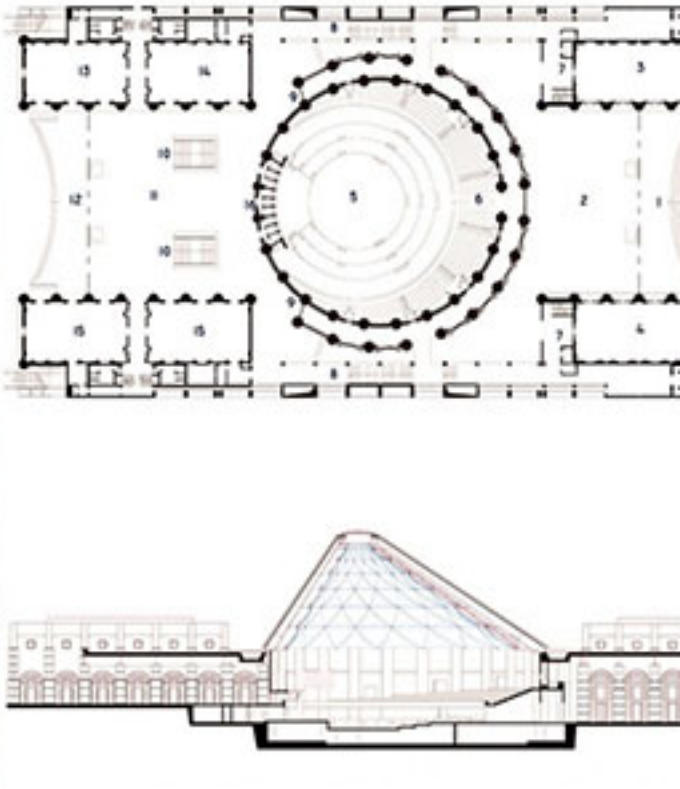
عنوان: **مفاهیم برنامه نویسی ـ مروری بر کلاس و شیء**
نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۱/۱۹ ۲۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: آموزش برنامه نویسی ,C#, OOP, Programming

من قصد دارم در قالب چند مطلب برخی از مفاهیم پایه و مهم برنامه نویسی را که پیش نیازی برای درک اکثر مطالب موجود در وب سایت است به زبان ساده بیان کنم تا دایره افرادی که می‌توانند از مطالب ارزشمند این وب سایت استفاده کنند وسعت بیشتری پیدا کند. لازم به توضیح است از آنجا که علاقه ندارم اینجا تبدیل به نسخه فارسی MSDN یا کتاب آنلاین آموزش برنامه نویسی شود این سری آموزش‌ها بیشتر شامل مفاهیم کلیدی خواهند بود.
این مطلب به عنوان اولین بخش از این سری مطالب منتشر می‌شود.

هدف این نوشته بررسی جزییات برنامه نویسی در رابطه با کلاس و شیء نیست. بلکه دریافتن چگونگی شکل گرفتن ایده شیء گرایی و علت مفید بودن آن است.

مشاهده مفاهیم شیء گرایی در پیرامون خود حتماً در دنیای برنامه نویسی شیء گرا بارها با کلمات کلاس و شیء روبرو شده اید. درک صحیح از این مفاهیم بسیار مهم و البته بسیار ساده است. کار را با یک مثال شروع می‌کنیم. به تصویر زیر نگاه کنید.



در سمت راست بخشی از نقشه یک ساختمان و در سمت چپ ساختمان ساخته شده بر اساس این نقشه را می‌بینید. ساختمان همان شیء است. و نقشه ساختمان کلاس آن است چراکه امکان ایجاد اشیائی که تحت عنوان ساختمان طبقه بندی (کلاس بندی) می‌شوند را فراهم می‌کند. به همین سادگی. کلاس‌ها طرح اولیه، نقشه یا قالبی هستند که جزییات یک شی را توصیف می‌کنند.
حتماً با من موافق هستید اگر بگویم:
در نقشه ساختمان نمی‌توانید زندگی کنید اما در خود ساختمان می‌توانید.
از روی یک نقشه می‌توان به تعداد دلخواه ساختمان ساخت.
هنگامی که در یک ساختمان زندگی می‌کنید نیازی نیست تا دقیقاً بدانید چگونه ساخته شده و مثلاً سیم کشی یا لوله کشی‌های آن

