آموزش برنامه نویسی شی گرا در سی شارپ (C#) قسمت ۲ : کلاس و شئ (نسخه PDF)

در قسمت قبلی آموزش زبان سی شارپ، به بررسی مفاهیم اولیه برنامه نویسی شئ گرا پرداختیم. در ادامه، بر اساس مفاهیم گفته شده در قسمت قبل، به صورت عملی با نحوه تعریف کلاس ها، ایجاد اشیاء از روی کلاس ها و همچنین نحوه تعریف فیلد، خصوصیت و رفتارها برای اشیاء آشنا خواهیم شد.همانطور که در قسمت قبل گفتیم، زمانی که قصد نوشتن برنامه ای به صورت شئ گرا را داریم، باید موجودیت های مورد استفاده را در برنامه مدل سازی کنیم. این موجودیت ها همان اشیاء هستند که در سیستم مورد استفاده قرار میگیرند. اما شیوه مدل سازی و استفاده از اشیاء چگونه خواهد بود؟ در اینجا باید با دو مفهوم آشنا شویم: ۱. کلاس ها و ۲. اشیاء.

۱. کلاس: نمونه ای از یک شئ که داخل برنامه طراحی می شود را کلاس می گویند. برای اینکه با مفهوم کلاس بیشتر آشنا شوید یک مثال از دنیای واقعی می زنیم. فرض کنید تصمیم به ساخت یک خانه دارید. اولین چیزی که به آن نیاز خواهید داشت نقشه خانه ایست که تصمیم دارید بسازید. نقشه یک طرح اولیه و مفهومی از ساختمان به شما می دهد و بعد از روی نقشه اقدام به ساخت خانه می کنید. نقشه شامل تمامی بخش های خانه است، اطاق پذیرایی، آشپزخانه، حمام، سرویس بهداشتی و سایر بخش ها. اما فقط یک نقشه در اختیار دارید. نمی توانید از اطاق پذیرایی داخل نقشه استفاده کنید. کلاس دقیقاً معادل نقشه ای است که شما برای ساختمان خود کشیده اید. کلاس یک نمونه اولیه از موجودیت ایست که باید اشیاء از روی آن ساخته شوند.
۲. شئ: باز هم به سراغ مثال قبلی می رویم. بعد از کشیدن نقشه ساختمان شما باید اقدام به ساخت خانه کنید.

بعد از اتمام عملیات ساخت، خانه شما قابل سکونت بوده و شما می توانید از آن استفاده کنید. همچنین از روی یک نقشه ساختمانی می توان چندین ساختمان ساخت. شئ دقیقاً معادل همان مفهوم ساختمانی است که از روی نقشه ساخته شده است. شما بعد از اینکه کلاس را تعریف کردید، باید از روی کلاس شئ بسازید تا بتوانید از آن استفاده کنید. در حقیقت کلاس به صورت مستقیم قابل استفاده نیست، مگر اینکه شامل اعضای static باشد که در بخش های بعدی با آنها آشنا خواهیم شد. همچنین می توان از روی یک کلاس، یک یا چندین شئ تعریف کرد.حال که با مفاهیم اولیه کلاس و شئ آشنا شدید، بهتر است با نحوه تعریف کلاس و ساخت شئ آشنا شویم. تعریف کلاس بوسیله کلمه کلیدی class در زبان C# انجام می شود. ساختار کلی این دستور به صورت زیر است:

```
{access-modifier} class {name}
{
}
```

قسمت access-modifier سطح دسترسی به کلاس را تعیین می کند. ما زمانی که اقدام به تعریف کلاس یا هر قطعه کدی در زبان c# می کنیم، می توانیم سطح دسترسی به آن کد را تعیین کنیم. اما سطح دسترسی به چه معناست؟ در قسمت های اولیه آموزش گفتیم که زمان ایجاد یک پروژه به زبان C#، برای شما یک solution ایجاد شده که هر solution می تواند شامل چندین پروژه باشد. برای مثال، کلاس یا اعضای یک کلاس را تعریف می کنیم، می توانیم مشخص کنیم که این کلاس از کدام قسمت های پروژه قابل دسترس باشد. سطوح دسترسی زیر در زبان سی شارپ تعریف شده اند:

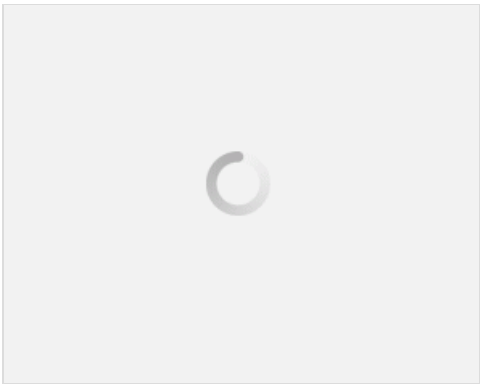
۱. private: این سطح دسترسی مشخص می کند که قطعه کد تعریف شده تنها داخل خود پروژه یا Scope مربوطه قابل دسترس باشند. برای مثال کلاسی که به صورت private تعریف شده باشد، تنها داخل همان پروژه قابل دسترس بوده و از سایر پروژه هایی که در solution تعریف شده قابل دسترس نخواهد بود، یا اعضای کلاسی که به صورت private تعریف شده اند، تنها در Scope همان کلاس که بین علامت های {} می باشد قابل دسترس خواهند بود.
۲. public: کدهایی که با این سطح دسترسی مشخص شده باشند، در تمامی قسمت های پروژه و سایر پروژه ها قابل دسترس خواهند بود.
۳. internal: سطوح دسترسی internal، تنها داخل همان پروژه قابل دسترس بوده و سایر پروژه ها به آنها دسترسی نخواهند داشت. این سطح دسترسی برای اعضای کلاس ها کاربرد زیادی دارد.
۴. protected: این سطح دسترسی زمانی که از مفهوم inheritance استفاده کنیم کاربرد دارد. در قسمت وراثت این سطح دسترسی را به تفصیل مورد بررسی قرار خواهیم داد.
۵. internal protected: همانند قسمت protected، این دسترسی نیز در قسمت وراثت توضیح داده خواهد شد که تفاوتی از دسترسی

۵. internal protected: ترکیبی از دسترسی protected و internal. این دسترسی نیز در کلاس وارث توضیح داده خواهد شد که تلفیقی از دسترسی های internal و protected می باشد.

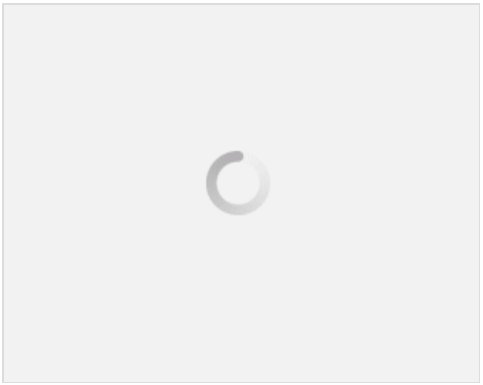
بعد از access-modifier، با کلمه کلیدی class می گوییم که قصد تعریف یک کلاس را داریم و بعد از کلمه کلیدی class در قسمت name نام کلاس را مشخص می کنیم. نام کلاس باید همیشه بر اساس قاعده PascalCase نام گذاری شود. ما دو شیوه نام گذاری داریم:

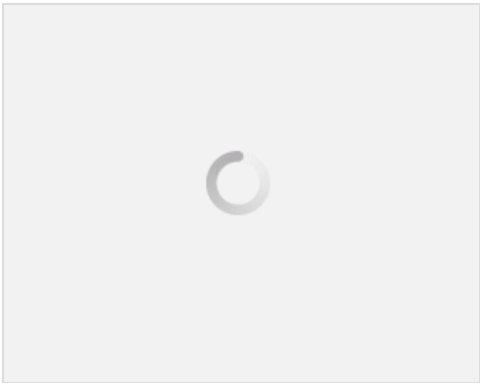
۱. camelCase: در این شیوه نام گذاری، کاراکتر ابتدای هر کلمه باید با حروف بزرگ نوشته شود غیر از کلمه اول. مانند: newEmployee، sampleDictionary.
۲. PascalCase: در این شیوه نام گذاری، کاراکتر ابتدای هر کلمه باید با حروف بزرگ نوشته شود، مانند: SampleDictionary، NewEmployee

حال، تصمیم داریم یک کلاس با نام Person تعریف کنیم. در قسمت های بعدی به این کلاس خصوصیات و رفتارهای مورد نظر را اضافه خواهیم کرد. برای تعریف کلاس، بر روی نام پروژه در پنجره Solution Explorer، با موس راست کلیک کرده و از منوی ظاهر شده از قسمت گزینه Class... را انتخاب می کنیم:

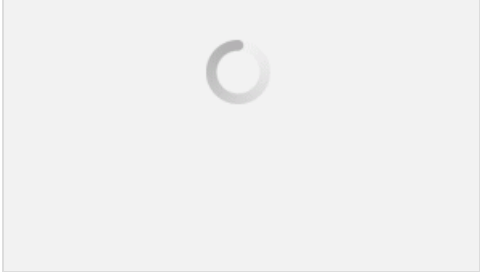


بعد از انتخاب این گزینه، نام کلاس مورد نظر را در پنجره Add New Item وارد کرده و روی دکمه Add کلیک می کنیم. در اینجا نام Person را وارد می کنیم. بعد از انجام این کار، فایل جدیدی با نام Person.cs به پروژه ما اضافه می شود:

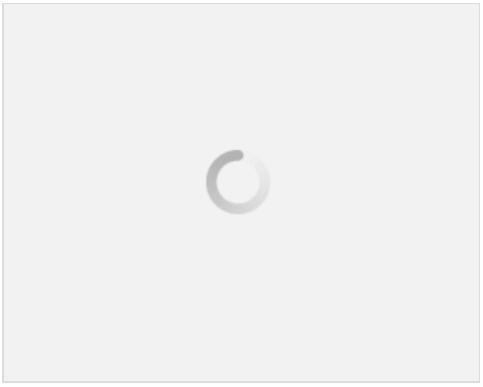




اگر بر روی فایل Person.cs دوبار کلیک کنیم، محتویات فایل مورد نظر به صورت زیر نمایش داده خواهد شد:



نکته: در قسـمت معرفی ابزارهای این مجموعه آموزشـی، درباره ابزاری به نام Resharper صـحبت کردیم. در صورتی که این ابزار را نصـب کرده باشـید، برای تعریف کلاس جدیـد، کافیست پروژه ای که قصـد تعریف کلاس داخل آن را دریـد، در پنجره Solution Explorer انتخاب کرده و کلیدهای Alt+Insert را فشار دهید. با اینکار منوی زیر نمایش داده می شود:



بعد از انتخاب گزینه کلاس از منوی ظاهر شده، نام کلاس از شما پرسیده شده و کلاس به پروژه شما اضافه می شود.

به سراغ محتویات فایل اضافه شده برویم:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpOOP
{
    class Person
    {
    }
}
```

قسمت using مربوط به استفاده کلاس هایی است که در namespace های دیگر تعریف شده اند. namespace ها برای دسته بندی کدهای پروژه مورد استفاده قرار میگیرند، در حقیقت شما می توانید از لحاظ کاربردی کدهای خود را در زبان سی شارپ بوسیله namespace تقسیم بنـدی کنید. برای مثال، در کد بالا، کلاس Person، در namespace یا فضای نام CSharpOOP تعریف شـده است. زمانی که پروژه ای ایجاد می کنید، فضای نام پیش فرض بر اساس نام پروژه ایجاد شده و تمام کدهای شما داخل این فضای نام تعریف خواهند شد. همچنین کلیه کلاـس هایی که به صورت پیش فرض در دات نت تعریف شـده انـد، در فضای نام System قرار دارنـد. در حقیقت System فضای نام پایه برای کلیه کلاـس های موجود در دات نت می باشـد. همچنین می توان برای هر فضای نام یک فضای نام زیر مجموعه تعریف کرد که این جـداسازی بوسـیله کـاراکتر . انجـام می شود. برای مثـال، برای فضـای نام CSharpOOP می خواهیم یـک فضـای نـام زیر مجموعه بـا نام DataTools تعریف کنیم:

