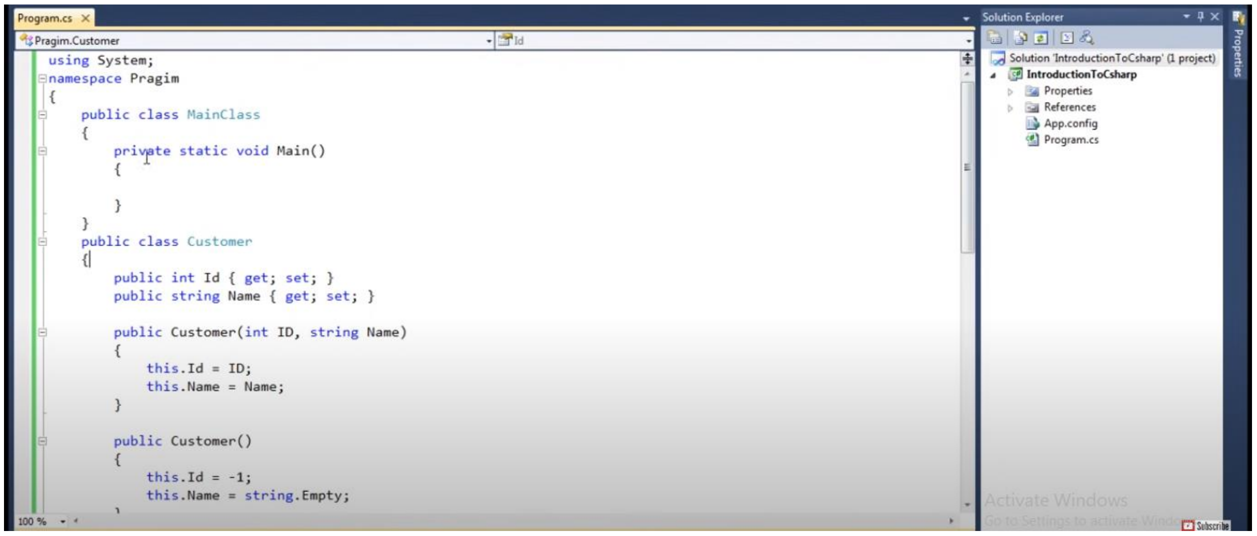
<u>Reflection</u>

- **Reflection** is ability of inspecting an assemblies metadata at runtime.
  Let's Understand,
  What do you mean by inspecting an Assemblies metadata at run time?
  Look at this simple console application...



```
using System;
namespace Pragim
{
    public class MainClass
    {
        private static void Main()
        {

        }
    }
    public class Customer
    {
        public int Id { get; set; }
        public string Name { get; set; }

        public Customer(int ID, string Name)
        {
            this.Id = ID;
            this.Name = Name;
        }

        public Customer()
        {
            this.Id = -1;
            this.Name = string.Empty;
```

