# LABSHEET-3

**Questions**

1. Create another static member to hold the count of ‘ClassTime’ objects currently active. Then display the value using a member function. Does the member function need to be static or non static? Why?

Use a destructor in your class and try to call it. What is the output? Why?

2. Passing Parameters

A function can return a single value at a time. This can be overcome by passing a number of parameters to the function and, let the method modify the parameters, and examine the result in the calling method.

Create a Time class and add a GetTime( ) method, which returns the hour, minutes, and

Seconds using three ref parameters. Replace them with value parameters, and see the difference.

The out modifier removes the requirement that a reference parameter be initialized. Try the GetTime() method with out parameters.

Create another member function

public void SetTime(int hr, out int min, ref int sec)

if the passed in sec is >= 60 increment the minute and set second to zero, otherwise leave both alone.

Then call GetTime and print the values.

3. Params

Create a class and add a method, ‘SetAddress()’ which takes a variable number of string arguments , each argument being the address line.

(Hint: Let the signature of the method be

public void SetAddress()(params string[] str)

)

Take each address line from the user into the string array and pass the string array as the argument to the SetAddress() method.

What do you see when you pass a null string? How many such address strings you can pass?

4. Create a class number. Add methods to perform arithmetical operations on two numbers. operations includes addition, subtraction , multiplication , division. Use separate methods for each operation. And use reference methods (out or ref) to pass values to methods.

5. Royal Orchid is a florist. They want to be alerted when stock of a flower goes below a particular level. The flowers are identified using name, price per kg and stock available (in kgs).Write a C# program to implement the above requirement.

Details of Flower class are given below:

Class name: Flower

Attributes (private) flower\_name price\_per\_kg stock\_available

Methods

(public) \_\_init\_\_() Create and initialize all instance variables to None(constructor) validate\_flower() Return true, if flower name is valid. Else, return false (Refer table for valid flower names) validate\_stock(required\_quantity) Accept the quantity required. Return true, if stock is available. Else return false. sell\_flower(required\_quantity)) Accept the quantity required. Validate flower name and stock. If both are valid, update stock available based on the quantity required check\_level() Check if available stock is below the order level If so, return true. Else, return false (Refer table for order level of each flower) setter methods Include setter methods for all instance variables to set its values getter methods Include getter methods for all instance variables to get its values

Flower

Name Level(in Kgs)

Orchid 15

Rose 25

Jasmine 40

**Answers**

**I have used a driver method to call all the questions(1-5), using switch case.**

**Program.cs-contains driver method to call other questions and also has the ClassTime Class of the first Question.**

using System;

using System.Collections.Specialized;

using System.Threading;

namespace Lab3

{

class ClassTime{

public static int count;

public ClassTime()

{

count++;

}

public void GetCount()

{

Console.WriteLine("Number of Instances :"+count);

}

~ClassTime()

{

Console.WriteLine("Destroyed");

//Destructors are special functions called upon destruction of Objects, here it isn’t called!

}

}

//Static variables can be called by non-static methods because they are shared by all //instances of the class, and can be called anywhere from the class. Moreover they are same for all objects.

class Program

{

static void Main(string[] args)

{

Console.WriteLine("1)Question 1\n2)Qeustion 2\n3)Question 3\n4)Qeustion 4\n5)Question 5");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1:

ClassTime classTime = new ClassTime();

ClassTime classTime1 = new ClassTime();

classTime.GetCount();

break;

case 2:

Time time = new Time();

int hour =10,sec = 60, min=61;

int hours = 10, seconds = 60, minutes = 61;

time.SetTime( hour, out min, ref sec);

Console.WriteLine("Three Refs Used {0}",time.getTime());

time.SetTimeAgain(ref hours, ref minutes, ref seconds);

Console.WriteLine("Using Ref and Out {0}",time.getTime());

break;

case 3:

Console.WriteLine("Number of Addressses");

int number = Convert.ToInt32(Console.ReadLine());

String[] address = new string[number];

for (int i=0;i<number;i++)

{

address[i] = Console.ReadLine();

}

Params params\_test = new Params();

params\_test.setAddress(address);

break;

case 4:

call\_question\_four();

break;

case 5:

call\_question\_five();

break;

default:

Console.WriteLine("Selected Question Number Doesn't Exist !!!");

break;

}

}

private static void call\_question\_four()

{

Number num = new Number();

int a = 12, b = 90;

num.add(ref a, ref b);

num.showResult();

num.sub(ref a, ref b);

num.showResult();

num.mul(ref a, ref b);

num.showResult();

num.div(ref a, ref b);

num.showResult();

}

private static void call\_question\_five()

{

Flower rose = new Flower(), orchid = new Flower(), jasmine = new Flower();

jasmine.\_\_init\_\_();

orchid.\_\_init\_\_();

rose.\_\_init\_\_();

jasmine.setFlowerName("Jasmine");

jasmine.setPrice(100.50);

jasmine.setStock(40);

rose.setFlowerName("Rose");

rose.setPrice(200.50);

rose.setStock(25);

orchid.setFlowerName("Orchid");

orchid.setPrice(150.50);

orchid.setStock(15);

Flower flower\_to\_buy = new Flower();

Console.WriteLine("Enter Flower name ");

String name = Console.ReadLine();

flower\_to\_buy.setFlowerName(name);

Console.WriteLine("Enter The Quantity !");

double qty = Convert.ToInt32(Console.ReadLine());

flower\_to\_buy.setStock(qty);

if (string.Equals(flower\_to\_buy.getFlowerName(), "Orchid", StringComparison.OrdinalIgnoreCase))

{

orchid.sell\_flower(flower\_to\_buy.getStock());

}

else if (string.Equals(flower\_to\_buy.getFlowerName(), "Jasmine", StringComparison.OrdinalIgnoreCase))

