12214994_SaurabhRana

Project Brief: GymStream Membership Validation System

1. Executive Summary

GymStream is an emerging fitness-tech startup. To ensure a seamless user experience, the company requires a robust C# backend module to handle new membership enrollments. The system focuses on **Business Rule Validation** using custom exception handling and automated billing calculations based on loyalty-tier discounts.

---

2. Technical Specifications

A. Exception Architecture

To maintain clean code and specific error reporting, the system utilizes a custom exception class.

- **Custom Exception:** `InvalidTierException`
- **Inheritance:** Must inherit from the system `Exception` class.
- **Purpose:** To be thrown when a user inputs a subscription tier not supported by the platform.

B. Core Model: `Membership` Class

The system is built around the following data structure:

Property Name	Data Type	Description
`Tier`	`string`	The subscription plan (Basic, Premium, or Elite).
`DurationInMonths`	`int`	Length of the membership contract.

Property Name	Data Type	Description
`BasePricePerMonth`	`double`	The standard monthly rate before discounts.

C. Functional Methods

Method 1: `ValidateEnrollment()`

- **Return Type:** `bool`
- **Logic Gate 1:** If `Tier` is not "Basic", "Premium", or "Elite", throw `InvalidTierException`.
- **Logic Gate 2:** If `DurationInMonths` ≤ 0, throw a standard `Exception`.
- **Success:** Return `true` if all business rules are satisfied.

Method 2: `CalculateTotalBill()`

- **Return Type:** `double`
- **Formula:** $Total = (Duration \times Price)$
- **Discounting Logic:**
  - **Basic:** 2% Discount
  - **Premium:** 7% Discount
  - **Elite:** 12% Discount
- **Final Output:** $Total - (Total \times \text{Discount Rate})$

---

3. Implementation Workflow

1. **Input Phase:** Capture user data for Tier, Duration, and Price.
2. **Try Block:** Attempt to call `ValidateEnrollment()`.
3. **Calculation:** If no exception is thrown, invoke `CalculateTotalBill()`.
4. **Catch Blocks:** * Intercept `InvalidTierException` to display specific tier errors.
   - Intercept general `Exception` for duration errors.
5. **Output:** Display the final bill formatted to two decimal places.

---

4. Sample Test Cases

Scenario	Tier Input	Duration	Result
Success	Premium	12	Enrollment Successful.
Invalid Input	Gold	6	Error: Tier not recognized.
Zero Duration	Basic	0	Error: Duration must be at least one month.

Here is the **NUnit Test Suite** for the **GymStream Membership System**. These tests are designed to cover the "Happy Path" (successful enrollments) and the "Error Paths" (exception handling).

C# Unit Test Implementation

C#

```
using NUnit.Framework;
using GymStream;
using System;

namespace GymStream.Tests
{
    [TestFixture]
    public class MembershipTests
    {
        private Membership _membership;

        [SetUp]
        public void Setup()
        {
            _membership = new Membership();
        }

        // --- VALIDATION TESTS ---

        [Test]
        public void ValidateEnrollment_ValidTiers_ReturnsTrue()
        {
            // Arrange
            _membership.Tier = "Premium";
            _membership.DurationInMonths = 6;

            // Act & Assert
            Assert.IsTrue(_membership.ValidateEnrollment());
```

```
        }

        [Test]
        public void
ValidateEnrollment_InvalidTier_ThrowsInvalidTierException()
        {
            // Arrange
            _membership.Tier = "Gold"; // Invalid tier
            _membership.DurationInMonths = 12;

            // Act & Assert
            var ex = Assert.Throws<InvalidTierException>(() =>
_membership.ValidateEnrollment());
            Assert.AreEqual("Tier not recognized. Please choose an available
membership plan.", ex.Message);
        }

        [Test]
        public void ValidateEnrollment_ZeroDuration_ThrowsGeneralException()
        {
            // Arrange
            _membership.Tier = "Basic";
            _membership.DurationInMonths = 0; // Invalid duration

            // Act & Assert
            var ex = Assert.Throws<Exception>(() =>
_membership.ValidateEnrollment());
            Assert.AreEqual("Duration must be at least one month.",
ex.Message);
        }

        // --- CALCULATION TESTS ---

        [Test]
        [TestCase("Basic", 10, 100, 980)]    // 1000 - 2% = 980
        [TestCase("Premium", 10, 100, 930)]  // 1000 - 7% = 930
        [TestCase("Elite", 10, 100, 880)]    // 1000 - 12% = 880
        public void CalculateTotalBill_CorrectDiscountsApplied(string tier,
int duration, double price, double expected)
        {
            // Arrange
            _membership.Tier = tier;
            _membership.DurationInMonths = duration;
            _membership.BasePricePerMonth = price;

            // Act
            double actual = _membership.CalculateTotalBill();

            // Assert
            Assert.AreEqual(expected, actual, 0.001);
        }
    }
}
```