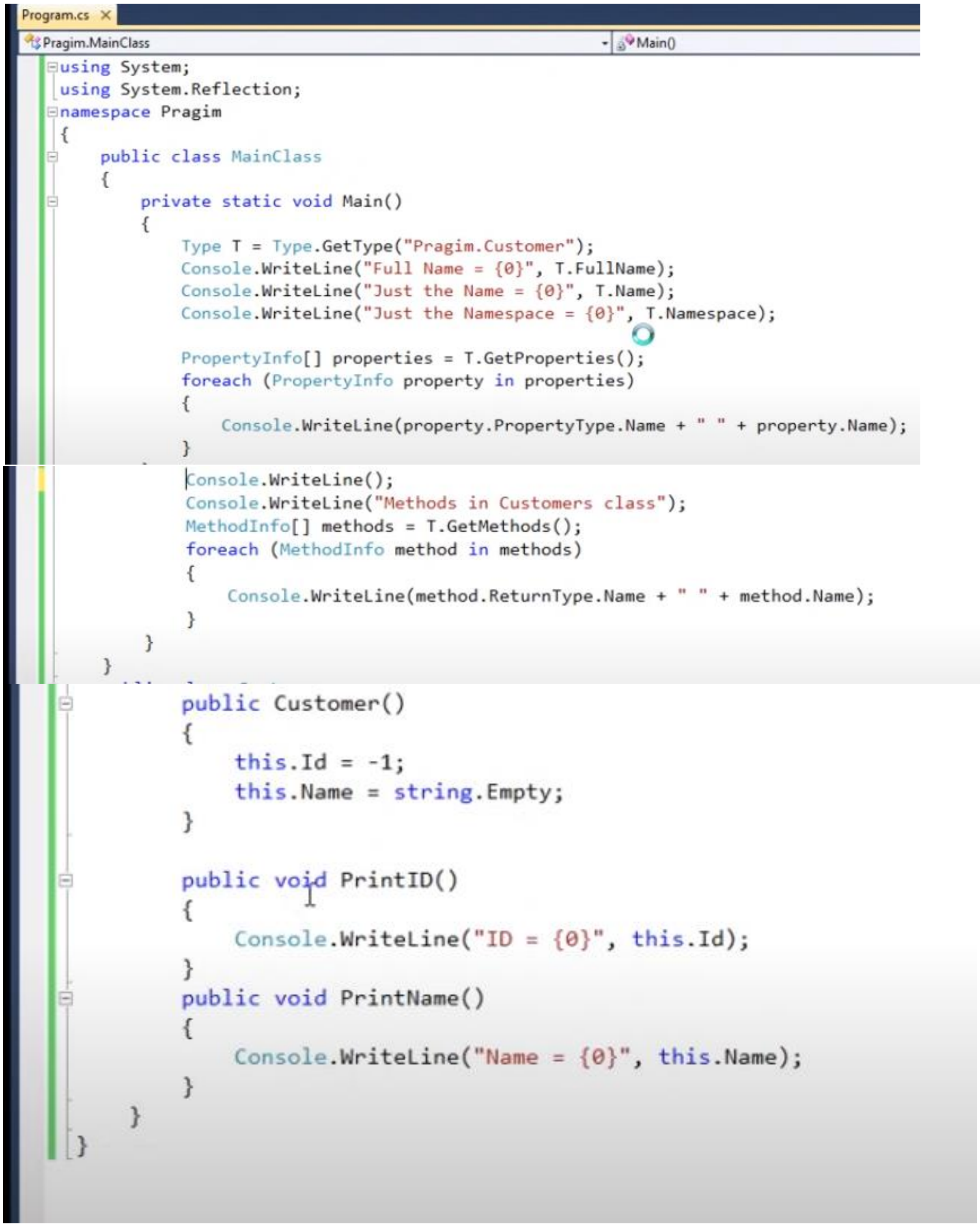
- This application has Main class and customer class,
- **When we build this app what going happened is these?**
  ⇨ Two classes are compiled into **intermediate language** and package into something called an **assembly**.
  ⇨ When we look at assembly,
    ➢ **Assembly** consists of **two** parts,
      ▪ One is **intermediate language** and
      ▪ Other one is the **metadata**.

- **What does this metadata contain?**
  ⇨ It contains the information about the **types** within that assembly.

- **What is the name, what are the different types here with in this assembly ?**
  ⇨ Main class and customer class and if you take customer class what does it have in it.
  See What are the members of this customer class?
  If you look at that customer class has
    - Two public **properties,**
    - Two **constructors,**
    - Two **methods.**

