Assignment 1 Week 1

**Part – A**

2. A character string is said to have period k if it can be formed by concatenating one or more repetitions of another string of length k. For example, the string "abcabcabcabc" has period 3, since it is formed by 4 repetitions of the string "abc". It also has periods 6 (two repetitions of "abcabc") and 12 (one repetition of "abcabcabcabc").

Write a program to read a character string and determine its smallest period.

**Input**

A single character string of up to 80 non-blank characters.

**Output**

An integer denoting the smallest period of the input string.

**Sample Input**

HoHoHo

**Sample Output**

2

Program:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1Assignment1in13

{

class Program

{

static void Main(string[] args)

{

bool continued = false;

do

{

start:

char[] charArray = new char[100];

string[] stringArray = new string[100];

Console.Write("Enter the string to find the period of repetition:\t");

string str = Console.ReadLine();

stringArray = str.Split(' ');

if (stringArray.Length == 1) //checks if there is no spaces

charArray = str.ToCharArray();

else

{

Console.WriteLine("Input has blank spaces in it please provide valid input");

goto start; //when user input is wrong

}

if (charArray.Length < 81)

{

int n = charArray.Length, ans = n;

//loop from start of the array till element in array

for (int iteration = 0; iteration < n; iteration++)

{

bool f = true;

//number has to be divisible by size taken

if (n % (iteration + 1) == 0)

{

//nested for loop with element taken to compare

for (int j = iteration + 1; j < n; j++)

{

//remainder of initial element

if (charArray[j] == charArray[j % (iteration + 1)])

{

}

else

{

f = false;

break;

}

}

//when all the above condition fails and f reatains true

if (f)

{

//the output is the position where the for loop terminates

ans = iteration + 1;

break;

}

}

else

continue;

}

Console.WriteLine("The period of repetition for the string --{0}-- is --{1}--", str, ans);

Console.ReadKey();

}

else

{

Console.WriteLine("Enter a valid input less than or equal to 80 characters");

goto start;

}

Console.Write("Press 1 to continue and anyother key to exit:");

continued=(Console.ReadLine()=="1"?true:false);

} while (continued);

}

}

}

END

Yes

No

No

Yes

Answer=pos

F==true

yes

i++

J++

S[j]==s[j%pos]

J=pos

j<size(array)

NO

Size%pos==0

Get the user input as string and convert it to char array

Int i=0

For

I<size(array)

START

ALGORITHM:

1. Get the input and validate on the space and whether it is below 80 characters long.
2. Use for loop to traverse from start of the string after converting to a character array.
3. Check if the string count is divisible by the position to continue with next step else go to step 2 after incrementing the position
4. Another for loop which checks the element in next position to the remainder of the size to position.
5. If the elements doesn’t match the loop breaks and the position gives the smallest period of repetition.

Input:

deedee

abcabcabc

HoHoHo

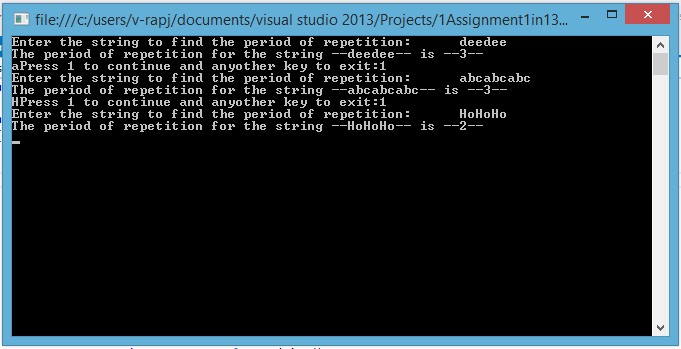
Output:

3

3

2

Output:



Part- B

1. Without changing the actual function, call it in such a way that it prints "True"

Program:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1Assignment1in13

{

class Class1

{

void Puzzle(out int x, out int y)

{

x = 0;

y = 1;

Console.WriteLine(x == y);

}

static void Main(string[] args)

{

Class1 objBool = new Class1();

int a = 0;

objBool.Puzzle(out a,out a); //give same out parameter you will get true

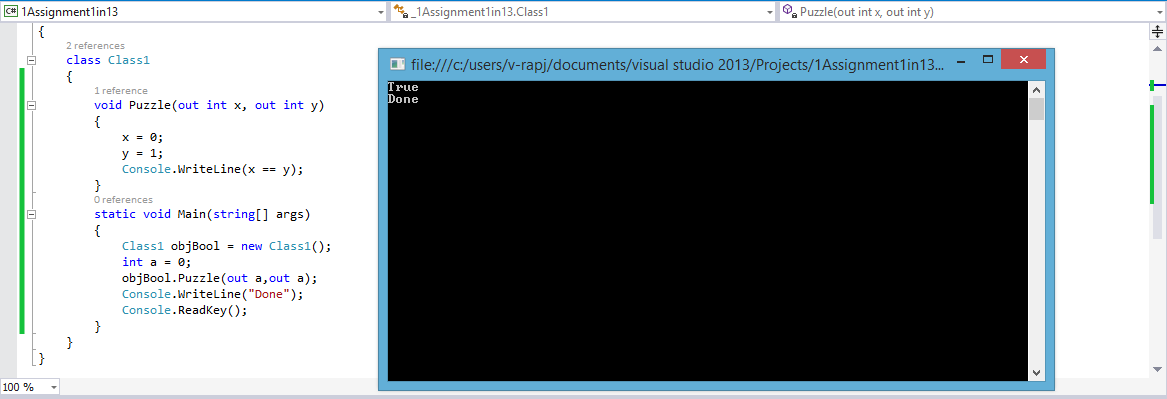
Console.WriteLine("Done");

Console.ReadKey();

}

}

}



1. In this programming puzzle, you are to produce Exceptions that can be caught but are thrown again at the end of the catch block.

static void Main(string[] args)

{

try

{

while (true)

try

{

int b = 0, c = 1 / b; //add int which divide by zero

//add try block which redirects the below catch block

try

{

}

catch (Exception ex2)

{

}

}

catch (Exception ex1)

{

Console.WriteLine("You won!"); // for C#

Console.ReadLine();

}

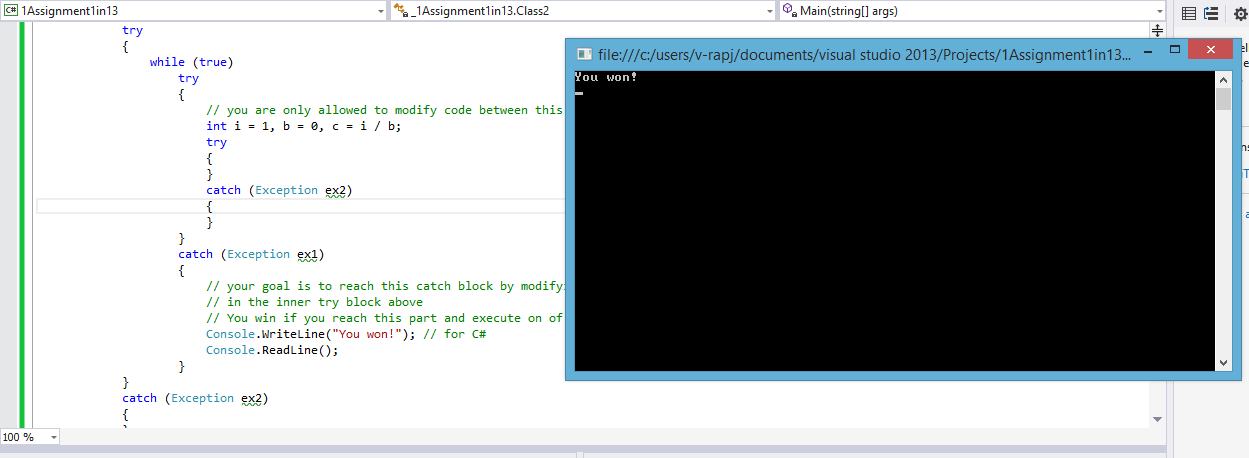
}

catch (Exception ex2) //add catch block after it

{

}

}



1. Given the following C# program outputting False, inject a 'malicious' line of code such that the program outputs True.

class Program

{

static void Main()

{ System.Console.Write("False");

;

}

}

Your answer should consist of a string that replaces the second semicolon, and causes the program to output True, the whole True and nothing but True (not even a newline). The shortest answer wins.

class Program

{

static void Main()

{

System.Console.Write("False");

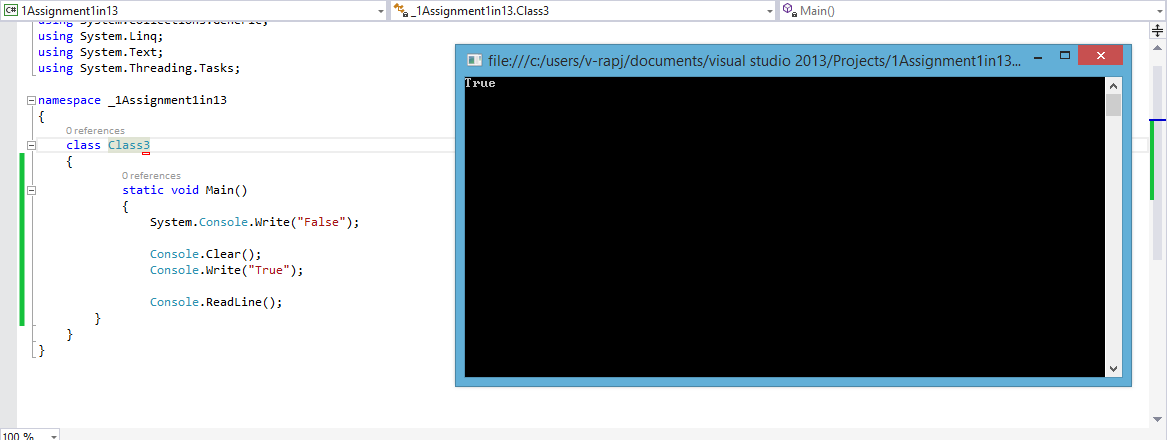
Console.Clear(); //clear the console and print required text

Console.Write("True");

Console.ReadLine();

}

}



1. Give an example of a situation where the C# method below returns false:

public class Giraffe : Animal

{ public bool Test()

{

return this is Giraffe;

}

}

Rules: The code lines above must not be changed in any way, but you would put the code into an application so that the "project" compiles and the method is called. When run, the method must return false

class Program

{

public class Animal //add animal class and inside it declare giraffe

{

public class Giraffe

{

}

}

public class Giraffe : Animal

{

public bool Test()

{

return this is Giraffe;

}

}

static void Main(string[] args)

