به نام خداوند بخشنده‌ی مهربان

شماره‌ی ۵ ۱۳ فروردین ۱۴۰۰

# کد نامه

ویژه‌ی دانش‌جویان برنامه‌سازی پیشرفته نیم‌سال دوم ۱۴۰۰-۱۳۹۹ دانشکده‌ی مهندسی کامپیوتر دانشگاه صنعتی شریف

در این شماره از کدنامه، می‌خوانید:


چگونگی هم‌کاری تیمی به کمک Trello

مدیریت نیازهای پروژه با Maven

قواعد پیشرفته‌ی UML



## هم‌کاری تیمی و انجام پروژه

### پروژه و تیم خود را به خوبی و با ابزارهای مناسب، مدیریت کنید

با این که ممکن است هم‌کاری تیمی در انجام پروژه، به خصوص در دوران شیوع ویروس کرونا و آموزش مجازی، سخت و حتی غیرممکن به نظر برسد، می‌توان از ابزارهایی برای پیش‌برد موثر کارهای تیمی به صورت مجازی استفاده کرد، به گونه‌ای که مطمئن باشیم هر یک از اعضای تیم، کار خود را به درستی انجام می‌دهند و پروژه در مسیر صحیح خود پیش می‌رود. با کدنامه همراه باشید.

## آیا محدودیت «حداکثر ۳۰ خط کد در هر تابع» برای تمیزی توابع کافی است؟

همان‌طور که احتمالا حدس زده‌اید، هدف از این محدودیت، تک‌کاربره کردن توابع بوده است؛ در نتیجه، اگر تابعی این محدودیت را رعایت کرده باشد، اما چندین کار را با هم انجام دهد. (که در صورت کوتاه و مختصر بودن کارها، کاملا محتمل است)، لازم است با وجود آن‌که محدودیت عددی («کمی») تعداد خط در تابع را رعایت کرده، محدودیت «کیفی» آن (که انجام دادن یک و تنها یک عمل در تابع است) را نیز رعایت کند. توجه کنید که رعایت این نکته، به خصوص در main، بسیار مهم است؛ main **نباید** به غیر از صدا زدن توابع دیگر برنامه، کار دیگری انجام دهد.

# چگونگی هم‌کاری تیمی به کمک Trello

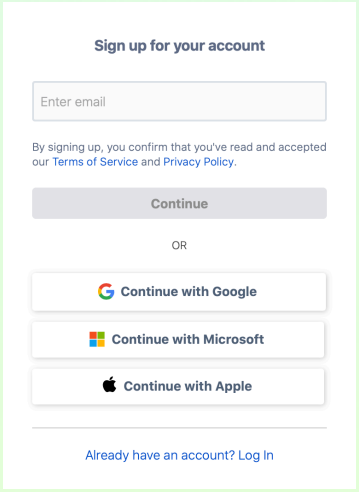
**سید پارسا نشایی**

## چرا به یک نرم‌افزار مدیریت task ها نیازمندیم؟

وقتی یک تیم در یک شبکه‌ی اجتماعی اقدام به هماهنگی و پیش‌برد فعالیت‌ها می‌کند، احتمال گم شدن پیام‌های مهم بر اثر چت‌های متعدد در گروه، بسیار زیاد است. برخی از شبکه‌های اجتماعی، امکان «تگ‌گذاری» پیام‌ها را در اختیار اعضای تیم قرار می‌دهند؛ با این حال، هم‌چنان محدودیت‌های زیادی در استفاده از این امکانات وجود دارد و در یک نگاه، روند کلی و میزان پیش‌برد پروژه، مشخص نخواهد بود. پروژه‌هایی مانند پروژه‌ی درس برنامه‌سازی پیشرفته، شامل اجزا و ماژول‌های بسیار زیادی هستند که معمولا اعضای تیم، پیاده‌سازی آن‌ها را میان خود تقسیم می‌کنند؛ در صورت عدم یک‌پارچگی تیمی، پروژه به خوبی ساخته نخواهد شد و در نسخه‌ی نهایی، دارای باگ‌ها و اشکالات متعددی خواهد بود. در نتیجه، استفاده از یک نرم‌افزار مدیریت task ها در تیم، امری ضروری است. در این مقاله، به توضیح مقدماتی یکی از مشهورترین نرم‌افزارها در این حوزه به نام Trello خواهیم پرداخت.