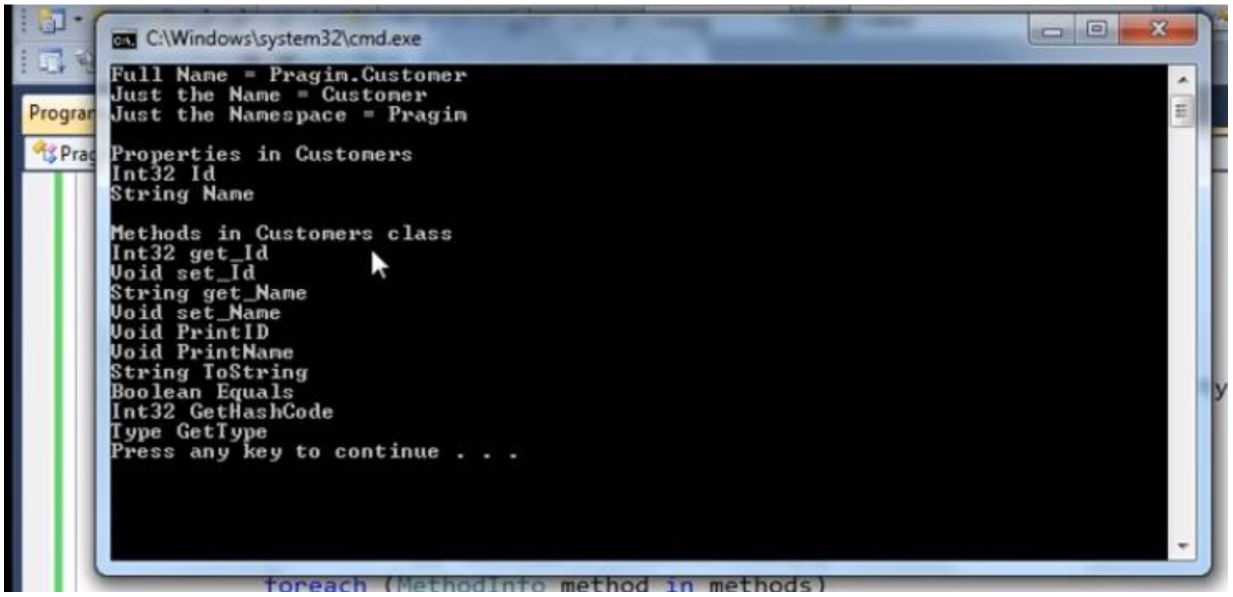
  So if you look at, these are members of this **customer class,**
  ⇨ So all this information about these **types** are actually packaged into that **assembly** in the form of **metadata**.
- **So what is Reflection?**
  Reflection is actually taking that assembly and then **inspecting** the definitely to **find out how many classes** has this assembly got,
  How many enums or **structure** has this assembly got?
  And each class what are the different **members** that each class or **enum/ structure** has gone.
  So doing this inspecting an assemblies contents by looking at its metadata runtime is nothing but in a reflection.

Lets see Example : 1



```
using System;
using System.Reflection;
namespace Pragim
{
    public class MainClass
    {
        private static void Main()
        {
            Type T = Type.GetType("Pragim.Customer");
            Console.WriteLine("Full Name = {0}", T.FullName);
            Console.WriteLine("Just the Name = {0}", T.Name);
            Console.WriteLine("Just the Namespace = {0}", T.Namespace);

            PropertyInfo[] properties = T.GetProperties();
            foreach (PropertyInfo property in properties)
            {
                Console.WriteLine(property.PropertyType.Name + " " + property.Name);
            }
            Console.WriteLine();
            Console.WriteLine("Methods in Customers class");
            MethodInfo[] methods = T.GetMethods();
            foreach (MethodInfo method in methods)
            {
                Console.WriteLine(method.ReturnType.Name + " " + method.Name);
            }
        }
    }
        public Customer()
        {
            this.Id = -1;
            this.Name = string.Empty;
        }

        public void PrintID()
        {
            Console.WriteLine("ID = {0}", this.Id);
        }
        public void PrintName()
        {
            Console.WriteLine("Name = {0}", this.Name);
        }
    }
}
```

run this code….

