



Class section

UML

By: M·Madadyar

[HTTP://WWW.STUDENTS.MADADYAR.COM](http://www.students.madadyar.com)

UML چیست ؟

○ UML

- زبانی استاندارد به منظور مشخص نمودن، ایجاد و مستندسازی تولیدات نرم افزاری.
- مجموعه ای است از بهترین امکانات مهندسی به منظور استفاده در مدلسازی سیستم های بزرگ.
- UML یک ابزار ویژوال بوده که از انواع متفاوتی از نمودارها استفاده می کند و هریک از نمودارهای آن امکان مشاهده یک سیستم نرم افزاری را از دیدگاههای متفاوت و با توجه به درجات متفاوت تجرید (Abstraction) در اختیار پیاده کنندگان قرار می دهد.

○ UML مکانیزمی برای استفاده برنامه نویسان نرم افزار در:

✓ درستی دریافت درخواست مشتری.

✓ جلوگیری از ابهام و دوباره کاری در نوشتن برنامه.

○ مهندسی رو به جلو

○ نگاشت از مدل‌های UML به کد زبانهای برنامه نویسی.

○ مهندسی معکوس

○ بدست آوردن مدل‌های UML از یک برنامه به زبان شی گرا.

○ مزیت استفاده از UML تفکر مبتنی بر برنامه نویسی شی گراست.

○ تعریف.

- شبیه سازی یک محیط با اندازه های متفاوت از محیط واقعی و احتمالا مواد ومصالحی متمایز از جنس مواد ومصالح محیط مدل شده.

○ اهداف

✓ شناخت (Exploration)

✓ هدف شناخت محیطی مورد مدل.

✓ تبیین (Specification)

✓ معرفی و ارائه خصوصیات موجودیت واقعی یک مدل.

✓ رفع ایرادات قبل از ساخت.

چرا مدلسازی می کنیم؟

- ضروری بودن مدل‌های خوب، برای ارتباط افراد در گروه های پروژه با یکدیگر و نیز اطمینان از قوت معماری.

○ وظیفه UML

- با وجود موثر بودن عوامل متعدد در موفقیت پروژه، داشتن یک زبان استاندارد مدلسازی واحد یکی از عوامل ضروری است و این همان چیزی است که UML فراهم می کند.

ابزارهای موجود برای طراحی شی گرا

- Microsoft Visio

- UML star

- ★ **Enterprise Architect (Sparx)**

- VP Suite Windows

- ◎ **Rational Rose**

-

○ یکی از دیدگاههای موجود در UML می باشد.

○ این نما شامل نیازمندی های عملیاتی سیستم می باشد که به کلاسها و ارتباط بین آنها می پردازد.

○ این نما شامل دو دیاگرام زیر می باشد:

- دیاگرامهای کلاسها (class diagrams)

- دیاگرامهای حالت (state chart diagrams)

Class Diagram

○ این دیاگرام به شما کمک می کند تا نمای ساختاری سیستم تان را بصورت بصری (visual) در آورید.

• پایه و اساس دیاگرامهای بعدی در UML است.

در روش های شیء گرا، مرکزیت با دیاگرام کلاس است.

- دیاگرام کلاس، انواع اشیاء درون سیستم و انواع مختلف ارتباطات بین آنها را نمایش می دهد.

- دیاگرام کلاس، صفات و اعمال یک کلاس و محدودیت هایی که در ارتباط با کلاس های دیگر دارد را نیز نشان می دهد.

به طور کلی عناصر یک دیاگرام کلاس عبارت است از کلاس و روابط بین آنها.

Stereotype

boundary ○

- اجزای لازم برای برقراری ارتباط سیستم با یک بازیگر را در خود دارند . (در واقع کلاس های تعریف واسط کاربری)

control ○

- این کلاسها معمولا اشیا دیگر و رفتارهای تعبیه شده در یک مورد کاری را کنترل می کنند.

entity ○

- این کلاسها اطلاعاتی را که باید توسط سیستم ذخیره گردند را در خود نگهداری می کنند .

Class name

Attributes

Methods

توضیحاتی در مورد کلاس

class System

Class4

This is a Sample Description About Class

{ Capacity=10 Or 90 }

- کلیشه ها برای توصیف صفات و عملیات (>> کلیشه <<)

○ چند تایی (Multiplicity)

○ چند تایی در کلاس ها، تعداد اشیاء در سیستم را مشخص می کند.

○ به عنوان مثال ممکن است در سیستم مورد نظرتان کلاسی وجود داشته باشد که همیشه ۳ شیء دارد. در این صورت عدد ۳ را در گوشه بالا و سمت راست مستطیل کلاس می نویسیم.