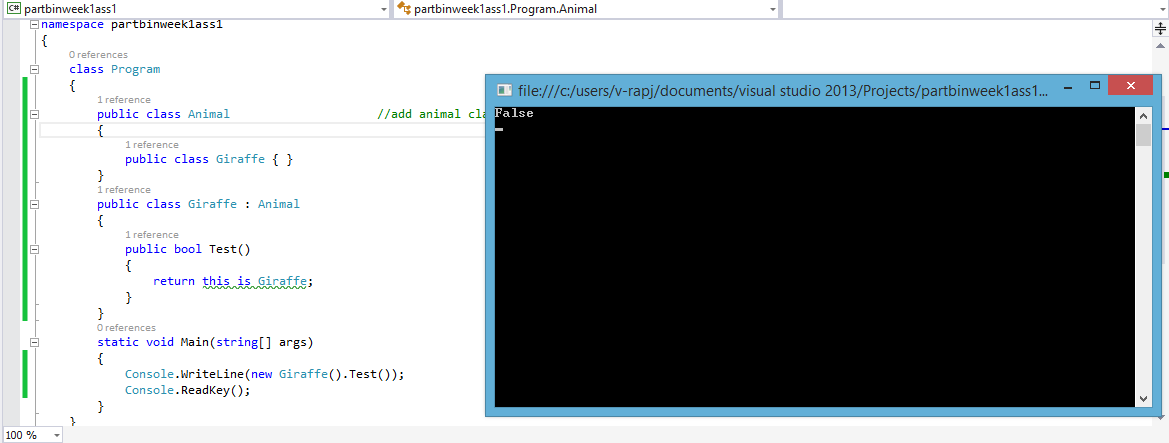
{

Console.WriteLine(new Giraffe().Test());

Console.ReadKey();

}

}



1. Write C# Program to Convert a 2D Array into 1D Array.

Input :

Given 2-D Array(Matrix) is :

1 4 3

7 3 8

Output :

Converted 1-D Array is : 1 4 3 7 3 8

static void Main(string[] args)

{

int[,] twoDim = new int[,]{{1,4,3},{7,3,8}};

int[] oneDim = new int[6];

int pos = 0;

//use foreach loop and add it inside the 1D array from start

foreach(int element in twoDim)

{

oneDim[pos++] = element;

}

for (int iteration = 0; iteration < oneDim.Length; iteration++)

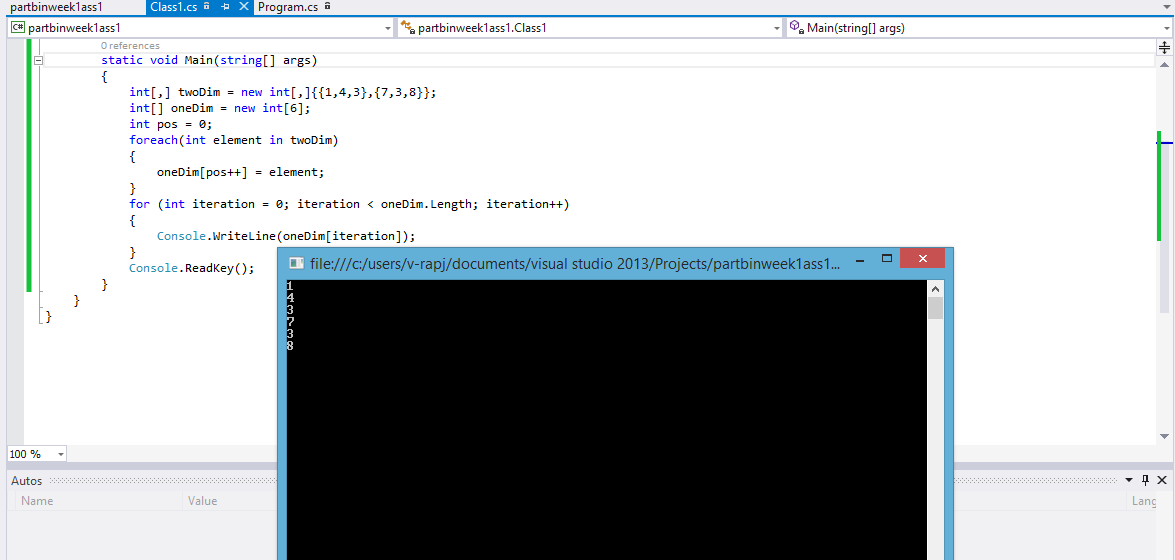
{

Console.WriteLine(oneDim[iteration]);

}

Console.ReadKey();

}



1. Write C# Program Displays the Abbreviation of a Text.

Input: Wipro Technologies Ltd

Output: W.T.L

static void Main(string[] args)

{

string abbreviation = "";

string fullForm = "Wipro Technologies Ltd";

char[] firstLetter = fullForm.ToCharArray();//Convert to char array

abbreviation += firstLetter[0]; //add first Letter

//iterate for full length

for (int position = 0; position < firstLetter.Length; position++)

{

//if space is seen add next element into a array preceeded by “.”

if(firstLetter[position]==' ')

{

abbreviation += "." + firstLetter[position + 1];

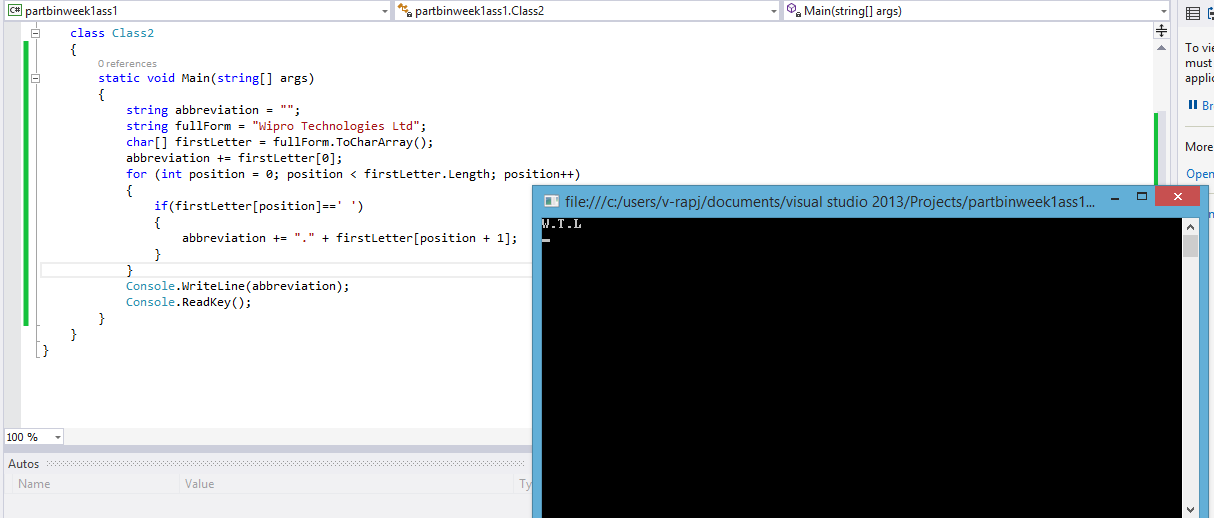
}

}

Console.WriteLine(abbreviation);

Console.ReadKey();

}



1. Write C# Program Reads a String and find the Sum of all Digits in the String.

Input:

Enter the Length of the sentence : 6

Enter the string1 containing both digits and alphabet : SAN193

Output

NO. of Digits in the string1 = 3

Sum of all digits = 13

static void Main(string[] args)

{

int count = 0;

int sum = 0;

Console.WriteLine("Enter the Length of the sentence : ");

int size = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter the string1 containing both digits and alphabet :"); //get input and change to char array

char[] comboChar = Console.ReadLine().ToCharArray();

foreach(char element in comboChar) //traverse

{

if(element>='0'&&element<='9') //if ASCII value is between 0-9

{

count += 1; //increment count and add it to sum

sum += Convert.ToInt32(element.ToString());

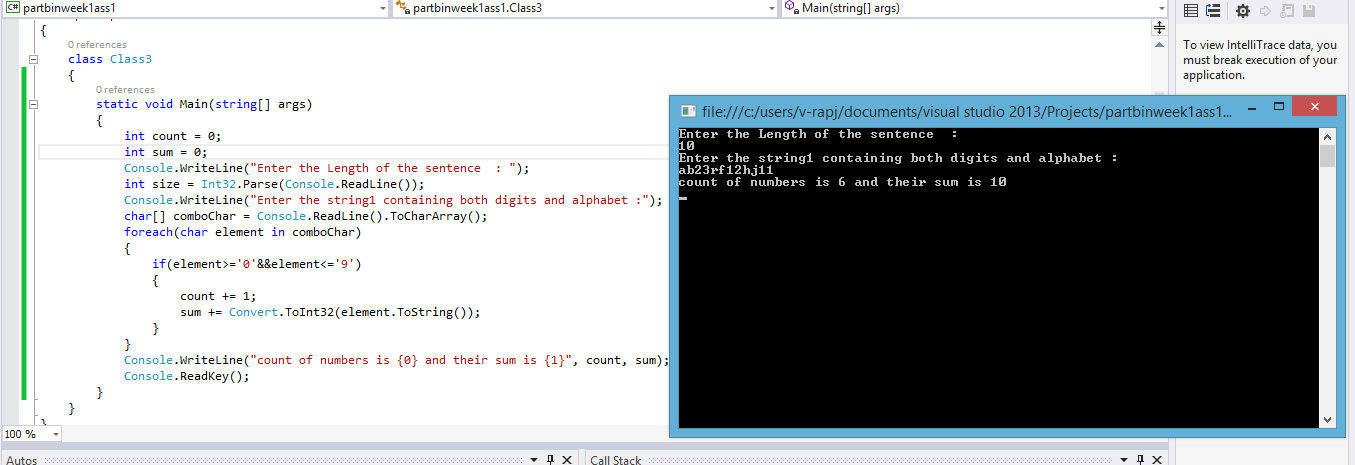
}

}

Console.WriteLine("count of numbers is {0} and their sum is {1}", count, sum);

Console.ReadKey();

}



2. Write C# Program Implements Use of Indexers. Indexers allow instances of a class or struct to be indexed just like arrays.

class Employee

{

public int id;

public string name;

public int sal;

public object this[int thing]

{

get

{

if (thing == 0) return this.id;

if (thing == 1) return this.name;

if (thing == 2) return this.sal;

else return null;

}

set

{

if (thing == 0) this.id = (int)value;

if (thing == 1) this.name = (string)value;

if (thing == 2) this.sal = (int)value;

}

}

}

static void Main()

{

Employee objEmp = new Employee();

objEmp[0] = 1;

objEmp[1] = "Raj";

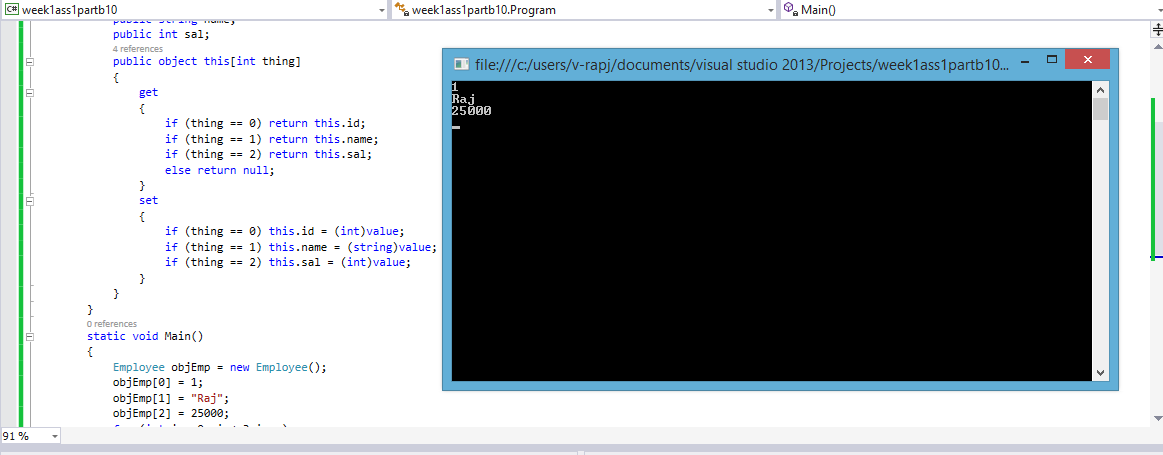
objEmp[2] = 25000;

for (int i = 0; i < 3;i++ )

