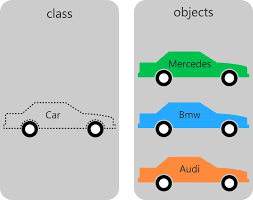
**Object and Class**

Any real-world entity which can have some characteristics or which can perform some work is called as Object. This object is also called as an instance of class.

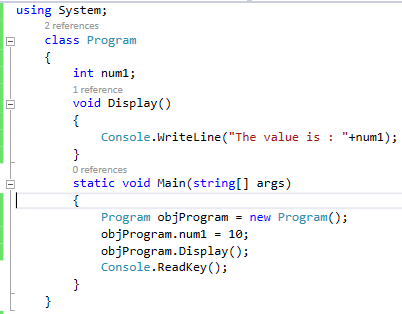
Class is a plan which describes the object. We call it as a blue print of how the object should be represented. Mainly a class would consist of a name, attributes & operations.

For example, consider we have a class of cars under which Mercedes, Audi and BMW represents individual objects. In this context each car will have its own Colour, model year, Engine power etc which form properties of the Car class and the associated actions i.e., objects functions like Start, Move, stop form the methods of Car class.



**Description**

First of all, a class is a group of related methods and variables. A class describes these things, and in most cases, you create an instance of this class, now referred to as an object. On this object, you use the defined methods and variables. Of course, you can create as many instances of your class as you want to.



**Constructors**

A special method of the class that will be automatically invoked when an instance of the class is created is called a constructor. The main use of constructors is to initialize private fields of the class while creating an instance for the class.

There can be two types of constructors in C#.

* **Default constructor**

A constructor without any parameters is called a default constructor; in other words, this type of constructor does not take parameters

* **Parameterized constructor**

A constructor with at least one parameter is called a parameterized constructor. The advantage of a parameterized constructor is that you can initialize each instance of the class to different values.

**Destructor**

A destructor runs after a class becomes unreachable. It has the special "~" character in its name. The exact time it is executed is not specified. But it always runs when the class is not reachable in memory by any references.

It destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.

C# destructor cannot have parameters. Moreover, modifiers can’t be applied on destructors.

