSavings(String name, String password, int initialBalance)`: Initializes the savings account with the name, password, and initial balance.
  - `authenticate(String inputPassword)`: Authenticates the account using the stored password.
  - `computeAndDepositInterest()`: Computes and adds interest to the balance.

    - `displayDetails()`: Displays account details (account type, name, account number, balance).
    - `deposit(int amount)`: Deposits an amount into the account.
    - `withdraw(int amount)`: Withdraws an amount from the account if sufficient balance is available.

`CurrentAccount` Class

- **Purpose**: Represents a current account, which is another type of `Account`. It also implements the `Authenticatable` interface and checks for minimum balance.
- **Attributes**:
    - `String password`: The account password.
    - `static final int MIN_BALANCE`: The required minimum balance (1000).
    - `static final int PENALTY`: Penalty imposed for not maintaining the minimum balance (100).
- **Methods**:
    - `initializeCurrent(String name, String password, int initialBalance)`: Initializes the current account with the name, password, and initial balance.
    - `authenticate(String inputPassword)`: Authenticates the account using the stored password.
    - `checkMinimumBalance()`: Checks if the balance is below the minimum required and imposes a penalty if necessary.
    - `displayDetails()`: Displays account details (account type, name, account number, balance).
    - `deposit(int amount)`: Deposits an amount into the account.
    - `withdraw(int amount)`: Withdraws an amount from the account if sufficient balance is available.

---

3. Main Class

The `Main` class manages the user interface and interaction with the bank account system. It allows users to:

1. Create savings or current accounts.
2. Deposit or withdraw money.
3. View account balance and details.
4. Perform specific functions like interest computation or checking minimum balance.

Key Methods:

- **findAccount(List accounts, Scanner in)**: A helper method to find an account by its account number.
- **Main Method (`public static void main(String[] args)`)**: The entry point of the program that provides a menu for users to interact with the system. It handles user choices for creating accounts, depositing, withdrawing, checking balance, and more.

Sample Usage

Upon running the program, users are presented with the following menu:

1. **Create Savings Account**: Users are prompted to enter their name, set a password, and provide an initial balance.
2. **Create Current Account**: Similar to creating a savings account, but for current accounts with a different set of rules (like minimum balance).
3. **Deposit Money**: Allows users to deposit money into their accounts.
4. **Withdraw Money**: Allows users to withdraw money from their accounts.
5. **Check Balance**: Displays the current balance in the selected account.
6. **Display Account Details**: Shows detailed information about the selected account.
7. **Compute Interest (Savings Only)**: Only available for savings accounts to compute and add interest.
8. **Check Minimum Balance (Current Only)**: Available for current accounts to check and ensure the minimum balance is maintained.
9. **Quit**: Exit the program.

Conclusion

This project demonstrates the basic principles of object-oriented programming such as inheritance, interfaces, and abstraction, applied to a bank account management system. It offers fundamental banking operations and showcases the power of Java to create extensible and maintainable systems.