```
namespace CSharpOOP.DataTools
{
}
```

به کد کلاس Person برگردیم. در ادامه کد، فضای نام CSharpOOP مشخص شده که داخل آن کلاس Person تعریف شده است. اگر دقت کنید، این کلاس access-modifier ندارد. کدهایی که برای آنها access-modifier مشخص نشده باشد، به صورت پیش فرض private در نظر گرفته می شوند. بعد از تعریف کلاس بوسیله {} محدوده کلاس مشخص شده است که کدهای مربوط به کلاس داخل آن نوشته می شوند.

خوب تا اینجا، ما با شـیوه تعریف یک کلاس ساده آشـنا شدیم. در مرحله بعد، باید از روی این کلاس یک شئ بسازیم. ساختار کلی تعریف شئ به صورت زیر است:

```
{class-name} {object-name} = new {class-name}();
```

در قسمت class-name، نام کلاس را مشـخص می کنیم، برای مثال Person و در قسمت object-name، نام شئ مورد نظر را مشخص می کنیم. در حقیقت object-name یک متغیر است که به شـئ ما اشاره می کند. بعد از علامت انتساب یا = باید عملیات ساخت شئ را انجام دهیم. بوسیله کلمه کلیدی new می گوییم که تصـمیم به ساخت یک شـئ جدید داریم و در مقال آن نام کلاسی که می خواهیم از روی آن شئ بسازیم را می نویسـیم. دقت کنید که بعد از نوشتن نام کلاس در مقال کلمه کلیدی new باید () حتماً نوشته شود، در غیر اینصورت با پیغام خطا مواجه خواهید شد. با توضیحات بالا، می توان گفت عملیات ساخت شئ در دو مرحله انجام می شود:

۱. تعریف متغیری که شئ داخل آن نگهداری می شود (دستورات قبل از عملیات انتساب).
۲. ساخت شئ و قرار دادن آن داخل متغیر مربوطه (دستورات بعد از عملیات انتساب).

حال از روی کلاس Person یک شئ ایجاد می کنیم. کد متد Main در فایل Program.cs را به صورت زیر تغییر دهید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpOOP
{
    class Program
    {
        static void Main(string[] args)
        {
            Person person = new Person();
        }
    }
}
```

بوسـیله کـد بالا، عملیات ساخت شـئ انجام شـد. همانطور که قبلا گفتیم، ما می توانیم چندین شـئ با نام های متفاوت از روی یک کلاس ایجاد کنیم:

```
Person person1 = new Person();
Person person2 = new Person();
```

```
Person person2 = new Person();
Person person3 = new Person();
```

دقت کنید، کلاس Program در فایل Program.cs نیز داخل فضای نام CSharpOOP قرار دارد. شـما می توانید در فایل های متفاوت فضای نام همنام داشـته باشـید، بدین معنی که کلیه کدها در همان فضای نام قرار خواهند گرفت. در صورتی که شـما در متد Main نام فضای نام CSharpOOP را تایپ کنید و پس از آن کلید . را بزنید، لیستی که از محتویات آن فضای نام برای شما نمایش داده خواهد شد:

اما فرض کنید، کد ما در فایل Person.cs، در فضای نام دیگری با نام CSharpOOP.Entities تعریف شده بود:

```
namespace CSharpOOP.Entities
{
    class Person
    {
    }
}
```

