



× ثبت نام سریع برای استفاده از تمام امکانات

شماره موبایل خود را وارد کنید

ثبت نام

+ دنبال کنید فرشید عزیزی

Software Engineer

۹ ماه پیش / خواندن ۱۸ دقیقه



آموزش کاربردی UML برای توسعه دهندگان نرم افزار



زبان مدل سازی یکپارچه

هر چه یک سیستم پیچیده تر باشد، نمایش بصری آن اهمیت بیشتری دارد. فرآیند ترسیم فیزیکی اجزاء، مشخص می کند که چه چیزی در کجا کار می کند، چه چیزی نیست، و کجا فرصت هایی برای بهبود وجود دارد. و استفاده از یک زبان مشترک، مانند نمودارهای UML، به تیم ها کمک می کند تا در این مسائل همکاری کنند.

این راهنما شما را با Unified Modeling Language و نمودارهایی که آن را نشان می دهد آشنا می کند. در کمترین زمان، شما و تیمتان می توانید از نمودارهای UML برای پروژه های خود استفاده کنید.

این راهنما برای موارد زیر مفید است:

هر کسی که علاقه مند به تجسم یک سیستم پیچیده است، معماران سیستم، مهندسان نرم افزار و توسعه دهندگان نرم افزار که به دنبال مقدمه ای برای نمودارهای UML، اصول UML و به روز رسانی به UML 2.5 (آخرین نسخه موجود تا زمان نگارش این مطلب) هستند مفید است.

یک مرور کلی

مدلینگ (Modeling) چیست؟

مدل سازی راهی برای تجسم طراحی برنامه نرم افزاری شما و بررسی آن در برابر الزامات قبل از شروع تیم شما به کدنویسی است.

به همان شیوه ای که یک معمار قبل از شروع ساخت و ساز بر روی یک آسمان خراش طرحی را ایجاد می کند، یک توسعه دهنده می تواند از نمودارهای مدل سازی برای تثبیت و آزمایش آنچه که قرار است قبل از شروع به کدنویسی ایجاد کند، استفاده کند.

ترسیم نقشه اولین گام برای هر پروژه ای است و مدل سازی بخشی ضروری از هر پروژه نرم افزاری کوچک یا بزرگ است. برای اینکه یک برنامه به خوبی کار کند، باید به گونه ای طراحی شود که مقیاس پذیری (scalability) را فراهم کند.

با استفاده از نمودارهای زبان مدلسازی یکپارچه (UML)، می‌توانید طراحی‌های سیستم‌های نرم‌افزاری خود را قبل از اجرای کد، تجسم و تأیید کنید.

تعریف UML چیست؟

طبق آخرین نسخه از مستندات UML 2.5، "هدف UML ارائه ابزارهایی برای معماران سیستم، مهندسان نرم افزار و توسعه دهندگان نرم افزار برای تجزیه و تحلیل، طراحی و پیاده سازی سیستم های مبتنی بر نرم افزار می باشد.

از UML برای مدل سازی کسب و کار و فرآیندهای مشابه می توان استفاده کرد

نظر خود را درباره این پست بنویسید

iman safari

۷ ماه پیش

فرشید جان واقعا مطالب خوبی میداری. من همیشه دنبال می کنم

۱

برخی ممکن است نگران باشند که استفاده از نمودارهای UML می تواند آنها را در یک سبک آشنایی توسعه نرم افزار قفل کند، این لزوما درست نیست. نمودارهای UML را می توان در مراحل مختلف توسعه ایجاد و استفاده کرد و می توان آنها را به طور مداوم به روز کرد و تکرار کرد.

سایر مزایای کلیدی عبارتند از:

- **تعدیل کار :** نمودارها دیدن کد اضافی را برای برنامه نویسان آسان تر می کند و به جای بازنویسی آن توابع، از بخش هایی از کدهایی که از قبل وجود دارد استفاده مجدد می کنند.
- **ساده سازی تغییرات :** ایجاد تغییرات در نمودارها در ابتدا بسیار ساده تر از برنامه ریزی مجدد کد در مرحله بعدی است. این باعث صرفه جویی در وقت و هزینه تیم شما می شود.
- **روشن شدن ابهامات :** شما فقط با مستندات طراحی می توانید پیش بروید. در دراز مدت، برای برقراری ارتباط با توسعه دهندگان جدید یا دورکار(remote) که با تاریخچه محصول شما آشنا نیستند، به آن نیاز خواهید داشت.