Console.WriteLine(objEmp[i]);

Console.ReadKey();

}



1. Write C# Program to Combine Two Delegates. Here when the multicast delegate is called, it invokes the delegates in the list, in order.

class Class1

{

public void method1(string a)

{

Console.WriteLine(a+"try of 1");

}

public void method2(string b)

{

Console.WriteLine(b+"try of 2");

}

delegate void Del(string str);

static void Main()

{

Class1 objClass = new Class1();

Del delObj1 = new Del(objClass.method1);

Del delObj2 = new Del(objClass.method2);

delObj1("this is method 1");

delObj2("this is method 2");

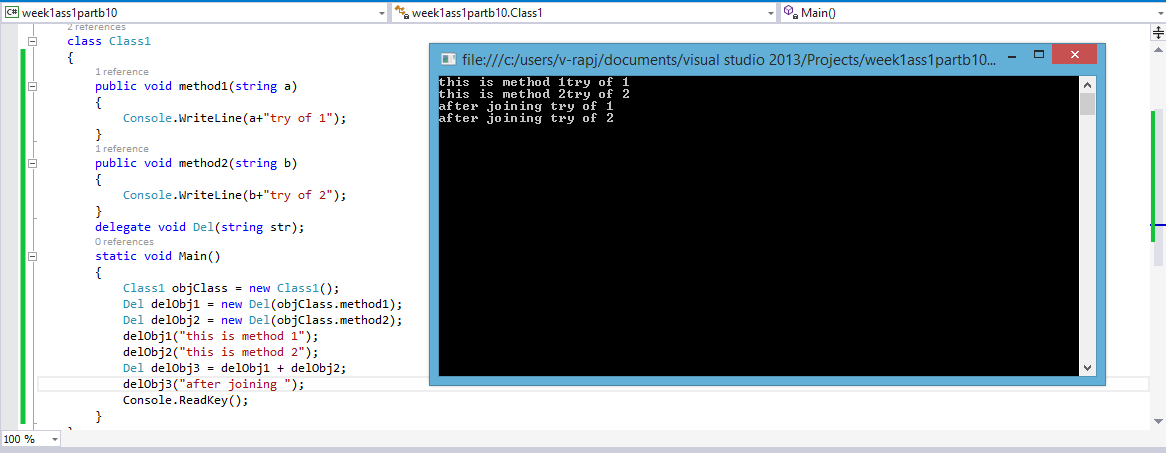
Del delObj3 = delObj1 + delObj2;

delObj3("after joining ");

Console.ReadKey();

}

}



1. Write C# Program to demonstrate Generic Delegate.

class Class2

{

public int a, b;

public void method1(string a)

{

Console.WriteLine("try of 1 of string:{0}",a);

}

public void method2(int b)

{

Console.WriteLine("try of 2 of integer:{0}",b);

}

delegate void Del <T>(T x);

static void Main()

{

Class2 objClass = new Class2();

Del<string> delObj1 = new Del<string>(objClass.method1);

Del<int> delObj2 = new Del<int>(objClass.method2);

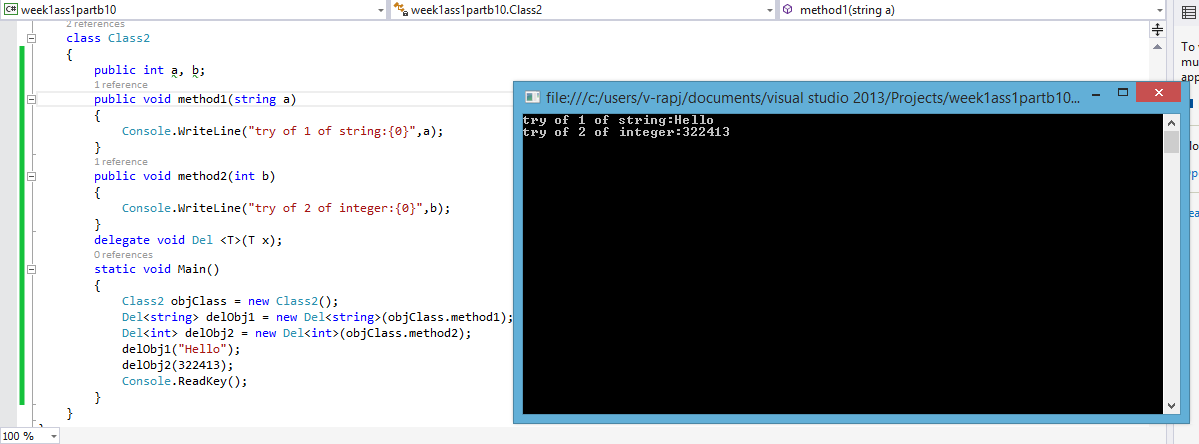
delObj1("Hello");

delObj2(322413);

Console.ReadKey();

}

}



1. Write C# Program Converts Feet to Inches using Delegates.

class Class3

{

public float method2(float b)

{

Console.WriteLine(b);

float ret = b \* 12;

return ret;

}

delegate float Del<T>(T x);

static void Main()

{

Class3 objClass=new Class3();

Del<float> delObj = new Del<float>(objClass.method2);

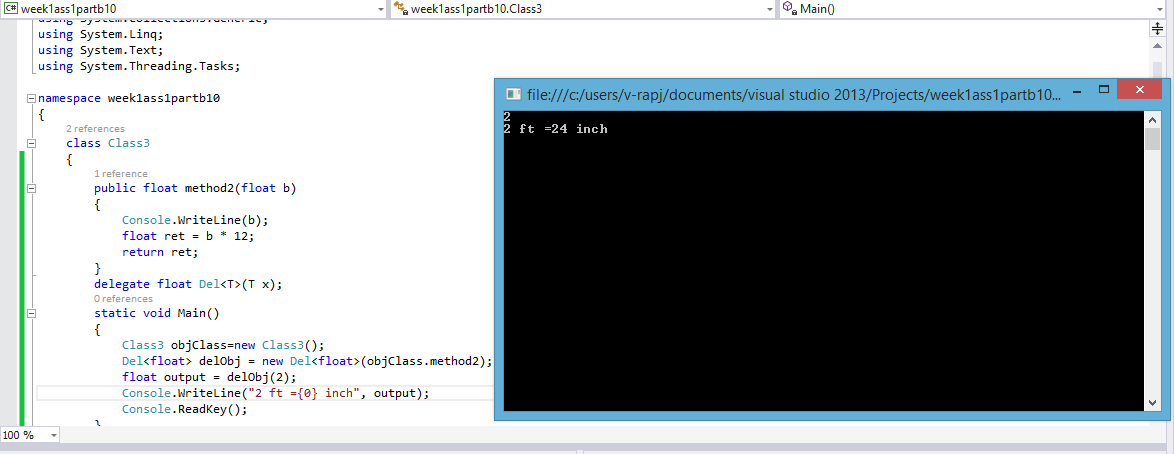
float output = delObj(2);

Console.WriteLine("2 ft ={0} inch", output);

Console.ReadKey();

}

}



1. Write C# Program Finds Largest Element in a Matrix.

static void Main()

{

int temp = 0;

int[, ] arr = new int[3, 3] { {1,2,3 }, {9,8,70 }, { 6,5,2} };

foreach(int element in arr)

{

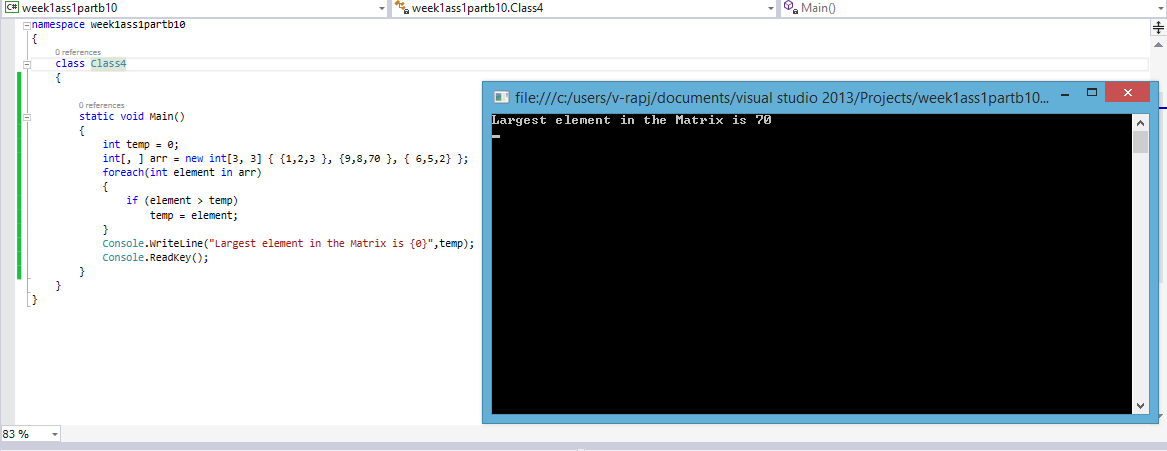
if (element > temp)

temp = element;

}

Console.WriteLine("Largest element in the Matrix is {0}",temp);

Console.ReadKey();

}

1. Write C# Program Implements Stack with Push and Pop operations: use interfaces, classes.

class Class5

{

public static void Main()

{

myimp st = new myimp();

st.pushing(30);

object s=st.popping();

Console.WriteLine(s+"is popped");

Console.WriteLine(st.popping());

Console.ReadKey();

}

}

interface stacking

{

void pushing(object element);

object popping();

}

class myimp : stacking

{

public int StackSize = 5;

public int top;

public Object[] item=new object[5];

public void pushing(object element)

{

if (top == (StackSize - 1))

{

Console.WriteLine("Stack is full!");

}

else

{

item[++top] = element;

Console.WriteLine("Item pushed successfully!");

}

}

public object popping()

{

bool ret=(top == -1 ? true : false);

if (ret)

{

Console.WriteLine("Stack is empty!");

return "No elements";

}

else

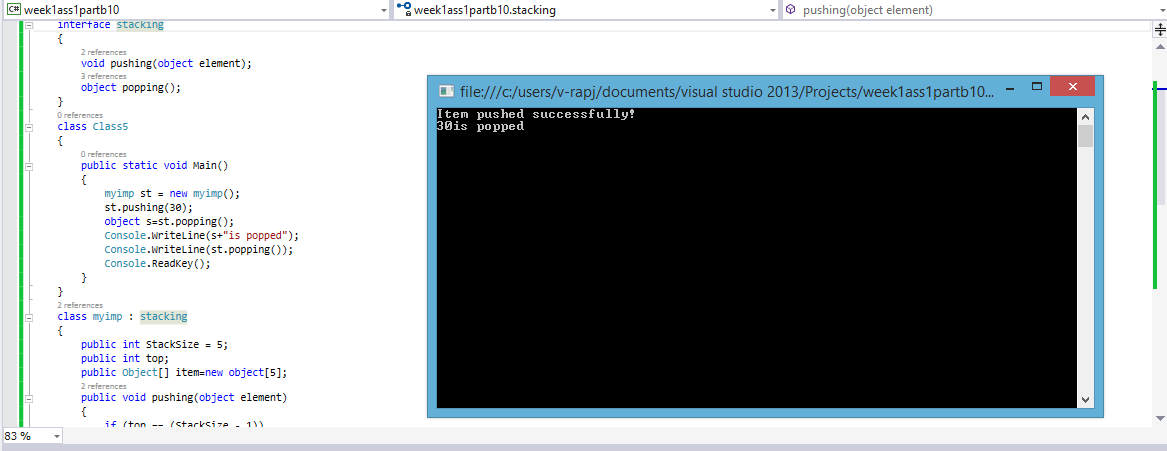
{

return item[top--];

