

Section 8: Constructors

10 July 2024 16:18



Harsha Vardhan

- › UI Expert
- › .NET Expert
- › Module Lead
- › Corporate Instructor



Constructors

Agenda

- 1 Understanding Constructors
What is Constructor and its features.
- 2 Rules of Constructors
What should be the constructor name?
Can a constructor receive parameters and return value?
- 3 Instance Constructor (vs) Static Constructor
Constructor that initializes instance fields (vs)
Constructor that initializes static fields.
- 4 Parameter-less Constructor (vs) Parameterized Constructor
Constructor without parameters (vs) Constructor with parameters.

Agenda

- 5 Implicit Constructor (vs) Explicit Constructor
Constructor provided by compiler (vs) Constructor defined by developer.
- 6 Constructor Overloading
Defining multiple constructors within the same class.
- 7 Object Initializer
Initialize fields of an object without constructor.

What

- Special method of class, which contains initialization logic of fields.

```
class Car
{
    string carBrand;
    string carModel;
    int carYear;
```

Declaration of fields

```
public Car( string carBrand, string carModel, int carYear )
{
    this.carBrand = carBrand;
    this.carModel = carModel;
    this.carYear = carYear;
}
```

**Initialization of fields****What**

- Special method of class, which contains initialization logic of fields.
- Constructor initializes the fields and also contains the additional initialization logic (if any).

```
class Car
{
    string carBrand;
    string carModel;
    int carYear;
```

Declaration of fields

```
public Car( string carBrand, string carModel, int carYear )
{
    this.carBrand = carBrand;
    this.carModel = carModel;
    this.carYear = carYear;
}
```

**Initialization of fields****Syntax**

- private
- protected
- private protected
- internal
- protected internal
- public

- static

```
accessModifier modifier ClassName( parameter1, parameter2, ... )
{
    Initialize fields here
}
```





- › Constructor's name should be same as class name.
- › Constructor is recommended to be "public" member or "internal" member;
- › if it is a "private member", it can be called within the same class only; so you can create object of a class only inside the same class; but not outside the class.
- › Constructor can have one or more parameters.
- › Constructor can't return any value; so no return type.
- › A class can have one or more constructors; but all the constructors of the class must have different types of parameters.

Agenda

1

Understanding Constructors

What is Constructor and its features.

2

Rules of Constructors

What should be the constructor name?

Can a constructor receive parameters and return value?

3

Instance Constructor (vs) Static Constructor

Constructor that initializes instance fields (vs)

Constructor that initializes static fields.

4

Parameter-less Constructor (vs) Parameterized Constructor

Constructor without parameters (vs) Constructor with parameters.

Instance Constructor

```
public ClassName( Parameter1, Parameter2, ... )
{
    ...
}
```

Static Constructor

```
static ClassName( )
{
    ...
}
```

- › Initializes instance fields.
- › Executes automatically every time when a new object is created for the class.
- › Initializes static fields.
- › Executes only once, i.e. when first object is created for the class or when the class is accessed for the first time during the execution of Main method.

Instance Constructor

```
public ClassName( Parameter1, Parameter2, ... )
{
    ...
}
```

Static Constructor

```
static ClassName( )
{
    ...
}
```

- › "private" by default;
We can use any of access modifiers.
- › Can contain any initialization logic, that should be executed every time when a new object is created for the class.
- › "public" by default;
Access modifier can't be changed.
- › Can contain any initialization logic, that should be executed only once i.e. when a new object is created for the class.

1

Understanding Constructors

What is Constructor and its features.

2

Rules of Constructors

What should be the constructor name?

Can a constructor receive parameters and return value?

3

Instance Constructor (vs) Static Constructor

Constructor that initializes instance fields (vs)

Constructor that initializes static fields.



4

Parameter-less Constructor (vs) Parameterized Constructor

Constructor without parameters (vs) Constructor with parameters.

Parameter-less Constructor

```
public ClassName()
{
    ...
}
```

- › Constructor without parameters.
- › It generally initializes fields with some literal values (or) contains some general-initialization logic of object.

Parameterized Constructor

```
public ClassName( Parameter1, Parameter2, ... )
{
    ...
}
```

- › Constructor with one or more parameters.
- › It generally initializes fields by assigning values of parameters into fields.

Implicit Constructor (after compilation)

```
public ClassName()
{
}
```

- › If there is a class without constructor, then the constructor automatically provides an empty constructor, while compilation, which initializes nothing. It is called as "Implicit Constructor" or "Default Constructor".
- › It is just to satisfy the rule "Class should have a constructor".

Explicit Constructor (While coding)

```
public ClassName( with or without parameters )
{
    ...
}
```

- › The constructor (parameter-less or parameterized) while is created by the developer is called as "Explicit Constructor".
- › In this case, the C# compiler doesn't provide any implicit constructor.

5

Implicit Constructor (vs) Explicit Constructor

Constructor provided by compiler (vs) Constructor defined by developer.

6

Constructor Overloading

Defining multiple constructors within the same class.

7

Object Initializer

Initialize fields of an object without constructor.

Constructor Overloading**Next: Object Initializer****What**

- › Write multiple constructors with same name in the class, with different set of parameters (just like 'method overloading').
- › It is recommended to write a parameter-less constructor in the class, in case of constructor overloading.

Constructor Overloading (multiple constructors in the same class)

```
public ClassName()
{
}

public ClassName( parameter1, parameter2, ... )
{
    ...
}
```

5

Implicit Constructor (vs) Explicit Constructor

Constructor provided by compiler (vs) Constructor defined by developer.

6

Constructor Overloading

Defining multiple constructors within the same class.

7

Object Initializer

Initialize fields of an object without constructor.

Object Initializer

| Harsha

What

- Special syntax to initialize fields / properties of class, along with creating the object.



- Executes after the constructor.

- It is only for initialization of fields / properties, after creating object; it can't have any initialization logic.

Execution Sequence

new Class() → Constructor → Object Initializer



Object Initializer

`new ClassName() { field1 = value, field2 = value, ... }`

Key Points to Remember

| Harsha

Key points to remember



- 'Instance constructor' initializes 'instance fields'; but also can access 'static fields'.
- 'Static constructor' initializes 'static fields'; can't access 'instance fields'.
- Default (empty constructor) is provided automatically by C# compiler, if the developer creates a class without any constructor.
- It is always recommended to write a parameter-less constructor first, if you are creating parameterized constructor.
- Use 'object initializer', if you want to initialize desired fields of an object, as soon as a new object is created.

