Industry Problem Statement

Banking Transaction Management System (Python)

Business Background

A bank wants to build a Python-based internal system to manage:

- Customer account details
- Deposits and withdrawals
- Balance validation
- Interest calculation
- Transaction summaries

You are required to implement this system **incrementally**, simulating a **real-world banking workflow** and following clean coding practices.

Task 1: Capture Customer & Account Details (Input Validation)

Objective

Collect and validate customer account information.

Requirements

Write a program to accept:

- Account Number
- Customer Name
- Account Type (Savings / Current)
- Initial Balance

Business Rules

- Initial balance must be **≥ ₹1,000**
- Account type must be valid
- Customer name must contain only alphabets

Expected Outcome

Validated bank account ready for transactions.

Task 2: Account Creation Confirmation

Objective

Confirm successful account creation.

Requirements

- Display account number and current balance

Expected Outcome

Customer receives account creation confirmation.

Task 3: Deposit Amount Validation

Objective

Ensure valid deposit transactions.

Requirements

- Accept deposit amount

Business Rules

- Deposit amount must be **greater than 0**

Expected Outcome

Valid deposit amount accepted.

Task 4: Deposit Processing

Objective

Update account balance after deposit.

Formula

```
New Balance = Current Balance + Deposit Amount
```

Expected Outcome

Updated account balance.

Task 5: Withdrawal Amount Validation

Objective

Prevent invalid withdrawals.

Requirements

- Accept withdrawal amount

Business Rules

- Withdrawal amount must be **> 0**
- Balance after withdrawal must not go below **₹1,000**

Expected Outcome

Safe and validated withdrawal.

Task 6: Withdrawal Processing

Objective

Update balance after withdrawal.

Formula

```
New Balance = Current Balance − Withdrawal Amount
```

Expected Outcome

Correct post-withdrawal balance.

Task 7: Balance Inquiry

Objective

Allow customer to check available balance.

Expected Outcome

Current balance displayed clearly.

Task 8: Interest Calculation (Savings Account)

Objective

Apply interest for savings accounts.

Rules

- Interest Rate = **4% annually**
- Apply interest only if account type is Savings

Formula

```
Interest = Balance × 4 / 100
```

Expected Outcome

Interest amount calculated.

Task 9: Transaction History Tracking

Objective

Maintain record of all transactions.

Requirements

- Store deposits and withdrawals in a list

Expected Outcome

Transaction history ready for statements.

Task 10: Mini Statement Generation (Procedural)

Objective

Generate a mini account statement.

Statement Should Include

- Last 5 transactions
- Current balance

Task 11: BankAccount Class Design (OOP)

Objective

Model bank account as a real-world object.

Create class `BankAccount` with:

Attributes

- account_number
- customer_name
- account_type
- balance
- transactions

Task 12: Deposit & Withdrawal Methods

Objective

Encapsulate transaction logic.

Methods

- deposit(amount)
- withdraw(amount)

Task 13: Interest Method

Objective

Encapsulate interest logic.

Method

- calculate_interest()

Task 14: Statement Generation Method

Objective

Generate account statement programmatically.

Method

- generate_statement()

Task 15: Final Account Summary Report

Objective

Generate a professional bank account summary.

Output Format (Example)

```
Account Number : 1023456789
Customer Name  : Rohit Mehta
Account Type   : Savings
Current Balance: ₹52,400
Interest Earned: ₹2,096
Recent Txns    : Deposit ₹10,000, Withdrawal ₹2,000
```