Bank Application (ATM Simulation) - Problem Statement

Goal:

To build a simple console-based banking system where a user can:

- Create a bank account
- Perform basic banking operations like:
  - Deposit
  - Withdraw
  - Check balance
  - View transaction history

Main Functionalities:

1. Create New Account

- User enters their name
- System generates a unique account number (using Random class or a sequence)
- Account is initialized with zero balance

2. Check Balance

- User can check their current account balance

3. Deposit Money

- User can deposit money into their account
- After deposit:
  - Balance is updated
  - Transaction is recorded

4. Withdraw Money

- User can withdraw money
- If balance is insufficient, show a warning
- If withdrawal is successful:
  - Balance is updated
  - Transaction is recorded

Bank Application (ATM Simulation) - Problem Statement

5. View Transaction History

- User can see a list of all transactions (both deposits and withdrawals)

6. Exit System

- User exits the application

- Program ends

Concepts Involved:

1. Classes and Objects:

- Use separate classes for Account and Transaction

- Include attributes:

  - Account: name, account number, balance, list of transactions

  - Transaction: type (deposit/withdraw), amount, timestamp (optional)

- Include methods: deposit(), withdraw(), checkBalance(), showTransactionHistory()

2. ArrayList:

- Store transaction history dynamically using ArrayList<Transaction>

3. Random Account Number:

- Use Random class or counter to assign unique account numbers

4. Loops and Conditions:

- Use do-while loop to keep showing the menu until the user exits

- Use switch-case to handle different user choices

5. Scanner:

- Use Scanner class to take input from the user

Flow Example:

Bank Application (ATM Simulation) - Problem Statement

1. Program starts

2. User is asked to enter their name

3. Account is created with:

   - Random account number

   - Zero balance

4. A menu is shown:

   - Deposit

   - Withdraw

   - Check Balance

   - View Transactions

   - Exit

5. User selects an option

6. Based on the selection:

   - Operation is performed

   - Balance and transaction history are updated accordingly

7. User can repeat operations

8. When the user selects Exit, the program terminates