## شروع به کار و تشکیل پروژه

برای شروع به کار، ابتدا وارد trello.com شده و از بخش بالا و سمت راست، گزینه‌ی Sign Up را انتخاب کنید و به روش دل‌خواه، در سایت ثبت‌نام کنید؛ سپس به سایت وارد شوید.




پس از ورود به سایت، به مکانی می‌رسید که می‌توانید board های خود را ببینید و یا board جدید بسازید. هر board مکانی برای نگه‌داری و مدیریت امور مربوط به یک پروژه است؛ در نتیجه، لازم است در ابتدای کار، یک board برای پروژه‌ی خود ایجاد کنید. هنگام ساخت board، می‌توانید آن را private (خصوصی و تنها برای اعضای تیم که شما اجازه‌ی دسترسی آن‌ها را صادر می‌کنید) و یا public (برای تمامی کاربران Trello) تعریف کنید. سپس، لازم است اعضای تیم و نیز راهنمای پروژه را به board خود دعوت کنید.

## لیست‌ها

در هر پروژه (board)، task های مورد نظر، سه حالت اصلی دارند که task های مربوط به هر یک از سه حالت، باید در لیست مربوط به خود قرار گیرند:

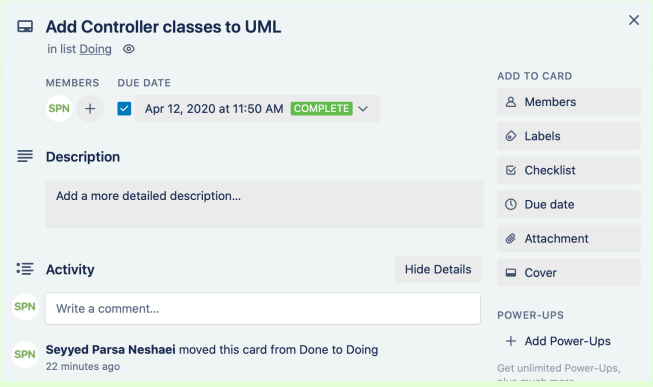
- To Do: برنامه‌ریزی برای آن‌ها انجام شده، ولی هنوز کاری در خصوص task مد نظر انجام نشده است.
- Doing: فردی که وظیفه‌اش انجام task مذکور بوده، شروع به انجام task کرده است.
- Done: مسئول انجام task، آن را به پایان رسانده است.

می‌توان task ها را به راحتی و با کشیدن و رها کردن، میان لیست‌ها جابه‌جا کرد.

## افزودن task ها

برای افزودن task به لیست مدنظرتان، لازم است بر روی گزینه‌ی Add (another) card در پایین لیست کلیک کنید. پس از زدن روی این دکمه، کادر کوچکی باز می‌شود که باید عنوان اصلی task را به طور خلاصه در این بخش وارد کنید و سپس دکمه‌ی Add Card را بزنید. سپس لازم است جزئیات کارت اضافه شده را مشخص کنید؛ برای این کار، نشانگر ماوس را به گوشه‌ی سمت راست کارت برده و دکمه Edit (که با آیکون مداد قرار دارد) را انتخاب کرده و سپس Open Card را بزنید. در این صفحه، می‌توانید موارد زیر را برای کارت خود مشخص کنید:

