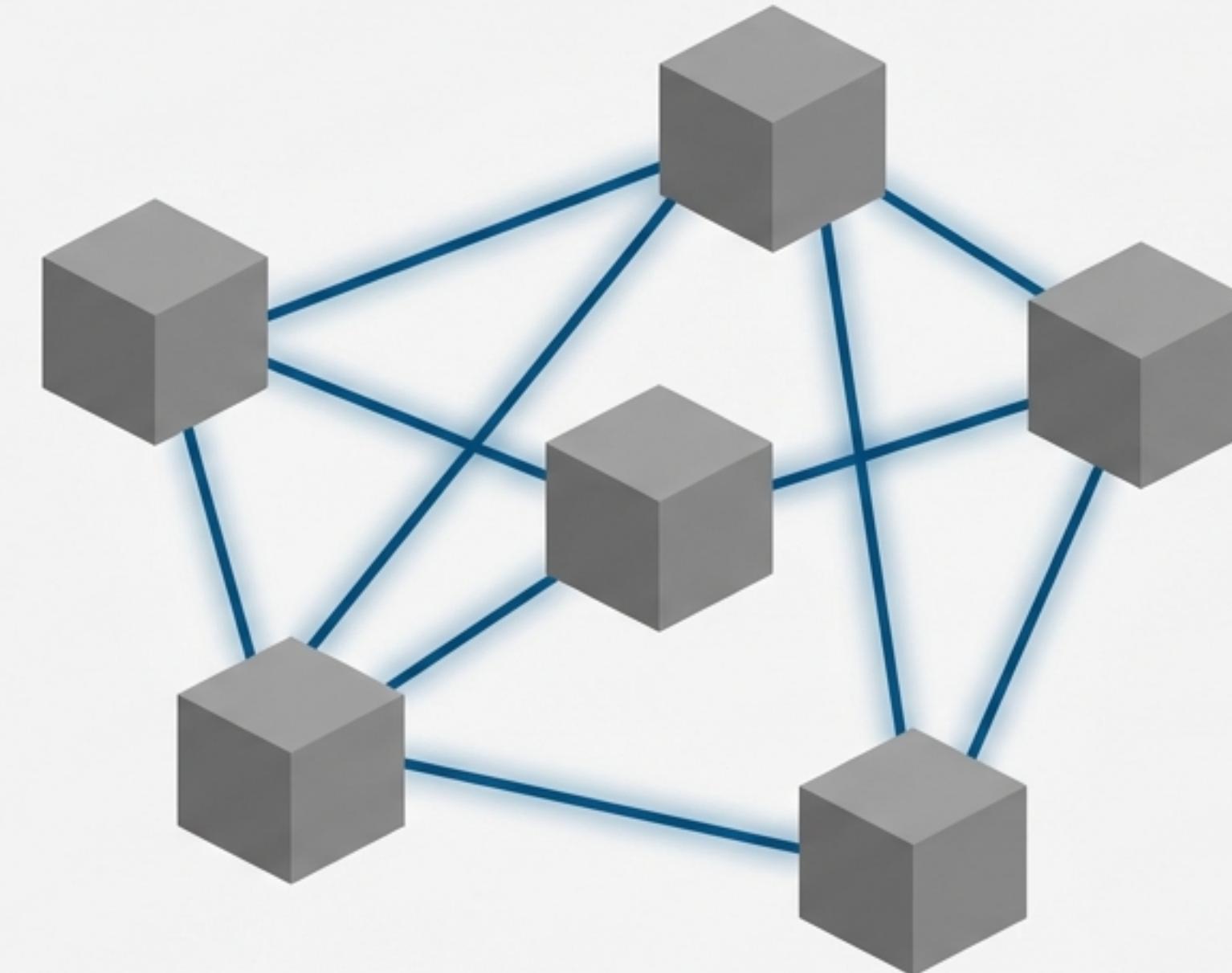


بخش ۸-ب: روابط در UML — داستان تعامل اشیاء

تهییه شده توسط: سید سجاد پیراھش

کلاس‌ها در نجی؛ در انزوا زندگی قدرت در تعامل است



در دنیای واقعی و برنامه‌نویسی شیء‌گرا، یک کلاس به تنها یی ارزش چندانی ندارد. قدرت واقعی سیستم‌های نرم‌افزاری از تعامل (Collaboration) و همکاری (Interaction) بین اشیاء ناشی می‌شود. خطوطی که کلاس‌ها را به هم وصل می‌کنند، داستان این این تعاملات را بازگو می‌کنند.