صفات (Attributes)

○ دید عمومی (علامت +) (Public)

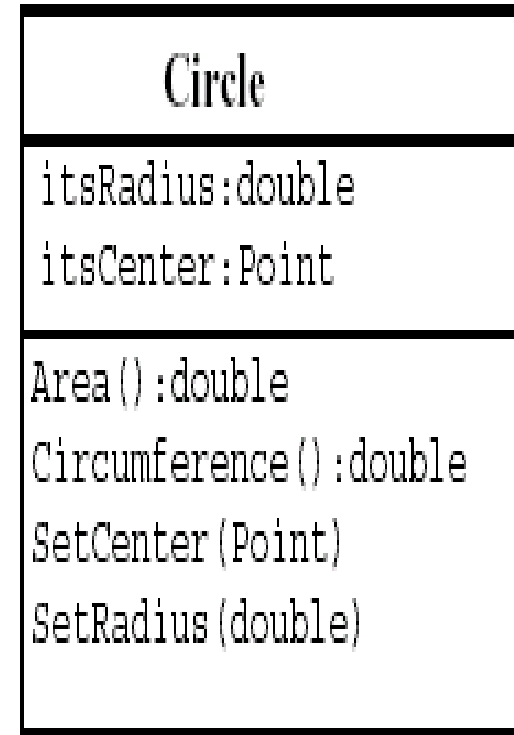
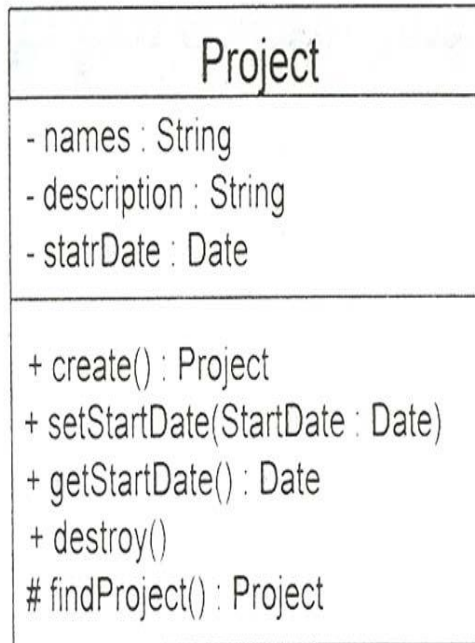
- دسترسی به این صفت برای همه عناصر سیستم مجاز است.

○ دید محافظت شده (علامت #) (Protected)

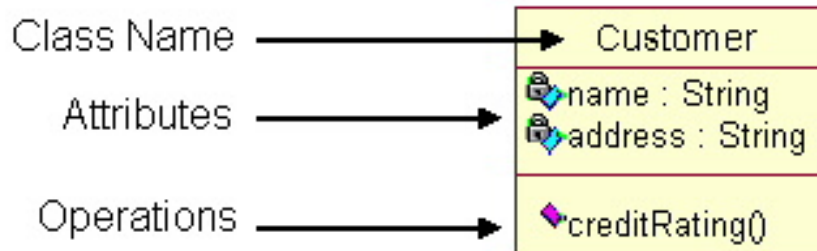
- دسترسی به این صفت برای همه زیر کلاس ها امکان پذیر است و غیر از این کلاس ها، دیگر کلاس های سیستم قادر به دیدن آنها نیستند.

○ دید خصوصی (علامت -) (Private)

- دسترسی به این صفت فقط برای متدهای همان کلاس امکانپذیر است.



Rational Rose Notation



❖ نوع در صفات

بیان گر نوع و ماهیت صفت است. انواع مختلفی که متداول هستند عبارتند از:

Boolean و Long, Integer, String, UDT و ...

-Name: String
-Checked: Boolean
-Balance: Number
.....

۱. فهرست پارامترها در متدها

- اگر بخواهیم پارامترهای یک متد را نشان دهیم، فهرست پارامترهای آن متد را درون پرانتز جلوی متد قرار می دهیم.

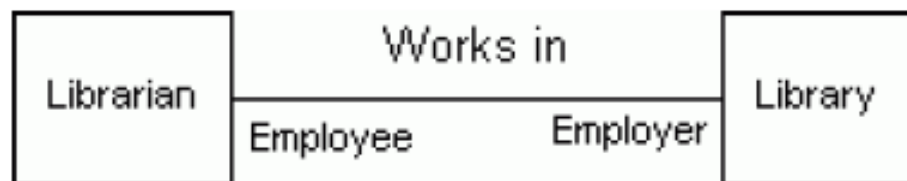
۲. نوع برگشتی در متدها

- بیان گر مقداری است که توسط تابع مربوطه به فرا خواننده برمی گردد.
- روبروی معرفی متد نوشته می شود. و در صورت عدم وجود این متد مقدار برگشتی نخواهد داشت (void).

○ رابطه تناظر

○ این نوع رابطه، رابطه ساختاری است و جدا از روابط ارث بری و وابستگی است.

○ یک رابطه تناظر می تواند دارای نام، چند تایی، نقش، جهت و قابلیت پیمایش است.



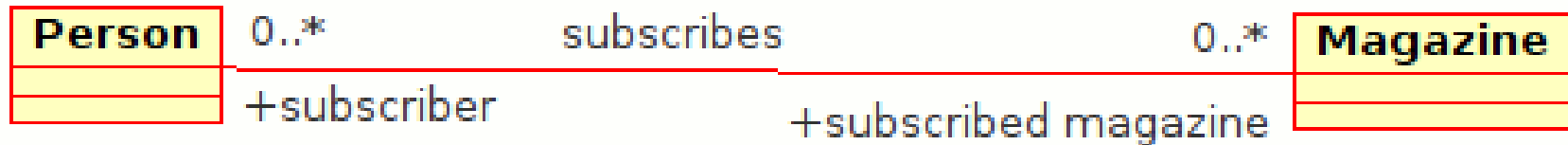
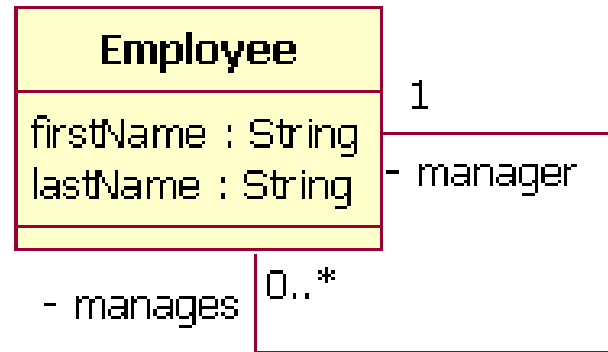
کاردینالیتی

به عددی که بیان گر تعداد اشیاء کلاس در یک رابطه با یک کلاس دیگر است، چندتایی یا کاردینالیتی گفته می شود.