- Members: نام کسانی که وظیفه‌ی انجام task مدنظر را به عهده دارند
- Labels: تگ‌گذاری روی card ها؛ به عنوان مثال، می‌توانید تگ‌هایی به رنگ‌های گوناگون، برای تفکیک «طراحی Model و کلاس‌ها»، «طراحی گرافیک (رابط کاربری)»، «طراحی سرور»، «طراحی کلاینت» و موارد مشابه از یک‌دیگر تعریف کنید.
- Checklist: تعریف زیرtask ها؛ اگر یک task پیچیده باشد، می‌توانید به کمک این گزینه، چند زیرtask برای task خود تعریف کرده و آن‌ها را به افراد مختلف اختصاص دهید.
- Due date: مهلت انجام task؛ در این بخش، می‌توانید deadline انجام task را مشخص کنید؛ Trello هنگام نزدیک شدن به ددلاین، اگر task مربوطه را انجام نداده باشید، یک ایمیل هشدار برای شما خواهد فرستاد. پس از فرارسیدن ددلاین، در صورت اتمام انجام task، لازم است که تیک جلوی آن را بزنید.
- Attachment: فایل‌های ضمیمه؛ اگر توضیحات بیش‌تری برای task مدنظر لازم است (به عنوان مثال، اسکرین‌شات بخشی از داک پروژه که مربوط به آن task است)، می‌توانید آن را در بخش attachment وارد کنید.
- Description: توضیحات بیش‌تر در خصوص task مدنظر
- Activity Comments: کامنت‌ها؛ در صورتی که پیش‌بردی در انجام task مدنظر انجام شده باشد، می‌توانید در این بخش، گزارش خود را از پیش‌برد ارائه کنید تا گزارش، مستند باشد و در میان چت‌های پیام‌رسان‌ها، گم نشود.




**چگونه، کار تیمی خود را مفید پیش ببریم؟**

- در ابتدای عرضه فاز پروژه، اعضای تیم به دقت داک را مطالعه کنند.
- یک جلسه‌ی آنلاین میان اعضای تیم تشکیل شود و task هایی که باید انجام شوند، مشخص شوند؛ این task ها باید با جزئیات کامل، تاریخ ددلاین دقیق و نیز فردی که مسئولیت انجام task به عهده‌ی او است، در لیست To Do در Trello وارد شوند.
- اعضای تیم، کار روی task ها را آغاز می‌کنند؛ لازم است اعضا در پایان هر روز، گزارشی از اقدامات انجام شده را ترجیحا در بخش کامنت‌های هر کارت در Trello وارد کنند.
- هر چند مدت یک بار، لازم است اعضای تیم جلسه‌ی آنلاین مجددی داشته و درباره‌ی هماهنگی ماژول‌های مختلف پروژه با یک‌دیگر و نیز چالش‌های احتمالی در انجام پروژه، با هم صحبت کنند.
- ددلاین task ها، لازم است با فاصله‌ی زمانی معقولی قبل از ددلاین فاز پروژه تعیین شود. توصیه می‌شود که در ۷ الی ۱۰ روز آخر، به دیباگ و اشکال‌زدایی پروژه بپردازید و بهتر است تا آن زمان، افزودن امکانات و ویژگی‌های ذکر شده در داک پروژه، به اتمام رسیده باشد. البته، می‌توانید در موارد محدودی، پیاده‌سازی برخی از بخش‌های امتیازی پروژه را برای ۱۰ روز انتهایی در نظر بگیرید.
- در طول روند پیاده‌سازی پروژه، دستیار آموزشی راهنمای پروژه، همواره در کنار شماست؛ در صورت برخوردن به چالش و یا بروز مشکل در انجام پروژه و هماهنگی تیمی، می‌توانید همواره با راهنمای تیم مشورت نمایید.

# مدیریت نیازهای پروژه با Maven

**علیرضا حسین‌خانی**

زمانی که در حال نوشتن یک پروژه‌ی برنامه‌نویسی نسبتا بزرگ هستیم، لزومی ندارد که از صفر تا صد نیازهای برنامه را خودمان بنویسیم! بسیاری از نیازها بین پروژه‌ها مشترک هستند و روش بهتر، آن است که تا حد امکان، از کدهای آماده‌ی استاندارد که قبلا برای نیازهای عمومی توسط سایر برنامه‌نویس‌ها نوشته شده‌اند (که آن‌ها را «کتاب‌خانه» می‌نامیم) استفاده کنیم تا زمان کمتری از ما گرفته شود و بتوانیم روی ویژگی‌های اختصاصی پروژه‌ی خود تمرکز کنیم. برای استفاده از این کدهای آماده، دو روش وجود دارد:

- به صورت دستی یک کتاب‌خانه را (از مکان‌هایی هم‌چون GitHub) دانلود و نصب کنیم.
- از ابزارهایی برای مدیریت خودکار این کتاب‌خانه‌ها استفاده کنیم.