چرا UML؟

زبان UML پرکاربردترین و قابل فهم ترین زبان مدل سازی است. UML تا حد زیادی محبوب ترین زبان مدل سازی است که امروزه استفاده می شود. این فراگیر بودن خود یک مزیت است زیرا اکثر توسعه دهندگان، صرف نظر از تخصص یا سابقه کاری خود، حداقل با برخی از نمودارهای UML آشنا هستند. و از آنجایی که آنها برای هر نوع برنامه نویسی قابل درک هستند، برای مفید بودن نیازی به خواندن یک زبان برنامه نویسی خاص ندارند.

سه مورد از محبوب ترین نمودارهای UML بیشتر نیازهای مدل سازی شما را پوشش می دهند. اگرچه 14 نوع مختلف نمودار UML برای مدل سازی برنامه ها وجود دارد، اما در عمل، توسعه دهندگان تنها از تعداد کمی برای مستندسازی سیستم های نرم افزاری خود استفاده می کنند. رایج ترین نمودارهای UML که می بینید و استفاده می کنید، نمودارهای کلاس(class diagrams)، نمودارهای توالی(sequence diagrams) و نمودارهای use case هستند. این بدان معناست که دانستن نحوه ایجاد و خواندن تنها 20 درصد از این زبان برای اکثر پروژه های شما کافی است. در این پست تنها به همان 20% یعنی موارد مورد نیاز یک توسعه دهنده نرم افزار پرداخته خواهد شد.

انواع نمودارهای UML

از UML 2.5، اکنون چهارده نمودار UML به طور رسمی شناخته شده است که به دو نوع اصلی تقسیم می شوند:

1. نمودارهای ساختاری(Structure diagrams) : نشان می دهد که چگونه بخش های استاتیک یک سیستم با یکدیگر ارتباط دارند. هر عنصر یک مفهوم خاص را نشان می دهد و ممکن است شامل عوامل انتزاعی(abstract)، دنیای واقعی و عوامل اجرایی باشد.

2. نمودارهای رفتاری(Behavior diagrams) : رفتار دینامیکی همه اشیاء در یک سیستم، از جمله تغییرات سیستم در طول زمان را نشان می دهد. نمودارهای تعامل(Interaction diagrams) را می توان به عنوان زیر مجموعه ای از نمودارهای رفتاری در نظر گرفت.

 انواع نمودارهای UML

انواع نمودارهای UML

نمودارهای ساختاری(Structure diagrams)

مطابق تصویر فوق هفت نمودار ساختاری در UML 2.5 وجود دارد. همانطور که بالاتر به آن اشاره شد بر حسب نیاز صرفا نمودارهای کلاس(Class diagrams) از مجموع هفت نمودار ساختاری را بررسی می کنیم.

نمودارهای کلاس(Class diagrams) :

در مهندسی نرم افزار، نمودار کلاس در زبان مدلسازی یکپارچه (UML) نوعی نمودار ساختار ایستا است که ساختار یک سیستم را با نشان دادن کلاس های سیستم، ویژگی ها(attributes)، عملیات (methods) و روابط

بین اشیاء و همچنین محدودیت های اعمال شده بر سیستم را توصیف می کند.

نمودار کلاس ستون فقرات ساختار مدل سازی شی گرا است. برای مدل سازی مفهومی کلی ساختار برنامه و تبدیل مدل ها به کد برنامه نویسی استفاده می شود. نمودارهای کلاس نیز می توانند برای مدل سازی داده ها استفاده شوند.

نمودار کلاس یک نمودار استاتیک است. نمای استاتیک یک برنامه را نشان می دهد. نمودار کلاس نه تنها برای تجسم، توصیف و مستندسازی جنبه های مختلف یک سیستم، بلکه برای ساخت کدهای اجرایی برنامه نرم افزاری نیز استفاده می شود. نمودارهای کلاس به طور گسترده ای در مدل سازی سیستم های شی گرا استفاده می شوند زیرا آنها تنها نمودارهای UML هستند که می توانند مستقیماً با زبان های شی گرا نگاشت شوند.

نمودارهای کلاس در میان مهندسان نرم افزار برای مستندسازی معماری نرم افزار بسیار محبوب هستند.