مثال:

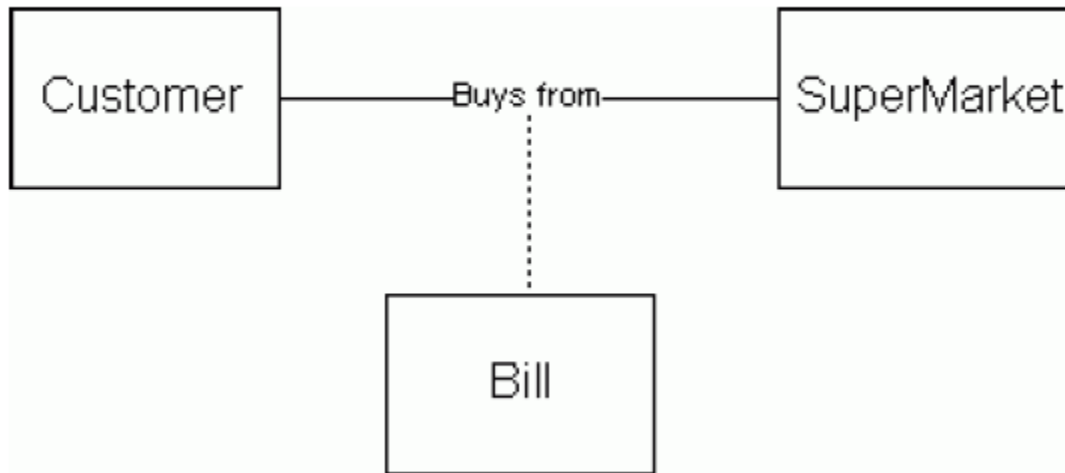
- ۲..۵: کلاس، همیشه حداقل ۲ و حداکثر ۵ شیء دارد.
- ۵ و ۲: کلاس مورد نظر، همیشه ۲ یا ۵ شیء دارد.
- ۰..*: کلاس مورد نظر، همیشه حداقل ۰ و حداکثر بی شمار شیء دارد.
- *: این علامت معادل ۰..* می باشد.
- ۸..۶ و ۳: کلاس همیشه ۳ شیء و یا از ۶ تا ۸ شیء در رابطه دارد.