طبعا روش دوم بسیار ساده‌تر، بهتر و زیباتر است، چرا که برخی کتاب‌خانه‌ها، خود به کتاب‌خانه‌های دیگری نیاز دارند و به کمک روش دوم، مدیریت کتاب‌خانه‌های مورد نیاز به سادگی انجام می‌شود. هم‌چنین، در صورت استفاده از این روش، می‌توان تمام کتاب‌خانه‌های شخص ثالث استفاده شده در برنامه‌مان را یک‌جا و در یک لیست (فایل) مشاهده کرده و در صورت تمایل، به آسانی هریک از آن‌را را حذف و یا به‌روزرسانی کرد.

Apache Maven، یکی از ابزارهای ساخت و مدیریت پروژه‌های جاوا است. یکی از معمول‌ترین کاربردهای Maven، مدیریت کتاب‌خانه‌های لازم برای نرم‌افزارهای جاوا است؛ برنامه‌نویس، به Maven اعلام می‌کند که کتابخانه‌ای با نام x و نسخه‌ی y را لازم دارد و سپس دانلود و نصب کتاب‌خانه توسط Maven انجام خواهد شد.

## روش استفاده از Maven در IntelliJ

برای راه‌اندازی Maven، مراحل زیر را به ترتیب، طی کنید:

- در دیالوگ New Project که هنگام ساخت پروژه‌ی جدید آن را مشاهده می‌کنید، گزینه‌ی Maven را در سمت چپ انتخاب کنید و سپس روی گزینه‌ی Next کلیک کنید.
- در صفحه‌ی بعدی، باید نام پروژه و مسیر ایجاد آن و در قسمت Artifact Coordinates اطلاعاتی درباره‌ی شرکت سازنده نرم‌افزار و نسخه‌ی نرم‌افزار وارد کنید. این اطلاعات، بعدا قابل ویرایش هستند.
- پس از ورود اطلاعات، روی گزینه‌ی Finish کلیک کنید.

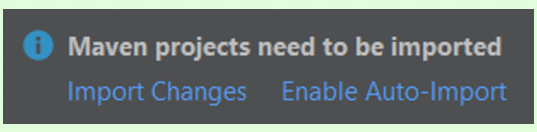
اگر تمامی مراحل قبل را به درستی انجام داده باشید، پروژه برای شما ایجاد شده و فایلی با نام pom.xml تشکیل می‌شود که در آن اطلاعاتی درباره‌ی پروژه به زبان xml نوشته شده است. برای افزودن کتاب‌خانه‌ها یا اصطلاحا Dependency های مورد نیاز برنامه، ابتدا باید یک تگ dependencies در تگ project به فایل pom.xml به صورت زیر اضافه کنیم. به خطوط ۱۰ تا ۱۲ دقت کنید:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/P
         xmlns:xsi="http://www.w3.org/200
         xsi:schemaLocation="http://maven
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>untitled3</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>

    </dependencies>
</project>
```

از این پس، باید قطعه کد مربوط به هر کتاب‌خانه را در تگ dependency ای که ساختیم اضافه کنیم. قطعه‌کد کتاب‌خانه‌ها به صورت آماده در این سایت موجود هستند. می‌توانید به کمک جست‌وجو در گوگل نیز قطعه‌کدهای لازم برای افزودن کتاب‌خانه‌های مختلف به پروژه‌ی خود را پیدا کنید. از الگوی «نام کتاب‌خانه + maven» برای جست‌وجو در گوگل استفاده کنید؛ مثلا برای افزودن کتابخانه‌ی Gson (که بعدا با آن آشنا می‌شوید)، عبارت Gson maven را جست‌وجو کنید و وارد وب‌سایت mvnrepository شوید؛ سپس از میان صفحه، نسخه‌ی مورد نیاز کتاب‌خانه را انتخاب کرده و در صفحه‌ی باز شده، قطعه کد مربوط به Maven را کپی کرده و در تگ dependencies که ایجاد کرده بودیم، paste کنید:

```xml
    <dependencies>
        <!-- https://mvnrepository.com/artifact/com.google.code.gson/gson
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.6</version>
        </dependency>