در این حالت، زمانی که شما در فایل Program.cs و متد Main، تصمیم دارید از روی کلاس Person شئ بسازید، باید آدرس کامل فضای نام را نیز هنگام ساخت شئ مشخص کنید، زیرا فضای نام کلاس های Program و Person دیگر یکسان نیستند:

```
CSharpOOP.Entities.Person person = new CSharpOOP.Entities.Person();
```

اما در اینجا نکته ای وجود دارد، چون ابتـدای فضای نام کلاس های Program و کلاس Person یکسان می باشد، یعنی فضای نام Entities زیر مجموعه CSharpOOP قرار دارد و کلاس Program نیز در فضای نام CSharpOOP تعریف شده، می توان از نوشـتن قسمت اول فضای نام یعنی CSharpOOP خودداری کرد:

```
Entities.Person person = new Entities.Person();
```

دقت کنیـد، اگر فضـای نـام را برای ایجاد شـئ ننویسـیم، با پیغام خطا مواجه خواهیم شـد. اما راهی وجود دارد که آدرس کامل کلاس را ننویسـیم، برای اینکار از دستور using اسـتفاده می کنیم که در بالا نیز به آن اشاره شـد. دسـتور using کلیه کـدهای داخل یک فضای نام را داخل فضای نام جاری قابل دسترس می کند. برای مثال بالا، کافیست در قسمت using فایل Program.cs، دستور زیر را بنویسیم:

```
using CSharpOOP.Entities;
```

با نوشتن دستور بالا، دیگر نیازی به نوشتن آدرس فضای نام هنگام ساخت شئ نخواهد بود. نمونه دیگر استفاده از دستور using، استفاده از دستورات کلاس Console می باشد که در قسمت های قبل با آن زیاد کار کردیم. کلاس Console داخل فضای نام System که فضای نام پایه کلیه کلاس های دات نت می باشـد تعریف شـده. اما بـدلیل اینکه در ابتـدای فایل Program.cs دستور using System; نوشته شده است،

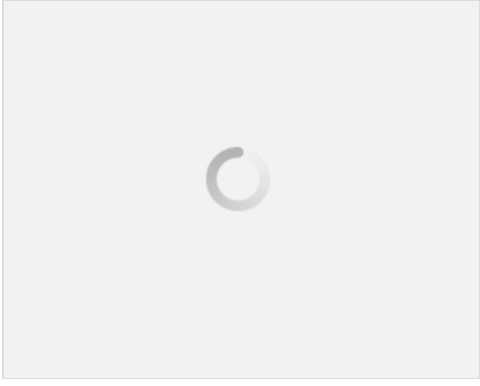
کافیست تنها نام کلاس Console را بنویسیم و نیازی به نوشتن آدرس کامل آن به صورت System.Console نمی باشد.

شما می توانید داخل یک فایل چندین کلاس را تعریف کنید. برای مثال، در فایل Program.cs می توانید بعد از اتمام کد کلاس Program.cs، اقدام به تعریف کلاس Person نمایید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CSharpOOP.Entities;

namespace CSharpOOP
{
    class Program
    {
        static void Main(string[] args)
        {
            Entities.Person person = new Entities.Person();
        }
    }