کاردینالی در رابطه تناظر

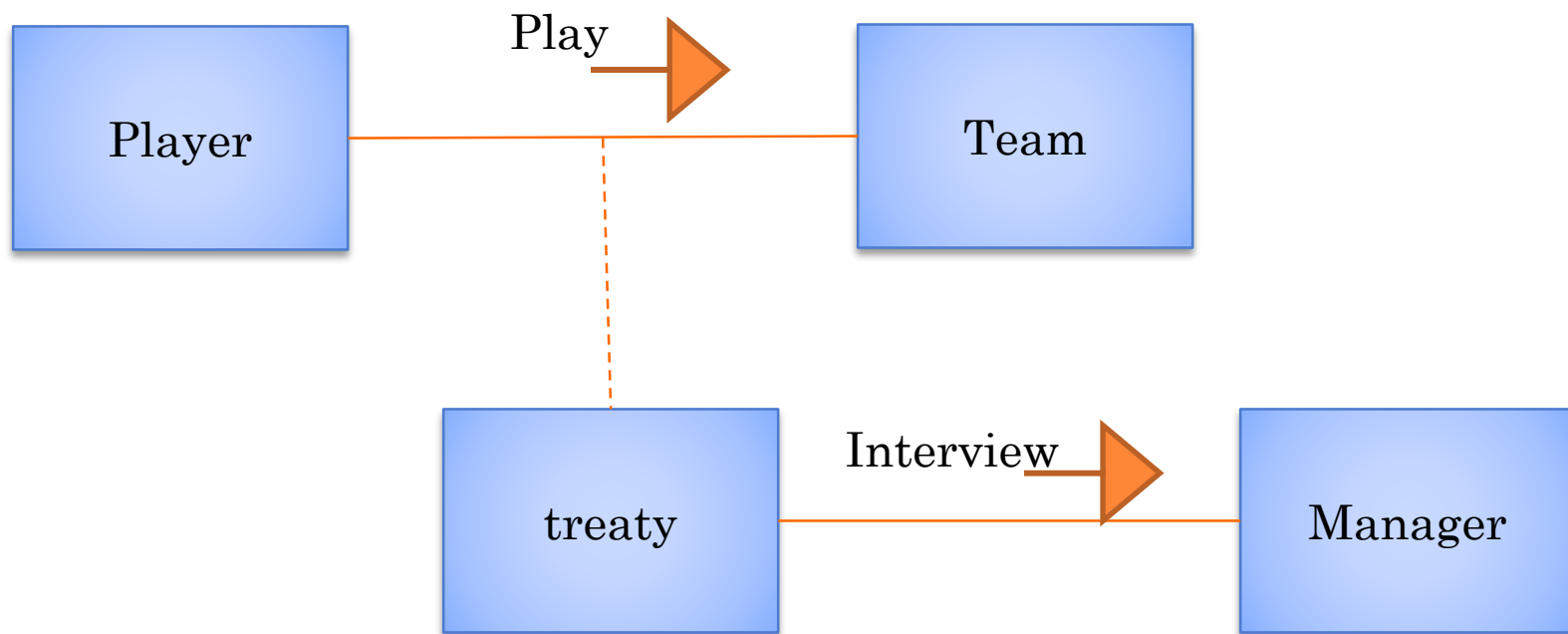


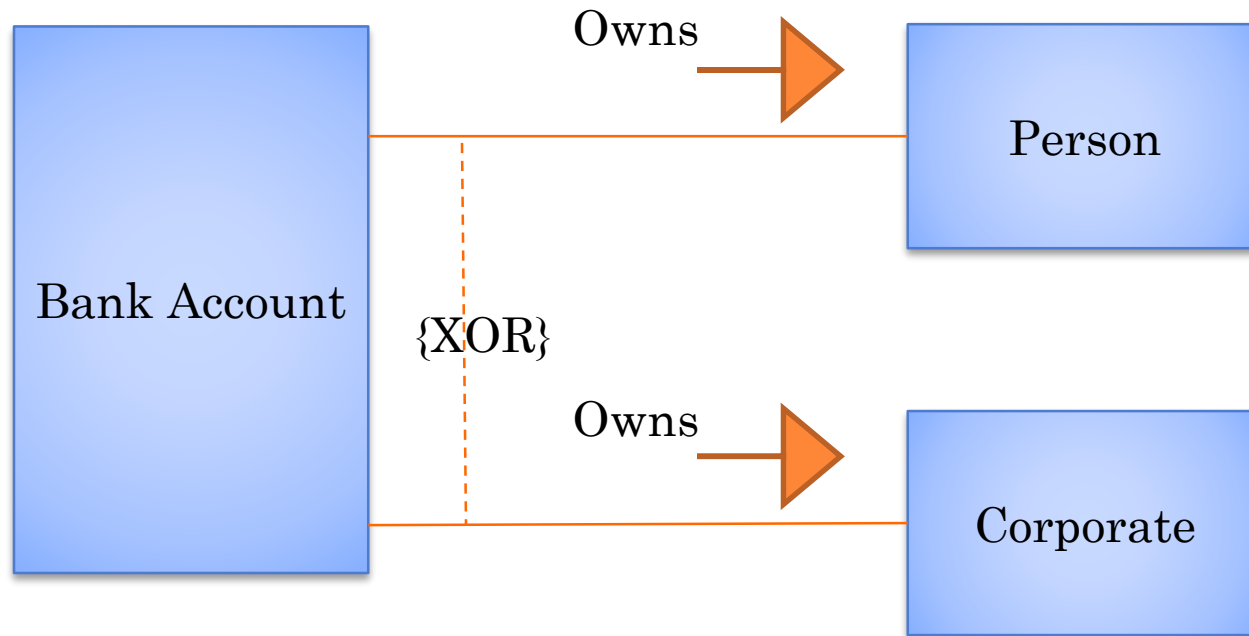
Association Class

- دقیقا مانند یک کلاس یک رابطه نیز می تواند صفات و عملیات داشته باشد که تشکیل **کلاس رابطه** می دهد.



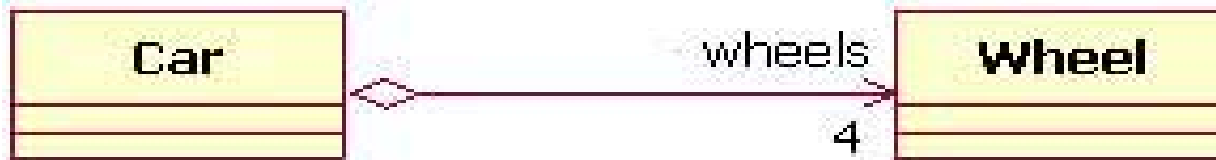
نقش ها در رابطه و جهت





○ رابطه تجمع یک حالت خاص از رابطه تناظر است و زمانی که بخواهیم نشان دهیم که از اجتماع چند شیء، یک شیء کلان تر به وجود می آید از این رابطه استفاده می کنیم.

