آموزش سی شارپ (C#) قسمت ۱۵ : آشنایی با متدها یک (نسخه PDF)

با سلام مجدد به دوستان عزیزم در انجمن تخصصی فناوری اطلاعات ایران. در ادامه مباحث آموزشی زبان سی شارپ به مبحث متدها و شیوه استفاده از آنها خواهیم پرداخت. اما متدها چه استفاده ای در زبان سی شارپ دارند؟ زمانی که ما برنامه ای را می نویسیم با زیاد شدن حجم کد نوشته شده، میزان خوانایی و درک از کد نوشته شده کاهش میابد. فرض کنید برنامه ای در کنسول نوشتیم که تعداد خطوط کد بالای ۵۰۰۰ خط بوده و کل کدهای آن را در متد Main که در بخش های ابتدایی این سری آموشی با آن آشنا شدیم نوشته ایم. پس از مدتی، قصد داریم قسمتی از کد را تغییر دهیم یا امکان جدیدی به برنامه اضافه کنیم. صد در صد کار نگهداری و خطایابی برنامه ای که نوشته ایم مشکل خواهد بود. زیرا تمامی کدها در یکجا نوشته شده و برنامه نویس برای پیدا کردن و مدیریت کدها به مشکل بر خواهد خورد. برای حل این مشکل باید کدها را از لحاظ کاربردی به قسمت های مختلف تقسیم کنیم. این تقسیم بندی با کمک متدها انجام می شوند. در حقیقت متدها قطعه کدهایی هستند که هر یک، وظیفه ای مشخص را در برنامه ما انجام می دهند. در این قسمت هدف آشنایی با کاربرد متدها می باشد که با مباحث زیر آشنا خواهیم شد:

۱. شیوه تعریف متدها
۲. پارامترها و مقادیر بازگشتی
۳. آشنایی با عبارات ref و out
۴. آشنایی با مفهوم method overloading
۵. نوشتن توابع بازگشتی (Recursive Functions)

شیوه تعریف متدها

در قسمت های قبلی، با متد Main که نقطه شروع برنامه های سی شارپ می باشد آشنا شدیم. در زبان سی شارپ ما می توانیم متدهای مورد نظر خودمان را ایجاد کنیم. زمان تعریف متدها به این نکته باید توجه داشت که هر متد باید یک وظیفه خاص را انجام دهد. در قسمت زیر با شیوه نوشتن متدها آشنا می شویم:

```
{modifier} [static] {return-type} {name}({parameters})
{
   // method body
}
```

۱. {modifier}: این بخش نمایانگر سطح دسترسی به متد می باشد. زمانی که ما کدی می نویسیم باید سطح دسترسی به آن قطعه کد را مشخص کنیم. برای مثال متد Main سطح دسترسی public دارد، یعنی از هر نقطه ای از برنامه می توان به آن دسترسی داشت. سطوح دسترسی به شیوه های مختلف تعریف می شوند که در قسمت های بعدی با آنها آشنا می شویم. در حال حاضر برای تمام متدها از کلمه کلیدی public یا عمومی استفاده می کنیم.
۲. [static]: در زبان سی شارپ، ما مفهومی داریم به نام کلاس ها که اشیاء از روی آنها ساخته می شوند، زمانی که ما متدی را با کلمه static مشخص می کنیم، می گوییم این متد خارج از نمونه های ساخته شده از آن کلاس قابل استفاده باشند. نوشتن کلمه کلیدی static برای متدها اختیاری می باشد. توجه داشته باشید. در حال حاضر ما تمامی متدهایی که می نویسیم زیر متد Main تعریف می شوند و باید از نوع static باشند، زیرا تنها متدهای static را می توان از داخل متد Main صدا زد.
۳. {return-type}: در این قسمت ما باید نوع بازگشتی متد را مشخص کنیم. متدها می توانند نوع بازگشتی داشته باشند یا نداشته باشند. برای مثال زمانی که ما متدی می نویسیم که نوع بازگشتی آن string است، جای {return-type} کلمه کلیدی string را می نویسیم. اگر متد نوع بازگشتی نداشته باشد از کلمه void برای مقدار بازگشتی استفاده می کنیم. مقدار مورد نظر داخل بدنه متد با کلمه کلیدی return برگردانده می شود. متدهایی که نوع بازگشتی دارند، باید داخل بدنه مقداری را برگردانند.
۴. {name}: در این قسمت نام متد را مشخص می کنیم، نام متد هر نامی می تواند باشد، اما قواعد نامگذاری متغیرها در قسمت متدها نیز صدق می کنند.

۵. {parameters}: در این قسمت ما پارامترهای ورودی متدها را مشخص می کنیم. یک متد می تواند شامل یک یا چندین پارامتر ورودی باشد یا اصلاً ورودی نداشته باشد.
۶. بدنه متد، مجموعه کدهایی است که با صدا زدن نام آن متد می خواهیم اجرا شوند، این کدها باید بین علامت های { و } قرار گیرند.