    class Person
    {

    }
}
```

اما بهتر است برای هر کلاس، یک فایل جداگانه در نظر بگیرید تا ساختار مناسب برای پروژه ای که تصمیم به انجام آن دارید حفظ شود.

یکی دیگر از قابلیت های موجود در Solution Explorer، قابلیت پوشه بندی فایل ها داخل پروژه می باشد. برای مثال، می توانید کلاس های مربوط به موجودیت های برنامه را داخل یک پوشه قرار دهید. برای اینکار، بر روی پروژه راست کلیک کرده، از قسمت Add گزینه New Folder را انتخاب کنید. با اینکار پوشه جدیدی به پروژه شما اضافه می شود که می توانید برای آن یک نام دلخواه انتخاب کنید:



در صورتی که ابزار Resharper را نصب کرده باشید، با زدن کلید های Alt+Insert بر روی پروژه داخل Solution Explorer از منوی ظاهر شده گزینه New Folder را برای افزودن پوشه جدید انتخاب کنید.

پس از تعریف پوشه، با انتخاب آن و تکرار مراحل قبل برای ایجاد کلاس، می توانید داخل آن پوشه یک فایل جدید ایجاد کنید. برای

پس از تعریف پوشه، با انتخاب آن و تکرار مراحل قبلی برای ایجاد کلاس، می توانید داخل آن پوشه یک فایل جدید ایجاد کنید. برای مثال، پوشه ای با نام Entities داخل پروژه تعریف کرده و کلاسی با نام Car داخل آن تعریف کنید. بعد از اینکار محتویات فایل شما به صورت زیر خواهد بود:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpOOP.Entities
{
    public class Car
    {

    }
}
```

به یک نکته توجه کنید که فضای نام یا namespace کلاس Car به صورت خودکار CSharpOOP.Entities انتخاب شده است، زیرا کلاس داخل پوشه Entities که در پروژه CSharpOOP قرار دارد اضافه شده. در صورتی که شما داخل یک پوشه، پوشه جدیدی اضافه کرده و داخل آن یک کلاس اضافه کنید، آدرس فضای نام مبتنی بر نام آن پوشه انتخاب خواهد شد. پس نکته بعدی که باید مد نظر داشته باشید، زمانی که قصد دارید کدهای خود را بوسیله فضاهای نام دسته بندی کنید، حدالامکان برای آنها پوشه ایجاد کنید، اجباری به اینکار نیست، اما برای حفظ ساختار و نظم پروژه اینکار توصیه می شود.

در این قسمت از سری آموزش سی شارپ، با مفاهیم کلاس، شئ، فضاهای نام، دستور using و پوشه بندی فایل ها داخل پروژه آشنا شدیم. در قسمت بعدی آموزش با نحوه تعریف خصوصیت و رفتار برای کلاس ها و شیوه استفاده از آنها بوسیله اشیاء ساخته شده آشنا خواهیم شد.

نویسنده : حسین احمدی

منبع : جزیره برنامه نویسی وب سایت توسینسو

هرگونه نشر و کپی برداری بدون ذکر منبع و نام نویسنده دارای اشکال اخلاقی است

#آموزش_زبان_سی_شارپ #برنامه_نویسی_شئ_گرا #تعریف_کلاس_و_شئ_در_زبان_سی_شارپ #شئ_در_برنامه_نویسی_شئ_گرا #دستور_using_در_سی_شارپ #namespace_در_سی_شارپ #فضاهای_نام_در_سی_شارپ #آموزش_برنامه_نویسی #کلاس_در_برنامه_نویسی_شئ_گرا #زبان_برنامه_نویسی_سی_شارپ

شهاب نوری گودرزی

سلام

تفاوت internal با private چی میشه؟

tivaproject

سلام. به طور معمول وقتی پروژه ای می سازیم یک namespace به نام آن پروژه ساخته میشود و از این به بعد تمام کلاسها و namespace های جدید در قالب آن فضای نام پروژه قرار می گیرد. حال اگر کلاسی را مستقیما در درون یک فضای نام بسازیم (نه در قالب کلاس دیگر) و سطح دسترسی private، protected و یا internal protected به آن بدهیم با خطا مواجه می شویم زیرا اصولا سطح

قالب کلاسی دیگر) و سطح دسترسی private، protected و یا internal protected به آن بدهیم با خطا مواجه می شویم زیرا اصولا سطح دسترسی private برای کلاس داخل فضای نام مفهوم ندارد.

در واقع کلاسی که مستقیما داخل یک فضای نام تعریف می شود یک نوع سطح بالا یا top level type است و فقط می تواند public یا internal باشد.

sanaz۷۳

namespace CSharpOOP.DataTools

بعد از CsharpOOp به چه معنی

و برای چی استفاده شده؟

مطلب اصلی