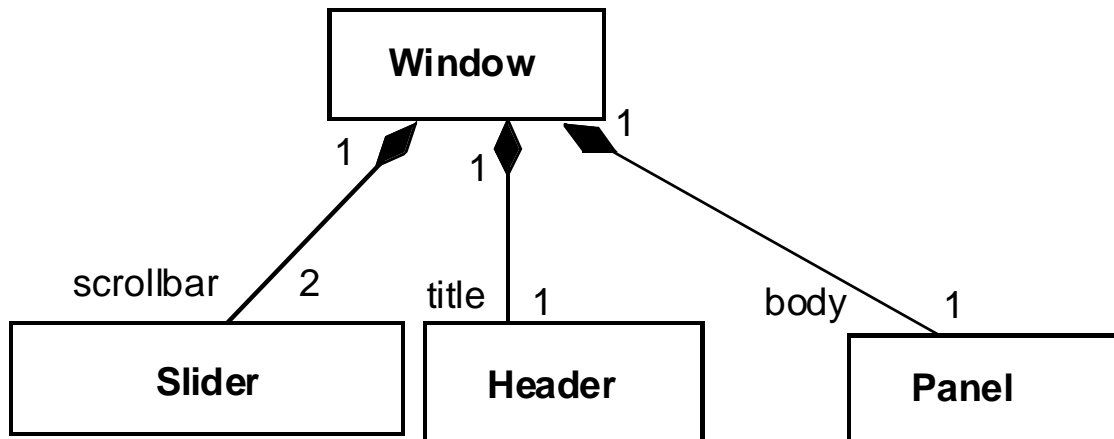
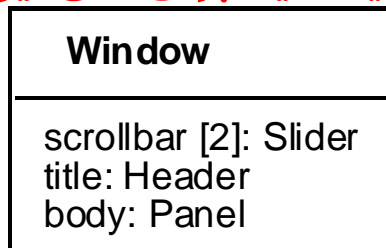
- علامت آن یک لوزی تو خالی است که به قسمت کل متصل است.



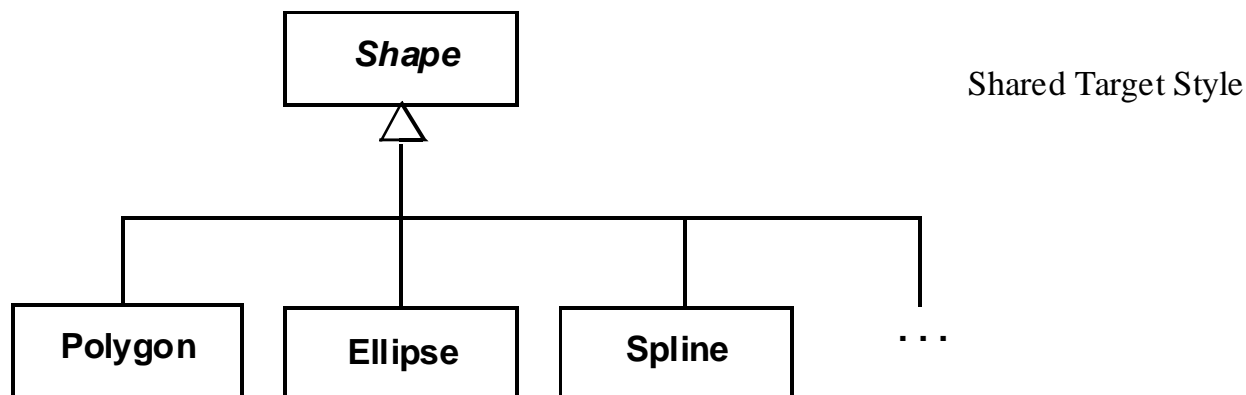
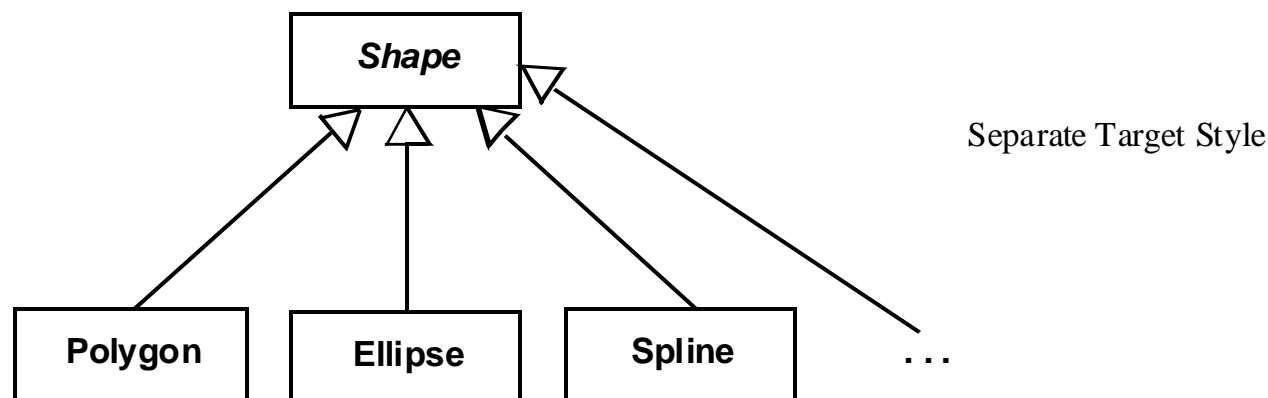
رابطه ترکیب

این رابطه حالت خاصی از رابطه تجمع است و بنابراین حالت خاصی از تناظر نیز می باشد. این رابطه علاوه بر مفهوم کلی رابطه تجمع که از اجتماع چند شیء یک شیء کلی تر تشکیل می شود،