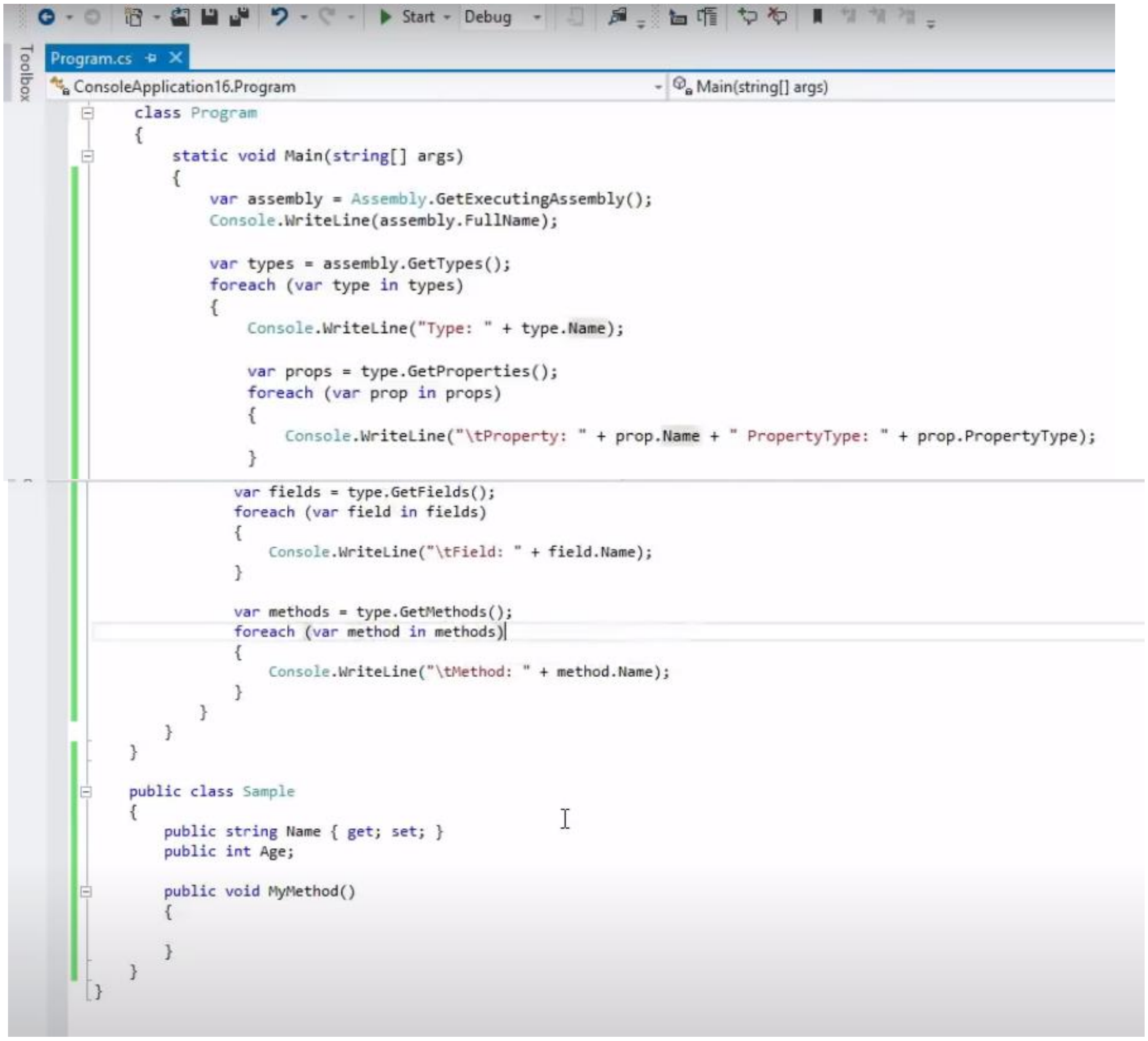


```
Full Name = Pragim.Customer
Just the Name = Customer
Just the Namespace = Pragim

Properties in Customers
Int32 Id
String Name

Methods in Customers class
Int32 get_Id
Void set_Id
String get_Name
Void set_Name
Void PrintID
Void PrintName
String ToString
Boolean Equals
Int32 GetHashCode
Type GetType
Press any key to continue . . .
```

```
foreach (MethodInfo method in methods)
```

