Assignment12

Created a Project called Assignment12 and created class for all the given steps. Program.cs remains same and used region to classify the code.

1. Part 1

DynamicArray.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

public class DynamicArray

{

//Data Members

public int[,] CustomArray;

public DynamicArray()

{

CustomArray = new int[3, 3]; // Define default size as 3 \* 3

}

public void Display()

{

Console.WriteLine("Printing the Given Array in Console");

for (int i = 0; i <=CustomArray.Rank; i++)

{

Console.WriteLine("");

for (int j = 0; j <= CustomArray.Rank; j++)

{

Console.Write("\t" + CustomArray[i, j]);

}

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

class Program

{

static void Main(string[] args)

{

#region Part 1

Console.WriteLine("Console Program Assignment 12 Part 1 - Array and Printing the same in Console");

DynamicArray da = new DynamicArray();

da.CustomArray = new int[3, 3] { { 5, 8, 9 }, { 7, 8, 9 }, { 1, 2, 3 } };

da.Display();

Console.WriteLine();

#endregion

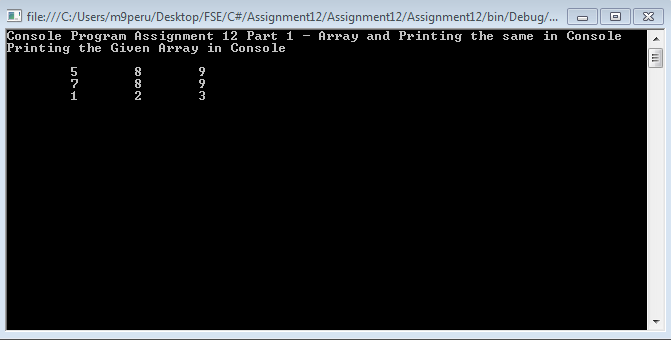
Console.ReadKey();

}

}

}

Output



1. Part 2

MultipleInheritance.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

public interface Addition

{

int Add(int Number1, int Number2);

}

public interface Subtraction

{

int Subtract(int Number1, int Number2);

}

public class PartialCalculator : Addition

{

public int Multiplication(int Number1, int Number2)

{

return Number1 \* Number2;

}

public virtual int Division(int Number1, int Number2)

{

return Number1 / Number2;

}

public virtual int Add(int Number1, int Number2)

{

return Number1 + Number2;

}

}

public class MultipleInheritance : PartialCalculator, Subtraction

{

public override int Add(int Number1, int Number2)

{

return Number1 + Number2;

}

public override int Division(int Number1, int Number2)

{

try

{

return Number1 / Number2;

}

catch

{

throw;

}

}

public int Subtract(int Number1, int Number2)

{

return Number1 - Number2;

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

class Program

{

static void Main(string[] args)

{

#region Part 2

Console.WriteLine("Console Program Assignment 12 Part 2 Multiple Inheritance Concept Calculator Program");

MultipleInheritance mi = new MultipleInheritance();

Console.WriteLine("Add 5 + 8 ="+ mi.Add(5, 8));

Console.WriteLine("Division 8 / 4 = " + mi.Division(8, 4));

Console.WriteLine("Multiplication 10 \* 2 = "+ mi.Multiplication(10, 2));

Console.WriteLine("Subtract 240 -5 =" + mi.Subtract(240,5));

Console.WriteLine("");

#endregion

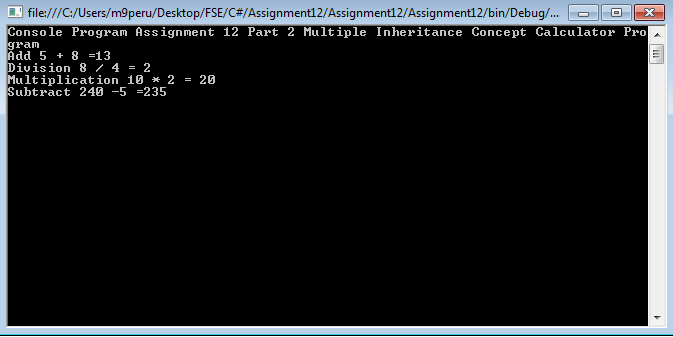
Console.ReadKey();

}

}

}

Output



1. Part 3

CollectionExcercise.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

class CollectionExcercise

{

public List<int> CollectionList;

public CollectionExcercise()

{

CollectionList = new List<int>();

