**Exercises**

You have to design at-least 3 Mutable and immutable variables in coffee shop application.

**Solution:**

**Coffee Project Library:**

using System;

namespace CoffeeProject

{

public class CoffeeProject

{

private string CustomerName;

**Three Immutable Variable:**

**private** readonly string FirstValue;

**private** readonly string SecondValue;

**private** readonly string ThridValue;

**Constructor of immutable variable:**

public CoffeeProject(string First, string Second, string Third)

{

FirstValue = First;

SecondValue= Second;

ThirdValue = Third;

}

**Getter Method on immutable variable:**

public string GetStrfirst

{

get { return FirstValue; }

}

public string GetStrSecond

{

get { return SecondValue; }

}

public string GetStrThrid

{

get { return ThirdValue; }

}

private double Customer\_Creditbalance;

public const string NoOrder = "Cancellation of order";

public const string NoMoney = "No money For Coffee";

private CoffeeProject()

{

}

public CoffeeProject(string customerName, double balance)

{

CustomerName = customerName;

Customer\_Creditbalance = balance;

}

public string CustomerName

{

get { return CustomerName; }

}

public double Balance

{

get { return Customer\_Creditbalance; }

}

public void Add\_Coffee(double Totalamount)

{

if (Totalamount = 100)

{

throw new Exception("Order Of Coffee");

}

if (Totalamount > Customer\_Creditbalance)

{

throw new ArgumentOutOfRangeException("amount", Totalamount, DebitAmountExceedsBalanceMessage);

}

if (Totalamount < 0)

{

throw new ArgumentOutOfRangeException("amount", Totalamount, DebitAmountLessThanZeroMessage);

}

Customer\_Creditbalance += Totalamount;

}

public void Cancelation\_Of\_CoffeeOrder(double Totalamount)

{

if (TotalAmount< 50)

{

throw new Exception("Cancel coffee Cup");

}

if (Totalamount < 0)

{

throw new ArgumentOutOfRangeException("amount less for coffee");

}

Customer\_Creditbalance += Totalamount;

}

private void BillGenerate\_ForCoffee()

{

TotalAmount = true;

}

private void OrderRequire\_ForCoffee()

{

TotalAmount = false;

}

public static void Main()

{

**Three Mutable Variable:**

CoffeeProject ba = new CoffeeProject("Jatinder Kumar", 80);

CoffeeProject object1 = new CoffeeProject("this is");

CoffeeProject object2 = new CoffeeProject("this is");

CoffeeProject object2 = new CoffeeProject("this is");

object1.append("Coffee");

object2.append("Bill of Coofee");

object3.append("Cup of coffee");

ba.Add\_Coffee(5.77);

ba.Cancelation\_Of\_CoffeeOrder(11.22);

Console.WriteLine("Current balance is ${0}", ba.TotalAmount);

}

}

}

**CoffeeTestCase:**

using System;

using CoffeShopNS;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CoffeShopNS

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void AddCoffee\_tothecup\_CustomerHasNotEnoughMoney\_ForCoffee()

{

double CoffeeCust\_Money = 15;

double Coffee\_Price = 7;

double Coffeechange =10;

Coffeshop coffee = new Coffeshop("Jatinder Kumar", AmountPaid);

coffee.AddCoffee(Amountdeducted);

double Coffeeactual = coffee.Balance;

Assert.AreEqual(change, actual, 0.001, "Sorry, Costumer has not enough money for Coffee ");

}

[TestMethod]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void Generate\_price\_list\_ShouldThrowArgumentOutOfRange()

{

double CoffeeCust\_Money = 15;

double Coffee\_Price =

Coffeshop account = new Coffeshop("Jatinder Kumar", CoffeeAmountPaid);

coffee.AddCoffee(CoffeeAmountdeducted)

}

[TestMethod]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void CoffePrice\_TransactionNotSuccesfull\_CoffeeShouldThrowArgumentOutOfRange()

{

double CoffeeAmountPaid = 11.99;

double CoffeeAccountIsEmpty = -100.00;

Coffeshop account = new Coffeshop("JAtinder kumar", CoffeeAmountPaid);

coffee.AddCoffee(CoffeeAccountEmpty)

}

[TestMethod]

public void CoffePrice\_CoffeeWhenAmountIsMoreThanBalance\_ShouldThrowArgumentOutOfRange()

{

double CoffeeAmountPaid = 11.99;

double CoffeeActualAmount = 20.0;

Coffeshop account = new Coffeshop("Jatinder kumar", CoffeeAmountPaid);

try

{

coffee.AddCoffee(CoffeeAccountEmpty)

}

catch (ArgumentOutOfRangeException e)

{

StringAssert.Contains(e.Message, Coffeshop.DebitAmountExceedsBalanceMessage);

}

}

}

}

**Output:**



