# How to make Optional Parameters in C#

به صورت پیش فرض تمامی پارامتر های مربوط به متد ها در سی شارپ required هستند. اما در نسخه های جدید تر سی شارپ این امکان اضافه شد که پارامتر های آپشنال ایجاد شوند برای متد که در صورتی که آن ها را پاس ندهیم به هنگام اجرا حذف می شوند.

Mandatory نیست یعنی اجباری نیست.

در constractor , delegate , method , indexer قابل استفاده هستند.

**نکته : هر پارامتر آپشنال در سی شارپ دارای یک مقدار Default است که در صورتی که مقداری داده نشود دارای آن است.**

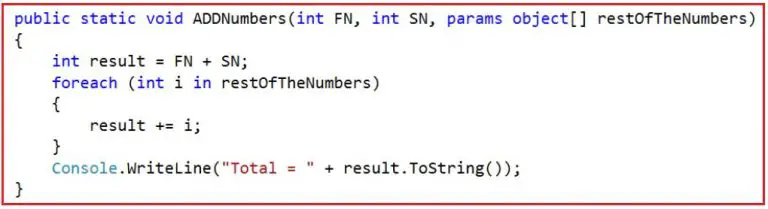
**نکته : پارامتر های آپشنال باید در انتهای لیست پارامتر های تعریف شوند.**

**راه های ایجاد پارمتر آپشنال**

1. **Using Parameter Array**
2. **Method Overloading**
3. **Specify Parameter Defaults**
4. **Using OptionalAttribute**

##### **Using Parameter Array to Make Optional Parameter**

استفاده از params int [] input به عنوان پارمتر



**ADDNumbers(10, 20);**

**ADDNumbers(10, 20, 30, 40, 50);**

**ADDNumbers(10, 20, new object[]{30, 40, 50});**

##### **Using Method Overloading to Make Optional Parameter**

با استفاده از روش overload کردن متد ها نیز می توان پارمتر آپشنال داشت.

##### **Making Method Parameters Optional by Specifying Parameter Defaults**

با تعریف یک پارامتر دارای مقدار پیش فرض یا default value می توان آن را به صورت آپشنال تعریف کرد.

public static void ADDNumbers(int FN, int SN, int[] restOfTheNumbers = null)

{

int result = FN + SN;

//Loop through the restOfTheNumbers only if it is not null

//else we will get runtime error

if (restOfTheNumbers != null)

{

foreach (int i in restOfTheNumbers)

{

result += i;

}

}

##### **Named Parameters in C#:**

با استفاده از یک امکام می توان مقدار پارمتر های متد را با استفاده از نام آن ارسال کرد

public static void Test(int a, int b = 10, int c = 20)

{

Console.WriteLine($"a = {a}, b = {b}, c= {c}");

}

Test(b:1, c: 2, a:10);

Test(1, c: 2)

##### **How to make Parameter Optional Using OptionalAttribute**

با استفاده از attribute که در سیگنیچر پارامتر های متد قرار می گیرد می توانیم یک پارامتر ار آپشنال کنیم.

[optional]

public static void ADDNumbers(int FN, int SN, **[Optional]** int[] restOfTheNumbers)

# Indexers in C#

یک پروپرتی مربوط به کلاس است به متغییر های یک کلاس با ویژگی مانند آرایه ها و با استفاده از indexer دسترسی داشته باشیم.

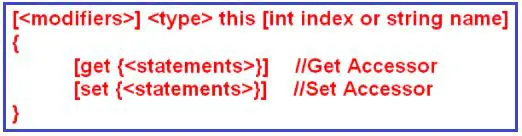
به این معنی است که indexer عضوی از کلاس است که اگر آن را تعریف کنیم class می تواند مثل یک virtual array رفتار کند. این امکان را می دهد که instance ما از کلاس مانند آرایه قابل دسترس باشد

//Accessing Employee Properties using Indexers i.e. using Index positions

Console.WriteLine("EID = " + emp[0]);

Console.WriteLine("Name = " + emp[1]);

اگربه صورت معلول از یک کلاس که دارای پروپرتی های است یک instance ایجاد کنیم و بخواهیم با indexer به اعضای آن دسترسی داشته باشیم با خطا مواجه می شویم به دلیل این که باید indexer را برای کلاس از قبل تعریف کنیم.



1. **Modifiers**: The Modifiers are nothing but the access specifiers such as public, private, protected, internal, etc.
2. **Type**: In our class, as we are dealing with the integer (ID), string (Name, Job, Department, Location, and Gender), and double (Salary) type of values. So, here we need to use the type as an Object because the Object type can return any type of values.
3. **This**: This keyword tells that we are defining an indexer on the current class, in this case, the current class is Employee.
4. **Int index or String name:** The int index or string name is used to specify whether you want to access the values by using its integer index position or by using the string name.
5. **Get and Set:** The get accessor is used for returning the value and the set accessor is used for assigning the value.