چگونه است! تنها کافیست بدانید برای روشن شدن لامپ باید کلید آن را بزنید.
ساختمان دارای ویژگی هایی مانند متراژ، ضخامت دیوار، تعداد پنجره و ابعاد هر یک و ... است که در هنگام ساخت و بر اساس اطلاعات موجود در نقشه تعیین شده اند.
ساختمان دارای کارکرد هایی است. مانند بالا و پایین رفتن آسانسور و یا باز و بسته شدن درب پارکینگ. هر یک از این کارکردها نیز بر اساس اطلاعات موجود در نقشه پیاده سازی و ساخته شده اند.
ساختمان تمام اجزای لازم برای اینکه از آن بتوانیم استفاده کنیم و به عبارتی در آن بتوانیم زندگی کنیم را در خود دارد.
در محیط پیرامون ما تقریباً هر چیزی را می‌توان در یک دیدگاه شیء تصور کرد. به عبارتی هر چیزی که بتوانید به صورت مستقل در ذهن بیاورید و سپس برخی ویژگی‌ها و رفتارها یا کارکردهای آن‌را برشمارید تا آن چیز را قابل شناسایی کند شیء است. مثلاً من به شما می‌گویم موجودی چهار پا دارد، مو... مو... می‌کند و شیر می‌دهد و ... . شما خواهید گفت گاو! و نمی‌گویید گربه. چرا؟ چون توانستید در ذهن خود موجودیتی را به صورت مستقل تصور کنید و از روی ویژگی‌ها و رفتارش آن‌را دقیقاً شناسایی کنید.
سوال: کلاس یا نقشه ایجاد گاو چیست؟ اگر از من بپرسید خواهم گفت طرح اولیه گاو هم ممکن است وجود داشته باشد البته در اختیار خداوند و با سطح دسترسی ملکوت!
اتومبیل، تلویزیون و ... همگی مثال هایی از اشیاء پیرامون ما در دنیای واقعی هستند که حتماً می‌توانید کلاس یا نقشه ایجاد آن‌ها را نیز بدست آورید و یا ویژگی‌ها و کارکردهای آن‌ها را برشمارید.

مفاهیم شیء گرایی در مهندسی نرم افزار مفاهیمی که تاکنون در مورد دنیای واقعی مرور کردیم همان چیزی است که در دنیای برنامه نویسی ـ به عقیده من دنیای واقعی‌تر از دنیای واقعی ـ با آن سر و کار داریم. علت این امر آن است که اصولاً ایده روش برنامه نویسی شیء گرا با مشاهده محیط پیرامون ما به وجود آمده است.
برای نوشتن برنامه جهت حل یک مسئله بزرگ باید بتوان آن مسئله را به بخش‌های کوچکتری تقسیم نمود. در این رابطه مفهوم شیء و کلاس با همان کیفیتی که در محیط پیرامون ما وجود دارد به صورت مناسبی امکان تقسیم یه مسئله بزرگ به بخش‌های کوچکتر را فراهم می‌کند. و سبب می‌شود هماهنگی و تقارن و تناظر خاصی بین اشیاء برنامه و دنیای واقعی بوجود آید که یکی از مزایای اصلی روش شیء گراست.
از آنجا که در یک برنامه اصولاً همه چیز و همه مفاهیم در قالب کدها و دستورات برنامه معنا دارد، کلاس و شیء نیز چیزی بیش از قطعاتی کد نیستند. قطعه کد هایی که بسته بندی شده اند تا تمام کار مربوط به هدفی که برای آن‌ها در نظر گرفته شده است را انجام دهند.
همان طور که در هر زبان برنامه نویسی دستوراتی برای کارهای مختلف مانند تعریف یک متغیر یا ایجاد یک حلقه و ... در نظر گرفته شده است، در زبان‌های برنامه نویسی شیء گرا نیز دستوراتی وجود دارد تا بتوان قطعه کدی را بر اساس مفهوم کلاس بسته بندی کرد.
به طور مثال قطعه کد زیر را در زبان برنامه نویسی سی شارپ در نظر بگیرید.

```
class Player
{
   public string Name;
   public int Age;
   public void Walk()
   {
      // کدهای مربوط به پیاده سازی راه رفتن
   }
   public void Run()
   {
      // کدهای مربوط به پیاده سازی دویدن
   }
}
```

