.Net Programming

Digital Assingment-1

Name: V. Sai Nikhil

Reg No: 16MIS0257

Create a DLL as your choice of any object. Write a Menu driven console application that should discover all the types that are available within the assembly. Use the concept of Multi File, Multicast delegates and store the count of types in registry. Provide an option for executing an method during runtime using the concept of Late Binding.

VB.Net Dll:

```
Public Class dog
    Sub height()
    End Sub

    Sub weight()
    End Sub

    Sub width()
    End Sub
End Class
```

C# Dll:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using da1vbclass;

namespace da1cclass
{
    //inheriting dog from VB.net class library with pet
    public class pet:dog
    {
        public string name, breed;
        //constructor
        public pet()
        {
        }
```

```
        //methods
        public void bark()
        {
        }
        public void laydown()
        {
        }
        public void sleep()
        {
        }
        public void playdead()
        {
        }

        //properties
        public string pbreed
        {
            get{ return breed; }
            set { breed = value; }
        }
        public string pname
        {
            get { return name; }
            set { name = value; }
        }
    }
}
```

Console Application:

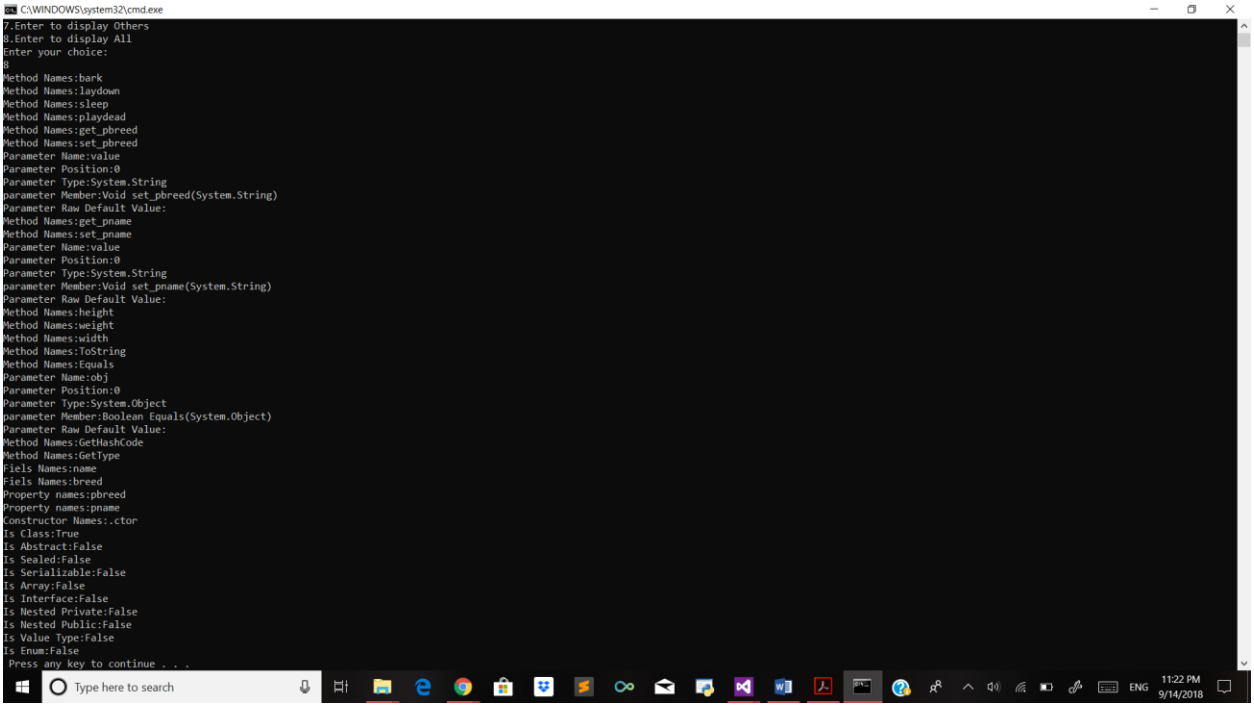
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using Microsoft.Win32;
using da1cclass;
using da1vbclass;

public delegate void sd(object obj);

namespace da1console
{
    public class general
    {
        RegistryKey rk;
        int mcount = 0, pmcount = 0, fcount = 0, ccount = 0, pcount = 0, procount = 0,
intcount = 0, mmcount = 0;
        public general()
        {
            rk = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\digital
assingment", true);
            if (rk == null)
                rk = Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\digital
assingment");
```

