Current Project: Databricks & SQL Assessment Banks

Coding Assessment — MediSure Clinic: Simple Patient Billing (No Arrays / No Lists / No Generics)

Background

MediSure Clinic requires a simple console-based billing utility for walk-in patients. The billing desk captures basic patient details and charges, then applies an insurance-based discount rule to compute the final payable amount. The solution must be implemented as a C# Console Application using a clean OOP structure and menu-driven interaction.

Assessment Focus

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

Domain / Business Scenario: Healthcare (Clinic Billing)

Core Entity Name: PatientBill

Key Metrics: GrossAmount, DiscountAmount, FinalPayable

Core Entity (Main Class)

Create the following ENTITY CLASS (must be created):

Class Name: PatientBill

- BillId (string) — unique identifier (example: BILL1001)
- PatientName (string)
- HasInsurance (bool) — true if the patient is insured
- ConsultationFee (decimal)
- LabCharges (decimal)
- MedicineCharges (decimal)
- GrossAmount (decimal) — calculated (not typed by user)
- DiscountAmount (decimal) — calculated (not typed by user)
- FinalPayable (decimal) — calculated (not typed by user)

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

The application must store at most ONE computed bill in memory for viewing purposes. Do NOT use arrays, List<T>, Dictionary<,>, or any other generic or collection types.

- static PatientBill LastBill
- static bool HasLastBill

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 Bill (Enter Patient Details)
2. 2. View Last Bill
3. 3. Clear Last Bill
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 patient billing inputs from the console, compute payable amounts, and store the result in LastBill.

Required console inputs:

- BillId
- PatientName
- HasInsurance (Y/N)
- ConsultationFee
- LabCharges
- MedicineCharges

Expected behavior:

- Validate BillId is non-empty.
- Validate fees: ConsultationFee must be > 0; LabCharges and MedicineCharges must be >= 0.
- Compute GrossAmount, DiscountAmount, and FinalPayable using the billing rules below.
- Store the computed object as LastBill and set HasLastBill = true.

2) View Method

Purpose: Print the LastBill in a clean, readable format.

If HasLastBill is false, print: "No bill available. Please create a new bill first."

3) Clear Method

Purpose: Clear LastBill and reset HasLastBill.

After clearing, print: "Last bill cleared."

Billing Rules

Compute the billing amounts using the following rules (keep the logic simple):

5. GrossAmount = ConsultationFee + LabCharges + MedicineCharges
6. If HasInsurance is true, apply a 10% discount on GrossAmount (DiscountAmount = GrossAmount * 0.10)
7. If HasInsurance is false, DiscountAmount = 0
8. FinalPayable = GrossAmount - DiscountAmount
9. Round displayed monetary values to 2 decimal places

Note

- If the user enters an invalid menu option, print a friendly message and re-display the menu.
- All output must be console-style (no GUI).
- 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 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.

```
================== MediSure Clinic Billing ==================
1. Create New Bill (Enter Patient Details)
2. View Last Bill
3. Clear Last Bill
4. Exit
Enter your option: <1>

Enter Bill Id: <BILL1001>
Enter Patient Name: <Divya>
Is the patient insured? (Y/N): <Y>
Enter Consultation Fee: <600>
Enter Lab Charges: <250>
Enter Medicine Charges: <150>

Bill created successfully.
Gross Amount: 1000.00
Discount Amount: 100.00
Final Payable: 900.00
------------------------------------------------------------

================== MediSure Clinic Billing ==================
1. Create New Bill (Enter Patient Details)
2. View Last Bill
3. Clear Last Bill
4. Exit
Enter your option: <2>

----------- Last Bill -----------
BillId: BILL1001
Patient: Divya
Insured: Yes
Consultation Fee: 600.00
Lab Charges: 250.00
Medicine Charges: 150.00
Gross Amount: 1000.00
Discount Amount: 100.00
Final Payable: 900.00
--------------------------------
------------------------------------------------------------

================== MediSure Clinic Billing ==================
1. Create New Bill (Enter Patient Details)
2. View Last Bill
3. Clear Last Bill
4. Exit
Enter your option: <4>

Thank you. Application closed normally.
```