در این قطعه کد با استفاده از کلمه کلیدی class در زبان سی شارپ کلاسی ایجاد شده است که دارای دو ویژگی نام و سن و دو رفتار راه رفتن و دویدن است.
این کلاس به چه دردی می‌خورد؟ کجا می‌توانیم از این کلاس استفاده کنیم؟
پاسخ این است که این کلاس ممکن است برای ما هیچ سودی نداشته باشد و هیچ کجا نتوانیم از آن استفاده کنیم. اما بیایید فرض کنیم برنامه نویسی هستیم که قصد داریم یک بازی فوتبال بنویسیم. به جای آنکه قطعات کد مجزایی برای هر یک از بازیکنان و کنترل رفتار و ویژگی‌های آنان بنویسیم با اندکی تفکر به این نکته پی می‌بریم که همه بازیکنان مشترکات بسیاری دارند و به عبارتی در یک گروه یا کلاس قابل دسته بندی هستند. پس سعی می‌کنیم نقشه یا قالبی برای بازیکن‌ها ایجاد کنیم که دربردارنده ویژگی‌ها و

رفتارهای آن‌ها باشد.
همان طور که در نقشه ساختمان نمی‌توانیم زندگی کنیم این کلاس هم هنوز آماده انجام کارهای واقعی نیست. چراکه برخی مقادیر هنوز برای آن تنظیم نشده است. مانند نام بازیکن و سن و ....
و همان طور که برای سکونت لازم است ابتدا یک ساختمان از روی نقشه ساختمان بسازیم برای استفاده واقعی از کلاس یاد شده نیز باید از روی آن شیء بسازیم. به این فرآیند وهله سازی یا نمونه سازی نیز می‌گویند. یک زبان برنامه نویسی شیء گرا دستوراتی را برای وهله سازی نیز در نظر گرفته است. در C# کلمه کلیدی new این وظیفه را به عهده دارد.

```
Player objPlayer = new Player();
objPlayer.Name = “Ali Karimi”;
objPlayer.Age = 30;
objPlayer.Run();
```

وقتی فرآیند وهله سازی صورت می‌گیرد یک نمونه یا شیء از آن کلاس در حافظه ساخته می‌شود که در حقیقت می‌توانید آنرا همان کدهای کلاس تصور کنید با این تفاوت که مقداردهی‌های لازم صورت گرفته است. به دلیل تعیین مقادیر لازم، حال شیء تولید شده می‌تواند به خوبی اعمال پیش بینی شده را انجام دهد. توجه نمایید در اینجا پیاده سازی داخلی رفتار دویدن و اینکه مثلاً در هنگام فراخوانی آن چه کدی باید اجرا شود تا تصویر یک بازیکن در حال دویدن در بازی نمایش یابد مد نظر و موضوع بحث ما نیست. بحث ما چگونگی سازماندهی کدها توسط مفهوم کلاس و شیء است. همان طور که مشاهده می‌کنید ما تمام جزییات بازیکن‌ها را یکبار در کلاس پیاده سازی کرده ایم اما به تعداد دلخواه می‌توانیم از روی آن بازیکن‌های مختلف را ایجاد کنیم. همچنین به راحتی رفتار دویدن یک بازیکن را فراخوانی میکنیم بدون آنکه پیاده سازی کامل آن در اختیار و جلوی چشم ما باشد.
تمام آنچه که بازیکن برای انجام امور مربوط به خود نیاز دارد در کلاس بازیکن کپسوله می‌شود. بدیهی است در یک برنامه واقعی ویژگی‌ها و رفتارهای بسیار بیشتری باید برای کلاس بازیکن در نظر گرفته شود. مانند پاس دادن، شوت زدن و غیره.
به این ترتیب ما برای هر برنامه می‌توانیم مسئله اصلی را به تعدادی مسئله کوچکتر تقسیم کنیم و وظیفه حل هر یک از مسائل کوچک را به یک شیء واگذار کنیم. و بر اساس اشیاء تشخیص داده شده کلاس‌های مربوطه را بنویسیم. برنامه نویسی شیء گرا سبب می‌شود تا مسئله توسط تعدادی شیء که دارای نمونه‌های متناظری در دنیای واقعی هستند حل شود که این امر زیبایی و خوانایی و قابلیت نگهداری و توسعه برنامه را بهبود می‌دهد.
احتمالاً تاکنون متوجه شده اید که برای نگهداری ویژگی‌های اشیاء از متغیرها و برای پیاده سازی رفتارها یا کارکردهای اشیاء از توابع استفاده میکنیم.
با توجه به این که هدف این مطلب بررسی مفهوم شیء گرائی بود و نه جزییات برنامه نویسی، بنابراین بیان برخی مفاهیم در این رابطه را که بیشتر در مهندسی نرم افزار معنا دارند تا در دنیای واقعی در مطالب بعدی بررسی می‌کنیم.

