به نام خدا

محمدجواد قمری

دانشجوی رشته فناوری اطلاعات

درس برنامه نویسی سمت سرور

استاد آقای میثاق یاریان

شماره دانشجویی 01221033720028

تحقیق در مورد YAGNI , KISS , DRY , SOLID و deconstructor , constructor , gc.collect

---

KISS

KISS مخفف عبارت "Keep It Simple, Stupid" است که در فارسی به معنای "ساده نگه دارید، احمق" است. این اصل در برنامه نویسی بیان می کند که کد باید تا حد امکان ساده و مختصر نوشته شود.

اصل KISS توصیفی است که کد را ساده و واضح نگه می دارد و درک آن را آسان می کند.بعلاوه باید توجه کنید زبانهای برنامه نویسی برای درک انسانهاست . کامپیوترها فقط 0 و 1 را می توانند بفهمند، بنابراین کدنویسی را ساده و آسان انجام دهید. متد ها و فانکشنهای خود را کوچک نگه دارید.

هر متد فقط باید یک مشکل کوچک را حل کند و باید برای حل یک مشکل یا انجام یک عملیات باشد. اگر شروط زیادی در متد دارید ، آنها را به متدهای کوچکتر تقسیم کنید. خواندن و نگهداری آن نه تنها آسان تر خواهد بود ، بلکه می توانید مشکلات و باگهای آنها را زود تر پیدا کنید.

اصل KISS مزایای زیادی دارد

کد ساده تر برای درک و نگهداری است.

کد ساده تر برای اشکال زدایی است.

کد ساده تر برای توسعه است.

کد ساده تر برای بهینه سازی است.

DRY

DRY مخفف عبارت "Don't Repeat Yourself" است که در فارسی به معنای "خودت را تکرار نکن" است. این اصل در برنامه نویسی بیان می کند که کد نباید تکراری باشد.

هدف اصل DRY کاهش اطلاعات تکراری است
اصل DRY توسط Andrew Hunt و David Thomas در کتاب The Pragmatic Programmer پایه گذاری شده 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.

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

با رعایت اصل DRY در برنامه نویسی، می توانید کدی بنویسید که ساده تر، قابل درک تر، قابل نگهداری تر و در نهایت با کیفیت تر باشد.

YAGNI

YAGNI مخفف عبارت "You Aren't Gonna Need It" است که در فارسی به معنای "شما به آن نیاز ندارید" است. این اصل در برنامه نویسی بیان می کند که نباید ویژگی هایی را در کد خود پیاده سازی کنید که در حال حاضر به آنها نیاز 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.

اصل YAGNI مزایای زیادی دارد. در اینجا چند مورد از این مزایا آورده شده است:

باعث صرفه جویی در زمان و تلاش می شود.

باعث کاهش پیچیدگی کد می شود.

باعث کاهش احتمال اشکالات می شود.

باعث افزایش قابلیت نگهداری کد می شود.

SOLID

SOLID مخفف پنج اصل طراحی شی گرا است که توسط Robert C. Martin در سال 2000 معرفی شد. این اصول برای بهبود کیفیت کد و سهولت توسعه و نگهداری آن طراحی شده اند.

اصول SOLID عبارتند از:

1: Single Responsibility Principle (SRP)

هر کلاس باید یک مسئولیت واحد داشته باشد.

2: Open-Closed Principle (OCP)

کلاس ها باید باز برای گسترش باشند، اما بسته برای اصلاح باشند.

3: Liskov Substitution Principle (LSP)

زیر کلاس ها باید بتوانند به جای کلاس های پایه خود استفاده شوند.

4: Interface Segregation Principle (ISP)

رابط ها باید به کوچکترین رابط های ممکن تقسیم شوند.

5: Dependency Inversion Principle (DIP)

وابستگی ها باید به سمت انتزاع ها هدایت شوند، نه جزئیات پیاده سازی.

Single Responsibility Principle (SRP)

اصل SRP بیان می کند که هر کلاس باید یک مسئولیت واحد داشته باشد. این بدان معناست که هر کلاس باید تنها یک کار را انجام دهد و به هیچ چیز غیر ضروری وابسته نباشد.

Open-Closed Principle (OCP)

اصل OCP بیان می کند که کلاس ها باید باز برای گسترش باشند، اما بسته برای اصلاح باشند. این بدان معناست که کلاس ها باید به گونه ای طراحی شوند که بتوان ویژگی های جدیدی را به آنها اضافه کرد، بدون اینکه نیاز به تغییر کد موجود باشد.