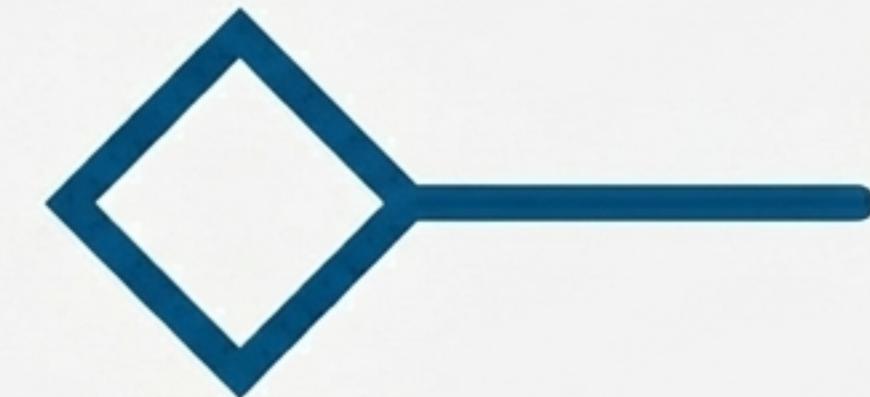
سه نوع رابطه کلیدی که هر معماری باید بشناسد

(Association) انجمن



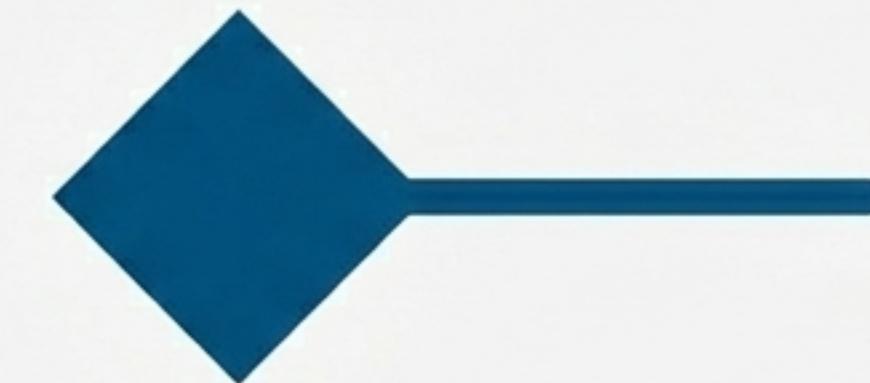
"استفاده می‌کند" (Uses-a)

(Aggregation) تجمیع



"دارای ... است" (Has-a)

(Composition) ترکیب



"متشكل از ... است" (Part-of)

درک دقیق تفاوت‌های ظریف این سه، مرز بین یک طراح معمولی و یک مهندس نرم‌افزار حرفه‌ای است.

۱. انجمن (Association): یک رابطه آزاد و مستقل

مفهوم کلیدی: "استفاده می‌کند" یا "ارتباط دارد" (Uses-a / Has a relationship with)

مثال: رابطه بین معلم (Teacher) و دانشآموز (Student)

- دو کلاس با هم کار می‌کنند، اما هیچگدام مالک دیگری نیست.

Independent حیات مستقل (Independent Lifecycle):

اگر معلم بازنشسته شود، دانشآموزان از بین نمی‌روند و بالعکس.



۲. تجمعیع (Aggregation): مالکیت ضعیف "کل و جزء"

مفهوم کلیدی: "دارای ... است" (Has-a)

مثال: رابطه بین گروه آموزشی (Department) و استاد (Professor)

- گروه آموزشی "دارای" تعدادی استاد است.
- **چرخه حیات مستقل:** اگر گروه آموزشی منحل شود، استاد همچنان به عنوان Professor شود، استاد همچنان به عنوان Professor شود، استاد همچنان به عنوان Professor در سیستم وجود دارند و می‌توانند به گروه دیگری منتقل شوند.