نظرات خوانندگان

نویسنده: Hesam
تاریخ: ۱۳۹۲/۰۱/۱۹ ۲۲:۴۵

بسیار عالی :)

نویسنده: من
تاریخ: ۱۳۹۲/۰۱/۲۰ ۱۴:۵۶

شروع خوبی بود، آفرین!
من هنوز وقتی جلسه‌ی اول کلاس میپرسم که پراید یک کلاس هست و یا یک آبجکت. همه میگن آبجکتی از کلاس ماشین!
این در حالیست که خیلی از این افراد سالهاست برنامه نویسی میکنند و هنوز نمیدونند heap یا stack چیه

نویسنده: علي
تاریخ: ۱۳۹۲/۰۱/۲۰ ۱۵:۴

حالا با توجه به توضیحات بالا که گفته شد « *در نقشه ساختمان نمی‌توانید زندگی کنید اما در خود ساختمان می‌توانید.* » با یک پراید می‌شود رانندگی کرد اما با ماشین که بیشتر یک مفهوم است خیر. نه؟

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۱/۲۰ ۱۵:۲۲

تشکر از نظر شما. متوجه صحبت شما هستم ولی این توضیح برای دوستان دیگه می‌تونه مفید باشه.
پراید به عنوان رده ای از اتومبیل‌ها کلاس است. اما اگر به کسی یک اتومبیل پراید در حال عبور را نشان دهید که دارای پلاک و ... است، آن شیء ای است از کلاس پراید. اگر در یک پارکینگ تعدادی اتومبیل باشد و از کسی بخواهیم بر اساس نوع گروه بندی یا کلاس بندی کند، بعد از چند دقیقه خواهیم دید اتومبیل‌های هم نوع را جدا کرده. مثلا همه اتومبیل‌های از نوع پراید را کنار هم قرار داده. این همان مفهوم کلاس بندی است. برای تولید اتومبیلی از نوع پراید کارخانه یک نقشه یا طرحی ایجاد می‌کند که بسیاری مشخصات و چگونگی ساخت را در خود دارد. چگونگی انجام برخی کارکردها مانند حرکت را دارد. این طرح کلاس نامیده می‌شود. اما آیا می‌توان برای آن پلاک در نظر گرفت؟ خیر چون فقط نقشه ایجاد اتومبیل است (یا به عبارتی فقط مفهوم است). همچنین کلاس پراید احتمالاً از کلاس اتومبیل یا ماشین برخی خصوصیات و رفتارها را با ارث برده است.