}

public void AddCollectionItems(int Value)

{

CollectionList.Add(Value);

}

public void DisplayCollectionItems()

{

Console.Write("Collection List Items : ");

foreach(int i in CollectionList)

{

Console.Write(" "+i);

}

Console.WriteLine();

}

public void DisplayCollectionItemsDivisibleByThree()

{

Console.Write("COllection List Items : ");

foreach (int i in CollectionList)

{

if (i % 3 == 0)

Console.Write(" " + i);

}

Console.WriteLine();

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

delegate void ListCollectionAdd(int Number);

delegate void DisplayItemDivisibleByThree();

class Program

{

static void Main(string[] args)

{

#region Part3

Console.WriteLine("Console Program Assignment 12 Part 3 Delegates and Lamda Expression");

CollectionExcercise ce = new CollectionExcercise();

ListCollectionAdd lc = new ListCollectionAdd(ce.AddCollectionItems);

DisplayItemDivisibleByThree dli = new DisplayItemDivisibleByThree(ce.DisplayCollectionItems);

lc(5);

lc(9);

lc(12);

lc(6);

lc(24);

lc(2);

dli();

Console.WriteLine("Using Delegate Results ");

dli = new DisplayItemDivisibleByThree(ce.DisplayCollectionItemsDivisibleByThree);

List<int> list = ce.CollectionList.FindAll(x => (x % 3) == 0);

Console.Write("Using Lamda Results ");

foreach (int i in list)

{

Console.Write(" " + i);

}

Console.WriteLine();

#endregion

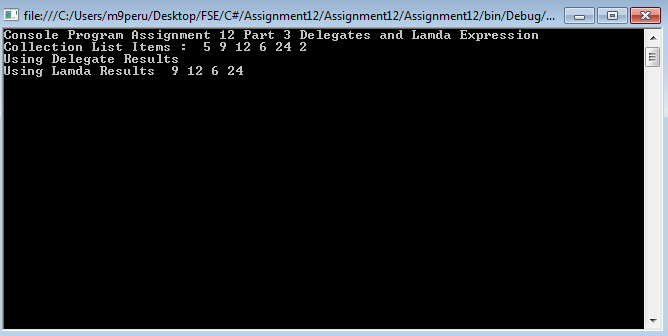
Console.ReadKey();

}

}

}

Output



1. Part 4

StringValidator.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace Assignment12

{

public static class StringValidator

{

public static bool IsEmail(this String Email)

{

const string Pattern = @"^([a-zA-Z0-9\_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

// Regular expression explained below.

/\*

\* We have 3 parts in a email.

\* ^- represents start of the character

\* first part ([a-zA-Z0-9\_\-\.]+)

\* which means we are allowing characters a-z , 0-9 numbers , special/other character such as \_-. .

\* We have should atleast those character one more more time

\* @ is the mandatory character in the string

\* in the second part ((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+)) number allowed 1 to 3 times

\* [a-zA-Z0-9\-]+ represents character from : a to z , 0 to 9 and - allowed 0 or more time and [a-z0-9])?.) part tells 0 to 9 , a to z and - allowed one or more time.

\* Similar to that number also allowed in the Email.

\* Third part represents ([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)

\* we should have only one sequence of characters a to z or A to Z , 0 to 9 one to 3 times.

\* $- end of the string

\* \*/

Regex re = new Regex(Pattern);

if (re.IsMatch(Email))

return (true);

else

return (false);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment12

{

delegate void ListCollectionAdd(int Number);

delegate void DisplayItemDivisibleByThree();

class Program

{

static void Main(string[] args)

{

#region "Part 4"

string[] emailAddresses = { "manoj.kumar@iiht.com", "m.p@server1.iiht.com",

"manoj@mk1.iiht.com", "m.@server1.iiht.com",

"p@iiht.com9", "pm#internal@iiht.com",

"m\_9@[129.126.118.1]", "j..s@iiht.com",

"ps\*@iiht.com", "js@iiht..com",

"js@iiht.com9", "j.s@server1.iiht.com",

"\"j\\\"s\\\"\"@iiht.com", "js@iiht.中国" };

foreach (var emailAddress in emailAddresses)

{

if (emailAddress.IsEmail())

Console.WriteLine("Valid: {0}", emailAddress);

else

Console.WriteLine("Invalid: {0}", emailAddress);

}

#endregion "Part 4"

Console.ReadKey();

}

}

}

Output

