1. Constructors
2. Destructors

# 1 Constructors

The constructor is a special method which is invoked automatically at the time of instance/ object creation of that class. Mostly, it is used to initialize the data members of the new object. The constructor in C# has the same name as of class or struct.

## 1.1 Class Constructors

Class constructors are basically of 5 types:

1) Default Constructor

2) Parameterized Constructor

3) Copy Constructor

4) Static Constructor

5) Private Constructor

### 1.1.1 Default Constructor

A constructor without any parameters is called a default constructor; in other words, this type of constructor does not take parameters. The drawback of a default constructor is that every instance of the class will be initialized to the same values and it is not possible to initialize each instance of the class with different values.

The default constructor initializes:

a) All numeric fields in the class to zero.

b) All string and object fields to null.

c) All date time objects are initialized to 01/01/1700

d) All Boolean values are initialized to false.

e) The enum types takes the first value in the enumeration.

Download Default Constructor files.

### 1.1.2 Parameterized Constructor

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 with a different value.

There can be any number of parameterized constructor in a class. The only criteria is that the signature should be different. This is called as Constructor Overloading.

Download Parameterized Constructor files.

### 1.1.3 Copy Constructor

The constructor which creates an object by copying variables from another object is called a copy constructor. The purpose of a copy constructor is to initialize a new instance to the values of an existing instance.

Download Copy Constructor files.

### 1.1.4 Static Constructor

When a constructor is created using a static keyword, it will be invoked only once for all of the instances of the class and it is invoked during the creation of the first instance of the class or the first reference to a static member in the class. A static constructor is used to initialize static fields of the class and to write the code that needs to be executed only once.

1. A static constructor does not take access modifiers or have parameters.
2. A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.
3. A static constructor cannot be called directly.
4. The user has no control over when the static constructor is executed in the program.
5. A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.

Download Static Constructor files.

### 1.1.5 Private Constructor

When a constructor is created with a private specifier, it is not possible for other classes to derive from this class, neither is it possible to create an instance of this class. They are usually used in classes that contain static members only. Some key points of a private constructor are:

1. One use of a private constructor is when we have only static members.
2. It provides an implementation of a singleton class pattern
3. Once we provide a constructor that is either private or public or any, the compiler will not add the parameter-less public constructor to the class.

Download Private Constructor files.

## 1.2 Struct Constructors

Structs has only parameterized constructors. The only requirement for a constructor in a struct is that *all the member fields should be initialized* during the construction process.

Download Struct constructor files

# 2 Destructors

Destructors in C# are methods inside the class used to destroy instances of that class when they are no longer needed. The Destructor is called implicitly by the .NET Framework’s Garbage collector and therefore programmer has no control as when to invoke the destructor. An instance variable or an object is eligible for destruction when it is no longer reachable.

1. A Destructor is *unique to its class* i.e. there *cannot be more than one destructor* in a class.
2. A Destructor has *no return type* and has exactly the same name as the class name (Including the same case).
3. It is distinguished apart from a constructor because of the Tilde symbol (**~**) prefixed to its name.
4. A Destructor *does not accept any parameters* and *modifiers*.
5. It cannot be defined in *Structures*. It is only used with *classes*.
6. It *cannot* be *overloaded* or *inherited*.
7. It is called when the program exits.
8. Internally, Destructor called the *Finalize*() on the base class of object.

Download Destructor files