# UML، چرا و چگونه؟

💉 محمدمهدی برقی و سجاد سلطانیان

کمت ر هنرمندی را میتوان یافت که قبل از رسم یک اثر هنری، نمونه اولیه آن را در ذهن و یا روی کاغذ نیاورده باشد. همچنان که هیچ خیاطی نیست که بدون الگو و پیاده سازی طرح روی کاغذ دست به کار شود و تقریبا هیچ معماری وجود ندارد که بدون نقشه کامل و اصولی خشت اول را بنهد. چون قطعا تا ثریا کج میرود! در این نوشته سعی شده است تا به صورت تجربه محور و علمی به بیان چرایی داشتن یک UML خوب بپردازیم و در ادامه، راهکارهایی برای رسیدن به یک نمونه خوب و اصولی از آن معرفی کنیم. به امید اینکه خشت اول را اصولی تر از همیشه بگذارید و یک پروژه خوب را تجربه کنید.

## پروژه در یک نگاه

نه تنها در طول این پروژه بلکه در تمامی پروژههایی که در آینده با آن مواجه میشوید، لازم است چندین بار آنها را برای افراد مختلف، از کارفرما گرفته تا راهنمای پروژه و همتیمیهایتان توضیح دهید. ارائله یک پروژه، بدون داشتن UML درست و اصولی، شبیه بیان آرزو و خیالاتمان از نتایجی است که میخواهیم حاصل شود؛ بدون راه حل عملی و فکر شده! یادمان نرود، تا زمانی که یک ایده فقط در ذهن ما باشد، فقط در ذهن ما باشد، فقط در ذهن کا است! و تا وقتی آن را از ذهن به کاغذ نیاوریم متولد نخواهد شد. در واقع، یک UML کامل، این امکان را به شما میدهد که ایده و طرح خود را به صورت ملموس و قابل درکی برای خود و دیگران توضیح دهید.

## جایی برای اندیشیدن

شــاید بتـــوان گفــت کــه از جملــه مهم تریـــن ویژگیهــای فــاز صفــر و طراحــی نمـــودار UML، اعمــال تغییــرات، بــه ســادگی جابهجایـــی چنـــد شــکل و کلمــه اســت، نــه تغییــر صدهــا خــط کــد!

ایــن موضــوع باعــث میشــود تــا ایــن مرحلــه از طراحــی بــه بهتریــن محــل بــرای ایــده پــردازی و بحــث بیــن اعضــای تیــم بــرای چالشهــای پــروژه باشــد و بتواننــد ســاعتها بــا صــرف کمتریــن هزینــه، بــه بهتریــن ســاختار بــرای پیادهســازی پــروژه برســند.

ایــن ویژگـی را اصــلا دســت کــم نگیریــد چــون هــر چــه پیشتــر برویــم، تغییــر دادن ســاختار و طراحــی پــروژه دشــوارتر خواهــد شــد؛ تــا جایــی کــه بــه یــک امــر ناممکــن تبدیــل میشــود و هــر تغییــر بــه ظاهــر کوچــک ممکــن اســت بــه قیمــت بازنویســی و پــاک کــردن دههــا و صدهــا خــط کــد تمــام شــود.

## چه حکمتی است که در آغاز، نگاه من به سرانجام است؟

نمـودار UML، بـه شـما یـک نگـرش و دیـد کلـی از پــروژه و کــد نهایــی آن میدهــد حتـی بــدون آنکـه یـک خـط کــد زده باشــید! در طراحـی UML، مـا قــرار اســت، قســمت اصلـی فکــر کــردن و بررســی ایدههـای مختلـف را روی کاغــذ انجــام بدهیــم و در فازهــای بعــدی، کار مــا تبدیــل ایــن طراحــی بــه کــد خواهــد بــود.

و بهتریـــن اتفـــاق ممکــن بــرای پــروژه شــما ایــن اســت کــه بدانیــد در نهایـت قــرار اســت بــه کجــا برســد و عناصــر مختلــف آن چگونــه بــا یکدیگــر ارتبــاط برقــرار کننــد.