۳. ترکیب (Composition): یک پیوند حیاتی و جدانشدنی

مفهوم کلیدی: "متشكل از ... است" (Part-of)

مثال: رابطه بین خانه (House) و اتاق (Room)

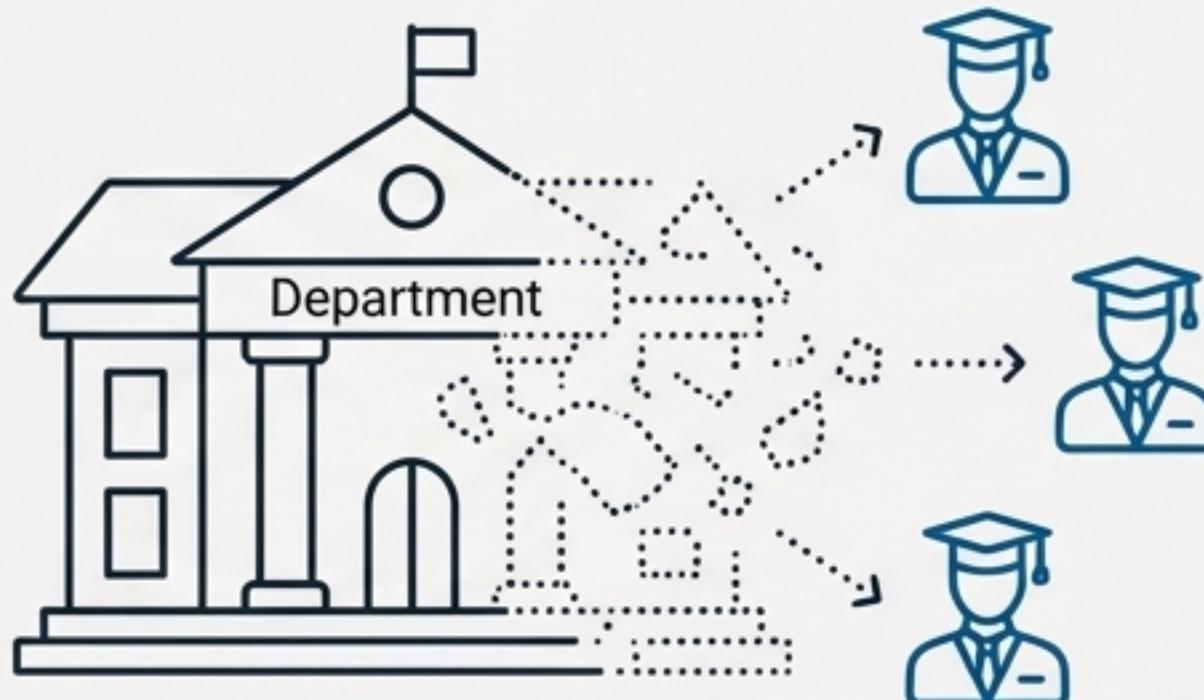
- "جزء" هیچ هویت مستقلی خارج از "کل" ندارد.
- **چرخه حیات وابسته** (Dependent Lifecycle)
اگر خانه تخریب شود، تمام اتاق‌هایش هم نابود می‌شوند. "کل" مسئول تولد و مرگ "جزء" است.



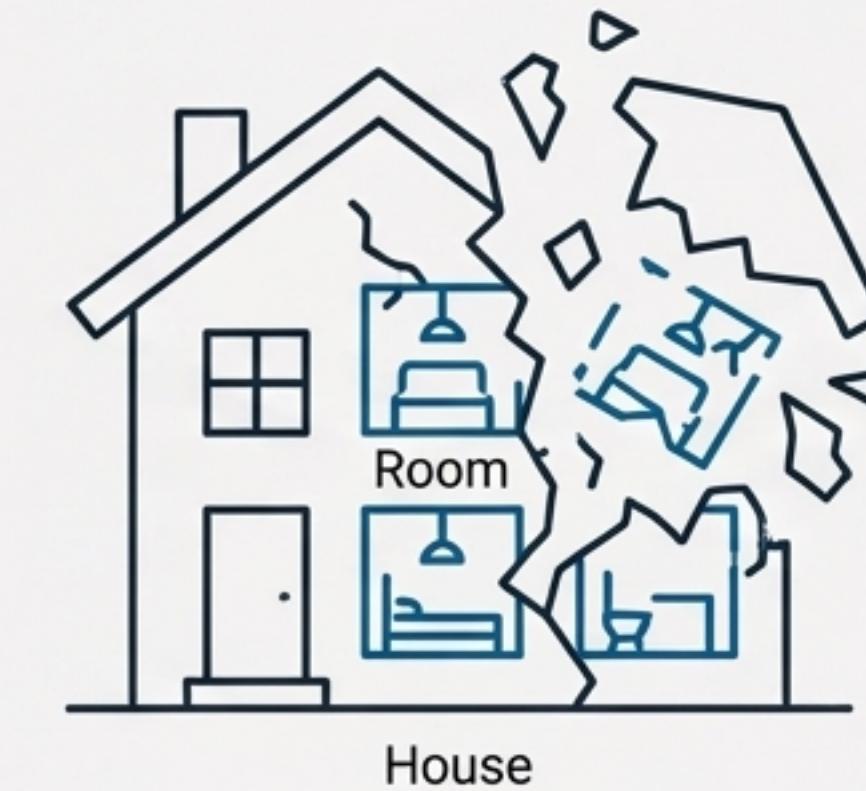
آزمون مرگ: چگونه Composition را از Aggregation تشخیص دهیم؟