}

}

}



1. Write a C# Program to find the Frequency of the Word ʺisʺ in a given Sentence.

class Class6

{

string strInput = "";

int count = 0;

void get()

{

Console.WriteLine("Enter the string to see number of \"is \" repeated");

strInput= Console.ReadLine();

}

void findIs()

{

for(int iteration=0;iteration<strInput.Length;iteration++)

{

if (iteration + 3 < strInput.Length)

{

if (strInput.Substring(iteration + 1, 1) == "i" && strInput.Substring(iteration + 2, 1) == "s")

{

count += 1;

}

}

}

Console.Write("{0} is the number of times \"is\" repeated", count);

}

public static void Main()

{

Class6 objClss=new Class6();

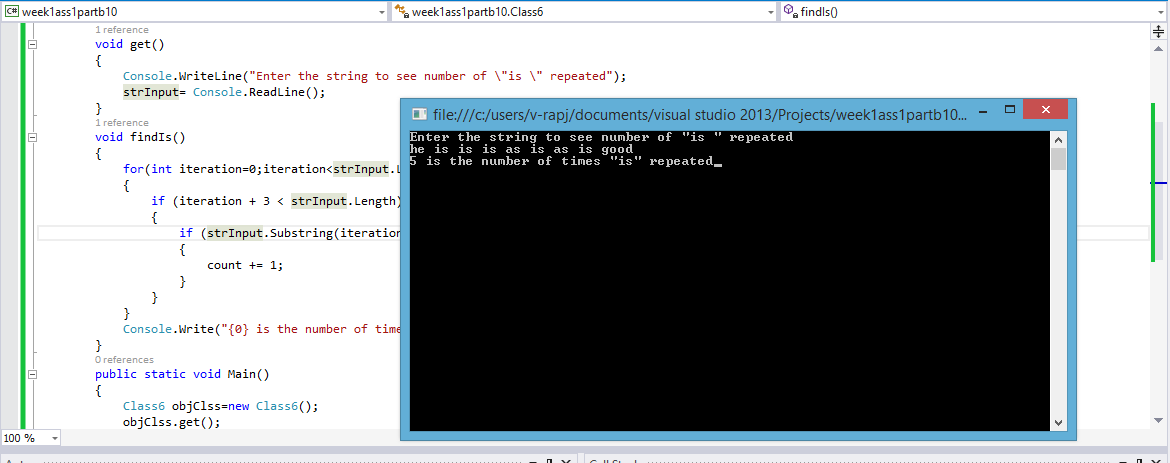
objClss.get();

objClss.findIs();

Console.ReadKey();

}

}



1. Write a C# Program to generate Strings randomly

class Program

{

public int count;

static void Main(string[] args)

{

string mystring = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

Program objProg = new Program();

objProg.get();

char[] mychar=mystring.ToCharArray();

char[] array = new char[objProg.count];

Random rando=new Random();

for (int iteration = 0; iteration < objProg.count; iteration++)

{

array[iteration] = mychar[rando.Next(mychar.Length-1)];

}

string outp = "";

outp=new string(array);

Console.WriteLine(outp);

Console.ReadKey();

}

void get()

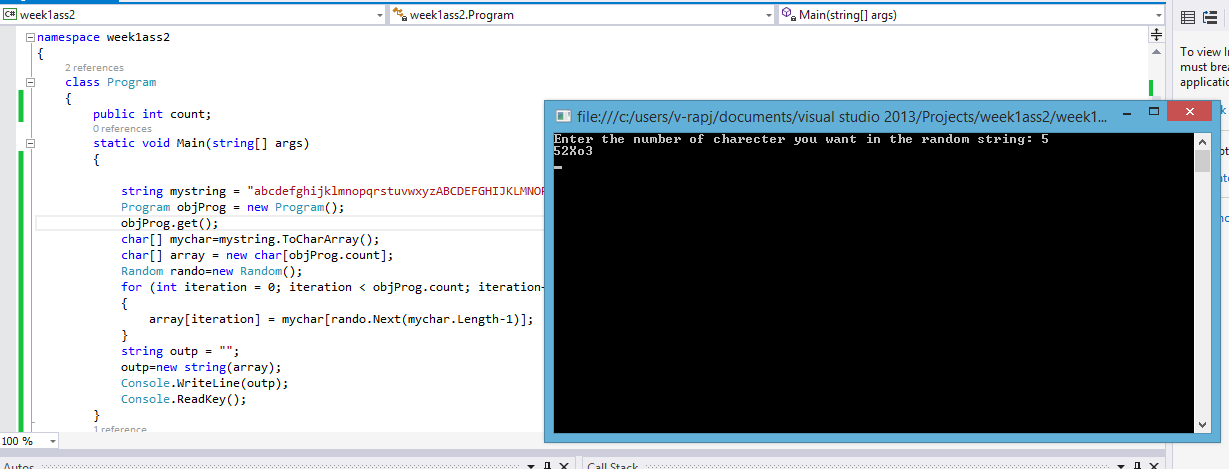
{

Console.Write("Enter the number of charecter you want in the random string: ");

count = Convert.ToInt32(Console.ReadLine());

}

}



1. Write a C# Program to lists all Substrings in a given String

class Class1

{

string intake="";

int innerInt, OuterInt;

void process()

{

Console.WriteLine("Enter the string to get all possible substrings: ");

intake=Console.ReadLine();

string[] myarr = new string[intake.Length];

for(OuterInt=1;OuterInt<=intake.Length;OuterInt++)

{

for(innerInt=0;innerInt<=intake.Length-OuterInt;innerInt++)

{

myarr[innerInt] = intake.Substring(innerInt, OuterInt);

Console.WriteLine(myarr[innerInt]);

myarr[innerInt] = "";

}

}

}

static void Main()

{

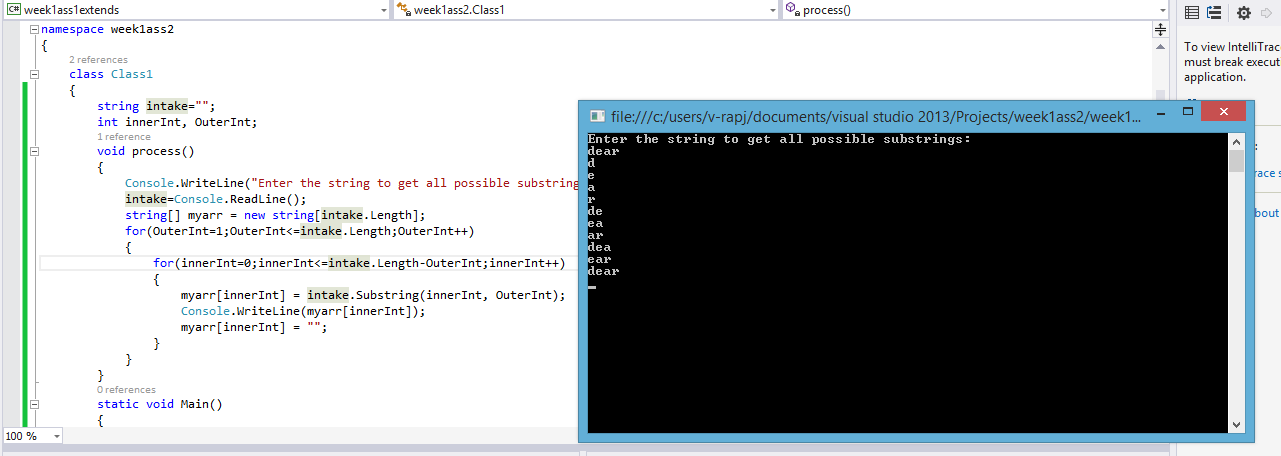
Class1 newobj = new Class1();

newobj.process();

Console.ReadKey();

}

}



1. Write a C# Program to demonstrate Properties of the Interface

interface Intof

{

int Id { get; set; }

string Name { get; set; }

}

class Propof : Intof

{

public int Id

{

get;

set;

}

string \_name;

public string Name

{

get { return this.\_name; }

set { this.\_name = value; }

}

}

class Article : Intof

{

public int Id

{

get;set;

}

string \_name;

public string Name

{

get { return this.\_name; }

set { this.\_name = value.ToUpper(); }

}

}

class Class2

{

static void Main()

{

Intof use1 = new Propof();

Intof use2 = new Article();

use1.Id++;

use2.Id++;

use1.Name = "Raj";

use2.Name = "Eshwar";

Console.WriteLine(use1.Name);

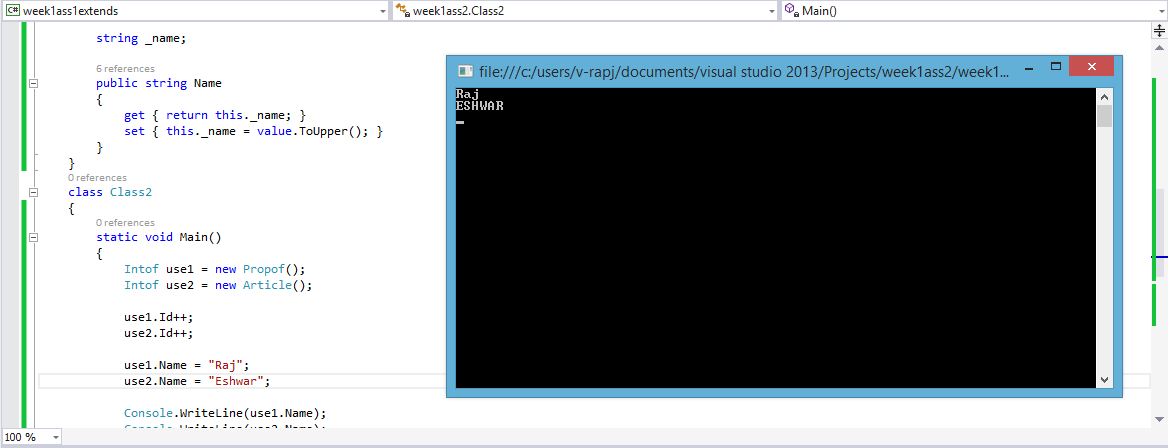
Console.WriteLine(use2.Name);

Console.ReadLine();

}

}

}



1. Write a C# Program to demonstrate IDumpable Interface

interface IDumpable

{

int ID { get; set; }

void dumpout();

}

class Myclass : IDumpable

{

int id;

string name;

public Myclass(int num,string val)

{

this.id=num;

this.name=val;

}

public int ID

{ get { return id; } set { this.id = value; } }

public void dumpout()

{

Console.WriteLine("he is {0}",ID);

}

}

class Class3

{

static void Main()

{

IDumpable a = new Myclass(1, "Raj");

a.ID = 322413;