## که چه؟

داشتن یک UML کامل علاوه بر تمام مزایایی که گفته شد دو ویژگی بسیار مهم نیز دارد. تا وقتی دید کلی و جامعی به پروژه نداشته باشیم، تقسیم وظایف معمولا به دو صورت انجام میشود: فرایند محور، به این معنا که مثلا یک نفر بخش لاگین و یک نفر قسمت گیم پلی را پیادهسازی کند، یا در حالت دیگر به صورت کلاس محور، به این معنا که مثلا یک نفر کل کلاس یوزر و متعلقات آن را و دیگری کلاس نیروهای داخل بازی را بزند!

طبیعــی اســت کــه هــر دو روش معقــول نیســت و مشــکلاتی را ناشــی می شــو ند .

اولیــن مشــکل آن اســت کــه قبــل از طراحــی UML، دشــواری و بــار کاری هــر قســمت مشـخص نیســت و لــذا معیــار مناســبی بــرای تقســیم کارهــا نداریــم.

دلیــل دوم، وابســتگی قســمتهای مختلــف پــروژه و اتصــال آنهــا بــه یکدیگــر اســت. بــه طــوری کــه مثــلا کلاس user بایــد بــا ده جــای دیگــر مرتبــط باشــد و تــا وقتــی کــه ایــن ارتبـاط بیــن اجــزای برنامــه، مشــخص نشــده باشــند، پیادهســازی مســتقل و مجــزای کلاسهــا و توابــع، مثــل قرارگیــری تعــداد زیــادی آجــر و مشــتی ســیمان اســت کــه دیـــواری را تشــکیل ندادهانـــد!

مزیت دیگر ترسیم UML آن است که با داشتن یک دید جامع از کل پروژه، اعضای تیم به نیازهای هر قسمت و هر فرایند از بازی مطلع هستند و میدانند که تکه کدی که در حال نوشتن آن هستند قرار است در کجا و چگونه میورد استفاده قرار بگیرد. در نتیجه داشتن ایس آگاهی، برای قسمتهای مختلف برنامه به نحوی برنامه ریزی و کدنویسی میکنند که در آینده و در فرایندهای آتی دچار مشکل نشوند.

## و اما بعد...

تــا اینجـــای کار ســعی در تفهیـــم و بیـــان اهمیــت وافــر طراحــی UML و فــاز صفــر پــروژه داشــتیم کــه حاصــل تجربــه و ســوز دلهــای نســلهای برنامــه نویســی پیشــرفته بودنــد.

از اینجــا بــه بعــد ســعی شــده تــا از بعــد علمــی بــه مســئله بپردازیــم و از دیـــد یــک مهنــدس نرمافــزار، ویژگیهـــا و راههـــای طراحــی یــک UML خــوب را بیــان کنیــم.

## ِ باز هم باگ!

در دنیای کامپیوتــر هــر چیــزی باگــی دارد و بــاگ در نمــودار UML آن اســت کــه نســبت بــه ســناریوهای برنامــه و برخــی قســمتهای آن غافــل باشــید. میتــوان گفـت مهــم تریــن وظیفــه شــما، تصــور و شبیهســازی ســناریوهای مختلــف یـک برنامــه اســت تــا نیازمندیهــای آن را شناســایی کنیــد و بــرای هــر کــدام از نیازهــا چــارهای بیندیشــید.

در ادامــه بــه بررســی برخــی راهکارهــا بــرای کشــف ایـــن نیازمندیهــا میپردازیـــم.

#### قصه کاربر (user story)

خلاصــه بگوییــم! خودتــان را بــه جــای کاربــر قــرار دهیــد و خاطــره اســتفاده از برنامــه تــان را مــو بــه مــو بنویســید. شــاید خنــدهدار یــا بیهــوده بـه نظـر برســد امــا بــاور کنیــد دیــدی کــه از ایــن طریــق نســبت بــه نــرم افزارتــان پیــدا خواهیــد کــرد در رفـع بســیاری از مشــکلات شــما بــه کمکتــان خواهــد آمــد.

