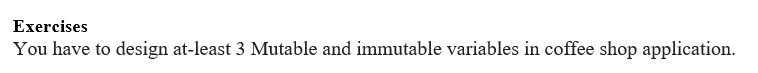


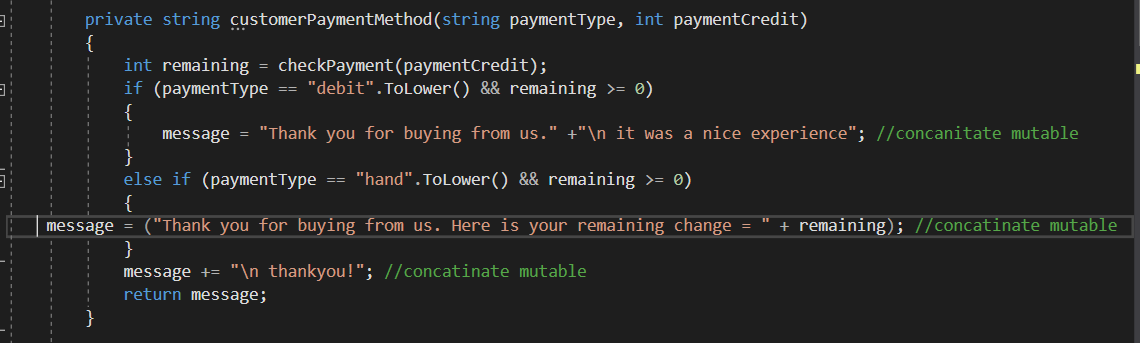
NAME: **Muhammad Arsalan**

CLASS: **BSE 5(B)**

Enrollment#: **02-131182-026**



**MUTABLE:**



**IMMUTABLE:**



**CODE:**

using System;

namespace CofeeShop

{

public class CoffeeShop

{

private int numberOfChocolateCoffee = 50;

private int numberOfVanillaCoffee = 50;

private readonly int priceOfCoffee = 60; //immutables

private static int totalAmount;

private string message;

public CoffeeShop(int amountVanilla, int amountChocolate, string paymentType, int paymentCredit)

{

totalAmountOfCoffee(amountVanilla, amountChocolate);

Console.WriteLine(customerPaymentMethod(paymentType, paymentCredit));

}

private string customerPaymentMethod(string paymentType, int paymentCredit)

{

int remaining = checkPayment(paymentCredit);

if (paymentType == "debit".ToLower() && remaining >= 0)

{

message = "Thank you for buying from us." +"\n it was a nice experience"; //concanitate mutable

}

else if (paymentType == "hand".ToLower() && remaining >= 0)

{

message = ("Thank you for buying from us. Here is your remaining change = " + remaining); //concatinate mutable

}

message += "\n thankyou!"; //concatinate mutable

return message;

}

private int checkPayment(int paymentCredit)

{

if (paymentCredit >= totalAmount)

{

paymentCredit -= totalAmount;

}

else

{

throw new Exception("Not enough credit provided.");

}

return paymentCredit;

}

private int totalAmountOfCoffee(int amount1, int amount2)

{

int totalCoffees = 100 - totalNumberOfCoffeesInStock(amount1, amount2);

totalAmount = totalCoffees \* priceOfCoffee;

return totalAmount;

}

private int totalNumberOfCoffeesInStock(int amountOfVanillaCoffee, int amountOfChocolateCoffee)

{

numberOfChocolateCoffee -= amountOfChocolateCoffee;

numberOfVanillaCoffee -= amountOfVanillaCoffee;

int totalNumberOfCoffee = numberOfVanillaCoffee + numberOfChocolateCoffee;

return totalNumberOfCoffee;

}

public static void Main()

{

int amountOfVanilla, amountOfChocolate, paymentCredit;

string paymentType;

Console.WriteLine("Please enter amount of Vanilla Coffee you want to buy : ");

amountOfVanilla = int.Parse(Console.ReadLine());

Console.WriteLine("Please enter amount of Chocolate Coffee you want to buy : ");

amountOfChocolate = int.Parse(Console.ReadLine());

Console.WriteLine("Please enter your payment method (debit/hand) : ");

paymentType = (Console.ReadLine());

Console.WriteLine("Please enter currency amount you are paying : ");

paymentCredit = int.Parse(Console.ReadLine());

CoffeeShop customerOne = new CoffeeShop(amountOfVanilla, amountOfChocolate, paymentType, paymentCredit);

}

}

}

**Note**: after we added mutable and immutable we still have the same answer after applying the test case:

**TESTCASE\_CODE:**

using System;

using Cofee\_shop1;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace testofcofeeshop

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void CorrestResponseOfRemainingAmount()

{

int amountOfVanilla = 5;

int amountOfChocolate = 5;

string paymentType = "hand";

int paymentCredit = 1000;

int expectedAmount = 400;

CoffeeShopClass customer1 = new CoffeeShopClass(amountOfVanilla, amountOfChocolate, paymentType, paymentCredit);

int actualAmount = customer1.checkPayment(paymentCredit);

Assert.AreEqual(expectedAmount, actualAmount, 1, "Something is wrong!");

}

[TestMethod]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void ThrowingAppropriateException()

{

int amountOfVanilla = 5;

int amountOfChocolate = 5;

string paymentType = "hand";

int paymentCredit = 500;

CoffeeShopClass customer1 = new CoffeeShopClass(amountOfVanilla, amountOfChocolate, paymentType, paymentCredit);

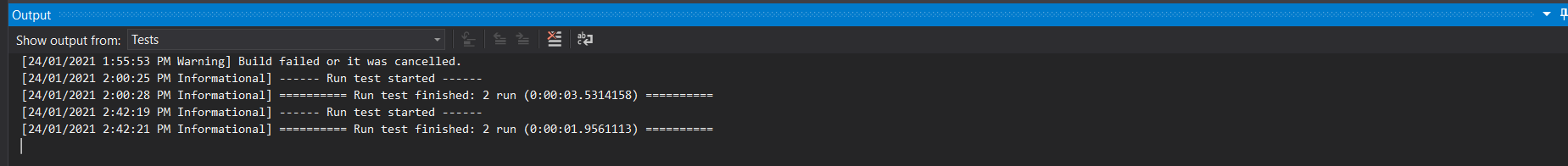
customer1.checkPayment(paymentCredit);

}

}

}

**Solution:**



**Output:**