یک نمودار کلاس UML از این موارد تشکیل شده است: مجموعه ای از کلاس ها و مجموعه ای از روابط بین کلاس ها

هدف از نمودارهای کلاس(Class diagrams) :

- تجزیه و تحلیل و طراحی نمای استاتیک یک برنامه.
- شرح وظایف یک سیستم
- مهندسی مستقیم(رو به جلو) و معکوس.(Forward and reverse engineering)

مهندسی مستقیم، فرآیند حرکت سنتی از مفاهیم انتزاعی سطح بالا و طراحی های منطقی به اجرای فیزیکی یک سامانه است

مهندسی معکوس می تواند به عنوان فرآیند تجزیه و تحلیل یک سامانه به موارد زیر به در نظر گرفته شود:

شناسایی اجزای سامانه و ارتباط آن ها

ایجاد نمایه هایی از سامانه به صورت سطح بالا

ایجاد نمایه ای فیزیکی از آن سامانه

در برنامه نویسی شی گرا، یک کلاس طرحی است برای ایجاد اشیا (یک ساختار داده خاص)، ارائه مقادیر اولیه برای حالت (متغیرها یا ویژگی های عضو)، و اجرای رفتار (توابع یا متدهای های عضو). اشیاء تعریف شده توسط کاربر با استفاده از کلمه کلیدی کلاس ایجاد می شوند.

چگونه Class diagram رسم کنیم؟

نمودارهای کلاس دارای ویژگی های زیادی هستند که باید در هنگام ترسیم در نظر گرفته شوند.

نمودار کلاس اساساً یک نمایش گرافیکی از نمای ایستا سیستم است و جنبه های مختلف برنامه را نشان می دهد. مجموعه ای از نمودارهای کلاس کل سیستم را نشان می دهد.

هنگام رسم نمودار کلاس نکات زیر را باید به خاطر بسپارید

- نام نمودار کلاس باید برای توصیف آن جنبه از سیستم معنی دار باشد.
- هر عنصر و روابط آنها باید از قبل مشخص شود.
- مسئولیت (ویژگی ها و متدهای) هر کلاس باید به وضوح مشخص شود.
- برای هر کلاس، حداقل تعداد ویژگی ها باید مشخص شود، زیرا ویژگی های غیر ضروری نمودار را پیچیده می کند.
- هر زمان که لازم است برای توصیف برخی از جنبه های نمودار از یادداشت ها استفاده کنید. در پایان ترسیم باید برای توسعه دهنده/کدنویس قابل درک باشد.
- در نهایت، قبل از ساختن نسخه نهایی، نمودار باید بر روی کاغذ ساده ترسیم شود و تا آنجا که ممکن است دوباره بر روی آن کار شود تا تصویر بهتری درست شود.

اجزای اصلی Class diagram

- **بخش بالایی(Upper section):** شامل نام کلاس است. به صورت پررنگ و وسط چاپ می شود و حرف اول آن بزرگ است.
- **بخش میانی(Middle section) :** شامل ویژگی های(attributes) کلاس است.(از این بخش برای توصیف کیفیت کلاس استفاده کنید. این فقط در هنگام توصیف یک نمونه خاص از یک کلاس مورد نیاز است در واقع ویژگی ها بر روی متغیرها/variables یا data members در کد نگاشت می شوند.) آنها تراز چپ هستند و حرف اول آن کوچک است.
- **بخش پایینی(Bottom section) :** شامل عملیات هایی(متدها/Methods) است که کلاس می تواند اجرا کند(عملیات نحوه تعامل یک کلاس با داده ها را توصیف می کند). آنها همچنین تراز چپ هستند و حرف اول آن کوچک است.

BankAccount
owner : String balance : Dollars = 0
deposit (amount : Dollars) withdrawal (amount : Dollars)

کلاسی با سه بخش

تعیین سطح دسترسی اعضا (Member access modifiers)

همه کلاس ها سطوح دسترسی متفاوتی دارند(visibility). سطوح دسترسی با نمادهای مربوطه در اینجا آمده است:

- Public (+)
- Private (-)
- Protected (#)
- Package (~)

برای مشخص کردن قابلیت مشاهده(visibility) یک عضو کلاس (یعنی هر ویژگی یا متد)، این نمادها باید قبل از نام آن عضو قرار گیرند.