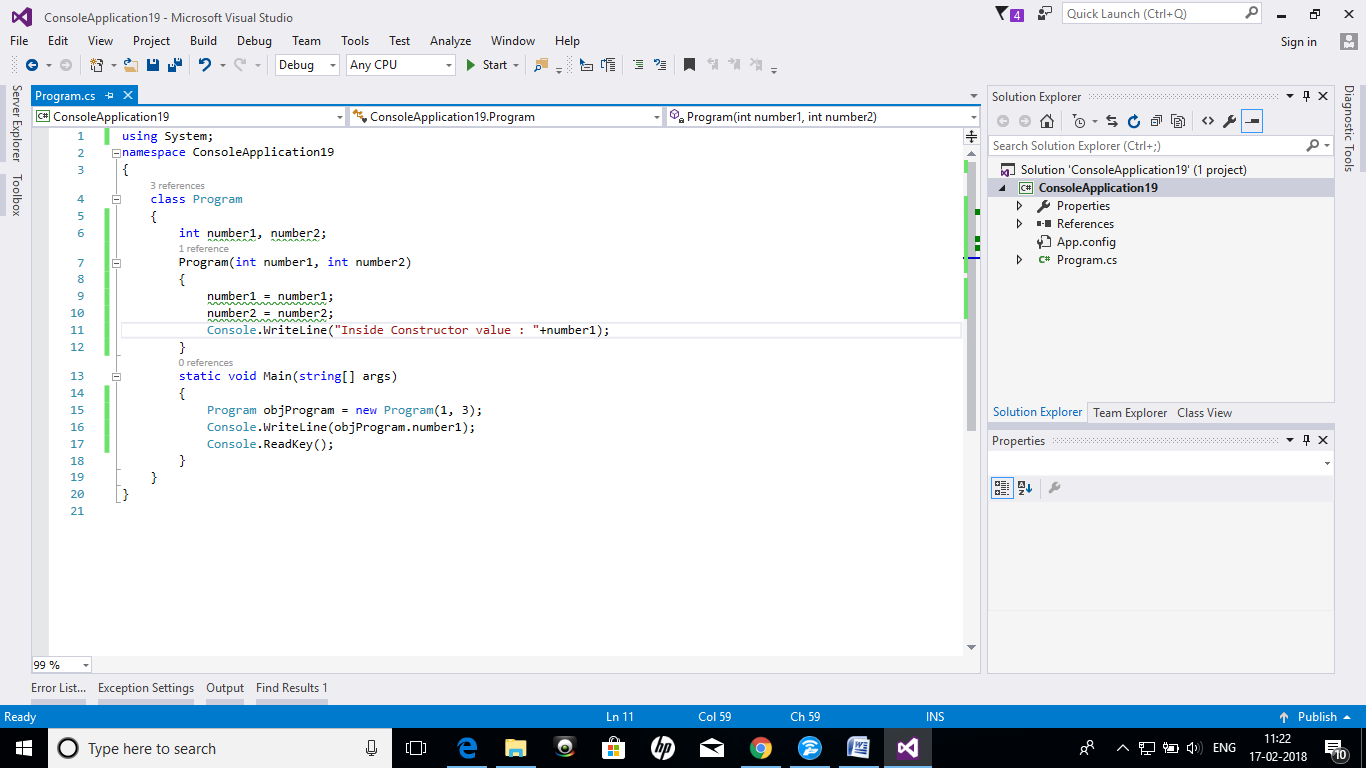
**this keyword**

In c# programming, this is a keyword that refers to the current instance of the class.

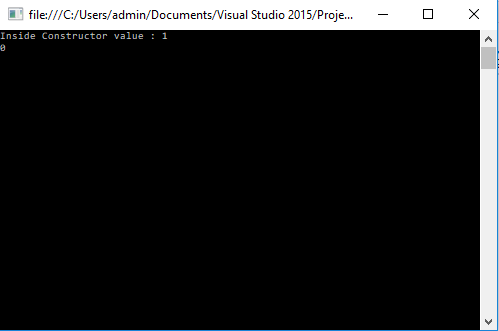
It can be used **to refer current class instance variable**. It is used if field names (instance variables) and parameter names are same, that is why both can be distinguish easily.

It can be used to pass current object as a parameter to another method.

**Problem without this keyword**



**Output**

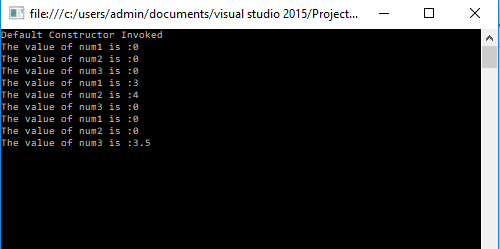


In this example, value is never assigned to objProgram.number1. So, it prints 0 as its value i.e. a default value.

**Source Code**



**OUTPUT:**



**Static Keyword**

In C#, static is a keyword or modifier that belongs to the type not instance. So instance is not required to access the static members. In C#, static can be variable, method, class and constructor.

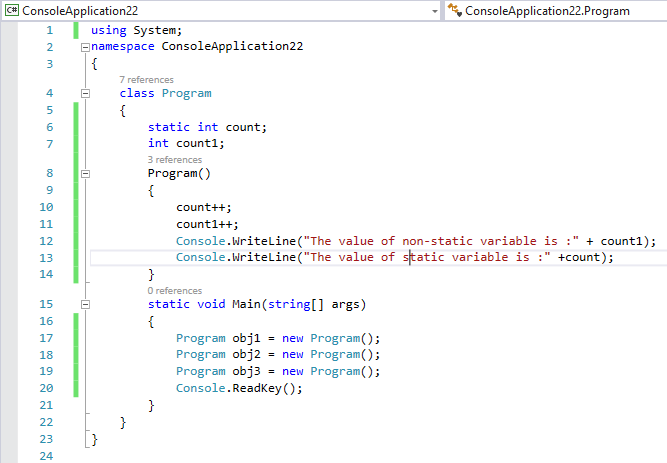
**Static Variables**

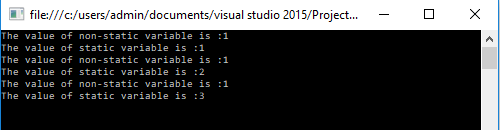
A C# class can contain both static and non-static members. When we declare a member with the help of the keyword static, it becomes a static member.

A static member belongs to the class rather than to the objects of the class. Hence static data members are also known as class members.

We cannot use this keyword with static.

**Memory Allocation to static variables**





**Static Constructor**

C# static constructor is used to initialize static fields. It can also be used to perform any action that is to be performed only once. It is invoked automatically before first instance is created or any static member is referenced.

C# static constructor cannot have any modifier or parameter.

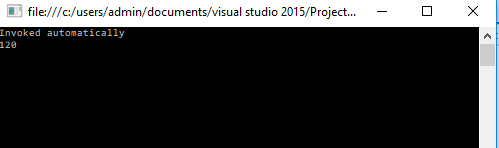
C# static constructor is invoked implicitly. It can't be called explicitly.

