Project 4 (Day 4): Advanced Features – LoanAccount, Transaction Logging, & Person Class

Description

Enhance the system by introducing:

- A **LoanAccount** that models a loan, including repayment and interest calculation.
- A **Transaction** class to record each operation.
- A **Person** class to represent an account holder, who may own one or more bank accounts and

may have relationships (such as a spouse or co-owner).

Additional Attributes

- **LoanAccount:**
  - **loanAmount** (double): The total amount of the loan.
  - **interestRate** (double): The interest rate applied on the loan.
- **Transaction:**
  - **transactionID** (String): A unique identifier for the transaction.
  - **accountNumber** (String): The account involved.
  - **type** (String): The kind of transaction (e.g., “DEPOSIT”, “WITHDRAWAL”, “TRANSFER”).
  - **amount** (double): The transaction amount.
  - **transactionDate** (Date): The date/time of the transaction.
- **Person:**
  - **personID** (String): A unique identifier for the person.
  - **name** (String): The person’s name.
  - **accounts** (Collection of Account objects): Accounts owned by the person.
  - **relationships** (Collection of Person objects): Other persons related to this person (e.g., spouse, family members).

Additional Methods

1. **LoanAccount:**
   - **repay(amount)**
     - **Purpose:** Deduct a repayment amount from the outstanding loan.
     - **Parameters:** `amount` (double) – must be positive and less than or equal to the outstanding loan.
     - **Return:** Void.
   - **calculateInterest()**
     - **Purpose:** Compute the interest on the remaining loan.
     - **Return:** Interest amount (double).
2. **Person:**
   - **addAccount(account)**

        - **Purpose:** Link an account to the person's account list.
        - **Return:** Void.
    - **addRelationship(person)**
        - **Purpose:** Establish a relationship with another person (e.g., to represent joint account holders).
        - **Return:** Void.

3. **Transaction:**

    - **toString()**
        - **Purpose:** Provide a string representation of the transaction details.
        - **Return:** A formatted String showing date, type, amount, and account involved.

Manual Test Cases (Using if/else Logic)

- **Test Case 1: LoanAccount Repayment & Interest**
    - **Action:** Create a LoanAccount with a `loanAmount` of 5000 and an `interestRate` of 0.10.
    - **Steps:**
        - Call `repay(1000)` on the LoanAccount.
        - Then call `calculateInterest()`.
    - **If/Else Check:**
        - **If** the outstanding loan is now 4000 and `calculateInterest()` returns 400 (i.e., 4000 * 0.10), **then** print "Loan repayment and interest calculation successful."
        - **Else** print "Error: Loan repayment or interest calculation incorrect."
- **Test Case 2: Linking a Person with an Account**
    - **Action:** Create a Person with a specific `personID` and name. Then create an account (of any type) for that person.
    - **If/Else Check:**
        - **If** the person's account list contains the newly created account, **then** print "Account successfully linked to person."
        - **Else** print "Error: Account not linked."
- **Test Case 3: Transaction Logging**
    - **Action:** After performing a deposit or withdrawal, create a Transaction record capturing the operation details.
    - **If/Else Check:**

- **If** the Transaction's string representation (via `toString()`) correctly displays the expected details, **then** print "Transaction logged correctly."
- **Else** print "Error: Transaction log details are incorrect."