    </dependencies>
</project>
```

هرگاه در طول انجام عملیاتی که گفته شد، پیامی در قسمت پایین سمت راست IntelliJ برای شما به صورت زیر ظاهر شد، با انتخاب گزینه‌ی Enable Auto-Import، کار دانلود و نصب dependency ها پس از هر بار اضافه کردن به تگ dependencies به طور اتوماتیک انجام خواهد شد:




برای کسب اطلاعات بیش‌تر، جست‌وجو در گوگل را فراموش نکنید!

# پاسخ به سوالات متداول

## خطاهای متداول در راه‌اندازی پروژه و کلید حل آن‌ها

### مشکلات ناشی از اتصال اینترنت

هنگامی که کتاب‌خانه‌های جدیدی به پروژه‌ی خود اضافه می‌کنید، ابزار Maven برای Import کردن این کتاب‌خانه‌ها، نیاز به دریافت نسخه‌ی مشخص شده از آن‌ها از اینترنت دارد؛ در نتیجه قبل از انجام عملیات Import، از صحت اتصال اینترنت خود، اطمینان حاصل کنید.

### ارور "java release not supported"

اگر به ارور فوق هنگام اجرای پروژه پس از راه‌اندازی Maven برمی‌خورید، قدم‌های زیر را به ترتیب طی کنید:

۱. کد زیر را به قبل از تگ پایانی `</project>` در `pom.xml` بیفزایید:

```
<properties>
    <maven.compiler.source>1.11</maven.compiler.source>
    <maven.compiler.target>1.11</maven.compiler.target>
</properties>
```

در صورت نمایش دیالوگ Import Changes، گزینه‌ی Import را بزنید.

۲. اقدام به اجرای پروژه (از تابع main) کنید. در صورت عدم اجرای صحیح، به قدم ۳ بروید.

۳. از منوی File، گزینه‌ی Project Structure را انتخاب کرده، سپس گزینه‌ی Modules و از بالا، تب Sources را انتخاب کنید.

۴. گزینه‌ی Language Level را روی مقداری تنظیم کنید که قبلا همان نسخه از Java را روی دستگاه خود نصب کرده‌اید؛ سپس OK را بزنید.

۵. روی فایل pom.xml در لیست فایل‌های پروژه در پنل سمت چپ IntelliJ، راست‌کلیک کرده و از زیرمنوی Maven، گزینه‌ی Reimport یا Refresh را انتخاب کنید.

۶. اقدام به build و اجرای مجدد پروژه‌ی خود نمایید.

### خطاهای مربوط به اتصال گیت و پروژه

توصیه می‌کنیم اگر در اتصال پروژه‌ی خود به repository ای که توسط تیم پروژه در اختیار شما قرار گرفته است، دچار مشکل شدید، از نو، طبق دستورالعمل زیر به پیش بروید:

۱. یک پوشه‌ی خالی در مکان مدنظر خود ایجاد کرده و سپس repository در اختیار قرار گرفته توسط تیم پروژه (که در ابتدا تنها شامل فایل‌هایی همچون README.md است) را به کمک command های گیت که در کارگاه گیت آموخته‌اید، در آن clone کنید.

۲. به کمک IntelliJ، در همان مکانی که repository محلی Git را ایجاد کرده‌اید، پروژه‌ی جدید بسازید.

۳. به کمک Git، تغییرات را commit کرده و سپس به repository مربوط به تیم پروژه در GitHub، push کنید.

# قواعد پیشرفته‌ی UML

**سید پارسا نشایی**

## نکات کلی

شما از طریق کلاس‌های حل تمرین و نیز انجام تمرین دوم درس برنامه‌سازی پیشرفته، با بسیاری از نکات مهم UML آشنا هستید، مانند:

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

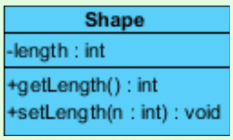
در طراحی یک پروژه، مشخص کردن «ارتباط» میان کلاس‌ها از اولویت فراوانی برخوردار است. در این مقاله، با نکاتی در این رابطه، آشنا می‌شوید.

## قواعد نوشتن «نام کلاس»

نام کلاس باید به شکل PascalCase در بالا و وسط نوشته شود. اگر کلاس از نوع abstract است، اسم آن باید *italic* نوشته شود.

## قواعد نوشتن خواص (property ها) و متدها

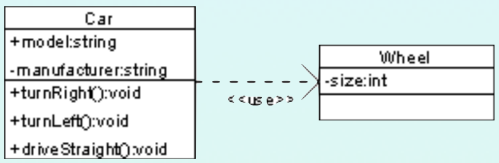
ابتدا باید سطح دسترسی خاصیت مشخص شود (+ برای public، - برای private، # برای protected و ~ برای package level) و سپس اسم داده نوشته شده و در نهایت پس از یک دو نقطه (:)، نوع آن مشخص می‌شود. اگر فیلد از نوع static است، زیر کل عبارت باید یک underline کشیده شود. قواعد نوشتن متدها نیز مشابه خواص است، با این تفاوت که پارامترهای ورودی نیز باید مشخص شوند. به مثال زیر توجه کنید:




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

### رابطه‌ی وابستگی

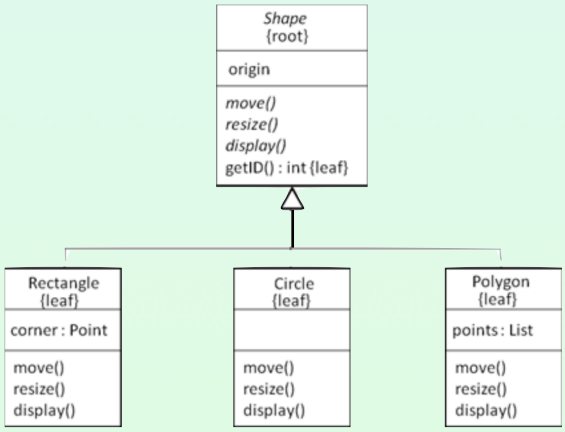
زمانی که دو کلاس از نظر مفهومی به هم وابسته‌اند و تغییر در خواص یک کلاس، ممکن است خواص کلاس دیگر را تغییر دهد و یا آن را دچار مشکل کند؛ به عنوان مثال، کلاس‌های Car و Wheel در شکل زیر:




نحوه‌ی نمایش این نوع از رابطه، یک فلش خط‌چین‌دار از کلاسی است که نیازمند کلاس دیگر است، به سمت کلاسی که نیازش را برطرف می‌کند. در زیر این خط چین، <<use>> نوشته می‌شود.