#### ی بسا کاری که اول صعب گشت!

حـالا بــه عنــوان يــک برنامهنويــس خبــره بــه ســراغ برنامــه برويــد و

سـعی کنیـد ابتـدا داک پـروژه را کامـل مطالعـه کنیـد و قسـمت هایـی کـه بـه نظرتــان چالشبرانگیــز خواهــد بــود را مشـخص کنیــد! توجـه کنیــد کـه خــوب اســت ایــن کار را هــر یــک از اعضــا انجــام دهنــد. از مشـخص کــردن چالــش هــا میتواننــد مشـخص کــردن چالــش هــا میتواننــد در نظــر هــر کســی از ســادهترین مــوارد تــا پیچیــده تریــن قســمتهای بــازی متغیــر باشــند!

حـالا وقـت جلسـه و همفکـری اسـت، بـرای ایـن فــاز بایــد تقریبــا هــر روز باهــم جلســه داشــته باشــید و بــرای کل پــروژه از جملــه چالشهــا ایدهپــردازی کنیــد و بــا هــم بحـث کنیــد! (دقــت کنیــم کــه حجــم زیــادی از یادگیــری مــا بــه همیــن بحثهــا وابســته اســت.)

مطمئین شـوید کـه تمامـی اعضـای تیـم در نهایـت بعـد از ایـده دادن و انتخـاب یـک راه حـل نهایـی، دیـد و برداشـت یکسـانی از راه حـل دارنـد و کل تیـم میداننـد کـه قـرار اسـت آن مشـکل چطـور حـل شـود.

#### کلاس بندی کنید

حـالا شــما برنامــهای کــه قــرار هســت بنویســید را یــک بــار از دیــد کاربــر و یــک بــار از دیــد برنامهنویــس مــرور کردهایــد. امیدواریــم تــا اینجــای کار توانســته باشــید نیازمندیهــا و چالشهــای مســیر را درک و راه حــل مناســبی بــرای آنهــا پیــدا کنید.(توجــه کنیــد کــه بــا هــر بــار بررســی مراحــل بــالا، دیــد عمیقتــر و شــفافـتری از نتیجــه نهایــی پــروژه کســب خواهیــد کــرد.)

حال وقت آن رسیده که با ساده ترین روش، مسئله را روی کاغذ بیاورید و کلاسها، متغیرها و توابعی که نیاز دارید را ایجاد کنید. در ایس مرحله، روی هر تابع، کلاس و متغیری که مینویسید بازنگری کنید چون ممکن است مواردی که برای مسئلهتان مشخص کردهاید، برخی از چالش و نیازهای شما را پوشش ندهند. پس سعی کنید به صورت تیمی چندین مرتبه کلاسها، روابط بیان آنها، پارامترها و توابعتان را مرور کنید تا علاوه بر رسیدن به یک دید جامع و مشترک در تیم، برای گام بعدی پروژه آماده شوید!

#### به وقت طراحي

در مرحلـه آخــر و حتــی در هنــگام مــرور کــردن، هــم بــرای داشــتن نظــم بیشــتر و داشــتن یــک نقشــه راه UML ای کــه بــر روی کاغــذ طراحــی کردید را بـا اســتفاده از نــرم افزارهـا و ســایت هایــی مثــل:

- Software ideas modeler <
  - وب سایت draw.io
  - وب سایت lucidchart
    - Microsoft visio <

و این قبیل موارد طراحی و پیاده سازی کنید.

خـب! همـه چیـز حاضـر اسـت. پـروژه شـما تقریبـا تمـام شــده و شــما آن را یکبـار بـا تمـام جزئیـات داخـل ذهنتـان سـاختهاید. فقـط مانــده نوشــتن کدهـا بــر اســاس ایــن نقشــهراهی کــه طراحــی کردهایــد.

امیدواریــم ایــن متــن توانســته باشــد اهمیــت UML را بــه شــما برســاند و جــای هیــچ حســرتی بــرای UML بهتــر در آخــر پــروژه را برایتــان باقــی نگذاشــته باشـــد!

# قواعد پیشرفته UML

سید پارسا نشایی

شــما از طریــق کلاسهــای حــل تمریــن و نیــز انجــام تمریــن دوم درس برنامهســازی پیشــرفته، بــا بســیاری از نــکات مهــم UML آشــنا هســتید، ماننــد:

- 🔾 نحوهی جداسازی نام کلاس، خواص و متدها
- نحوهی مشخص کردن سطح دسترسی و static بودن

در طراحــی یــک پــروژه، مشــخص کــردن «ارتبــاط» میــان کلاسهــا از اولویــت فراوانــی برخــوردار اســت. در ایــن مقالــه، بــا نکاتــی در ایــن رابطــه، آشــنا میشــوید.