سوال کلیدی: اگر "کل" بمیرد، چه بر سر "جزء" می آید؟

تجمعیع (Aggregation)



ترکیب (Composition)



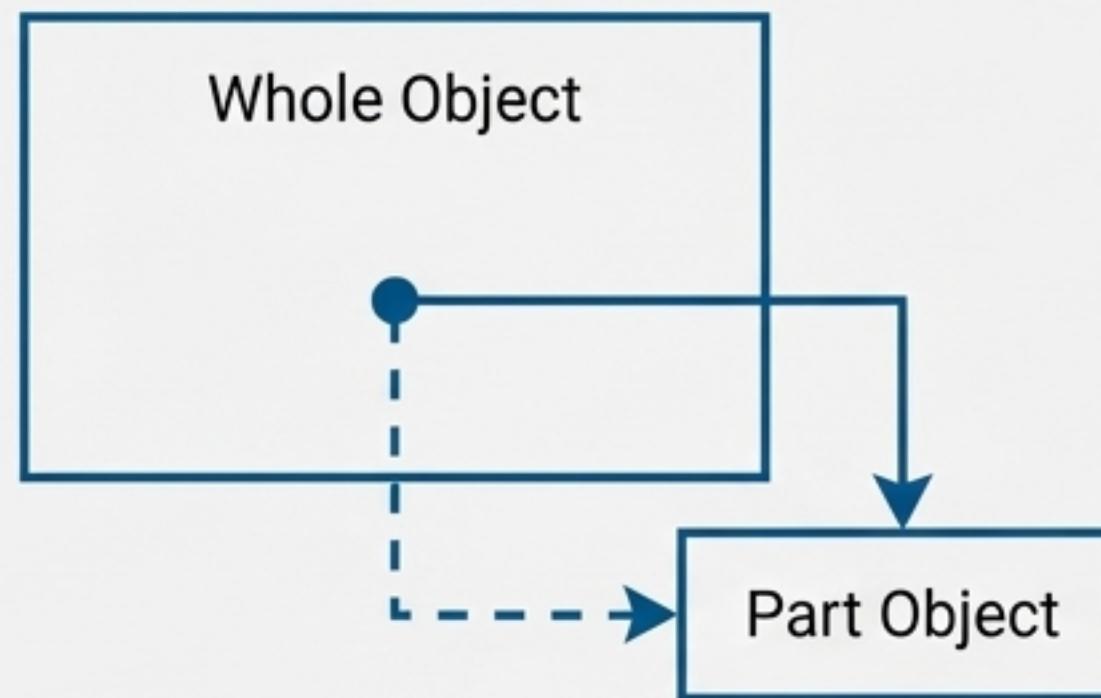
جزء زنده می‌ماند.

جزء نیز نابود می‌شود.

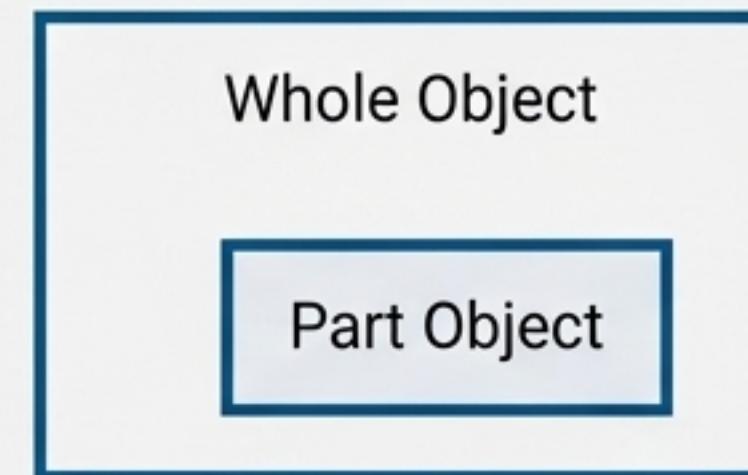
چرا این تفاوت مهم است؟ وابستگی چرخه حیات و مدیریت حافظه

انتخاب بین Composition و Aggregation یک تصمیم معماری مهم است.

