**TASK 7**

**Lab Exercise**

Using Reflection in C#: Demonstrate how to gather information on various types included in any assembly by using the System.Reflection namespace and some main.NET baseclasses.

**Reflection:**

* Reflection objects are used for obtaining type information at runtime.
* **in C#, reflection is a process to get metadata of a type at runtime.**
* The classes that give access to the metadata of a running program are in the  System.Reflection namespace.
* The System.Reflection namespace contains classes that allow you to obtain information about the application and to dynamically add types, values, and objects to the application.
* It allows view attribute information at runtime.
* It allows examining various types in an assembly and instantiate these types.
* It allows late binding to methods and properties

The System.Reflection namespace contains required classes for reflection such as:

* Type
* MemberInfo
* ConstructorInfo
* MethodInfo
* FieldInfo
* PropertyInfo
* TypeInfo
* EventInfo
* Module
* Assembly
* AssemblyName

**Program:**

using System;

using System.Reflection;

namespace ReflectionDemo

{

class Program

{

public int id;

public string name { get; set; }

public double salary;

public void Method1()

{

Console.WriteLine("SampleMethod1 called.");

}

public int Method2(int param1, string param2)

{

return param1 + param2.Length;

}

public static void Main(string[] args)

{

// Load the assembly (for this example, we'll use the current assembly)

Assembly assembly = Assembly.GetExecutingAssembly();

// Display the assembly's full name

Console.WriteLine($"Assembly Full Name: {assembly.FullName}\n");

// Get all types defined in the assembly

Type[] types = assembly.GetTypes();

foreach (Type type in types)

{

Console.WriteLine($"Type: {type.FullName}");

// Display all public methods of the type

MethodInfo[] methods = type.GetMethods();

foreach (MethodInfo method in methods)

{

Console.WriteLine($" Method: {method.Name}");

// Display method parameters

ParameterInfo[] parameters = method.GetParameters();

foreach (ParameterInfo parameter in parameters)

{

Console.WriteLi($"Parameter: {parameter.Name} of Type:" +

$"{parameter.ParameterType}");

}

// Display the return type of the method

Console.WriteLine($" Return Type: {method.ReturnType}");

}

// Display all public properties of the type

PropertyInfo[] properties = type.GetProperties();

foreach (PropertyInfo property in properties)

{

Console.WriteLine($" Property: {property.Name} of Type: {property.PropertyType}");

}

// Display all public fields of the type

FieldInfo[] fields = type.GetFields();

foreach (FieldInfo field in fields)

{

Console.WriteLine($" Field: {field.Name} of Type: {field.FieldType}");

}

} } }}

**Sample Program:**

using System;

using System.Reflection;

public class Program

{

public static void Main()

{

String studentName = "Abhishek ”;

// get the current type of studentName and class

Type t1 = studentName.GetType();

Type t2 = typeof(Program);

Console.WriteLine(t1);

Console.WriteLine(t1.Namespace);

Console.WriteLine(t1.BaseType);

Console.WriteLine(t2.Assembly);

Console.WriteLine(t2.FullName);

Console.WriteLine(t2.IsClass);

Console.WriteLine(t2.IsEnum);

Console.WriteLine(t2.IsInterface);

}

}