Individual Case Study - Bank System

You are to create an interface that allows you to simulate a bank to some degree.

Criteria:

- Code Format
- Clarity of Output (displays and error messages are clear)
- Adherence to specifications

Specifications

Your program must contain the following classes, each in a separate file:

- BankSystem.py
- BankAccount.py
- BankClient.py

You are free to add convenience methods.

BankClient

Represents a client whose identity is known by your bank.

Fields:
- String lastName: The client's name. (ex. "David")
- int idNumber: A UNIQUE number for tracking purposes. (ex. 123)
  * Two or more clients CANNOT have the same ID number!
- BankAccount account: The account under the client's name.
  * MUST NOT BE NULL!

Methods:
- Constructor: BankClient(int id, String name, BankAccount a)
- getName() return String
- getIDNumber() return int
- BankAccount getAccount()
- printDetails() - Display ID number and account details.

BankAccount

Represents a BankAccount being maintained in your bank.

Fields:
- balance: Amount of money in the account. (ex. 1234.567)
- Two or more accounts CANNOT have the same ID number!

Methods:
- Constructor: BankAccount(int id, double initialDeposit, double initialIRate)
- getBalance() return double
- getInterestRate() return double
- getIDNumber() return int
- printDetails() - Display ID number, current balance
- deposit(double amount)
- withdraw(double amount)
  * Returns false if amount exceeds balance and no deduction takes place.
  * Returns true if amount is deducted from balance successfully.

BankSystem

Represents your bank. This will be your entry class, so it should have a main() method.

Fields:
- BankAccount account[]: An array of 5 accounts.
- BankClient client[]: An array of 5 clients.

Methods:
- createAccount()
  * Creates a BankAccount and adds it to the Account table.
  * Returns true if BankAccount is successfully created and added.
- createClient(int id, String name)
  * Creates a BankClient and adds it to the Client Table.
  * Returns true if BankClient is successfully created and added.

- BankAccount findAccount(int id)
  * Returns a BankAccount with a matching ID.
  * Returns null if no match is found.

- BankClient findClient(int id)
  * Returns a BankClient with a matching ID.
  * Returns null if no match is found.

Main Menu

1. Account Management
   *Go to Account Management menu
2. Client Management
   Go to Client Management menu
3. Quit
   *Quit program

1. Account Management

- New Account
  *Create a new account. Ask for ID number, balance, and interest rate.
  * Ask again if an invalid input is detected (Example: ID already in use).

- List All Accounts
  List ID numbers of all accounts.

- Find an Account
  Ask for an ID number, then print details of a matching account.
  * Print an error message instead if no match is found or input is invalid.

- Deposit to an Account
  Ask for an ID number, then an amount. Deposit amount account if valid.
  Display an error message if input is invalid.

- Withdraw from an Account
  Ask for an ID number, then an amount. Withdraw amount from account if valid.
  * Print an error message if withdrawal fails of input is invalid.

- Return to Main Menu
  Go to Main Menu

2. Client Management

- New Client

  Create a new client. Ask for ID number, name, and account ID number.

  * Display an error message if input is invalid
- List All Clients

  List ID numbers of all clients.
- Find a Client

  *Ask for an ID number, then display details of a matching client (and his/her account).

  * Display an error message instead if no match is found or input is invalid.
- Return to Main Menu

  Go to Main Menu

Good luck!