### رابطه‌ی تعمیم و وراثت

هر جا که کلاسی از پدرش ارث می‌برد (یا به عبارت دیگر، extend می‌کند)، لازم است از این رابطه استفاده شود.




نحوه‌ی نمایش این رابطه، یک فلش است که انتهای آن، مثلثی توخالی قرار دارد. در مثال فوق، همه‌ی انواع Polygon، Circle یا Rectangle، نوعی Shape (که یک کلاس abstract است و در نتیجه italic نوشته شده) هستند و در نتیجه، رابطه‌ی میان آن‌ها، از نوع وراثت است. با وراثت در کلاس‌های درس، بیش‌تر آشنا خواهید شد.

توجه کنید که این رابطه تنها معادل extend کردن است و نه معادل implement کردن که در interface ها استفاده می‌شود؛ برای واسطها، رابطه‌ای دیگر به زودی گفته خواهد شد.

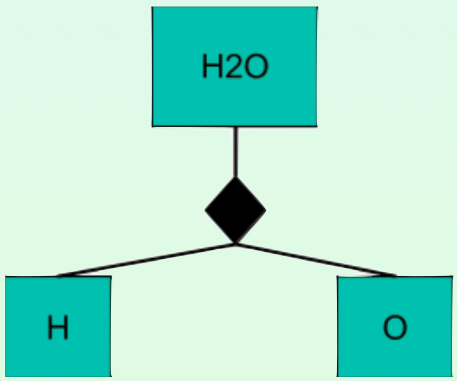
### رابطه‌ی تجمعی (Aggregation)

بسیاری از روابط، رابطه‌ی میان «کل» و «اجزا»ی یک پدیده را نشان می‌دهند؛ مثلا رابطه‌ی «دانشگاه» و «دانش‌جو». دیدیم اگر این رابطه به حدی شدید باشد که تغییر در یکی، دگرگونی بسیار دیگری را نتیجه دهد (مثلا اگر بخواهیم از نوع دیگری از چرخ استفاده کنیم، طراحی ماشین دگرگون می‌شود)، از رابطه‌ی وابستگی استفاده می‌شود، اما اگر این رابطه به گونه‌ای باشد که کلاس‌ها کاملا به هم وابسته نباشند، از رابطه‌ی تجمعی بهره گرفته می‌شود.


نحوه‌ی نمایش این رابطه، یک خط با یک شکل همانند لوزی توخالی است.

### رابطه‌ی ترکیب (Composition)

این رابطه، نسخه‌ی قوی‌تر رابطه‌ی تجمعی است، با این تفاوت که هیچ‌یک از دو کلاس دو طرف رابطه، بدون هم معنایی ندارند. در رابطه‌ی «وابستگی»، تنها پدر بدون فرزند بی‌معناست و فرزندان بدون پدر معنا دارند؛ اما در این رابطه (ترکیب)، فرزندان نیز بدون پدر معنایی ندارند.

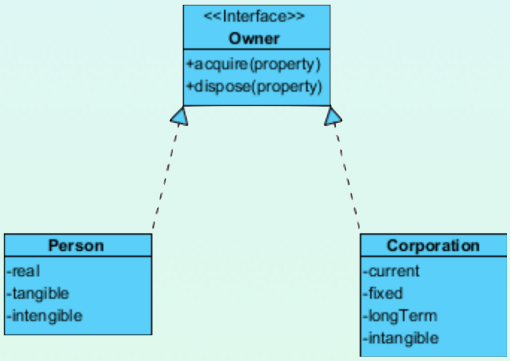



در مثال فوق، اگر در نظر بگیریم که برنامه‌مان تنها با «مولکول»ها کار دارد و نه با اتم‌ها، آن‌گاه H و O بدون مولکولی که شامل آن‌هاست، معنایی ندارند و خود مولکول نیز بدون اجزایش، معنایی ندارد. این رابطه با یک لوزی توپر روی خط نشان داده می‌شود.

### رابطه‌ی تفهیم یا تحقق (Realization)

از این رابطه، برای ارتباط دادن کلاس‌هایی به یک کلاس دیگر استفاده می‌شود که وظیفه‌ی تکمیل کردن آن کلاس دیگر را داشته باشند. برای ارتباط enum هایی که در یک کلاس استفاده می‌شوند به آن کلاس، از این رابطه بهره گرفته می‌شود. هم‌چنین، کلاس‌هایی که یک interface را پیاده می‌کنند نیز به کمک این رابطه به

متصل می‌شوند. نحوه‌ی نمایش interface و enum در نمودار UML، مشابه کلاس است، با این تفاوت که عبارت <<Interface>> و یا <<Enum>>، همان‌طور که در شکل مشخص است، در بالای box نوشته می‌شود.



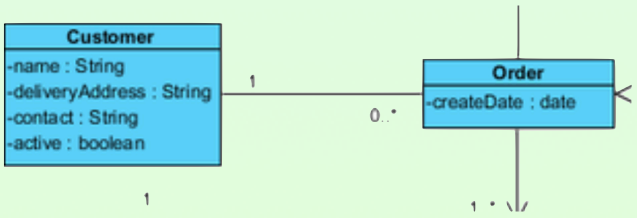

نحوه‌ی نمایش این رابطه، یک فلش خط‌چین‌دار با یک مثلث توپر است.

### رابطه‌ی عادی (Normal)

اگر رابطه‌ی میان دو کلاس در هیچ‌یک از حالات فوق جای نمی‌گرفت، از رابطه‌ی نرمال استفاده می‌کنیم که شامل یک خط بدون هیچ نمادی است.

### چندی (multiplicity) روابط

می‌توانید تعداد مواقعی که هر کلاس نیازمند دیگری است را با یک عدد روی خط نمایش دهید؛ مثلا هر customer می‌تواند تعداد زیادی order داشته باشد، اما هر order تنها به یک customer مرتبط است. در این حالت، در خط میان دو کلاس Customer و Order، دو عدد می‌نویسیم:




عدد ۱ روی خط یعنی هر سفارش فقط یک مشتری دارد، و عدد 0..* یعنی هر مشتری، از صفر تا بی‌نهایت سفارش دارد؛ اگر مثلا ماکسیمم ۴ سفارش ممکن بود، می‌توانستیم از 0..4 استفاده کنیم.