**Project Title:**

**Bank Management System**

**Project Description:**

The **Bank Management System** is a Python-based application that leverages **Object-Oriented Programming (OOP)** principles to manage various types of bank accounts and transactions. The system supports functionality such as account creation, deposits, withdrawals, applying interest, and displaying account details. It uses **CSV files** for persistent storage and separates each class into its own file for modularity.

**Objectives:**

* Demonstrate **OOP principles**: Encapsulation, Inheritance, Polymorphism, and Abstraction.
* Enable persistent data storage using **CSV files**.
* Create a user-friendly, menu-driven interface for account management.

**Features:**

1. **Add Accounts**:
   * Supports **Savings Account** and **Current Account** creation.
2. **Deposit Money**:
   * Allow deposits to any account.
3. **Withdraw Money**:
   * Handle withdrawals with type-specific rules:
     + Savings accounts cannot withdraw beyond their balance.
     + Current accounts support overdraft limits.
4. **Apply Interest**:
   * Apply an interest rate to savings accounts.
5. **View Account Details**:
   * Retrieve details such as account holder, balance, and account type.
6. **Persistent Storage**:
   * Save and retrieve account data from a CSV file.

**Key Technologies and Concepts:**

1. **Object-Oriented Programming**:
   * **Encapsulation**: Sensitive account data is private.
   * **Inheritance**: Savings and Current accounts inherit from a base account class.
   * **Polymorphism**: Different implementations of the withdraw method for account types.
   * **Abstraction**: Base account class serves as a template for derived classes.
2. **File Handling**:
   * **CSV**: Data is stored in and retrieved from a accounts.csv file.
3. **Modular Code**:
   * Each class resides in a separate file, improving maintainability and readability.

**Modules Used:**

* **csv**: For reading and writing account data to CSV files.

**Directory Structure:**

plaintext

Copy code

BankManagementSystem/

│

├── main.py # Main program file

├── bank\_account.py # Base class for all accounts

├── savings\_account.py # Savings account class

├── current\_account.py # Current account class

├── bank\_system.py # File handling and account management

└── accounts.csv # CSV file for persistent account data

**File Descriptions:**

1. **bank\_account.py**:
   * Contains the base class BankAccount, which defines common properties and methods for all account types.
   * Implements methods for depositing, updating balance, and retrieving account details.
   * Declares the abstract method withdraw, to be implemented by derived classes.
2. **savings\_account.py**:
   * Defines SavingsAccount, inheriting from BankAccount.
   * Adds an interest\_rate attribute and methods to apply interest.
   * Implements the withdraw method to restrict withdrawals beyond the account balance.
3. **current\_account.py**:
   * Defines CurrentAccount, inheriting from BankAccount.
   * Adds an overdraft\_limit attribute.
   * Implements the withdraw method to allow overdraft withdrawals within the specified limit.
4. **bank\_system.py**:
   * Handles account storage, retrieval, and management using CSV files.
   * Provides methods to load accounts from the CSV file, save accounts to it, add new accounts, and find accounts by account number.
5. **main.py**:
   * Entry point for the application.
   * Implements a menu-driven interface for user interaction.
   * Facilitates adding accounts, performing transactions, and displaying account details.

**System Requirements:**

* **Python 3.x**
* A CSV file (accounts.csv) for storing account data (created automatically if not found).

**Code Workflow:**

1. **Program Initialization**:
   * The program starts with main.py, which initializes an instance of BankSystem.
   * The BankSystem class attempts to load data from accounts.csv.
2. **Menu-Driven Interaction**:
   * The user is presented with a menu to perform operations:
     1. **Add Savings Account**:
        + Prompts the user for account details, creates a SavingsAccount instance, and saves it to the CSV file.
     2. **Add Current Account**:
        + Prompts the user for account details, creates a CurrentAccount instance, and saves it to the CSV file.
     3. **Deposit Money**:
        + Finds the account by number and allows depositing money.
     4. **Withdraw Money**:
        + Finds the account by number and performs withdrawals using the respective account type’s rules.
     5. **Apply Interest**:
        + Applies interest to all savings accounts.
     6. **View Account Details**:
        + Displays details for a specific account.
     7. **Exit**:
        + Terminates the program.
3. **Persistent Storage**:
   * Changes to account data (e.g., deposits, withdrawals) are immediately saved to the accounts.csv file.

**File Format for accounts.csv:**

The CSV file contains the following fields:

* **Account Number**: Unique identifier for the account.
* **Account Holder**: Name of the account holder.
* **Balance**: Current balance in the account.
* **Type**: Type of account (Savings or Current).
* **Interest Rate**: Interest rate for savings accounts (empty for current accounts).
* **Overdraft Limit**: Overdraft limit for current accounts (empty for savings accounts).

**Sample accounts.csv Data:**

Account Number,Account Holder,Balance,Type,Interest Rate,Overdraft Limit

101,John Doe,1500,Savings,0.05,

102,Jane Smith,2000,Current,,500

**Classes and Methods:**

1. **BankAccount (Abstract Class)**:
   * **Attributes**:
     + \_\_account\_number: Account number (private).
     + \_\_account\_holder: Account holder's name (private).
     + \_\_balance: Account balance (private).
   * **Methods**:
     + get\_account\_details(): Returns account details as a dictionary.
     + deposit(amount): Adds money to the balance.
     + withdraw(amount): Abstract method for withdrawing money.
     + get\_balance(): Returns the current balance.
     + update\_balance(new\_balance): Updates the balance.
2. **SavingsAccount**:
   * **Additional Attribute**:
     + interest\_rate: Annual interest rate.
   * **Additional Methods**:
     + apply\_interest(): Applies interest to the balance.
     + withdraw(amount): Restricts withdrawals beyond the balance.
3. **CurrentAccount**:
   * **Additional Attribute**:
     + overdraft\_limit: Maximum allowable overdraft.
   * **Additional Method**:
     + withdraw(amount): Allows withdrawals within the overdraft limit.
4. **BankSystem**:
   * **Attributes**:
     + accounts: List of account instances.
     + filename: Path to the CSV file.
   * **Methods**:
     + load\_accounts(): Reads account data from the CSV file.
     + save\_accounts(): Writes account data to the CSV file.
     + add\_account(account): Adds a new account to the system.
     + find\_account(account\_number): Finds an account by its number.