Liskov Substitution Principle (LSP)

اصل LSP بیان می کند که زیر کلاس ها باید بتوانند به جای کلاس های پایه خود استفاده شوند. این بدان معناست که زیر کلاس ها باید رفتار کلاس پایه را حفظ کنند.

Interface Segregation Principle (ISP)

اصل ISP بیان می کند که رابط ها باید به کوچکترین رابط های ممکن تقسیم شوند. این بدان معناست که هر رابط باید فقط یک مجموعه کوچک از مسئولیت ها را تعریف کند.

Dependency Inversion Principle (DIP)

اصل DIP بیان می کند که وابستگی ها باید به سمت انتزاع ها هدایت شوند، نه جزئیات پیاده سازی. این بدان معناست که کلاس ها باید به انتزاع ها وابسته باشند، نه جزئیات پیاده سازی.

Constructor

در برنامه نویسی شی گرا، سازنده constructor یک متد ویژه است که به محض ایجاد یک شی از یک کلاس، فراخوانی می شود. سازنده معمولاً برای مقداردهی اولیه فیلدهای یک شی استفاده می شود.

(همچنین یکی دیگر از کاربردهای کانستراکتور، مقداردهی کردن پراپرتی‌ها است.)

به طور معمول، ما از متد Constructor برای دادن مقادیر اولیه به متغیر های نمونه که درون کلاس قرار دارند، استفاده میکنیم. همچنین همه اعمال اولیه ای که نیاز داریم انجام بدهیم تا یک Object یا شئ ساخته شود را درون آن قرار میدهیم.

چه شما متد سازنده را برای کلاس تعریف کنید چه نکنید، این متد برای کلاس شما ساخته میشود. حتی اگر جاوا هم متد Constructor را بصورت اتوماتیک برای کلاس شما معین بکند، تمامی متغیر های نمونه که در کلاس شما وجود دارند، برابر با "صفر" مقدار دهی اولیه خواهند شد. اما اگر خودتان متد Constructor را برای کلاس بسازید، دیگر از مقدار صفر استفاده نمیشود.

Deconstructor

در برنامه نویسی شی گرا، تخریب کننده یک متد ویژه است که به محض حذف یک شی از حافظه، فراخوانی می شود. تخریب کننده معمولاً برای آزادسازی منابعی که توسط شی استفاده می شود، استفاده می شود.

در زبان های برنامه نویسی شی گرا، هر کلاسی می تواند یک تخریب کننده داشته باشد. اگر تخریب کننده ای برای یک کلاس تعریف نشود، کامپایلر هیچ تخریب کننده ای برای آن کلاس ایجاد نمی کند.

تخریب کنندگان می توانند با یا بدون پارامتر تعریف شوند. تخریب کنندگان بدون پارامتر ساده ترین نوع تخریب کننده هستند. آنها معمولاً برای آزادسازی منابعی مانند حافظه استفاده می شوند.

تخریب کنندگان به برنامه نویس اجازه می دهد تا یک آرایه یا شیء را به اجزای آن تجزیه کند. این به برنامه نویس امکان می دهد تا به راحتی به اجزای یک آرایه یا شیء دسترسی پیدا کند و از آنها استفاده کند.

gc.collect

در برنامه نویسی، gc.collect() یک تابع است که در کتابخانه استاندارد زبان های برنامه نویسی شی گرا مانند جاوا، سی شارپ، و پایتون وجود دارد. این تابع برای جمع آوری زباله (garbage collection) استفاده می شود.

جمع آوری زباله یک فرآیند خودکار است که حافظه تخصیص داده شده به اشیای غیرقابل دسترسی را آزاد می کند. این فرآیند به برنامه نویس اجازه می دهد تا بدون نیاز به مدیریت دستی حافظه، از اشیای شی گرا استفاده کند.

تابع gc.collect() می تواند به صورت دستی فراخوانی شود یا به صورت خودکار توسط جمع آوری زباله فراخوانی شود. فراخوانی دستی این تابع می تواند برای افزایش عملکرد برنامه استفاده شود.

تابع gc.collect() یک ابزار قدرتمند است که می تواند به بهبود عملکرد و کارایی برنامه های شی گرا کمک کند.