Current Project: Databricks & SQL Assessment Banks

Coding Assessment — QuickMart Traders: Profit Calculator (No Arrays / No Lists / No Generics / No SQL)

Background

QuickMart Traders is a small retail distributor that sells items to walk-in customers. The owner wants a minimal console-based utility that captures a single sale transaction, computes profit or loss, and prints a clean invoice-style summary. This assessment must be implemented as a C# Console Application using OOP structure and menu-driven interaction.

Assessment Focus

Technical Topic: C# If-Else + Switch + Methods + Classes (Arithmetic + Console I/O validation)

Domain / Business Scenario: Retail Sales (Purchase vs Selling Profit/Loss)

Core Entity Name: SaleTransaction

Key Metrics: PurchaseAmount, SellingAmount, ProfitOrLossAmount, ProfitOrLossStatus, ProfitMarginPercent

Core Entity (Main Class)

Create the following ENTITY CLASS (must be created):

Class Name: SaleTransaction

- InvoiceNo (string) — unique identifier (example: INV1001)
- CustomerName (string)
- ItemName (string)
- Quantity (int)
- PurchaseAmount (decimal) — total purchase cost for the invoice (not per-unit)
- SellingAmount (decimal) — total selling amount for the invoice (not per-unit)
- ProfitOrLossStatus (string) — PROFIT / LOSS / BREAK-EVEN (calculated)
- ProfitOrLossAmount (decimal) — calculated
- ProfitMarginPercent (decimal) — calculated (relative to PurchaseAmount)

Static Storage Requirement (No Arrays / No Lists / No Generics)

The application must store at most ONE computed transaction in memory for viewing and recalculation. Do NOT use arrays, List<T>, Dictionary<,>, or any other generic or collection types. Do NOT use SQL or any database integration.

- static SaleTransaction LastTransaction
- static bool HasLastTransaction

Functionalities

Your application must be menu-driven and repeatedly show the menu until the user exits.

Menu Options (must be numbered exactly):

1. 1. Create New Transaction (Enter Purchase & Selling Details)
2. 2. View Last Transaction
3. 3. Calculate Profit/Loss (Recompute & Print)
4. 4. Exit

Required Public Methods (Assessment Expectations)

Implement the following PUBLIC methods (names may vary, but behavior must match).

1) Create/Register Method

Purpose: Capture transaction inputs from the console, compute profit/loss values, and store the result in LastTransaction.

Required console inputs:

- InvoiceNo
- CustomerName
- ItemName
- Quantity
- PurchaseAmount (total)
- SellingAmount (total)

Expected behavior:

- Validate InvoiceNo is non-empty.
- Validate Quantity must be > 0.
- Validate PurchaseAmount must be > 0.
- Validate SellingAmount must be >= 0.
- Compute ProfitOrLossStatus, ProfitOrLossAmount, and ProfitMarginPercent as per rules below.

- Store the computed object as LastTransaction and set HasLastTransaction = true.

2) View Method

Purpose: Print LastTransaction in a clean, readable invoice-like format.

If HasLastTransaction is false, print: "No transaction available. Please create a new transaction first."

3) Calculation Method

Purpose: Recompute profit/loss for LastTransaction and print the computed output again.

If HasLastTransaction is false, print: "No transaction available. Please create a new transaction first."

Calculation Rules

Compute Profit/Loss using the following rules (keep the logic simple):

5. If SellingAmount > PurchaseAmount: ProfitOrLossStatus = PROFIT; ProfitOrLossAmount = SellingAmount - PurchaseAmount
6. If SellingAmount < PurchaseAmount: ProfitOrLossStatus = LOSS; ProfitOrLossAmount = PurchaseAmount - SellingAmount
7. If SellingAmount == PurchaseAmount: ProfitOrLossStatus = BREAK-EVEN; ProfitOrLossAmount = 0
8. ProfitMarginPercent = (ProfitOrLossAmount / PurchaseAmount) * 100
9. Round displayed monetary values and percentage values to 2 decimal places

Note

- All output must be console-style (no GUI).
- If the user enters an invalid menu option, print a friendly message and re-display the menu.
- Do not terminate the program using forced termination APIs.

CONSTRAINTS (Do Not Modify)

- Do not use Environment.Exit() or any forced termination approach.
- Do not use file/database/network storage; use only in-memory static variables.
- Do not use SQL, database access libraries, or ORM frameworks.
- Do not use any third-party libraries.
- The application must be fully menu-driven via Console input/output.
- Validate inputs and handle invalid menu selections without crashing.
- Use OOPS structure: entity class + manager/service methods + menu in Main().
- Do not use arrays, List<T>, Dictionary<,>, LINQ over collections, or any generic collection types.

Input / Output (Illustrative Console Run)

User inputs are shown in angle brackets like <...> for clarity.

```
================== QuickMart Traders ==================
1. Create New Transaction (Enter Purchase & Selling Details)
2. View Last Transaction
3. Calculate Profit/Loss (Recompute & Print)
4. Exit
Enter your option: <1>

Enter Invoice No: <INV1001>
Enter Customer Name: <Sanjay>
Enter Item Name: <Mixer Grinder>
Enter Quantity: <2>
Enter Purchase Amount (total): <4200>
Enter Selling Amount (total): <5000>

Transaction saved successfully.
Status: PROFIT
Profit/Loss Amount: 800.00
Profit Margin (%): 19.05
-------------------------------------------------------

================== QuickMart Traders ==================
1. Create New Transaction (Enter Purchase & Selling Details)
2. View Last Transaction
3. Calculate Profit/Loss (Recompute & Print)
4. Exit
Enter your option: <2>

-------------- Last Transaction --------------
InvoiceNo: INV1001
Customer: Sanjay
Item: Mixer Grinder
Quantity: 2
Purchase Amount: 4200.00
Selling Amount: 5000.00
Status: PROFIT
Profit/Loss Amount: 800.00
Profit Margin (%): 19.05
-------------------------------------------
-------------------------------------------------------

================== QuickMart Traders ==================
1. Create New Transaction (Enter Purchase & Selling Details)
2. View Last Transaction
3. Calculate Profit/Loss (Recompute & Print)
4. Exit
Enter your option: <4>

Thank you. Application closed normally.
```