بیاد داشته باشیم هر فردی در هر سطحی از دانش هم مسلماً بسیاری چیزها را هنوز نمی‌داند و دانسته‌های ما در مقابل نادانسته‌ها قطره ای بیش نیست. تا جایی که می‌توان گفت همه ما از نظر نادانسته‌ها برابریم. تفاوت در دانسته هاست. کسانی که سال‌ها برنامه نویسی می‌کنند هم حتما دانسته هایی دارند که می‌توانند این کار را ادامه دهند. پس کافی است آنچه نمی‌دانند را سعی کنیم به اشتراک بگذاریم تا بدانند و از دانسته هایشان استفاده کنیم.

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۱/۲۰ ۱۵:۲۵

جمله با پراید می‌شود رانندگی کرد ابهام دارد. ابهام آن به این صورت رفع می‌شود که من می‌دانم منظور شما از پراید به عنوان یک اسم عام و یک مفهوم و **نام رده یا کلاسی از اتومبیل ها** نیست. بلکه منظور شما با اتومبیل پرایدی است که دارای یک پلاک مشخص است و مثلاً کسی به تازگی گنجی پیدا کرده و رفته یک پراید خریده! آن اتومبیل پراید مشخص یک شیء است از کلاس پراید. بله با آن شی می‌توان رانندگی کرد. اما با مفهوم یا نقشه یا کلاس یا رده یا گروه یا طرح تولید خودروی پراید یا هر ماشین دیگری نمی‌توان رانندگی کرد.

نویسنده: سید ایوب کوکبی
تاریخ: ۱۳۹۲/۰۱/۲۱ ۱۵:۴۸

ممنون، خیلی خوب بررسی کرده بودید و مثالهایی که هم ارائه کردید به دلم نشست، امیدوارم برای سایر مباحث هم به همین صورت ادامه بدید. البته به نظر بنده نیازی هم نیست خیلی روی این مبحث تناظر بین دنیای واقعی و دنیای شی گرا حساس باشیم، چون اگر به همین نحو بخواهیم این دو را با هم مقایسه کنیم شاید بعضی جاها با مشکل مواجه بشیم. این تناظر فقط برای اینه که دید و تجسمی از این مفهوم داشته باشیم تا بهتر بتوانیم آن را بپذیریم.
یادمون نره که هدف ما یادگیری مفهوم شی گرایی است نه یادگیری تناظر آن با دنیای واقعی، این تناظر فقط یک ابزاره برای دسترسی سریعتر به این هدف!

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۱/۲۱ ۱۷:۳۵

سلام و ممنون از نظر شما.
اتفاق جالبی افتاد و آن این بود که هم اکنون داشتم در OneNote بخشی برای مطلب بعدی می‌نوشتم. دقیقاً داشتم پاراگرافی را می‌نوشتم که جلوی این که ذهن خواننده به سمتی برود که گویی "الزاماً هر مورد در مهندسی نرم افزار را باید پس از یافتن مصداق آن در محیط اطراف یاد گرفت" را بگیرم.
دقیقاً صحیح است. تاکید بر این تناظر در این بخش به دلیل یافتن درک عمیق‌تر از شیء گرایی و علت مفید بودن آن و چگونگی شکل گیری ایده آن است. این درک عمیق‌تر امکان استفاده بهتر و صحیح‌تر این مفاهیم در برنامه را فراهم می‌کند. و سبب می‌شود برنامه نویس شیءگرایی را ابزاری برای حل مسئله بیاد نه راه و روشی که همه میگن خوبه پس باید رعایت کرد. حال آنکه چون درک دقیقی از آن ندارد در حقیقت مسئله را با آن روشی که بهتر بلد است حل می‌کنند و فقط تعدادی کلاس و شیء در برنامه وجود دارد.

نویسنده: آریانا
تاریخ: ۱۳۹۲/۰۲/۰۷ ۱۱:۵۶

بسیار بسیار عالی بود مرسی

نویسنده: سحابی
تاریخ: ۱۳۹۲/۰۷/۱۴ ۱۴:۵۹

سلام
برای یادگیری Desing pattern منابعی وجود دارد ؟ لطفا در صورت امکان اگر منابع و یا لینک کارآمدی معرفی کنید .

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۱۴ ۱۷:۱۵

برچسب‌های سایت رو مرور کنید به اندازه کافی در این زمینه مطلب هست. مثلا اینجا