Aggregation



Composition



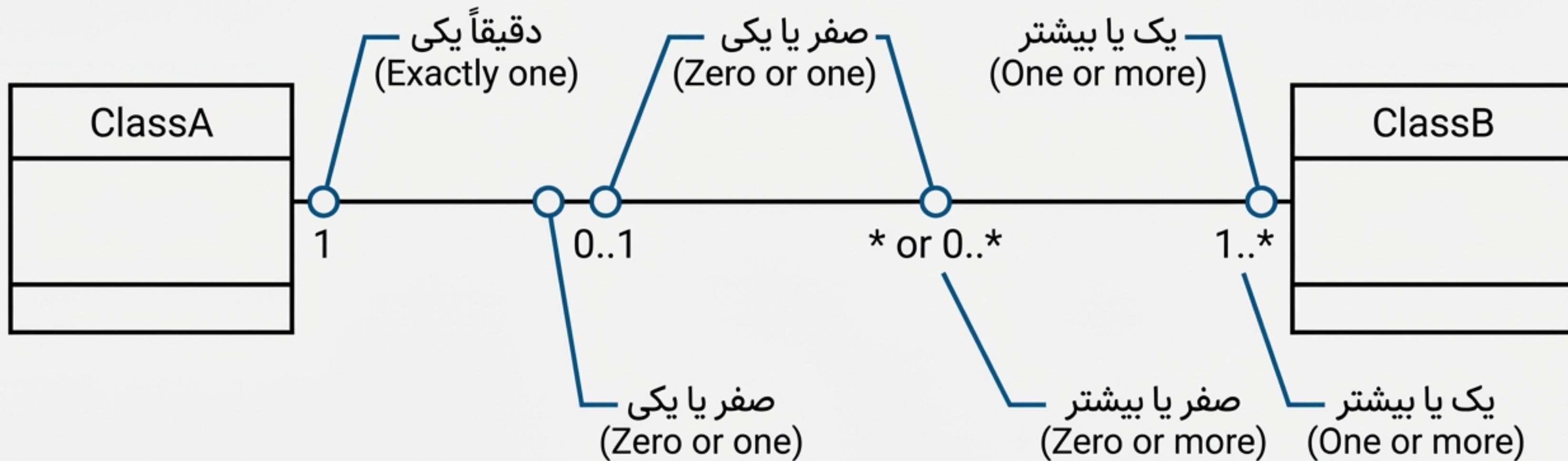
یک انتخاب امن‌تر برای مدیریت حافظه است. وقتی شیء "کل" از حافظه پاک می‌شود (Garbage Collection)، تمام اجزاء داخلی آن نیز به طور خودکار پاک می‌شوند. این طراحی از نشت حافظه (Memory Leaks) و وابستگی‌های ناخواسته در سیستم جلوگیری می‌کند.