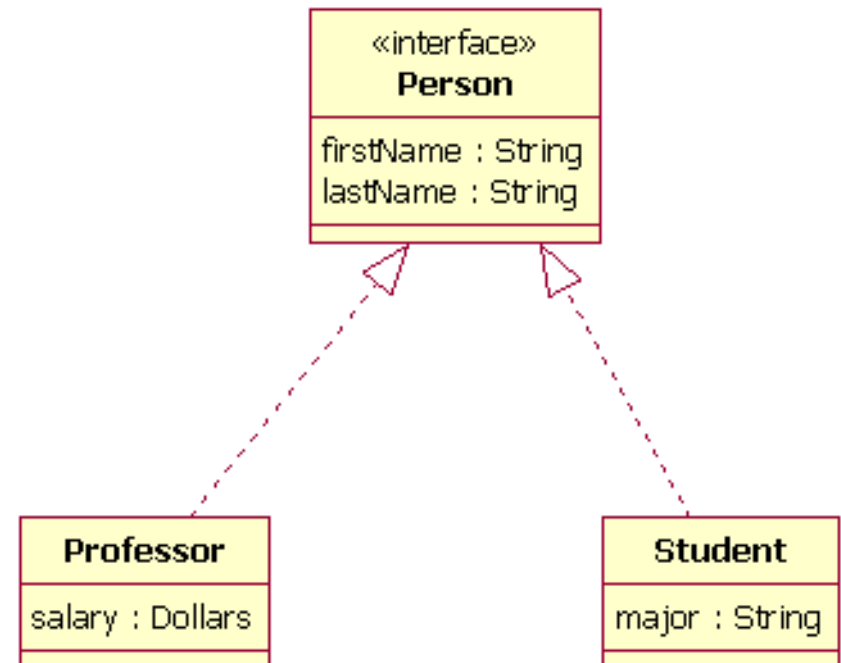
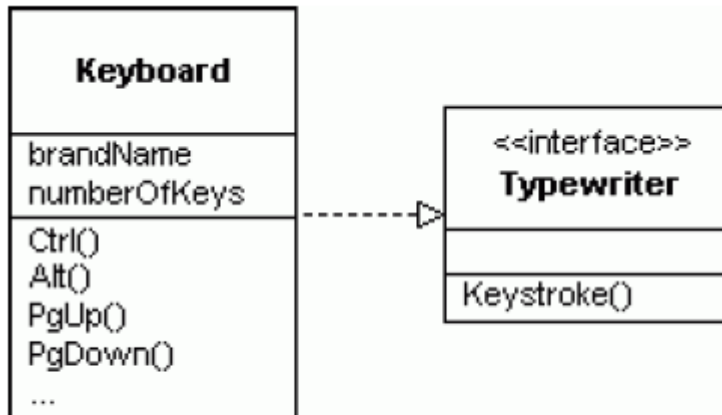
• بیان گر آن است که با از بین رفتن شیء کلی تر، کلیه اشیاء جزئی اش نیز از بین می



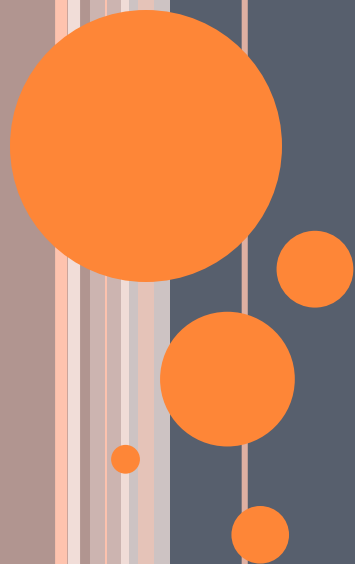
در صورت وجود صفات و عملیات مشترک می توان از کلاس ها ارث بری نمود.