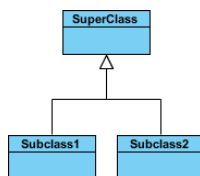
 تعیین سطح دسترسی اعضا (Member access modifiers)

تعیین سطح دسترسی اعضا (Member access modifiers)

روابط (Relationships) در Class Diagram ها

یک کلاس ممکن است در یک یا چند رابطه با کلاس های دیگر درگیر باشد. UML روابط زیر را تعریف می کند:

- **رابطه وراثت (Inheritance) :**



رابطه وراثت (Inheritance)



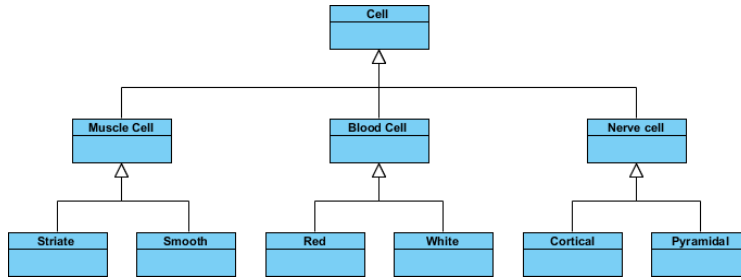
مثال از رابطه وراثت

به نوعی رابطه اشاره دارد که در آن یک کلاس مرتبط به دلیل فرض عملکردهای یکسان کلاس والد، فرزند دیگری است. به عبارت دیگر کلاس فرزند نوع خاصی از کلاس والد است. برای نشان دادن وراثت در نمودار UML، یک

خط ثابت از کلاس فرزند به کلاس والد با استفاده از یک سر پیکان پر نشده رسم می شود.

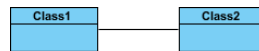
نشان دهنده یک رابطه "is-a" است.

فرض کنید کلاسی داریم به نام Person و کلاسی دیگر به نام Teacher که از کلاس Person مشتق شده است یعنی Teacher یک Person است و اصطلاحاً می گویم: Teacher is a person



مثالی دیگر از رابطه وراثت

• رابطه Simple Association



رابطه Simple Association



مثال از Simple Association


یک اصطلاح گسترده است که تقریباً هر ارتباط یا رابطه منطقی بین کلاس ها را در بر می گیرد. به عنوان مثال، مسافر و شرکت هواپیمایی ممکن است مانند بالا مرتبط شوند

• رابطه Directed Association



مثال از Directed Association

رابطه به جهتی اشاره دارد که با یک خط با نوک پیکان نشان داده می شود. نوک پیکان یک جریان جهت دار را نشان می دهد. می توان گفت که بین یک مسافر و هواپیما ارتباط وجود دارد. هواپیما برای مسافر کار می کند. در اینجا هواپیما برای مسافر کار می کند و نه مسافر برای یک هواپیما.

مثالی دیگر از Directed Association 

مثالی دیگر از Directed Association

یا در تصویر فوق می توان گفت که بین یک شخص و شرکت ارتباط وجود دارد. شخص برای شرکت کار می کند. در اینجا شخص برای شرکت کار می کند و نه شرکت برای یک شخص.

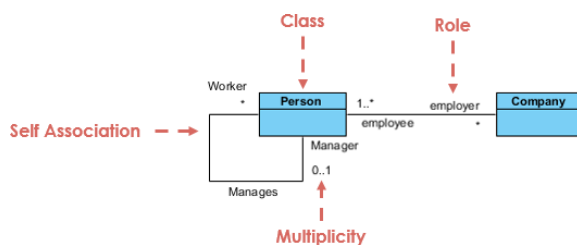
ارتباط یک رابطه ساختاری است که نشان می دهد چگونه دو موجودیت در یک سیستم به یکدیگر مرتبط یا متصل می شوند. می تواند چندین نوع رابطه را تشکیل دهد، مانند یک به یک، یک به چند، چند به یک و چند به چند.

• رابطه Self Association یا Reflexive Association



این زمانی اتفاق می افتد که یک کلاس ممکن است چندین عملکرد یا مسئولیت داشته باشد. به عنوان مثال، یکی از کارکنانی که در فرودگاه کار می کند ممکن است خلبان، مهندس هوانوردی، نگهبان یا خدمه تعمیر و نگهداری باشد. اگر خدمه تعمیر و نگهداری توسط مهندس هوانوردی مدیریت شود، ممکن است در دو نمونه از یک کلاس، یک رابطه مدیریت شده وجود داشته باشد.

بیا یک مثال واضح تر ببینیم



به نظر کاملا مشخص است و نیازی به توضیح اضافه تر ندارد :

• رابطه کثرت (Multiplicity)



به عنوان مثال، یک ناوگان ممکن است شامل چندین هواپیما باشد، در حالی که یک هواپیمای تجاری ممکن است شامل صفر تا تعداد زیادی مسافر باشد. علامت 0..* در نمودار به معنای "صفر تا بسیاری" است.