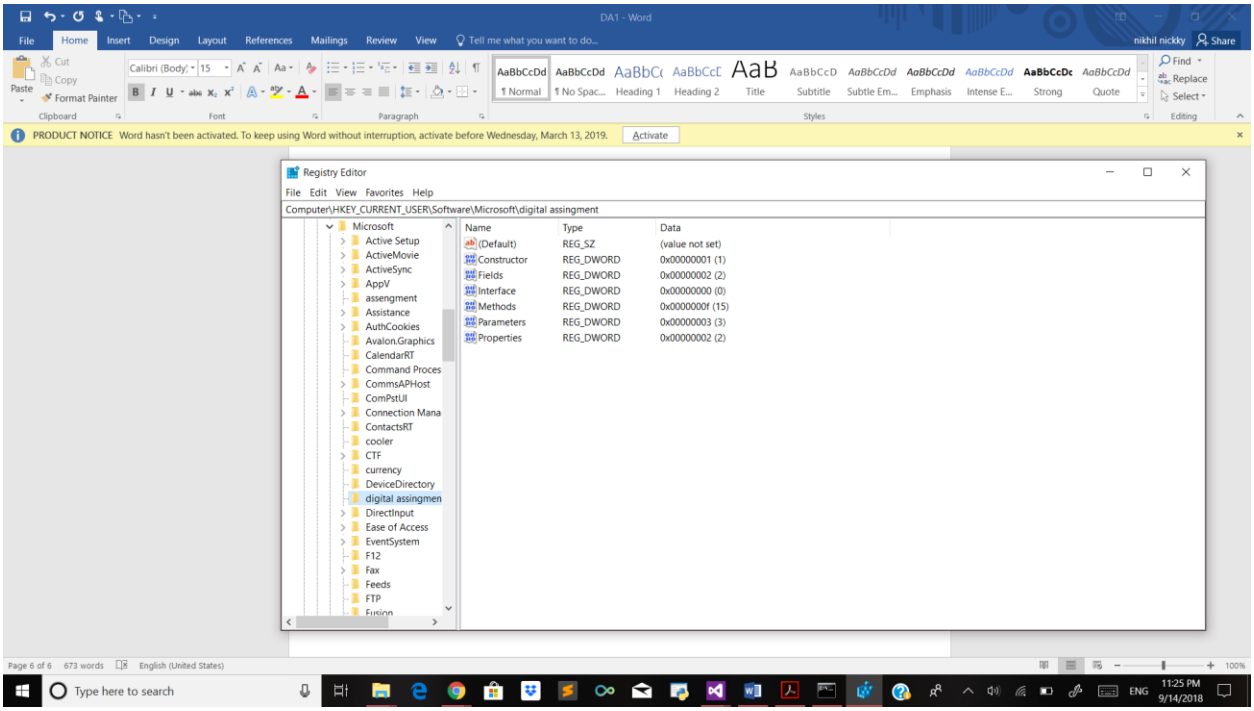
```
        }
        public void get_m(object obj)
        {
            Type t = obj.GetType();
            MethodInfo[] mi = t.GetMethods();
            foreach (MethodInfo m in mi)
            {
                Console.WriteLine("Method Names:{0}", m.Name);
                mcount++;
                ParameterInfo[] pi = m.GetParameters();
                foreach (ParameterInfo p in pi)
                {
                    Console.WriteLine("Parameter Name:{0}", p.Name);
                    Console.WriteLine("Parameter Position:{0}", p.Position);
                    Console.WriteLine("Parameter Type:{0}", p.ParameterType);
                    Console.WriteLine("parameter Member:{0}", p.Member);
                    Console.WriteLine("Parameter Raw Default Value:{0}",
p.RawDefaultValue);
                    pmcount++;
                }
                rk.SetValue("Parameters", pmcount);
            }
            rk.SetValue("Methods", mcount);
        }
        public void get_f(object obj)
        {
            Type t = obj.GetType();
            FieldInfo[] fi = t.GetFields();
            foreach (FieldInfo f in fi)
            {
                Console.WriteLine("Fiels Names:{0}", f.Name);
                fcount++;
            }
            rk.SetValue("Fields", fcount);
        }
        public void get_p(object obj)
        {
            Type t = obj.GetType();
            PropertyInfo[] ppi = t.GetProperties();
            foreach (PropertyInfo p in ppi)
            {
                Console.WriteLine("Property names:{0}", p.Name);
                pcount++;
            }
            rk.SetValue("Properties", pcount);
        }
        public void get_c(object obj)
        {
            Type t = obj.GetType();
            ConstructorInfo[] ci = t.GetConstructors();
            foreach (ConstructorInfo c in ci)
            {
                Console.WriteLine("Constructor Names:{0}", c.Name);
                ccount++;
            }
            rk.SetValue("Constructor", ccount);
        }
        public void get_i(object obj)
```

```
    {
        Type t = obj.GetType();
        Type[] ti = t.GetInterfaces();
        foreach (Type o in ti)
        {
            Console.WriteLine(o.Name);
            MethodInfo[] mi = o.GetMethods();
            foreach (MethodInfo m in mi)
                Console.WriteLine("Interface Method Names:{0}", m.Name);
            FieldInfo[] fi = o.GetFields();
            foreach (FieldInfo f in fi)
                Console.WriteLine("Interface Fiels Namws:{0}", f.Name);
            PropertyInfo[] ppi = t.GetProperties();
            foreach (PropertyInfo p in ppi)
                Console.WriteLine("Interface Property Names:{0}", p.Name);
            ConstructorInfo[] ci = t.GetConstructors();
            foreach (ConstructorInfo c in ci)
                Console.WriteLine("Interface COnstructor Names:{0}", c.Name);
            intcount++;
        }
        rk.SetValue("Interface", intcount);
    }
    public void get_oth(object obj)
    {
        Type t = obj.GetType();
        Console.WriteLine("Is Class:{0}", t.IsClass);
        Console.WriteLine("Is Abstract:{0}", t.IsAbstract);
        Console.WriteLine("Is Sealed:{0}", t.IsSealed);
        Console.WriteLine("Is Serializable:{0}", t.IsSerializable);
        Console.WriteLine("Is Array:{0}", t.IsArray);
        Console.WriteLine("Is Interface:{0}", t.IsInterface);
        Console.WriteLine("Is Nested Private:{0}", t.IsNestedPrivate);
        Console.WriteLine("Is Nested Public:{0}", t.IsNestedPublic);
        Console.WriteLine("Is Value Type:{0}", t.IsValueType);
        Console.WriteLine("Is Enum:{0}", t.IsEnum);

    }

}
class Program
{
    static void Main(string[] args)
    {
        Assembly a = null;
        try
        {
            a = Assembly.Load("da1cclass");
        }
        catch (Exception e)
        {
            Console.WriteLine(e);
        }
        general g = new general();
        dog d = new dog();
        Type t = a.GetType("da1cclass.pet");
        Console.WriteLine("1.Enter to display Methods");
        Console.WriteLine("2.Enter to display Constructors");
        Console.WriteLine("3.Enter to display Fields");
```

```
            Console.WriteLine("4.Enter to display Properties");
            Console.WriteLine("5.Enter to display Interface");
            Console.WriteLine("6.Enter to display VB.Net Program");
            Console.WriteLine("7.Enter to display Others");
            Console.WriteLine("8.Enter to display All");

            Console.WriteLine("Enter your choice:");
            int r = Convert.ToInt32(Console.ReadLine());

            switch (r)
            {
                case 1:
                    object c1 = Activator.CreateInstance(t);
                    g.get_m(c1);
                    break;
                case 2:
                    object c2 = Activator.CreateInstance(t);
                    g.get_c(c2);
                    break;
                case 3:
                    object c3 = Activator.CreateInstance(t);
                    g.get_f(c3);
                    break;
                case 4:
                    object c4 = Activator.CreateInstance(t);
                    g.get_p(c4);
                    break;
                case 5:
                    object c5 = Activator.CreateInstance(t);
                    g.get_i(c5);
                    break;
                case 6:
                    g.get_m(d);
                    break;
                case 7:
                    object c7 = Activator.CreateInstance(t);
                    g.get_oth(c7);
                    break;
                case 8:
                    object all = Activator.CreateInstance(t);
                    g.get_m(all);
                    g.get_f(all);
                    g.get_p(all);
                    g.get_c(all);
                    g.get_i(all);
                    g.get_oth(all);
                    break;
                default:
                    Console.WriteLine("Invalid choice");
                    break;
            }
            Console.ReadKey();
        }
    }
}
```