{

jasmine.sell\_flower(flower\_to\_buy.getStock());

}

else if (string.Equals(flower\_to\_buy.getFlowerName(), "Rose", StringComparison.OrdinalIgnoreCase))

{

rose.sell\_flower(flower\_to\_buy.getStock());

}

}

}

}

**Time.cs**

**//When using ref, we can achieve a bi-directional call and we can change the value of the variable using its reference. When passing by value , only a copy of the variable is passed.**

**When using out, we cannot achieve a bi-directional control, and value need to be initialized in the callee function.**

using System;

namespace Lab3

{

class Time

{

int hour, min, sec;

public String getTime()

{

return "The Time is :" + Convert.ToString(hour)+":"+Convert.ToString(min)+ ":"+Convert.ToString(sec);

}

public void SetTime(int hr, out int minutes, ref int second)

{

minutes = 0;

min = minutes;

hour = hr;

sec = second;

if (sec >= 60)

{

min = ++minutes;

sec %= 60;

if (min >= 60)

{

hour = ++hr;

min %= 60;

}

}

}

public void SetTimeAgain(ref int hr, ref int minutes, ref int second)

{

min = minutes;

hour = hr;

sec = second;

if (sec>=60)

{

min = ++minutes;

sec %= 60;

if (min >= 60)

{

hour = ++hr;

min %= 60;

}

}

}

}

}

**Params.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab3

{

class Params

{

public void setAddress(params string[] str)

{

Console.WriteLine("Setted Addresses:");

foreach (String address in str)

{

Console.WriteLine(address);

//Null String can be passed, and can be passed as many times depending on the size of array.

}

}

}

}

**Number.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab3

{

class Number

{

int result;

public void showResult()

{

Console.WriteLine("The Answer is {0}",result);

}

public void add(ref int first,ref int second)

{

result = first + second;

}

public void sub(ref int first, ref int second)

{

result = first - second;

}

public void mul(ref int first, ref int second)

{

result = first \* second;

}

public void div(ref int first, ref int second)

{

result = first / second;

}

}

}

**Flower.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace Lab3

{

class Flower

{

private String flower\_name;

private double price\_per\_kg,stock\_avail;

public void \_\_init\_\_()

{

flower\_name = null;

price\_per\_kg = 0.0;

stock\_avail=0.0;

}

public void setStock(double stocks)

{

stock\_avail = stocks;

}

public void setFlowerName(String flowerName)

{

flower\_name = flowerName;

}

public void setPrice(double flowerPrice)

{

price\_per\_kg = flowerPrice;

}

public double getStock()

{

return stock\_avail;

}

public String getFlowerName()

{

return flower\_name;

}

public double getPrice()

{

return price\_per\_kg;

}

public void sell\_flower(double required)

{

if (validate\_flower())

{

if (validate\_stock(required))

{

stock\_avail -= required;

double price = price\_per\_kg \* required;

Console.WriteLine("The Stock is Avaialable for Sale Net price: {0}",price);

Console.WriteLine("Current Stock {0}",stock\_avail);

}

else

{

Console.WriteLine("Sorry The Required Quantity Is Unavailable !");

}

}

else

{

Console.WriteLine("Sorry Invalid Flower Name !");

}

}

public Boolean validate\_flower()

{

if (string.Equals(flower\_name, "Orchid", StringComparison.OrdinalIgnoreCase) || string.Equals(flower\_name, "Rose", StringComparison.OrdinalIgnoreCase)|| string.Equals(flower\_name, "Jasmine", StringComparison.OrdinalIgnoreCase))

return true;

return false;

}

public Boolean validate\_stock(double stock\_entered)

{

if((stock\_entered<=stock\_avail)&& check\_level())

return true;

return false;

}

public Boolean check\_level()

{

Flower flower = new Flower();

if (this.getStock() >= flower.getStock())

return true;

return false;

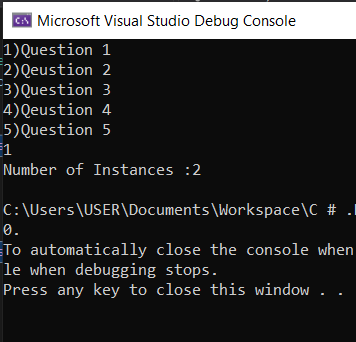
}

}

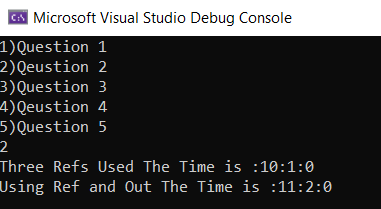
}

**Outputs:**

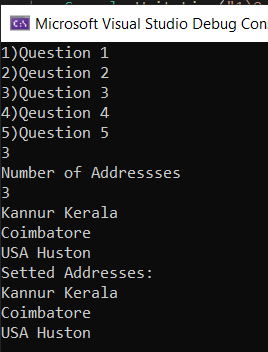
**1)**

****

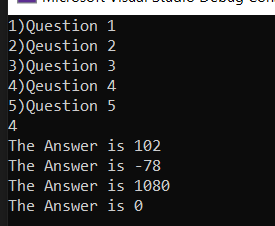
**2)**

****

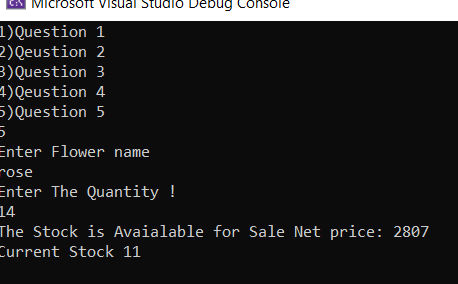
**3)**

****

**4)**

****

**5)**

****