Test Documentation for the Project Brief

Test Name	Logic Tested	Expected Result
`ValidateEnrollment_ValidTiers`	Checks if Basic/Premium/Elite pass.	Returns `true`.
`InvalidTier_ThrowsException`	Checks custom exception handling.	Throws `InvalidTierException`.
`ZeroDuration_ThrowsException`	Checks numeric boundary validation.	Throws `System.Exception`.
`CalculateTotalBill_Correct`	Verifies 2%, 7%, and 12% math.	Matches calculated discounted total.

Why these tests are important:

1. **Ensures Business Rules:** It guarantees that "Gold" or "Silver" tiers cannot be accidentally processed by the system.
2. **Prevents Revenue Loss:** It confirms the discount logic is exactly 2%, 7%, or 12%, ensuring the shop doesn't over-discount.
3. **Documentation:** The tests act as "living documentation" for any other developers joining the project.

Here is the full, production-ready **Source Code** for the **GymStream Membership System**. You can copy this into a single `Program.cs` file to run the application and verify the logic against the requirements.

```
using System;


namespace GymStream
{
   // 1. CUSTOM EXCEPTION CLASS
   public class InvalidTierException : Exception
   {
      public InvalidTierException(string message) : base(message) { }
   }
```

```
// 2. CORE MEMBERSHIP CLASS
public class Membership
{
    public string Tier { get; set; }
    public int DurationInMonths { get; set; }
    public double BasePricePerMonth { get; set; }

    /// <summary>
    /// Validates business rules for Tier and Duration.
    /// Throws specific exceptions if rules are violated.
    /// </summary>
    public bool ValidateEnrollment()
    {
        // Tier Validation (Case Sensitive)
        if (Tier != "Basic" && Tier != "Premium" && Tier != "Elite")
        {
            throw new InvalidTierException("Tier not recognized. Please choose an available membership plan.");
        }

        // Duration Validation
        if (DurationInMonths <= 0)
        {
            throw new Exception("Duration must be at least one month.");
        }

        return true;
    }
```

```
    /// <summary>
    /// Calculates the final bill after applying Tier-based loyalty discounts.
    /// </summary>
    public double CalculateTotalBill()
    {
        double totalPrice = DurationInMonths * BasePricePerMonth;
        double discountPercentage = 0;

        switch (Tier)
        {
            case "Basic":
                discountPercentage = 2;
                break;
            case "Premium":
                discountPercentage = 7;
                break;
            case "Elite":
                discountPercentage = 12;
                break;
        }

        double discountAmount = totalPrice * (discountPercentage / 100);
        return totalPrice - discountAmount;
    }
}

// 3. PROGRAM ENTRY POINT
public class Program
```

```
{
    static void Main(string[] args)
    {
        Membership member = new Membership();

        try
        {
            Console.WriteLine("--- GymStream Enrollment Portal ---");

            Console.WriteLine("Enter membership tier (Basic/Premium/Elite):");
            member.Tier = Console.ReadLine();

            Console.WriteLine("Enter duration in months:");
            member.DurationInMonths = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Enter base price per month:");
            member.BasePricePerMonth = Convert.ToDouble(Console.ReadLine());

            // Perform Validation
            if (member.ValidateEnrollment())
            {
                Console.WriteLine("\nEnrollment Successful!");
                double finalBill = member.CalculateTotalBill();
                Console.WriteLine($"Total amount after discount: {finalBill:F2}");
            }
        }
        // Catch Block 1: Custom Exception for Tiers
        catch (InvalidTierException ex)
        {
```

```
            Console.WriteLine($"\nError: {ex.Message}");
        }
        // Catch Block 2: General Exception for Duration or numeric errors
        catch (Exception ex)
        {
            Console.WriteLine($"\nError: {ex.Message}");
        }

        Console.WriteLine("\nPress any key to close...");
        Console.ReadKey();
    }
  }
}
```