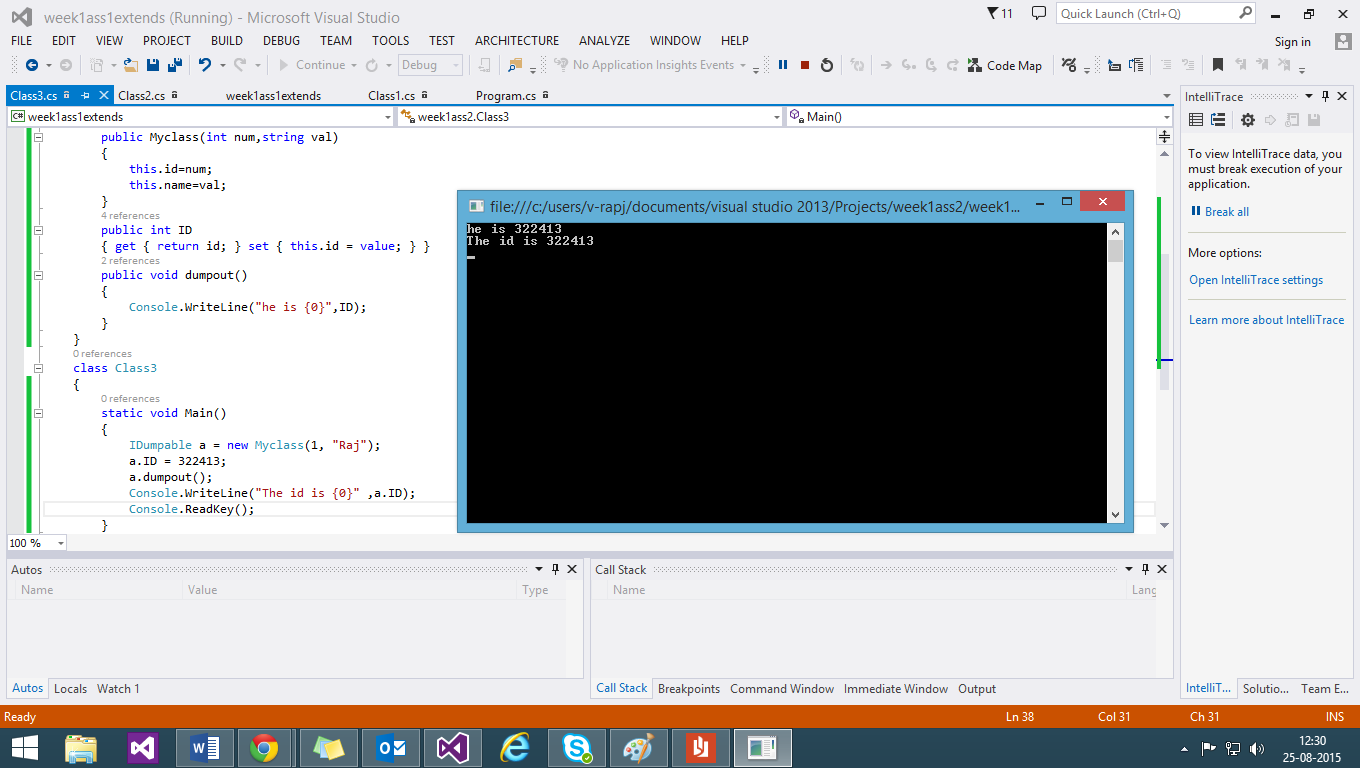
a.dumpout();

Console.WriteLine("The id is {0}" ,a.ID);

Console.ReadKey();

}

}



1. Write a C# Program to demonstrate IDisposable Interface

class mywork:IDisposable

{

int id;

string name;

public mywork(int num,string val)

{

this.id=num;

this.name=val;

}

public int ID

{ get { return id; } set { this.id = value; } }

public void dumpout()

{

Console.WriteLine("he is {0}, {1}, {2}",ID,id,name);

}

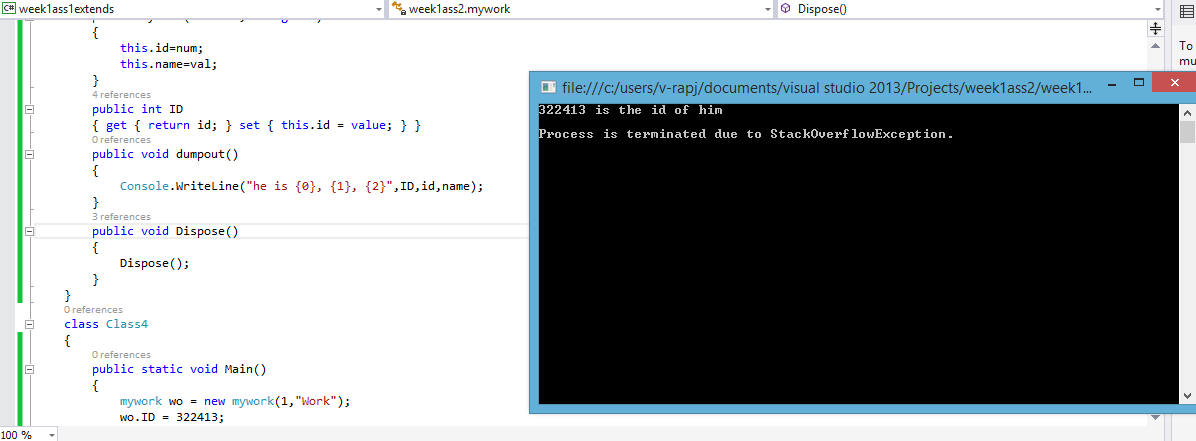
public void Dispose()

{

Dispose();

}

}



15. Write a C# Program Creates Anonymous Method

delegate int getsum(int a,int b);

class Class5

{

static void Main()

{

getsum objsum = delegate(int i, int j){return (i + j); };

Console.WriteLine("{0} is the sum",objsum(Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine())));

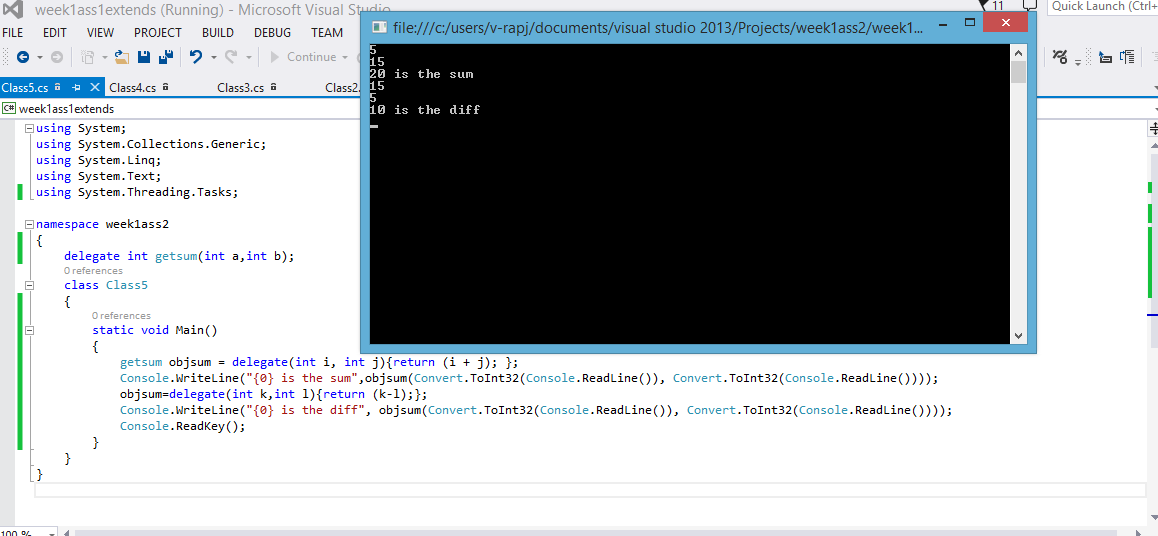
objsum=delegate(int k,int l){return (k-l);};

Console.WriteLine("{0} is the diff", objsum(Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine())));

Console.ReadKey();

}

}



16. Write a C# Program to demonstrate Method Hiding

public class hidedclass

{

public void hidedmethod()

{

Console.WriteLine("This is fetched from hided method of hidedclass");

}

}

class shownclass : hidedclass

{

}

class Class6

{

public static void Main()

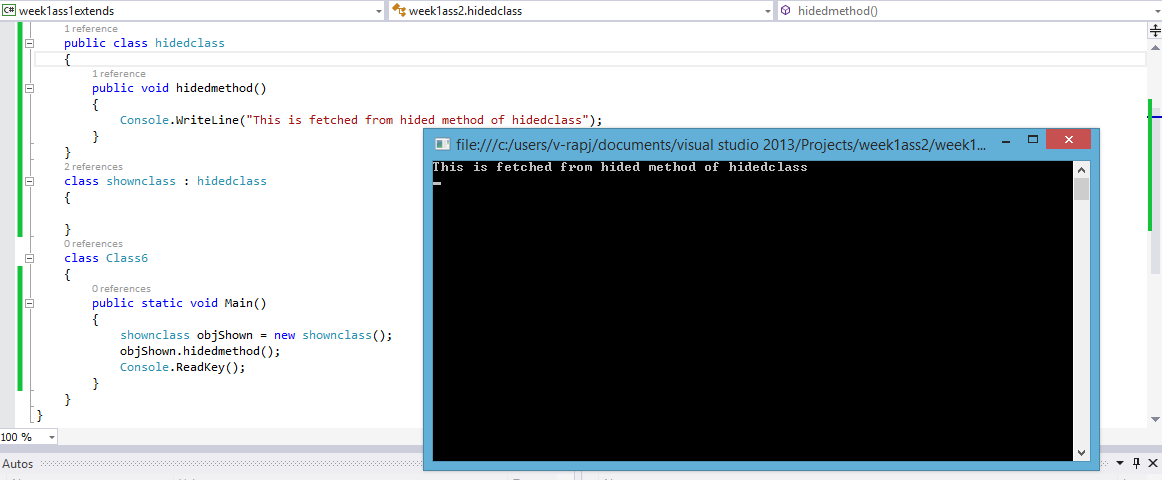
{

shownclass objShown = new shownclass();

objShown.hidedmethod();

Console.ReadKey();

}

}

17. Write a C# Program Creates Obsolete Class.

class Class7

{

static void Main()

{

old("My try");

Console.WriteLine("---------------------");

newest("My new try");

Console.ReadKey();

}

[Obsolete("this method is obsolete try using newest() method")]

static void old(string s)

{

Console.WriteLine(s);

}

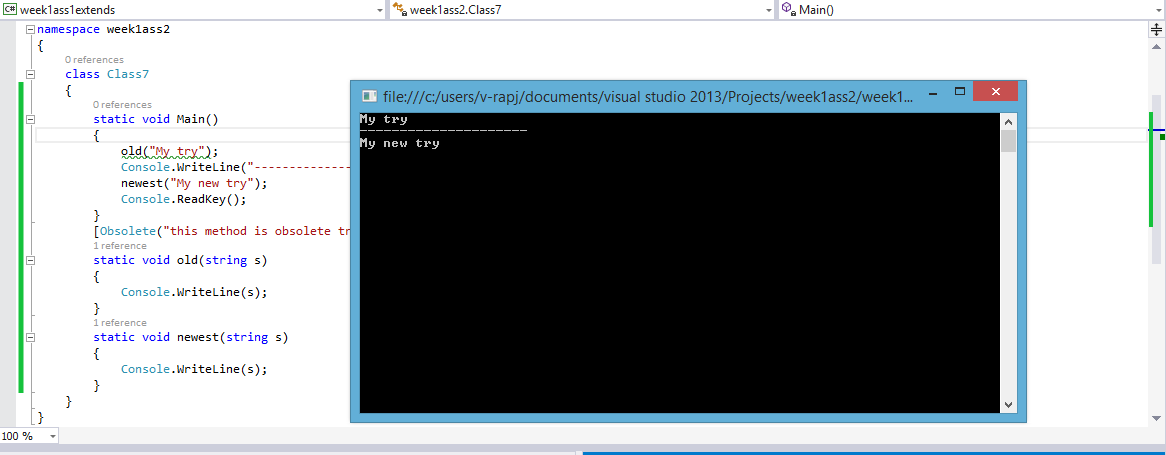
static void newest(string s)