Output:


C:\WINDOWS\system32\cmd.exe
7.Enter to display Others
8.Enter to display All
Enter your choice:
8
Method Names:bark
Method Names:laydown
Method Names:sleep
Method Names:playdead
Method Names:get_pbreed
Method Names:set_pbreed
Parameter Name:value
Parameter Position:0
Parameter Type:System.String
parameter Member:Void set_pbreed(System.String)
Parameter Raw Default Value:
Method Names:get_pname
Method Names:set_pname
Parameter Name:value
Parameter Position:0
Parameter Type:System.String
parameter Member:Void set_pname(System.String)
Parameter Raw Default Value:
Method Names:height
Method Names:weight
Method Names:width
Method Names:ToString
Method Names:Equals
Parameter Name:obj
Parameter Position:0
Parameter Type:System.Object
parameter Member:Boolean Equals(System.Object)
Parameter Raw Default Value:
Method Names:GetHashCode
Method Names:GetType
Fiels Names:name
Fiels Names:breed
Property names:pbreed
Property names:pname
Constructor Names:.ctor
Is Class:True
Is Abstract:False
Is Sealed:False
Is Serializable:False
Is Array:False
Is Interface:False
Is Nested Private:False
Is Nested Public:False
Is Value Type:False
Is Enum:False
Press any key to continue . . .
Type here to search
ENG
11:22 PM
9/14/2018


DA1 - Word
PRODUCT NOTICE Word hasn't been activated. To keep using Word without interruption, activate before Wednesday, March 13, 2019.
Activate
Registry Editor
File Edit View Favorites Help
Computer\HKEY_CURRENT_USER\Software\Microsoft\digital assingment
Name Type Data
(Default) REG_SZ (value not set)
Constructor REG_DWORD 0x00000001 (1)
Fields REG_DWORD 0x00000002 (2)
Interface REG_DWORD 0x00000000 (0)
Methods REG_DWORD 0x0000000f (15)
Parameters REG_DWORD 0x00000003 (3)
Properties REG_DWORD 0x00000002 (2)
Type here to search
ENG
11:25 PM
9/14/2018