**Static Class**

The C# static class is like the normal class but it cannot be instantiated. It can have only static members. The advantage of static class is that it provides you guarantee that instance of static class cannot be created.



**OUTPUT:**

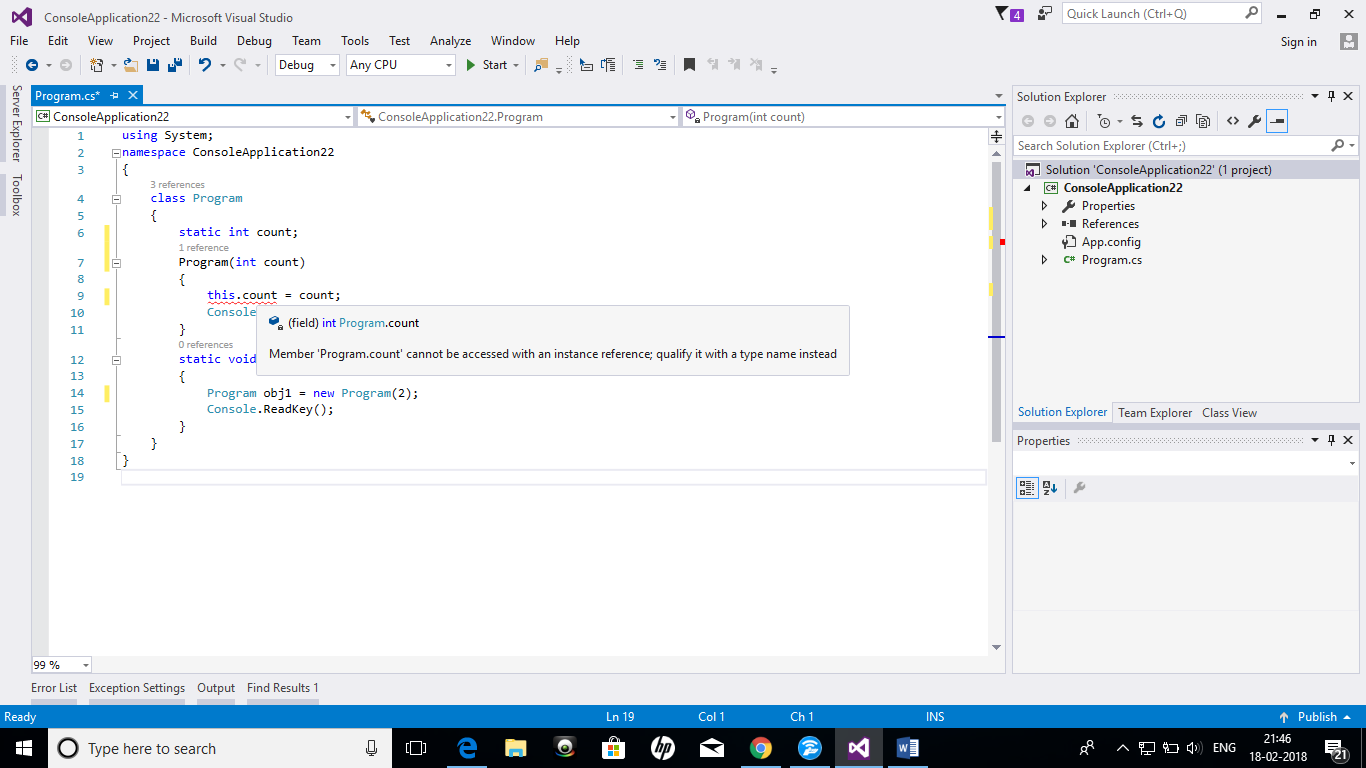


**Questionnaire**

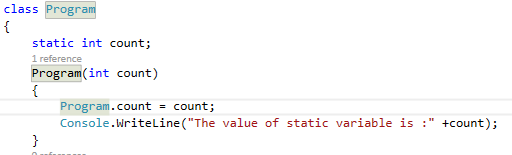
**Q. What if there is no constructor defines in the class?**

When you have not created a constructor in the class, the compiler will automatically create a default constructor in the class. The default constructor initializes all numeric fields in the class to zero and all strings and object fields to null.

**Q. How to resolve this keyword problem in case of static variables?**



So, this keyword can’t be used with static variable and to resolve the error we need to write Program.count as static variable always belongs to the class.



**Q. When a static constructor gets invoked?**

* A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.
* A static constructor cannot be called directly.
* The user has no control on when the static constructor is executed in the program.

**Q. If in a static class, we don’t have non-static members/functions then does it hold any default constructor?**

Yes, it does hold a constructor but it is also declared as static.