{

Console.WriteLine(s);

}

}



18. Write C# Program finds Product of 2 Numbers using Recursion.

static void Main()

{

Console.Write("Enter the two numbers for which you want to find the product of:");

Class8 objProd = new Class8();

Console.Write("{0} is the product",objProd.prod(Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine())));

Console.ReadKey();

}

int prod(int first,int second)

{

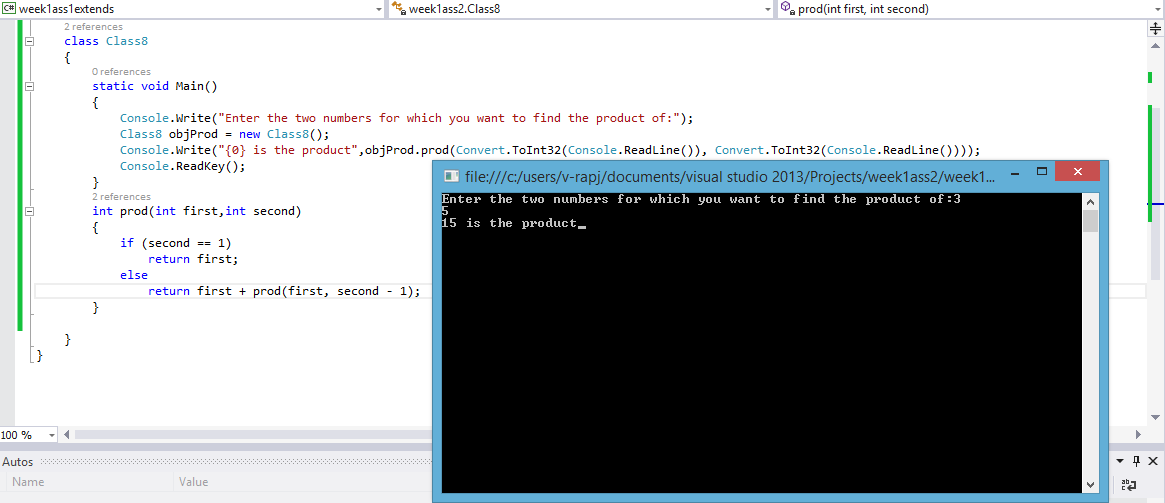
if (second == 1)

return first;

else

return first + prod(first, second - 1);

}



19. Write a C# Program to demonstrate Boxing Operations.

class Class9

{

public static void Main()

{

int sum = 0;

List<object> Mixed = new List<object>();

Mixed.Add("First set of integers");

Mixed.Add(1);

Mixed.Add(2);

Mixed.Add(3);

Mixed.Add(4);

for (int iteraton = 1; iteraton < 5;iteraton++ )

{

sum += (int)Mixed[iteraton];

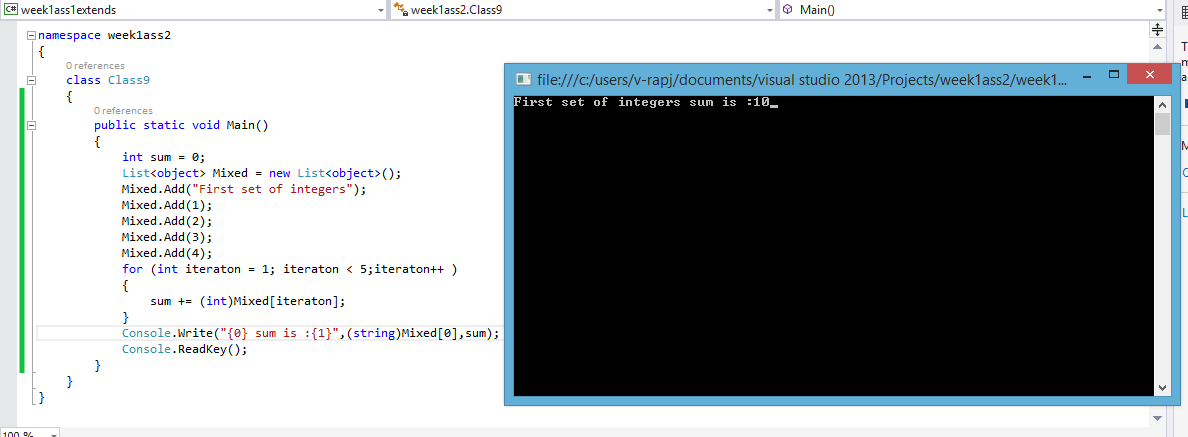
}

Console.Write("{0} sum is :{1}",(string)Mixed[0],sum);

Console.ReadKey();

}

}



20. Write a C# Program to demonstrate Multilevel Inheritance

class Class10

{

public static void Main()

{

ChildClass objchild = new ChildClass();

objchild.display();

objchild.displayParent();

objchild.displayGrand();

//now it is time for parent

Console.WriteLine("---------------------------");

ParentClass objPar = new ParentClass();

objPar.displayParent();

objPar.displayGrand();

//now for grand parent

Console.WriteLine("---------------------------");

GrandParentClass objGrand = new GrandParentClass();

objGrand.displayGrand();

Console.ReadKey();

}

}

class GrandParentClass

{

public void displayGrand()

{

Console.WriteLine("Grand Parent Class");

}

}

class ParentClass : GrandParentClass

{

public void displayParent()

{

Console.WriteLine("Parent Class");

}

}

class ChildClass : ParentClass

{

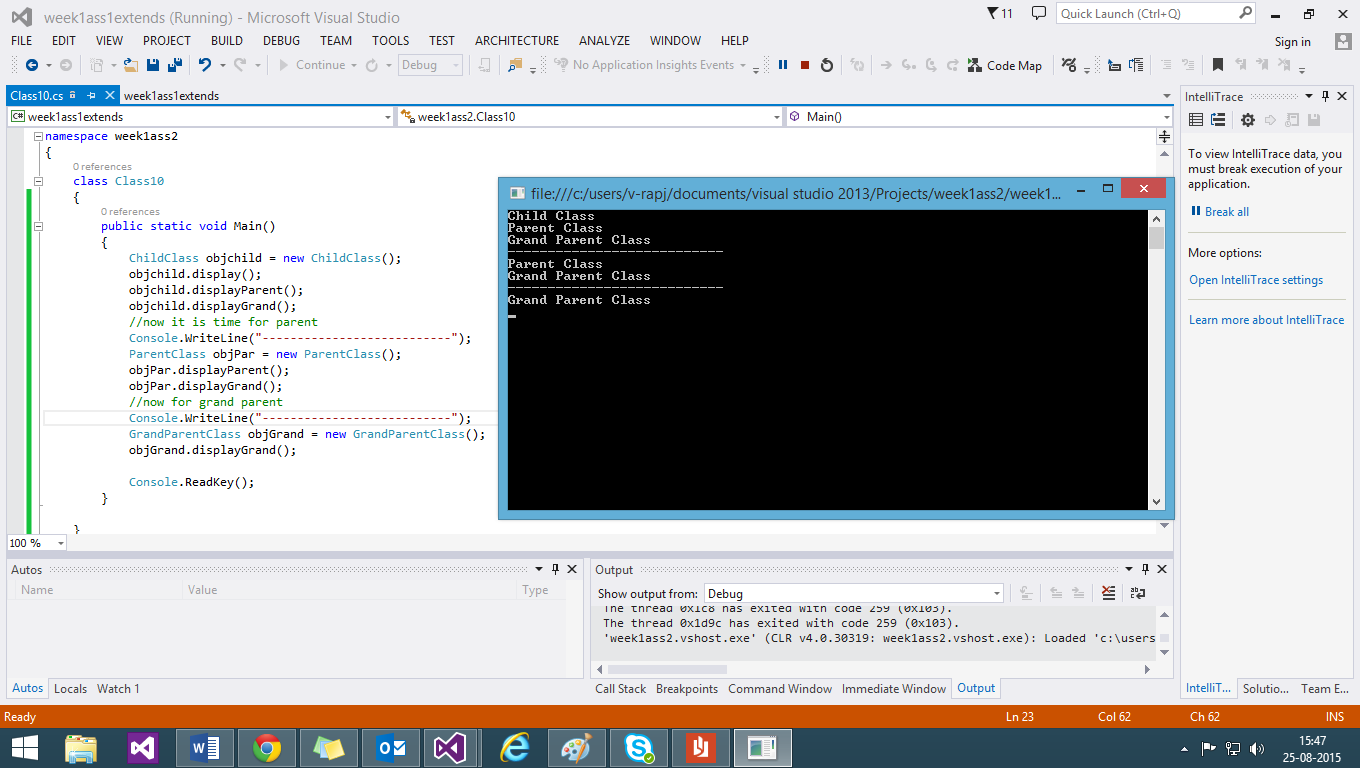
public void display()

{

Console.WriteLine("Child Class");

}

}



21. Write a C# Program to demonstrate Hierarchical Inheritance.

class Class11

{

public static void Main()

{

ChildClass1 objchild1 = new ChildClass1();

objchild1.displaychild1();

objchild1.displayParent();

Console.WriteLine("---------------------------");

ChildClass2 objchild2 = new ChildClass2();

objchild2.displaychild2();

objchild2.displayParent();

//now it is time for parent

Console.WriteLine("---------------------------");

ParentClasses objPar = new ParentClasses();

objPar.displayParent();

Console.ReadKey();

}

}

class ParentClasses

{

public void displayParent()

{

Console.WriteLine("Parent Class ");

}

}

class ChildClass2 : ParentClass

{

public void displaychild2()

{

Console.WriteLine("Child Class 2");

}

}

class ChildClass1 : ParentClass

{

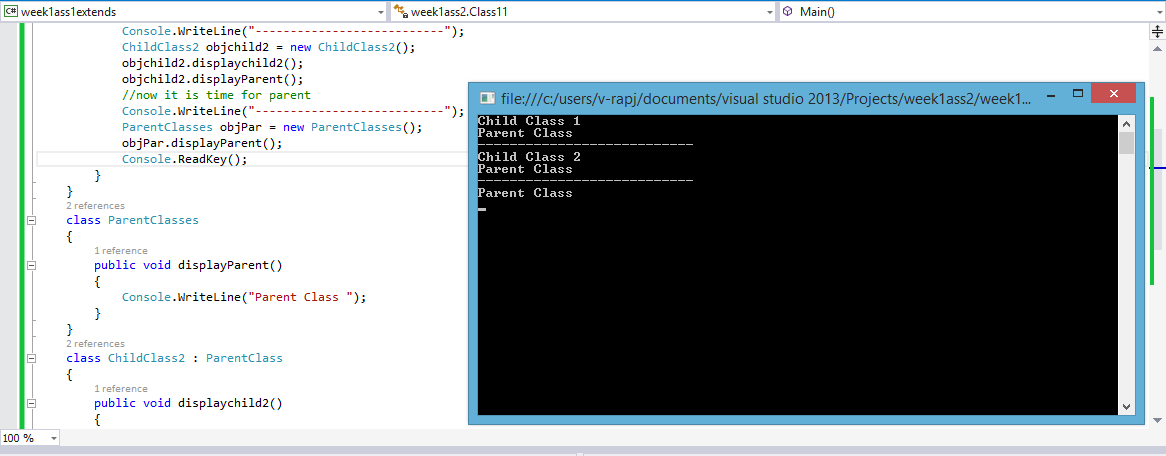
public void displaychild1()