طریقه ایجاد indexer برای یک کلاس

public object this[int index]

{

//Get accessor is used for returning a value

get

{

//based in the index position, return the appropriate property value

if (index == 0)

return ID;

else if (index == 1)

return Name;

else if (index == 2)

return Job;

else if (index == 3)

return Salary;

else if (index == 4)

return Location;

else if (index == 5)

return Department;

else if (index == 6)

return Gender;

else

return null;

}

//Set accessor is used to assigning a value

set

{

//based in the index position, set the appropriate property value

if (index == 0)

ID = Convert.ToInt32(value);

else if (index == 1)

Name = value.ToString();

else if (index == 2)

Job = value.ToString();

else if (index == 3)

Salary = Convert.ToDouble(value);

else if (index == 4)

Location = value.ToString();

else if (index == 5)

Department = value.ToString();

else if (index == 6)

Gender = value.ToString();

}

}

برای راحت تر شدن می توان indexer را از نوع string تعریف کرد

emp[0]

emp["ID"]

در حالت String باید پیاده سازی متد تغییر کند مثال بالا تبدیل به String شود

public object this[string Name]

{

//Get accessor is used for returning a value

get

{

//based in the index name, return the appropriate property value

if (Name == "ID")

return ID;

else if (Name == "Name")

return Name;

else if (Name == "Job")

return Job;

else if (Name == "Department")

return Department;

else if (Name == "Gender")

return Gender;

else

return null;

}

//Set accessor is used to assigning a value

set

{

//based in the index name, set the appropriate property value

if (Name == "ID")

ID = Convert.ToInt32(value);

else if (Name == "Name")

Name = value.ToString();

else if (Name == "Job")

Job = value.ToString();

else if (Name == "Salary")

else if (Name == "Department")

Department = value.ToString();

else if (Name == "Gender")

Gender = value.ToString();

}

}

##### **Indexer Overloading in C#:**

می توان با سر بارگذاری متد مربوط به indexer به گونه ای عمل کرد که هم حالت int و حالت string برای کارکردن با indexer مربوط به کلاس امکان پذیر باشد.

# Enums in C#

یک نوع ثابت است که DATA TYPE آن مشخص است strongly type named constant

فرض کنید جنسیت را با عدد int مشخص کرده اید مرد 1 و زن 2 و نامشخص 0 آیا این داده ها در حالت معمول قابل فهم اند ؟ پس بنابراین میتوانیم از enum در این مواقع استفاده کنیم.

public enum Gender

{

Unknown,

Male,

Female

}

نکات:

قابل enumerate هستند.

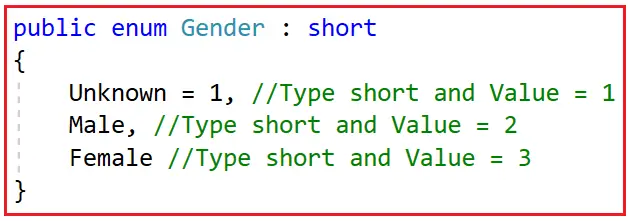
نیاز به explicit cast دارد.

به صورت پیش فرض مقدار int دارند.

مقدار اولین عنصر به صورت پیش فرض 0 است.

امکان تغییر نوع دارد

به صورت value type است.



امکان تبدیل صریح بین enum ها وجود دارد.

// The following line compiles as we have an explicit cast

Gender gender = (Gender)Season.Winter;

//Gender Enum

public enum Gender : int

{

Unknown = 1,

Male = 2,

Female = 3

}

//Season Enum

public enum Season : int

{

Winter = 1,

Spring = 2,

Summer = 3

}

##### **Understanding GetValues() and GetNames() Method**

getValue برای بازیابی یک آرایه از مقادیر ثابت که قابل پیمایش است کاربرد دارد.

getNames برای بازیابی یک ارایه از نام های enum به صورت قابل پیمایش کاربرد دارد.

**نکته : کلاس Enum به مانند Enumerable به صورت انفرادی قابل استفاده است و این متد ها را می دهد**

int[] EnumValues = (int[])Enum.GetValues(typeof(Gender));

آیا امکان ارث بری در enum ها وجود دارد ؟

خیر امکان ارث بری در enum وجود ندارد.

هیچ کلاسی نمی تواند از enum ارث ببرد

هیچ مقدار تکرای نمی تواند در enum وجود داشته باشد.