در قسمت زیر ما یک متد ساده که یک پیام خوش آمدگویی را در پنجره کنسول نمایش می دهد تعریف می کنیم. این متد مقدار بازگشتی نداشته و پارامتری نیز به عنوان ورودی نمیگیرد:

```
private static void Main(string[] args)
{
    DisplayMessage();
}

public static void DisplayMessage()
{
    Console.WriteLine("Welcome to C# Course...");
}
```

در نمونه کد بالا، متدی با نام DisplayMessage تعریف کردیم که داخل آن پیغامی بر روی صفحه کنسول نمایش داده می شود. در داخل متد Main هم متد DisplayMessage را فراخوانی کردیم. دقت کنید که برای فراخوانی متد باید پس از نام متد حتماً () را قرا دهیم. در این متد به دلیل اینکه ما پارامتری برای متد تعریف نکردیم داخل پرانتز هیچ چیز نمی نویسیم، اما اگر پارامتری تعریف شده بود، داخل پرانتز باید مقادیر هر پارامتر را مشخص می کردیم که در قسمت بعدی شیوه تعریف پارامترها را با هم بررسی می کنیم.

نحوه استفاده پارامتر ها در متدها

همانطور که در قسمت قبلی گفتیم، می توان برای هر متد یک یا چند پارامتر ورودی مشخص کرد و سپس داخل بدنه متد از پارامترها استفاده کرد. برای تعریف پارامترها، زمانی که نام متد را مشخص می کنیم داخل پرانتز باید لیست پارامترها را بنویسیم. این لیست بوسیله علامت , جدا شده و هر پارامتر شامل نوع پارامتر که می تواند یکی از انواع داده های اولیه یا کلاس های نوشته شده باشد و سپس نام پارامتر. برای آشنایی بیشتر به مثال زیر توجه کنید:

```
private static void Main(string[] args)
{
    DisplayMessage("Hossein","Ahmadi");
}

public static void DisplayMessage(string firstName, string lastName)
{
    Console.WriteLine("Welcome dear " + firstName + " " + lastName);
}
```

در نمونه کد بالا متدی تعریف کردیم با نام DisplayMessage که دو پارامتر از نوع string با نام های firstName و lastName به عنوان ورودی دریافت می کند. سپس داخل بدنه متد، پیغامی حاوی نام و نام خانوادگی که به عنوان پارامتر پاس داده شده است را در خروجی نمایش می دهیم.

برای فراخوانی متد، در متد Main داخل پرانتز پس از نام متد باید لیست مقادیر مورد نظر برای هر پارامتر را بنویسیم. دقت کنید که ترتیب نوشتن مقادیر برای پارامترها با لیست تعریف پارامترها باید یکسان باشد. همچنین باید تعداد مقادیر ورودی هنگام فراخوانی متد

با تعـداد پارامترهای مشـخص شده برای آن یکسان باشد. در غیر اینصورت با پیغام خطا مواجه خواهید شد. البته راهیی وجود دارد که از نوشتن مقادیر برای برخی پارامترها خودداری کرد و آن استفاده از مقادیر پیش فرض برای پارامترها می باشد.

مقادیر پیش فرض برای پارامترها

زمانی که قصـد تعریف کردن متدی را داریم، می توانیم برای پارامترهای آن متد مقادیر اولیه ای را مشـخص کنیم. برای آشـنایی بیشتر به مثال زیر توجه کنید:

```
private static void Main(string[] args)
{
   DisplayMessage("Hossein");
}

public static void DisplayMessage(string firstName, string lastName = "Ahmadi")
{
   Console.WriteLine("Welcome dear " + firstName + " " + lastName);
}
```

در مثال بالا، هنگام تعریف متد DisplayMessage برای پارامتر lastName مقدار پیش فرض Ahmadi را مشخص کردیم. با این کار می توانیم از نوشتن مقدار پارامتر lastName هنگام فراخوانی متد خودداری کنیم. در حقیقت نوشتن مقدار برای این پارامتر دلخواه بوده و در صورت نوشـته نشدن مقدار پیش فرض جایگزین خواهد شد. دقت کنید که زمان مشـخص کردن مقادیر پیش فرض، تنها انتهایی ترین پارامترها می توانند مقدار پیش فرض داشته باشند، در غیر اینصورت پیغام خطا دریافت خواهد کرد. به عنوان مثال:

```
public static void Test(int a, int b = 12, int c)
{
}
```

کد بالا صحیح نمی باشد، زیرا علاوه بر پارامتر b باید پارامتر c نیز مقدار اولیه داشته باشد تا پیغام خطا دریافت نکنیم.