انواع Multiplicity

0 بدون نمونه (نادر)

1..0 بدون نمونه، یا یک نمونه

1 دقیقا یک نمونه

1..1 دقیقا یک نمونه

*..0 صفر یا بیشتر نمونه

* صفر یا بیشتر نمونه

*..1 یک یا چند مورد

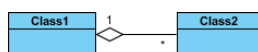
مثال از رابطه چند به چند Many to Many

مثال از رابطه چند به چند Many to Many

یک دانشجو می تواند دروس زیادی را بگذراند و دانشجویان زیادی می توانند در یک دوره ثبت نام کنند.

• رابطه تجمیع (Aggregation)

رابطه Aggregation شکل خاصی از رابطه Association است. در واقع بخشی از رابطه را به تصویر می کشد. این یک رابطه باینری تشکیل می دهد، به این معنی که نمی تواند بیش از دو کلاس را شامل شود. همچنین به عنوان رابطه Has-a شناخته می شود. در این رابطه، یک child می تواند مستقل از parent وجود داشته باشد.

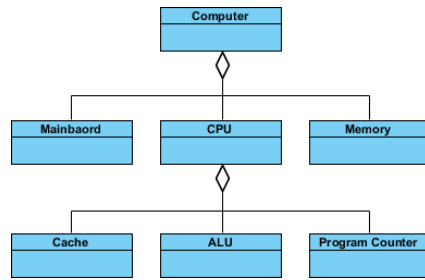


رابطه Aggregation



مثال از Aggregation

این رابطه به تشکیل یک کلاس خاص در نتیجه ادغام یا ساخت یک کلاس به عنوان یک مجموعه اشاره دارد. به عنوان مثال، کلاس "کتابخانه" از یک یا چند کتاب تشکیل شده است. در aggregation، کلاس های موجود قویا به یکدیگر وابسته نیستند. در همین مثال، حتی زمانی که کتابخانه منحل شود، کتاب ها باقی خواهند ماند. برای نشان دادن aggregation در نمودار، یک خط از کلاس والد به کلاس فرزند با شکل الماس نزدیک کلاس والد رسم کنید.



Aggregation ثالی دیگر از

رابطه ترکیب (Composition)



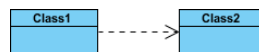
رابطه Composition



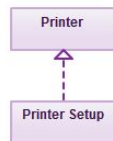
مثال از رابطه Composition

رابطه Composition بسیار شبیه به رابطه aggregation است. تنها تفاوت این است که اشیاء Class 2 با Class 1 زندگی می کنند و می میرند. Class 2 نمی تواند به تنهایی باقی بماند.

• رابطه وابستگی (Dependency) یا Realization



رابطه وابستگی (Dependency)



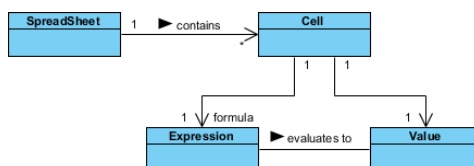
مثالی از رابطه وابستگی

در صورتی بین دو کلاس وجود دارد که تغییرات در تعریف یکی ممکن است باعث تغییر در دیگری شود (اما نه برعکس).

کلاس 1 به کلاس 2 بستگی دارد.

در مثال، تنظیمات برگزیده چاپ که با استفاده از رابط راه اندازی چاپگر تنظیم شده اند، توسط چاپگر پیاده سازی می شوند.

نام های رابطه (Relationship Names)



- نام روابط در وسط خط ارتباط نوشته می شود.
- نام های رابطه خوب زمانی معنا پیدا می کنند که آنها را با صدای بلند بخوانید:
"هر spreadsheet حاوی (contains) تعدادی cell است"
"یک عبارت (expression) به یک مقدار (Value) ارزیابی (evaluates to) می شود"
- آنها اغلب دارای یک نوک پیکان کوچک برای نشان دادن جهت خواندن رابطه هستند، به عنوان مثال،
expression به Value ارزیابی (evaluates to) می شوند، اما مقادیر به عبارت ها ارزیابی نمی شوند.

نقش‌های رابطه (Relationship Roles)

نقش‌ها در انتهای یک خط ارتباطی نوشته می‌شوند و هدف آن کلاس را در رابطه توصیف می‌کنند. به عنوان مثال، در تصویر بالا یک cell به یک expression مرتبط است. ماهیت رابطه این است که بیان فرمول (formula) سلول است.