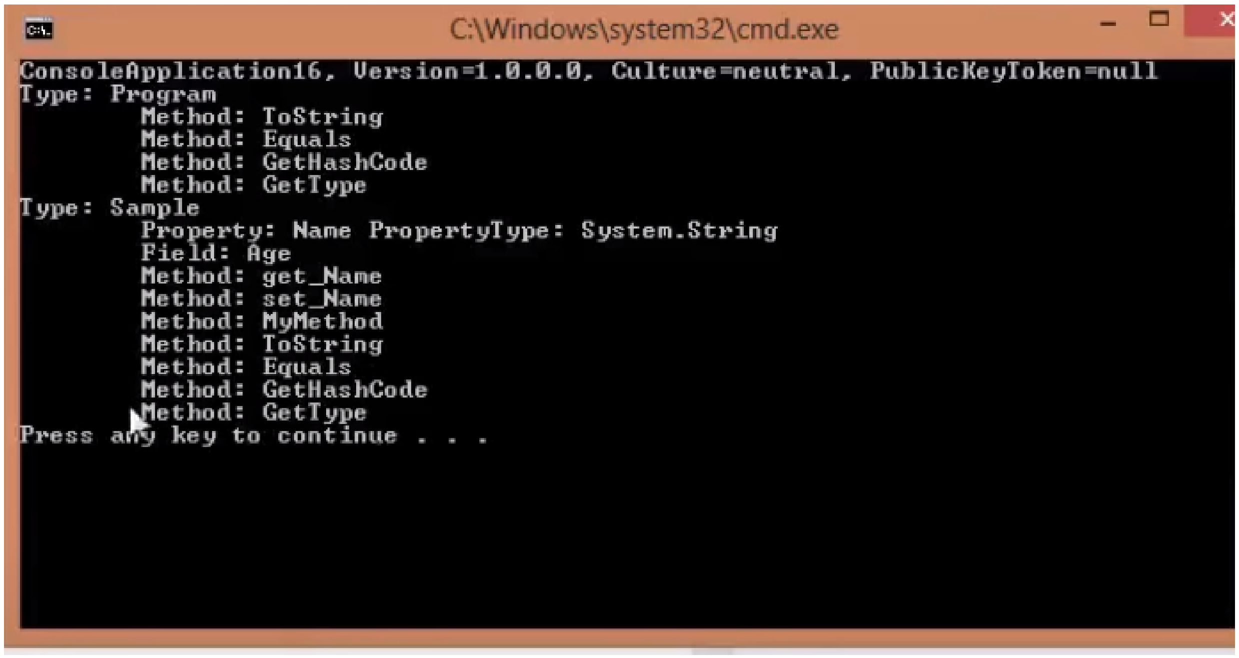
Example 2:

https://www.youtube.com/watch?v=3FvT6uNMT7M



```
class Program
{
    static void Main(string[] args)
    {
        var assembly = Assembly.GetExecutingAssembly();
        Console.WriteLine(assembly.FullName);

        var types = assembly.GetTypes();
        foreach (var type in types)
        {
            Console.WriteLine("Type: " + type.Name);

            var props = type.GetProperties();
            foreach (var prop in props)
            {
                Console.WriteLine("\tProperty: " + prop.Name + " PropertyType: " + prop.PropertyType);
            }
            var fields = type.GetFields();
            foreach (var field in fields)
            {
                Console.WriteLine("\tField: " + field.Name);
            }

            var methods = type.GetMethods();
            foreach (var method in methods)
            {
                Console.WriteLine("\tMethod: " + method.Name);
            }
        }
    }
}

public class Sample
{
    public string Name { get; set; }
    public int Age;

    public void MyMethod()
    {

    }
}
}
```

Run this code…



```
ConsoleApplication16, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Type: Program
        Method: ToString
        Method: Equals
        Method: GetHashCode
        Method: GetType
Type: Sample
        Property: Name PropertyType: System.String
        Field: Age
        Method: get_Name
        Method: set_Name
        Method: MyMethod
        Method: ToString
        Method: Equals
        Method: GetHashCode
        Method: GetType
Press any key to continue . . .
```