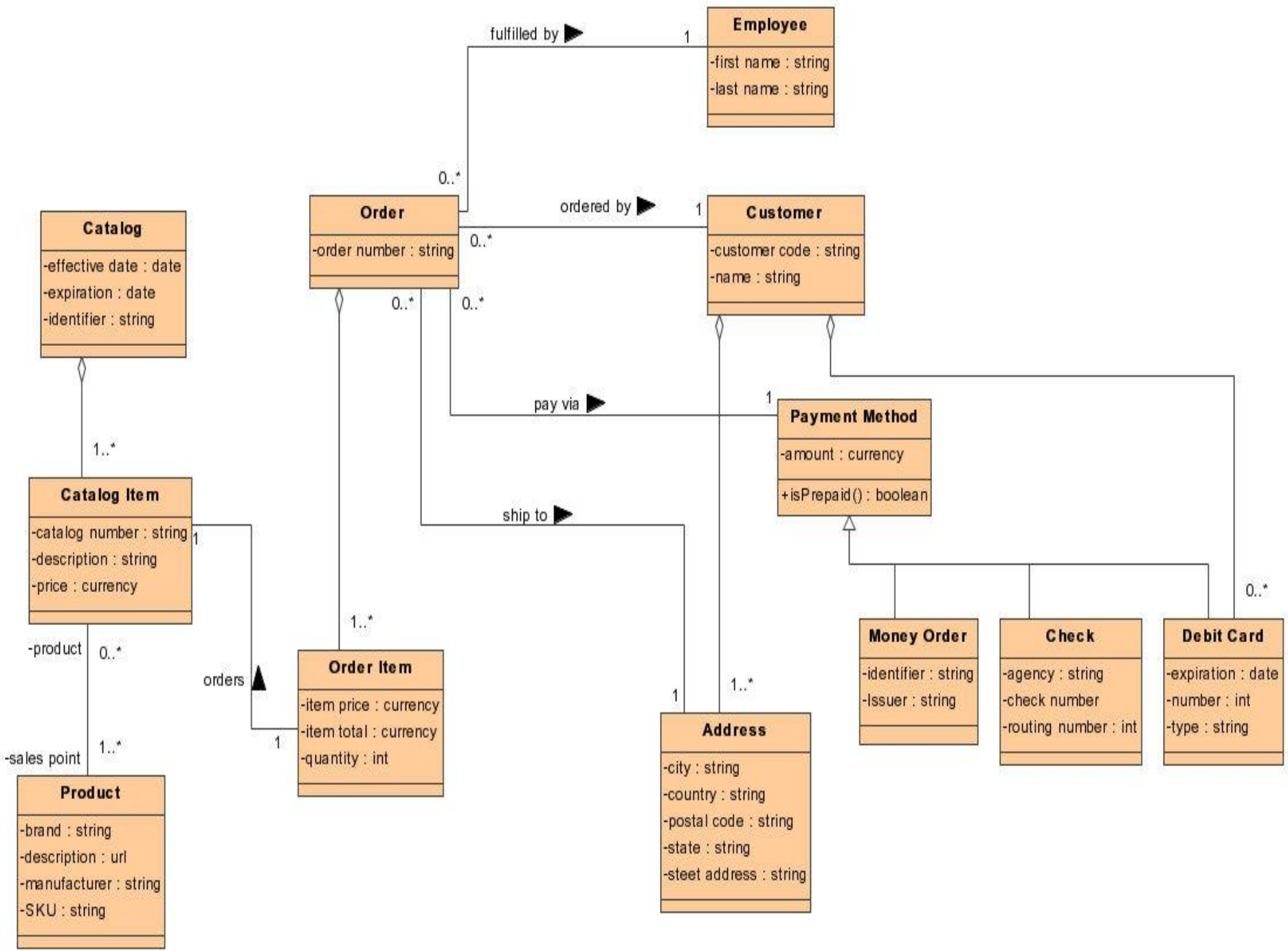


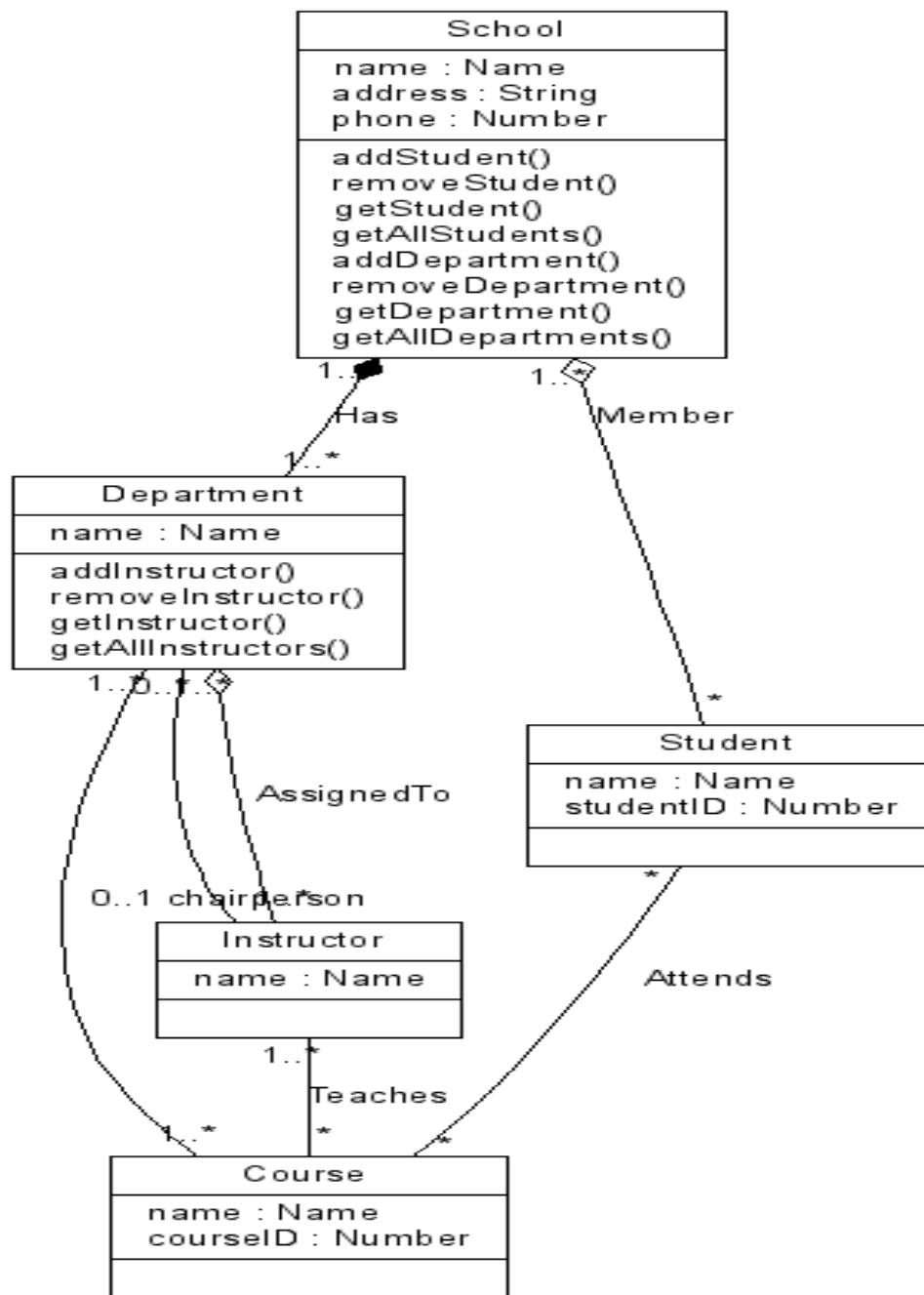
Interface

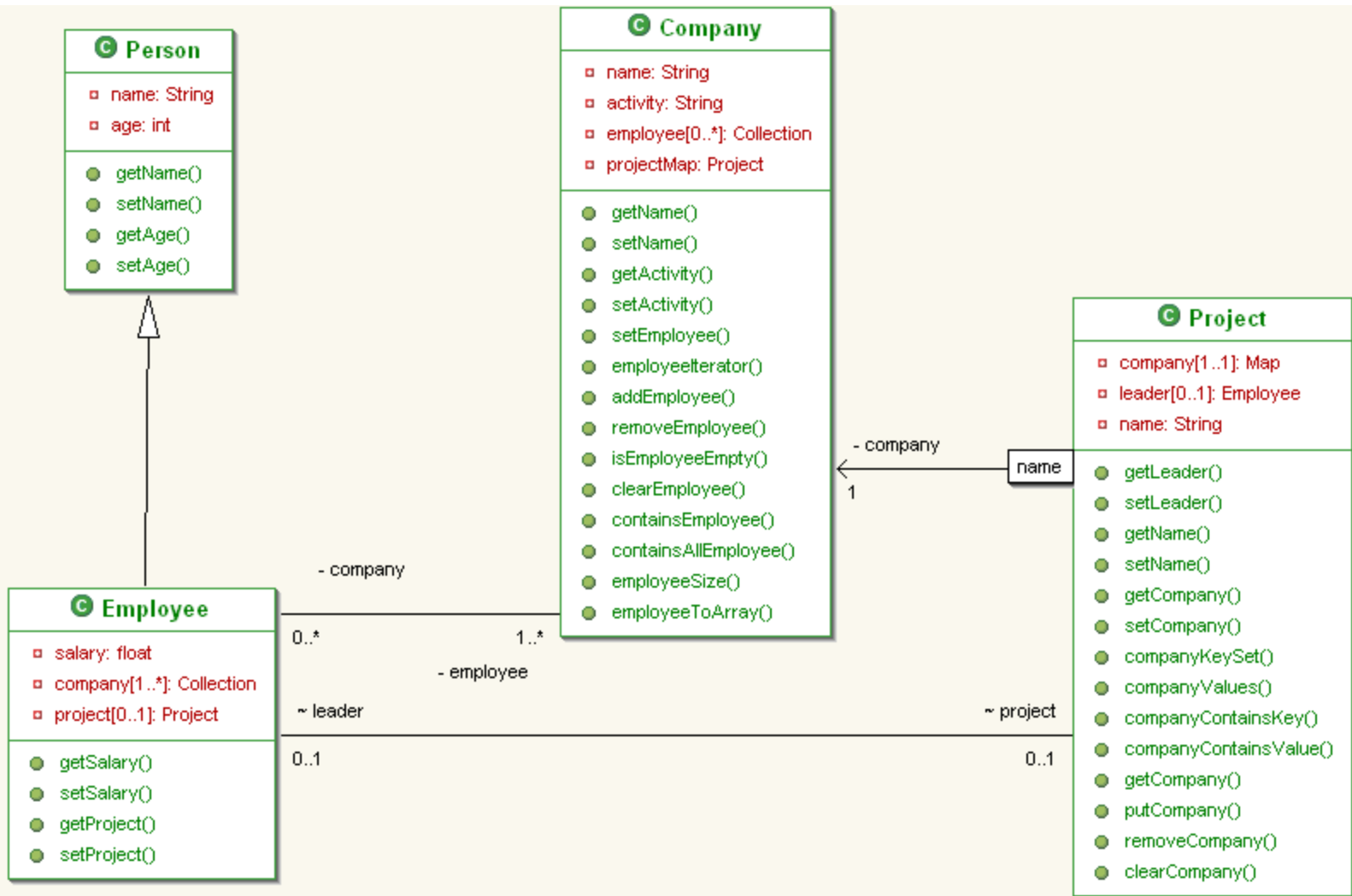


EXAMPLE S

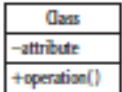


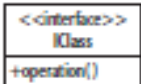








Summary


Class		Types and parameters specified when important; access indicated by + (public), (private), and # (protected).
-------	-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------


Interface		Name starts with I. Also used for abstract classes.
-----------	-----------------------------------------------------------------------------------	-----------------------------------------------------


Note		Any descriptive text.
------	-----------------------------------------------------------------------------------	-----------------------

Package		Grouping of classes and interfaces.
---------	-----------------------------------------------------------------------------------	-------------------------------------

Inheritance		B inherits from A.
-------------	-----------------------------------------------------------------------------------	--------------------

Realization		B implements A.
-------------	------------------------------------------------------------------------------------	-----------------

Association		A and B call and access each other's elements.
-------------	-------------------------------------------------------------------------------------	------------------------------------------------

Association (one way)		A can call and access B's elements, but not vice versa.
-----------------------	-------------------------------------------------------------------------------------	---------------------------------------------------------

Aggregation		A has a B, and B can outlive A.
-------------	-------------------------------------------------------------------------------------	---------------------------------

Composition		A has a B, and B depends on A.
-------------	-------------------------------------------------------------------------------------	--------------------------------