## قواعد نوشتن «نام کلاس»

نــام کلاس بایــد بــه شــکل PascalCase در بــالا و وســط نوشــته شــود. اگــر کلاس از نــوع abstract اســت، اســم آن بایــد italic نوشــته شــود.

## ِ قواعد نوشتن خواص (property ها) و متدها

ابتـدا بایــد ســطح دسترســی خاصیــت مشــخص شــود (+ بــرای package level) و ســپس – بــرای protected) و ســپس امــده و در نهایــت پــس از یــک دو نقطــه (:)، نــوع آن مشــخص میشــود. اگــر فیلــد از نــوع static اســت، زیــر کل عبــارت بایــد یک underline کشــیده شــود. قواعــد نوشــتن متدهــا نیــز مشــابه خــواص اســت، بــا ایــن تفــاوت کــه پارامترهــای ورودی نیــز بایــد مشــخص شــوند. بــه مثــال زیــر توجــه کنیــد:

Shape
-length: int
+getLength(): int
+setLength(n: int): void

## روابط میان کلاسها و نحوهی مشخص کردن آنها در UML

#### رابطەي وابستگى

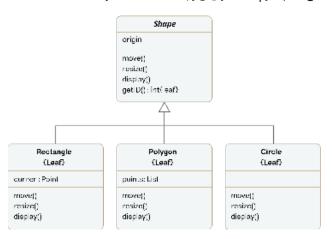
زمانــی کــه دو کلاس از نظــر مفهومــی بــه هــم وابســتهاند و تغییــر در خــواص یــک کلاس، ممکــن اســت خــواص کلاس دیگــر را تغییــر دهــد و یــا آن را دچــار مشــکل کنــد؛ بــه عنــوان مثــال، کلاسهــای Car و Wheel در شــکل زیــر:



نحـوهی نمایـش ایـن نـوع از رابطـه، یـک فلـش خطچیـندار از کلاسـی اسـت کـه نیازمنــد کلاس دیگــر اسـت، بـه ســمت کلاســی کــه نیــازش را برطــرف میکنــد. در زیــر ایــن خــط چیــن، ‹‹use›› نوشــته میشــود.

### ابطهی تعمیم و وراثت

هــر جــا کــه کلاســی از پــدرش ارث میبــرد (یــا بــه عبــارت دیگــر، extend میکنـــد)، لازم اســت از ایــن رابطــه اســتفاده شــود.



نحـوهی نمایـش ایـن رابطـه، یـک فلـش اسـت کـه انتهـای آن، مثلثـی توخالـی قــرار دارد. در مثــال فــوق، همــهی انــواع Polygon، Circle یــا Rectangle (کــه یــک کلاس abstract اســت و در نتیجــه italic نوشــته شــده) هســتند و در نتیجــه، رابطــهی میــان آنهـا، از نــوع وراثــت در کلاسهــای درس، بیشتــر آشــنا خواهیــد شــد.

توجــه کنیــد کــه ایــن رابطــه تنهــا معــادل extend کــردن اســـت و نــه معـــادل implement کـــردن کـــه در interface هـــا اســـتفاده میشـــود؛ بــرای واســـطها، رابطــهای دیگــر بـــه زودی گفتــه خواهـــد شـــد.

## رابطهی تجمعی (Aggregation)

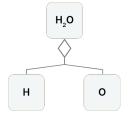
بســياری از روابــط، رابطــهی میــان «کل» و «اجــزا»ی یــک پدیــده را نشــان میدهنــد؛ مثــلا رابطــهی «دانشــگاه» و «دانشجـو». دیدیــم اگــر ایــن رابطــه بــه حــدی شــدید باشــد کــه تغییــر در یکــی، دگرگونــی بســیار دیگــری را نتیجــه دهــد (مثــلا اگــر بخواهیــم از نــوع دیگــری از چــرخ اســتفاده کنیــم، طراحــی ماشــین دگرگــون میشــود)، از رابطــهی وابســتگی اســتفاده میشــود، امــا اگــر ایــن رابطــه بــه گونــهای باشــد کـه کلاسهــا کامــلا بــه هــم وابســته نباشــند، از رابطــهی تجمعــی بهــره گرفتــه میشــود.