{

Console.WriteLine("Child Class 1");

}

}



22. Write a C# Program to demonstrate Multilevel Inheritance

class Class10

{

public static void Main()

{

ChildClass objchild = new ChildClass();

objchild.display();

objchild.displayParent();

objchild.displayGrand();

//now it is time for parent

Console.WriteLine("---------------------------");

ParentClass objPar = new ParentClass();

objPar.displayParent();

objPar.displayGrand();

//now for grand parent

Console.WriteLine("---------------------------");

GrandParentClass objGrand = new GrandParentClass();

objGrand.displayGrand();

Console.ReadKey();

}

}

class GrandParentClass

{

public void displayGrand()

{

Console.WriteLine("Grand Parent Class");

}

}

class ParentClass : GrandParentClass

{

public void displayParent()

{

Console.WriteLine("Parent Class");

}

}

class ChildClass : ParentClass

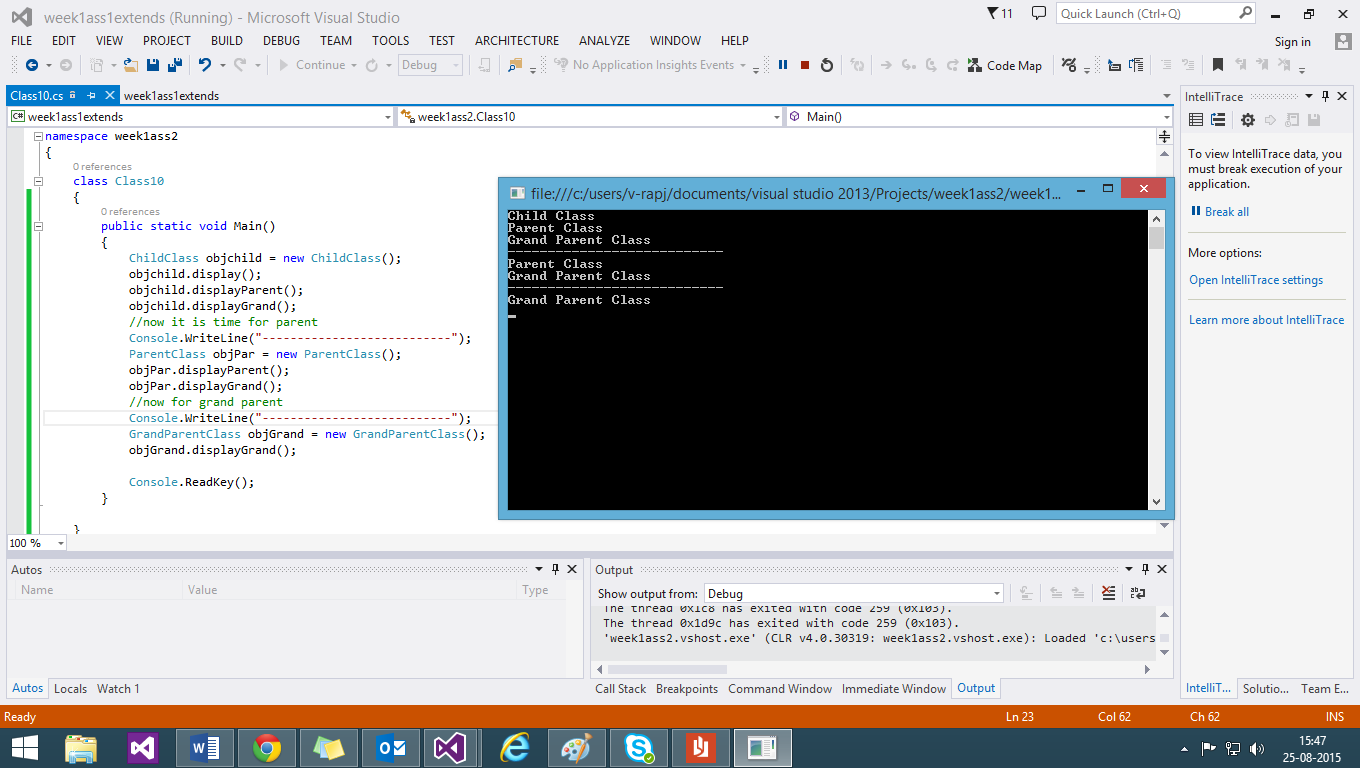
{

public void display()

{

Console.WriteLine("Child Class");

}

}

23. Write a C# Program to demonstrate Partial Class.

class Class12

{

public static void Main()

{

myPartial objPart = new myPartial();

Console.WriteLine("Enter two number to get Sum");

objPart.add(Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine()));

Console.WriteLine("Enter two number to get Multiplication");

objPart.Multi(Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine()));

Console.WriteLine("Enter two number to get Difference");

objPart.Sub(Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine()));

Console.ReadKey();

}

}

public partial class myPartial

{

public void add(int a , int b)

{

Console.WriteLine("sum is {0}\n from the add method", a + b);

}

}

public partial class myPartial

{

public void Multi(int a, int b)

{

Console.WriteLine("Multiplication results in {0}\n from the Multi method", a \* b);

}

}

public partial class myPartial

{

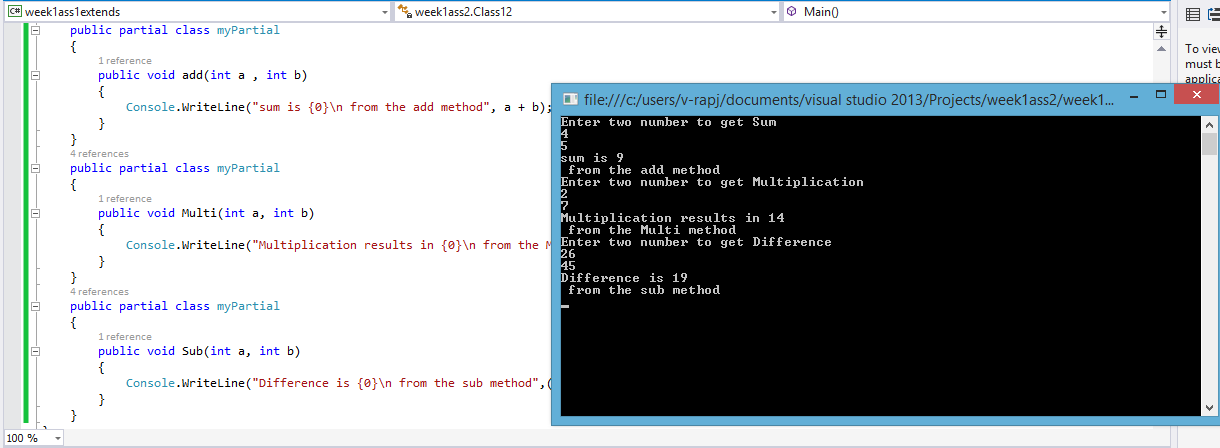
public void Sub(int a, int b)

{

Console.WriteLine("Difference is {0}\n from the sub method",(a>b? a - b:b-a));

}

}



24. Write a C# Program to demonstrate Sealed Class.

class Class13

{

public static void Main()

{

mysealed objdeal = new mysealed();

objdeal.print();

Console.ReadKey();

}

}

sealed class mysealed

{

int x = 10;

int y = 20;

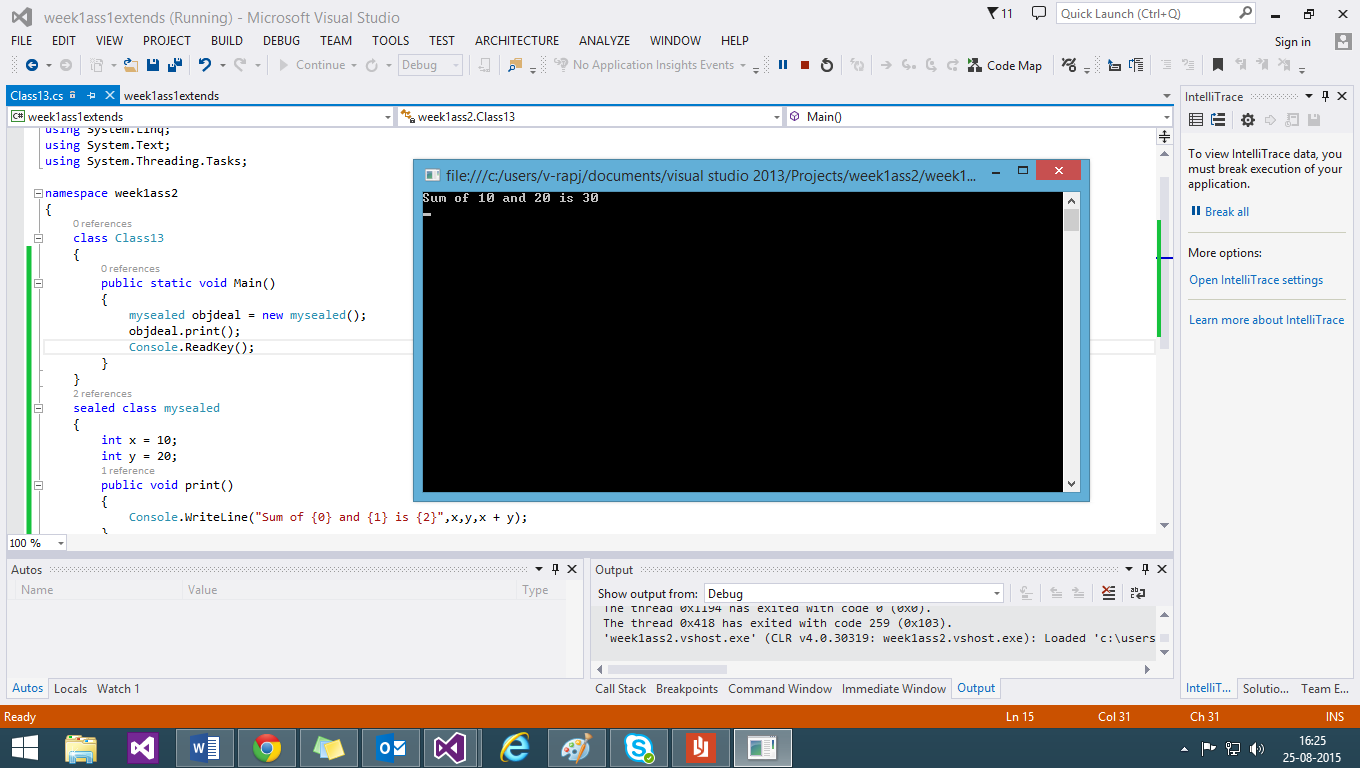
public void print()

{

Console.WriteLine("Sum of {0} and {1} is {2}",x,y,x + y);

}

}



25. Write a C# Program to demonstrate Dictionary, Hash Table Class.

class Class14

{

public static void Main()

{

Hashtableclass objHash = new Hashtableclass();

objHash.hash();

objHash.get();

objHash.print();

