بسم الله الرحمن الرحیم

نام دانشجو : محمد مهدی عبداللهی پیربداغ

نام درس : برنامه نویسی سمت سرور

نام استاد : مهندس میثاق یاریان

تمرین پنجم PDF

عناوین :

اصل Solid

Constructor – deconstructor – Yagni – Kiss – Dry – GC Collection

تابع سازنده یا Constructor متد ویژه ای است که هر بار به محض ساخته شدن یک شی یا object از کلاس، اجرا می شود. درواقع از تابع سازنده برای مقداردهی اولیه متغیرها یا اجرای یک کد ثابت استفاده می شود.
یک تابع سازنده هیچ گاه مقدار خروجی یا return ندارد و به همین دلیل در تعریف آن، هیچ نوع متغیری جهت return تعریف نمی شود. ساختار کلی تعریف تابع سازنده Constructor در کلاس های C# به صورت زیر است:

```
public string Describe()
```

همچنین تابع سازنده را به صورت زیر نیز می توانید تعریف کنید:

```
public Car()
```

- یک تابع سازنده، می تواند تابع سازنده دیگر را فراخوانی کند.
- تابع سازنده هیچ پارامتری را به عنوان ورودی ندارد، ابتدا اجرا می شود. از این حالت برای مقداردهی اشیا (objects) های یک کلاس با یک تابع سازنده پیش فرض استفاده می شود.
- اگر شما تابع سازنده ای که دارای ۲ پارامتر است را فراخوانی کنید، پارامتر اول برای فراخوانی تابع سازنده ای که دارای ۱ پارامتر است، استفاده می شود.

آموزش کار با تابع تخریب کننده یا Destractor

تابع تخریب کننده یا Destractor در زبان C# ، متدی است که در هنگام از بین رفتن یک شی از کلاس، اجرا می شود. زبان C# ، یک زبان پاک کننده خودکار سیستم یا collector garbage است، به این معنی که object هایی که دیگر در برنامه نیاز ندارید را جهت خالی کردن حافظه و آزاد نمودن سیستم، پاک می کند.
از طرف دیگر در برخی موارد شاید نیاز داشته باشید تا یک Clean up در سیستم انجام دهید، اینجاست که تابع های تخریب کننده Destructor به کار می آیند.
تابع های تخریب کننده چندان شبیه سایر متدها در زبان C# نیستند. در کد عملی زیر یک مثال از تابع تخریب کننده نشان داده شده است:

- ~Car()

```
Console.WriteLine("Out..");
```

به محض این که شی یا object ایجاد شده از کلاس، توسط تمیز کننده خودکار garbage collector جمع آوری شده، متد فوق فراخوانی می شود.

KISS

"Keep It Simple, Stupid"

اصل KISS در برنامه نویسی بسیار اهمیت دارد.سعی کنید این اصل را سعی کنید به خاطر بسپارید و برای حفظ آن تلاش کنید.هرچقدر کد ساده تر باشد نگهداری آن در آینده ساده تر است.افراد دیگری که بخواهند کد شما را مورد ارزیابی قرار دهند در آینده با استقبال بیشتری این کار را انجام میدهند.اصل KISS توسط Kelly Johnsonپایه گذاری شده است و سیستم های خوب به جای پیچیدگی، به سمت ساده سازی پیش میروند.از این رو سادگی، کلید طلایی طراحی است و باید از پیچیدگی های غیرضروری دوری کرد.

YAGNI

"You Aren't Gonna Need It"

گاهی اوقات تیم های توسعه و برنامه نویسان در مسیر پروژه تمرکز خود را بر روی قابلیت های اضافه ی پروژه که "فقط الان به آن نیاز دارند" یا "در نهایت به آن نیاز پیدا میکنند" میگذارند.در یک کلام :اشتباه است!در اکثر مواقع شما به آن نیاز پیدا ندارید و نخواهید داشت. "شما به آن نیازی ندارید."

اصل YAGNI قبل از کدنویسی بی انتها و بر پایه ی مفهموم "آیا ساده ترین چیزی است که می تواند احتمالا کار کند" قرار دارد.حتی اگر YAGNI را جزوی از کدنویسی بی انتها بدانیم، بر روی تمام روش ها و فرآیند های توسعه قابل اجرا است.با پیاده سازی ایده‌ی "شما به آن نیازی ندارید" میتوان از هدر رفتن وقت جلوگیری کرد و تنها رو به جلو و در مسیر پروژه پیش رفت.