مقدار دهی پارامتر ها با استفاده از نام

هنگـامی که متـدی با چنـدین پارامتر تعریف میکنیم، می توان مقـدار دهی پارامترها را بر اساس نام پارامتر انجام داد. متـد زیر را در نظر بگیرید:

```
public static void Test(int a, int b, int c, int d)
{
}
```

برای مقدار دهی پارامترها بر اساس نام هنگام فراخوانی به روش زیر عمل می کنیم:

```
Test(b: 12, d: 1, a: 18, c: 9);
```

همانطور که ملاحظه می کنید، زمانی که از نام پارامتر برای مقدار دهی آن استفاده می کنید، ترتیب نوشتن دیگر مهم نیست.

مقادیر بازگشتی از متدها

یکی از موارد اسـتفاده از متدها، گرفتن خروجی از یک متد است. یعنی متد بعد از انجام یکسـری کارها یک خروجی برای ما بر میگرداند. برای اسـتفاده از مقادیر خروجی، بایـد ابتدا نوع خروجی متد را هنگام تعریف آن مشـخص کرد. تا این لحظه متدهایی که نوشتیم مقادیر بازگشتی آنها از نوع void بود، یعنی مقـدار بازگشتی نداشـتند. اما فرض کنید می خواهیم متدی بنویسـیم که دو عدد را به عنوان ورودی دریافت کرده و به عنوان خروجی حاصل جمع آنها را برگرداند. برای اینکار نیاز به یک مقدار خروجی داریم. به مثال زیر توجه کنید:

```
public static int Sum(int n1, int n2)
{
    var result = n1 + n2;
    return result;
}
```

به تعریف بالا دقت کنید، به جای کلمه void از نوع int اسـتفاده کردیم که نشان دهنده نوع بازگشتی متد ما می باشد. اما نکته دیگری نیز وجود دارد که بایـد به آن توجه کنیم. زمانی که متـد ما مقـدار بازگشتی دارد، بایـد جایی داخل بـدنه متـد، مقداری را با کلمه کلید return بـازگردانیم. این مقـدار به عنوان خروجی متـد در نظر گرفته خواهـد شـد. در صورتی که متـد ما مقـدار بازگشتی داشـته و جایی از return استفاده نکنیم با پیغام خطا مواجه خواهیم شد. برای فراخوانی متد Sum و استفاده از نتیجه بازگشتی آن به صورت زیر عمل می کنیم:

```
Console.WriteLine(Sum(2,4));
```

کد بالا، در خروجی مقدار ۶ که حاصل جمع ۴ و ۲ می باشد را چاپ خواهد کرد.

برای آشنایی بیشتر با مقادیر بازگشتی، مثال دیگری را با هم بررسی می کنیم. در این مثال متدی خواهیم نوشت که حقوق و درصد مالیات را به عنوان ورودی دریافت کرده و مقدار مالیات را بر می گرداند:

```
public static decimal CalculateTax(decimal salary, byte tax)
{
    return (salary / 100) * tax;
}
```

و سپس می توانیم به صورت زیر کد بالا را فراخوانی کنیم:

```
Console.WriteLine(CalculateTax(2500000, 5));
```

تا این جای کار با مقدمات تعریف متدها، پارامترها و مقادیر بازگشتی متدها آشـنا شدیم. در بخش دوم از قسمت دهم سری آموزشی سی شارپ با کلمات کلیدی out و ref، مفهوم method overloading و توابع بازگشتی آشنا خواهیم شدیم. ITPRO باشید

نویسنده : حسین احمدی

منبع : جزیره برنامه نویسی وب سایت توسینسو

foz

آقا دمت گرم چقدر برای دانشگاهم دنبال این متد ها گشتم بلخره پیداش کردم خیلی ممنونم فقط یه مشکلی هست من مثالی از این متد ها نمیبینم یعنی مثالی ک جنبه امتحانی داشته باشه میخواستم ببینم از کجا میتونم پیدا کنم ؟

حسین احمدی

منظورتون از مثال امتحانی چی هست؟ اگر چیزی مد نظرتون هست در قالب یک سوال ارسال کنید بنده در خدمتتون هستم.

amir_۷۸m

سلام ممنون بابت آموزشهای عالی و مفید شما

اگر میشه دوره آموزشی فرم با سی شارپ هم قرار دهید

تشکر.

در ضـمن من متد فاکتوریل رو نوشـتم البته به شـکلی دیگ و میخوام در آخر برنامه از دسـتور readkey اسـتفاده کنم که برنامه با کلید کیبورد بسته شه ولی وقتی اینکارو میکنم کامپایلر خطا میده میشه مشکلشو بگید ؟!

```
;(()Console.WriteLine(fact

{

(public static int fact(int num = ۱

}

;(" : Console.WriteLine("Enter a Value

;(()int a = int.Parse(Console.ReadLine

(++for (int i = ۱; i <= a; i

;num *= i

;return num

Console.ReadKey(); //Error

{
```

حسین احمدی

سلامـ و عرض ادب، ممنون از شـما. دوره آموزشـی فرم هـا به صورت ویـدیویی داخل سایت هست و آموزش متنی راجع به Windows Forms نداریم.

مطلب اصلی