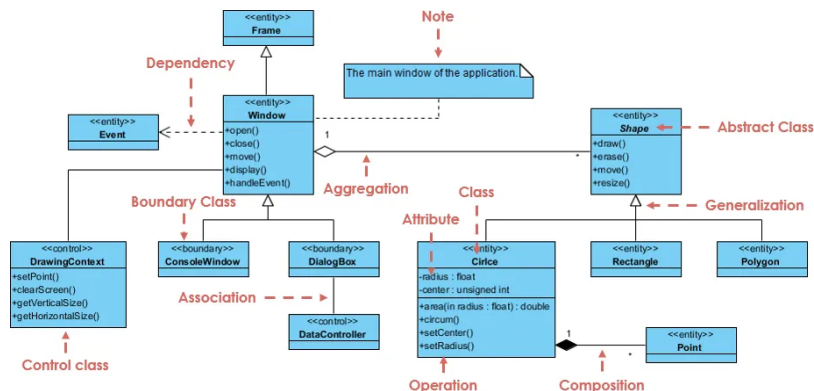
جهت رابطه (Relationship Navigability)

فلس‌ها نشان می‌دهد که آیا با توجه به یک نمونه شرکت‌کننده در یک رابطه، امکان تعیین نمونه‌های کلاس دیگر مرتبط با آن وجود دارد یا خیر.

- تصویر بالا نشان می دهد که با توجه به یک spreadsheet ، می توانیم همه سلول های موجود در آن را پیدا کنیم، اما ما نمی توانیم از روی یک سلول تعیین کنیم که در چه spreadsheet قرار دارد.
- با توجه به یک سلول، می توانیم عبارت و مقدار مربوطه را بدست آوریم، اما با توجه به یک Value یا expression نمی توانیم Cell ی را پیدا کنیم

بررسی یک Class Diagram

یک نمودار کلاس همچنین ممکن است دارای یادداشت هایی باشد که به کلاس ها یا روابط متصل شده اند.



در مثال بالا:

با خواندن نکات زیر می توانیم معنای نمودار کلاس فوق را تفسیر کنیم.

یک: **Shape** یک کلاس abstract است. با حروف کج نشان داده شده است.

دو : **Shape** یک Super Class است. Circle، Rectangle و Polygon از **Shape** گرفته شده اند. به عبارت دیگر، یک Circle یک **Shape** است. این یک رابطه تعمیم / وراثت است.

سه : ارتباطی (association) بین DialogBox و DataController وجود دارد.

چهار **Shape** بخشی از **Window** است. این یک رابطه تجمیع (aggregation) است. **Shape** می تواند بدون **Window** وجود داشته باشد.

پنج : **Point** بخشی از Circle است. این یک رابطه ترکیبی (composition) است. **Point** بدون Circle نمی تواند وجود داشته باشد.

شش : **Window** به **Event** وابسته است. با این حال، **Event** به **Window** وابسته نیست.

هفت : ویژگی های (attributes) دایره (radius) **Circle** و center هستند. این یک کلاس موجودیت (entity class) است.

هشت : نام متدهای **Circle** عبارتند از ()setCenter، ()circum، ()area و ()setRadius.

نه : پارامتر radius در **Circle** یک پارامتر از نوع float است.

ده : متد ()area کلاس **Circle** مقداری از نوع double را برمی گرداند.

یازده : attributes ها و نام متدهای **Rectangle** پنهان هستند. برخی از کلاس های دیگر در نمودار نیز دارای ویژگی ها و نام متدهای خود هستند.

برخورد با سیستم پیچیده - نمودار کلاس چندگانه یا تک؟

به ناچار، اگر شما در حال مدل سازی یک سیستم بزرگ یا یک منطقه تجاری بزرگ هستید، موجودیت های (entities) متعددی وجود خواهند داشت که باید در نظر بگیرید. آیا باید از نمودار چندگانه یا تک کلاسی (Multiple or Single Class Diagram) برای مدل سازی مسئله استفاده کنیم؟ پاسخ این است:

به جای مدل سازی هر موجودیت و روابط آن بر روی یک نمودار کلاس، بهتر است از نمودارهای کلاس چندگانه استفاده کنید.

تقسیم یک سیستم به نمودارهای کلاس چندگانه، درک سیستم را آسان تر می کند، به خصوص اگر هر نمودار یک نمایش گرافیکی از بخش خاصی از سیستم باشد.