هر زمان اضطراب ناشناخته ای در کد حس کردید نشانه ی یک امکان اضافی بدون مصرف در این زمان است.احتمالا شما فکر میکنید یک زمانی این امکان اضافی را نیاز دارید.آرامش خود را حفظ کنید! و تنها به کارهای موردنیاز پروژه در این لحظه نگاه کنید. شما نمیتوانید زمان خود را صرف بررسی آن امکان اضافی کنید

چون در نهایت مجبور به تغییر، حذف یا احتمالا پذیرفتن هستید ولی در نهایت جزو امکانات اصلی محصول شما نیست.

DRY

“Don’t Repeat Yourself”

تا الان چندین بار به کد های تکراری در پروژه برخورد کرده اید؟ اصل DRY توسط Andrew Huntو David Thomas در کتاب The Pragmatic Programmer پایه گذاری ده است.خلاصه ی این کتاب به این موضوع اشاره میکنید که "هر بخش از دانش شما در پروژه باید یک مرجع معتبر، یکپارچه و منحصربفرد داشته باشد". به عبارت دیگر شما باید سعی کنید رفتار سیستم را در یک بخش از کد مدیریت کنید.

از سوی دیگر زمانی که از اصل DRY پیروی نمیکنید، در حقیقت اصل WET که به معنای Write Everything Twiceیا We Enjoy Typing دامن گیر شما شده است! (لذت بردن از وقت تلف کردن(

استفاده از اصل DRY در برنامه نویسی بسیار کارآمد است. مخصوصا در پروژه های بزرگ که کد دائما در حال نگهداری و توسعه است

ممکن است علاقه ای به رعایت اصل DRY نداشته باشید ولی اصول قبلی (YAGNI, KISS) را حتما بخاطر بسپارید.

سولید (SOLID) یک کلمه مخفف برای پنج اصل اولیه طراحی شئ گرا است که رابرت سیسیل مارتین معروف به عمو باب (uncle bob) اون رو مطرح کرد.

این اصول زمانی که دست به دست هم میدن، کار گسترش یا اضافه کردن قابلیت های جدید به برنامه و نگهداری یا همون دیباگ یک برنامه رو برای برنامه نویس ها آسان می کنند.

- S - Single-responsiblity principle اصل مسئولیت واحد یا اصل تک مسئولتی بودن کلاس ها
- O - Open-closed principle اصل باز/بسته بودن کلاس ها
- L - Liskov substitution principle اصل جانشینی لیسکوف
- I - Interface segregation principle اصل تفکیک اینترفیس
- D - Dependency Inversion Principle اصل انطباق پذیری

Garbage Collection جمع آوری زباله در دات نت چیست؟

به طور کلی، Garbage Collection (GC)چیزی نیست جز به دست آوردن مجدد حافظه اختصاص داده شده به اشیایی که در حال حاضر در هیچ بخشی از برنامه ما استفاده نمی شوند .

هنگامی که یک شی در سی شارپ ایجاد می کنیم،یک حافظه برای شی در حافظه heap تخصیص داده می شود. حافظه heap به طور کامل توسط Common Language Runtime (CLR) در چارچوب دات نت مدیریت می شود. تخصیص حافظه و توزیع در heap توسط CLR انجام می شود. همیشه برای هر چیزی محدودیتی وجود دارد، در چنین مواردی حافظه نیز محدود است. ما باید مقداری حافظه را در heap پاک کنیم تا CLR بتواند حافظه را به اشیاء تازه ایجاد شده اختصاص دهد.

Win۳۲ DLL (Dynamic Link Library)در سیستم عامل بسته به نسخه سیستم عامل ۳۲ بیتی یا ۶۴ بیتی وظیفه مراقبت از تخصیص حافظه را دارد. برای سیستم عامل های ۳۲ بیتی، حداکثر رمی که می تواند پشتیبانی کند تا ۴ گیگابایت است. تخصیص حافظه مجازی به حداکثر ۲ گیگابایت توسط Win۳۲ DLL محدود شده است. حافظه Heap نیز از طریق این حافظه مجازی مدیریت می شود.

Garbage Collection چه زمانی اتفاق می افتد؟

Garbage Collection در صورتی اتفاق می افتد که حداقل یکی از شرایط زیر برآورده شود. این شرایط به شرح زیر است:

- اگر سیستم حافظه فیزیکی پایینی دارد، Garbage Collection ضروری است.
- اگر حافظه تخصیص داده شده به اشیاء مختلف در حافظه heap از یک آستانه از پیش تعیین شده فراتر رود، Garbage Collection اتفاق می افتد.
- اگر متد GC.Collect فراخوانی شود، Garbage Collection اتفاق می افتد. با این حال، این روش تنها در شرایط غیرعادی فراخوانی می شود، زیرا به طور معمول Garbage Collection به طور خودکار انجام می شود.