نحـوهی نمایـش ایـن رابطـه، یـک خـط بـا یـک شــکل هماننــد لـوزی توخالــی اســت.

#### رابطهی ترکیب (Composition)

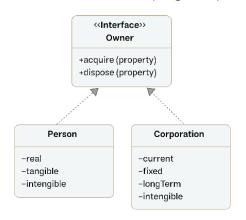
ایــن رابطــه، نســخهی قویتــر رابطــهی تجمعــی اســت، بــا ایــن تفــاوت کــه هیچیــک از دو کلاس دو طــرف رابطــه، بــدون هـــم معنایـــی ندارنــد. در رابطـــهی «وابســـتگی»، تنهـــا پـــدر بـــدون فرزنـــد بیمعناســـت و فرزنــدان بــدون پــدر معنـا دارنــد؛ امـا در ایــن رابطــه (ترکیــب)، فرزنــدان نیــز بــدون پــدر معنایــی ندارنــد.



در مثـــال فـــوق، اگـــر در نظــر بگیریـــم کــه برنامهمـــان تنهـــا بــا «مولکول»هــا کار دارد و نــه بــا اتمهـا، آنگاه H و O بــدون مولکولــی کــه شــامل آنهاست، معنایــی ندارنــد و خــود مولکــول نیــز بــدون اجزایــش، معنایــی نــدارد. ایـــن رابطــه بــا یــک لــوزی توپــر روی خــط نشـــان داده میشـــود.

### رابطهی تفهیم یا تحقق (Realization)

از ایــن رابطــه، بــرای ارتبــاط دادن کلاسهایــی بــه یــک کلاس دیگــر اداشــته اســتفاده میشــود کــه وظیفــهی تکمیــل کــردن آن کلاس دیگــر را داشــته باشــند. بــرای ارتبــاط enum هایــی کــه در یــک کلاس اســتفاده میشــوند بــه آن کلاس، از ایـــن رابطــه بهـــره گرفتــه میشـــود. همچنیـــن، کلاسهایــی کــه یــک ایــن interface کلاسهایــی کــه یــک interface رابطــه بــه کمــک ایــن وابطــه بــه عمـــود. نحــوهی نمایــش interface رابطــه بــه عبـارت «enum در نمــودار UML، مشــابه کلاس اســت، بــا ایــن تفــاوت کــه عبـارت «Interface»، همانطــور کــه در شــکل مشـخص اســت، در بــالای box نوشـــته میشــود.



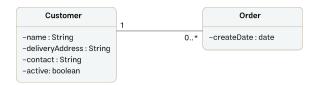
نحـوهی نمایــش ایــن رابطــه، یـک فلــش خطچیــندار بــا یــک مثلــث توپــر اســت.

#### رابطهی عادی (Normal)

اگــر رابطــهی میـــان دو کلاس در هیچیــک از حـــالات فـــوق جـــای نمیگرفــت، از رابطــهی نرمــال اســتفاده میکنیـــم کــه شــامل یــک خــط بـــدون هیــچ نمــادی اســـت.

#### چندی (multiplicity) روابط

میتوانیــد تعــداد مواقعــی کــه هــر کلاس نیازمنــد دیگــری اســت را بــا یــک عــدد روی خــط نمایــش دهیــد؛ مثــلا هــر customer میتوانــد تعــداد زیــادی order داشـــته باشــد، امــا هـــر order تنهــا بــه یــک customer مرتبــط اســـت. در ایـــن حالــت، در خــط میـــان دو کلاس Customer و Order، دو عــدد مینویســـیم:



عــدد ۱ روی خــط یعنــی هــر ســفارش فقــط یــک مشــتری دارد، و عــدد 
۰..\* یعنــی هــر مشــتری، از صفــر تــا بینهایــت ســفارش دارد؛ اگــر مثــلا 
ماکســـیمم ۴ ســفارش ممکـــن بـــود، میتوانســـتیم از ۶...۶ اســـتفاده 
کنـــم.