ما می توانیم Class Diagram ها را در سه دیدگاه مختلف و به طور تدریجی و با

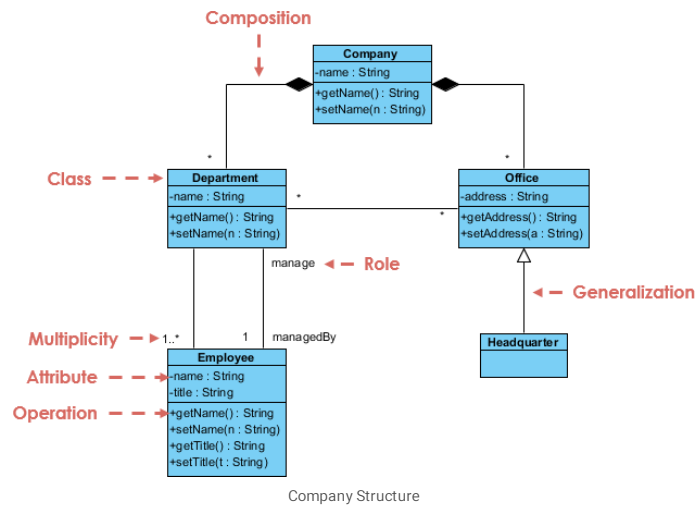
حرکت رو به جلو در مراحل مختلف چرخه عمر توسعه نرم افزار استفاده کنیم:

دیدگاه مفهومی: نمودارها به عنوان توصیف چیزهایی در دنیای واقعی تفسیر می شوند. بنابراین، اگر از دیدگاه مفهومی استفاده کنید، نموداری را ترسیم می کنید که مفاهیم موجود در حوزه مورد مطالعه را نشان می دهد. این مفاهیم به طور طبیعی به کلاس هایی که آنها را پیاده سازی می کنند مربوط می شود. دیدگاه مفهومی مستقل از زبان در نظر گرفته می شود.

دیدگاه مشخصات: نمودارها به عنوان توصیف انتزاعات یا مؤلفه های نرم افزار با مشخصات و رابط ها اما بدون تعهد به یک پیاده سازی خاص تفسیر می شوند. بنابراین، اگر دیدگاه مشخصات را در نظر بگیرید، ما به رابط های نرم افزار نگاه می کنیم، نه پیاده سازی.

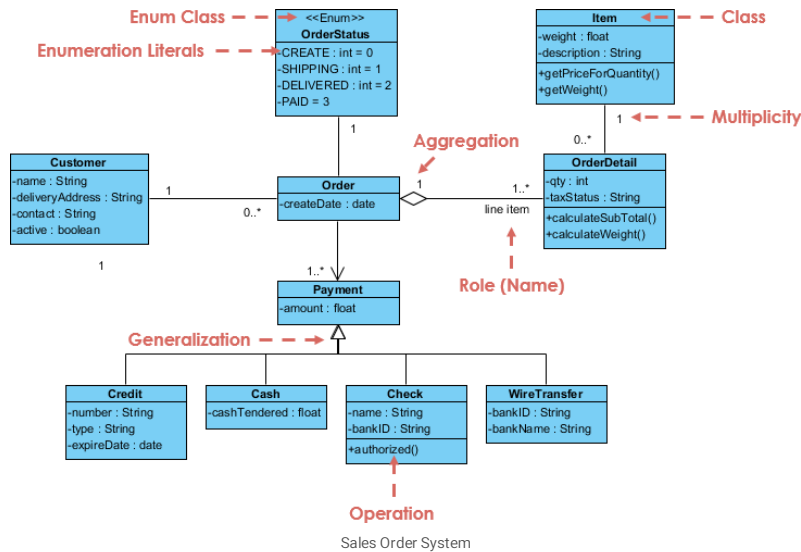
دیدگاه پیاده سازی: نمودارها به عنوان توصیف پیاده سازی نرم افزار در یک فناوری و زبان خاص تفسیر می شوند. بنابراین، اگر دیدگاه پیاده سازی را در نظر بگیرید، ما به پیاده سازی نرم افزار نگاه می کنیم.

در ادامه نمونه هایی از نمودار کلاس را خواهیم دید که به شما نشان می دهد چگونه ساختار سیستم را با یک نمودار کلاس UML مدل کنید. نمودار کلاس UML طرحی از کلاس ها (سطح کد) مورد نیاز برای ساختن یک سیستم نرم افزاری است. برنامه نویسان یک سیستم نرم افزاری را با کمک نمودار کلاس و مشخصات کلاس پیاده سازی می کنند.