Dictionary<int, string> objDic = new Dictionary<int, string>();

objDic.Add(2, "run");

objDic.Add(6, "runing");

foreach (KeyValuePair<int, string> val in objDic)

{

Console.WriteLine("Key: {0} has the value :{1}",val.Key,val.Value);

}

Console.ReadKey();

}

}

class Hashtableclass

{

Hashtable mytable = new Hashtable();

public void hash()

{

mytable.Add("1", "Akshay");

mytable.Add("2", "Anita");

mytable.Add("3", "Harsh");

mytable.Add("4", "Nishanth");

mytable.Add("5", "Rajeshwaran");

}

private void Add(object key,object value)

{

if(mytable.ContainsKey(key)||mytable.ContainsValue(value))

{

Console.WriteLine("The value or key already exist");

}

else

{

mytable.Add(key, value);

Console.WriteLine("The value and key inserted");

}

}

public void get()

{

Console.WriteLine("Enter the key to insert :");

object key = Console.ReadLine();

Console.WriteLine("Enter the corresponding value :");

object value = Console.ReadLine();

Add(key, value);

}

public void print()

{

ICollection foundkey=mytable.Keys;

foreach (DictionaryEntry keyval in mytable)

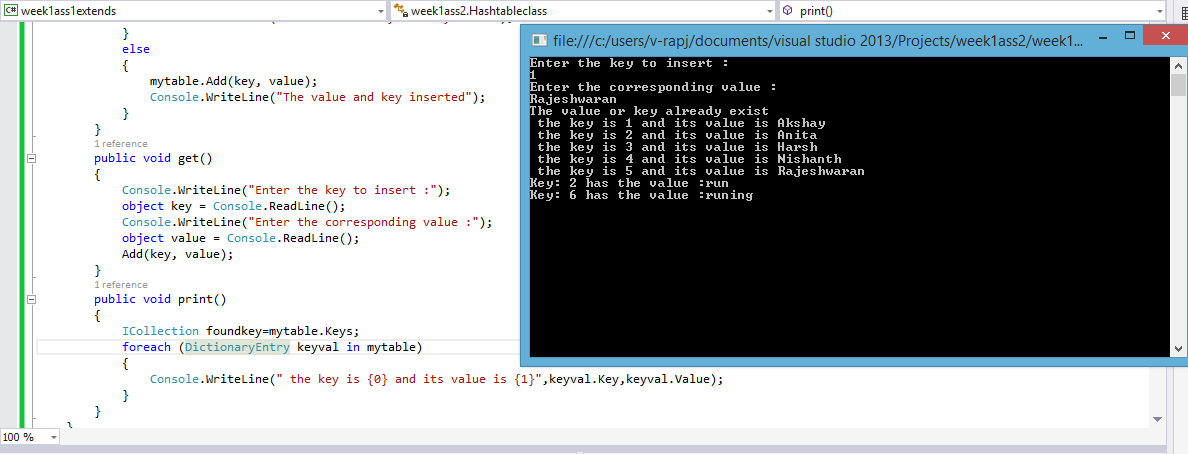
{

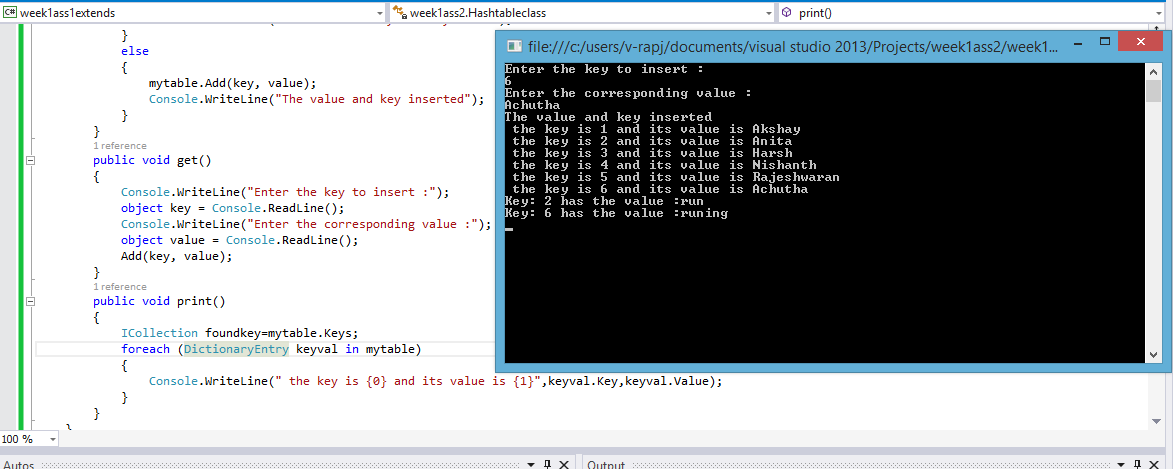
Console.WriteLine(" the key is {0} and its value is {1}",keyval.Key,keyval.Value);

}

}

}





26. Write a C# Program to demonstrate Generic Classes and Constraints.

class Class16

{

static void Main()

{

generic<int> objInt = new generic<int>();

Console.WriteLine("{0}",objInt.First(4,41));

generic<string> objString = new generic<string>();

Console.WriteLine("{0}",objString.First("Second is not first","this is second"));

generic<double> objDouble = new generic<double>();

Console.WriteLine("{0}",objDouble.First(2.12,4.21));

Console.Read();

}

}

class generic<T> //declaring the generic class

{

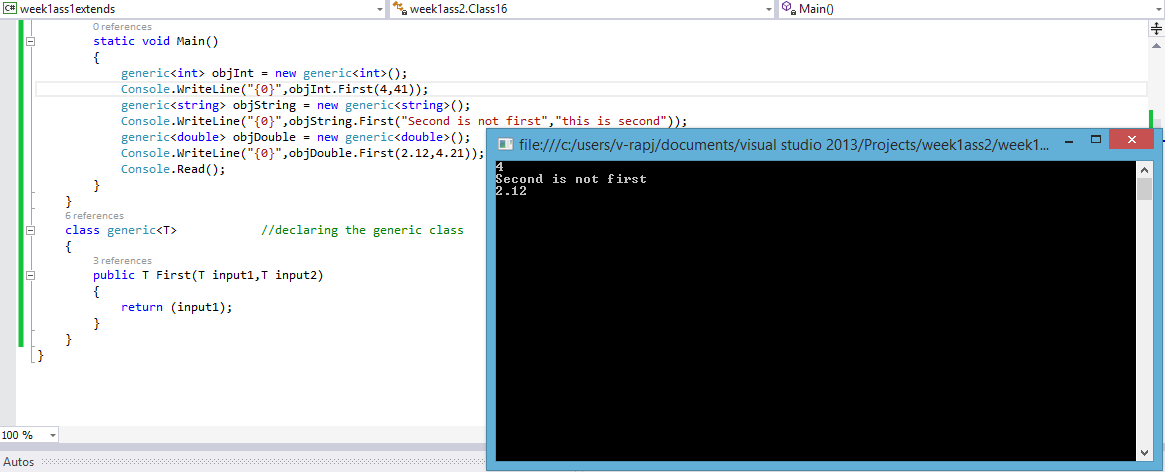
public T First(T input1,T input2)

{

return (input1);

}

}



27. Write a C# Program to demonstrate Structure and Enum.

public enum Shift { day, night };

struct Employee

{

public int Id;

public string Name;

public Int64 number;

};

class Class17

{

static void Main()

{

Employee e1;

Employee e2;

e1.Id = 1;

e1.Name = "Raj";

e1.number = 923812373;

e2.Id = 2;

e2.Name = "Noone";

e2.number = 412349827;

Shift s = Shift.day;

Shift n = Shift.night;

if(s==Shift.day) //true so the below code in console printed

{

Console.WriteLine("S==Day Shift");

Console.WriteLine(e1.Name);

}

if (n == Shift.day) //false so the below code in console not printed

{

Console.WriteLine("N!=Day Shift");

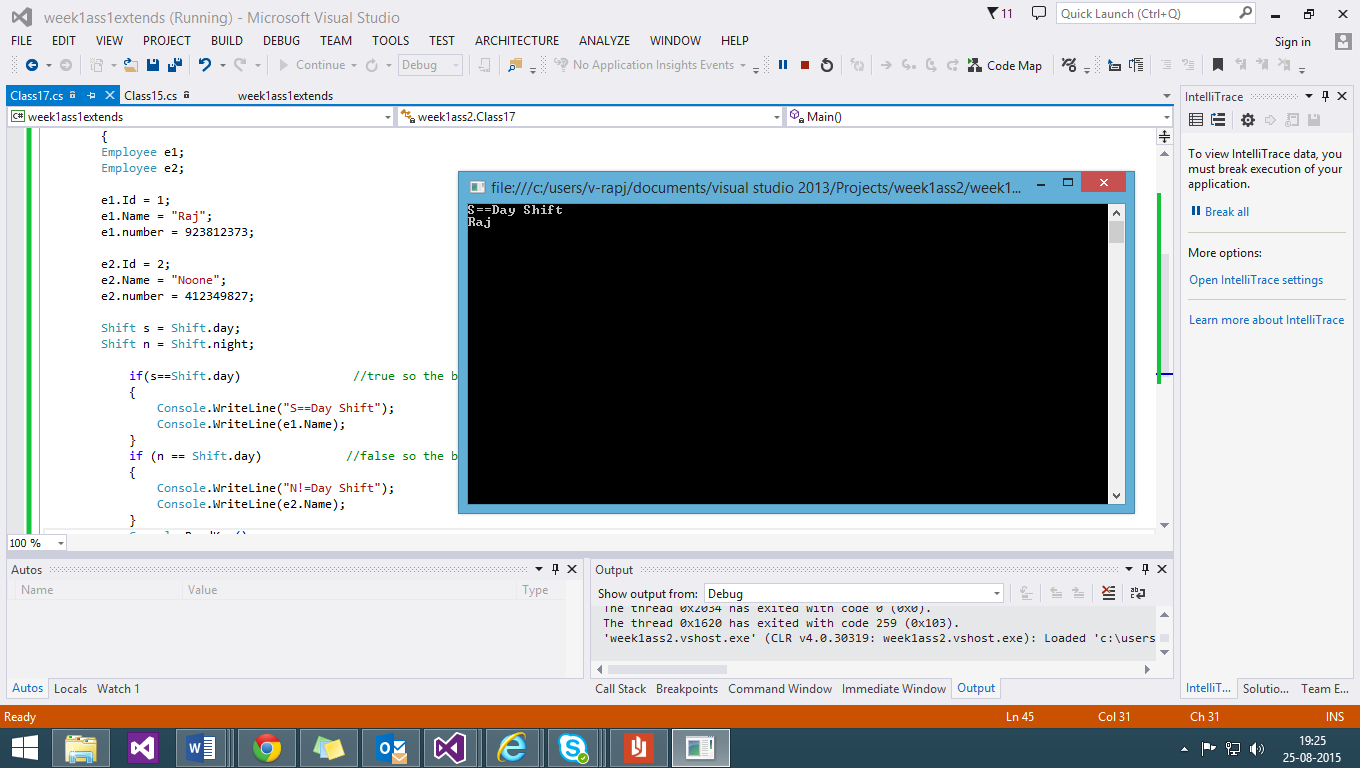
Console.WriteLine(e2.Name);

}

Console.ReadKey();

}

}



Thanks,

Rajeshwaran