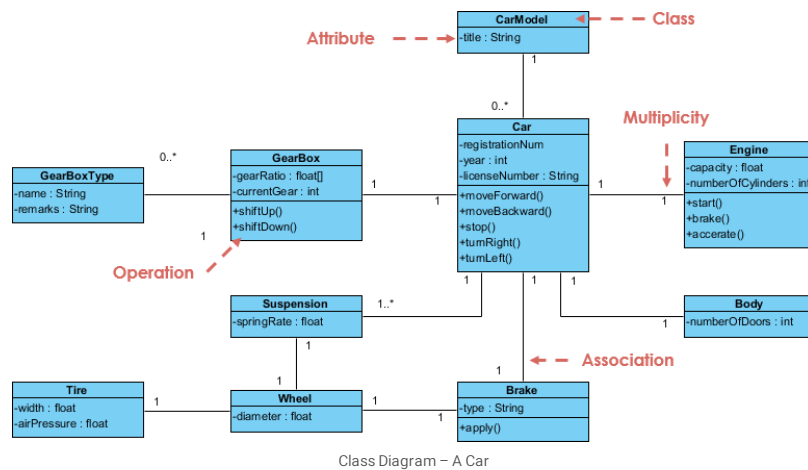
این یک نمونه نمودار کلاس است که نشان می دهد چگونه موجودیت های Company می توانند مدل شوند.

یک شرکت متشکل از بخش ها است. دپارتمان ها در یک یا چند دفتر مستقر هستند. یک دفتر به عنوان یک دفتر مرکزی عمل می کند. هر بخش یک مدیر دارد که از مجموعه کارمندان استخدام می شود. وظیفه شما مدل سازی سیستم برای شرکت است.



نمودار کلاس با نشان دادن کلاس های آن و روابط بین آنها، نمای کلی از یک سیستم را ارائه می دهد. نمودارهای کلاس ایستا هستند -- آنها آنچه را که با هم تعامل دارند را نشان می دهند اما آنچه را که هنگام تعامل اتفاق می افتد را نشان نمی دهند.

این نمودار کلاس یک سفارش مشتری را از کاتالوگ خرده فروشی مدل می کند. کلاس مرکزی Order است. مشتری که خرید و پرداخت را انجام می دهد با آن مرتبط است. پرداخت یکی از چهار نوع است: Cash, Check, Credit or Wire Transfer. سفارش شامل OrderDetails یعنی (line items) است، که هر کدام مورد مرتبط با خود را دارند.



یک خودرو از اجزای ساختاری مختلفی مانند موتور، بدنه، سیستم تعلیق، گیربکس و غیره تشکیل شده است. هر جزء به نوبه خود دارای ویژگی‌ها و عملکردهای خاص خود است. به عنوان مثال، موتور ظرفیت خود را دارد و می‌توان آن را روشن یا خاموش کرد. ما می‌توانیم ماشین را با نمودار کلاس نشان دهیم. مثال نمودار کلاس یک مدل ساختاری ساده شده از یک خودرو را در یک نمودار کلاس نشان می‌دهد.

مهم است که شروع کنید به مدلسازی سیستم فرضی خود اما نیاز به یک ابزار دارید. در اینجا لیستی از محبوبترین‌ها آورده شده است :

- 1) Adobe Spark
- 2) Edraw Max
- 3) Moqups
- 4) Visio
- 5) Lucidchart
- 6) Visual Paradigm
- 7) PlantUML

قسمت دوم آموزش کاربردی UML - قسمت دوم (use case diagrams)

قسمت سوم آموزش کاربردی UML - قسمت سوم (sequence diagrams)

بیشتر بخوانید : نقشه راه توسعه دهندگان Asp.NET Core

<https://zarinp.al/farshidazizi>

oop

شی گرای

زبان مدلسازی یکپارچه

uml

<https://vrgl.ir/xXpWX>



1 نظر

دنبال کردن

فرشید عزیزی

Software Engineer



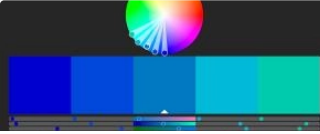
شاید از این نوشته‌ها هم خوشتان بیاید



بر اساس علایق شما

انتشارات در ویترگول

ویترگول
خواندن ۵ دقیقه



مطلبی دیگر در همین موضوع

انتخاب رنگ مناسب در طراحی

علی آجودانیان
خواندن ۶ دقیقه



مطلبی دیگر از این نویسنده

اصول طراحی نرم افزار Software Design Principles

فرشید عزیزی
خواندن ۲ دقیقه

پاسخ‌ها

نظر خود را درباره این پست بنویسید

فرشید جان واقعا مطالب خوبی میداری. من همیشه دنبال می‌کنم

۱