جدول مقایسه جامع روابط

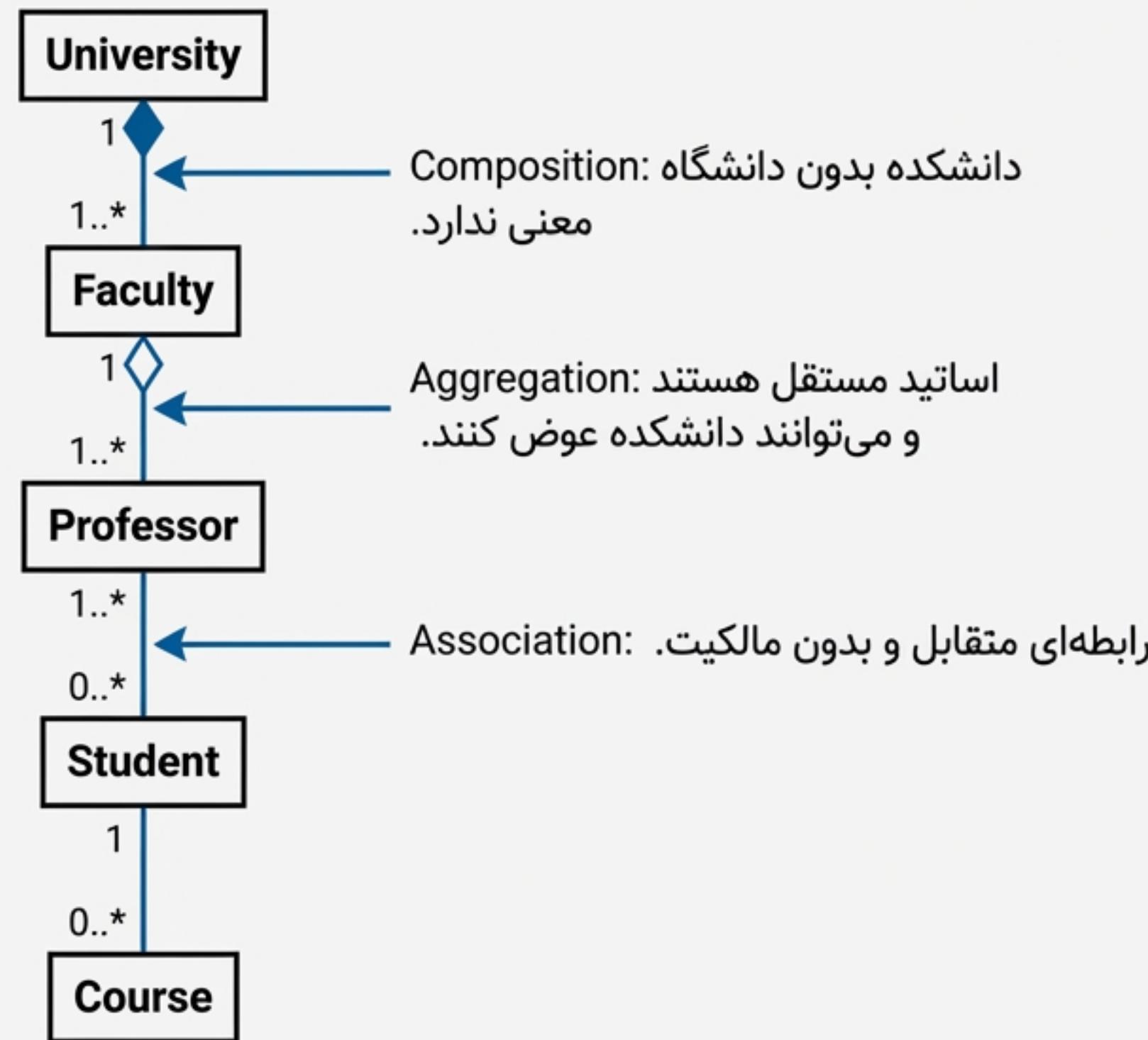
Composition (ترکیب)	Aggregation (تجمیع)	Association (انجمن)	معیار مقایسه
			نماد UML
"متشكل از ... است" (Part-of)	"دارای ... است" (Has-a)	"استفاده می‌کند" / "ارتباط دارد"	مفهوم کلیدی
مالکیت انحصاری / قوی	مالکیت اشتراکی / ضعیف	بدون مالکیت (همتا به همتو)	نوع مالکیت
وابسته: کل بمیرد، جزء هم می‌میرد.	مستقل: جزء بدون کل زنده می‌ماند.	مستقل: نابودی یکی بی‌تأثیر است.	چرخه حیات (Lifecycle)
بسیار قوی	متوسط	ضعیف	شدت رابطه
خانه و اتاق	تیم فوتبال و بازیکن	علم و دانشآموز	مثال واقعی

تعدد (Multiplicity): قوانین تعداد در یک رابطه

این اعداد، قوانین کسب و کار سیستم شما را تعریف می کنند و به ما می گویند چند شیء می توانند در یک رابطه شرکت کنند.



سناریوی جامع: طراحی سیستم مدیریت دانشگاه



تمرین ۱: کارآگاه روابط

برای هر جفت موجودیت زیر، نوع رابطه (Composition و Aggregation) را مشخص کنید و دلیل خود را با استفاده از "آزمون مرگ" بنویسید.

?



۱. انسان (Human) و مغز (Brain)

?



۲. پوشه (Folder) و فایل (File)

?



۳. کتابخانه (Library) و عضو (Member)

?



۴. ماشین (Car) و چرخ (Wheel)

تمرین ۲: چالش طراحی فروشگاه آنلاین

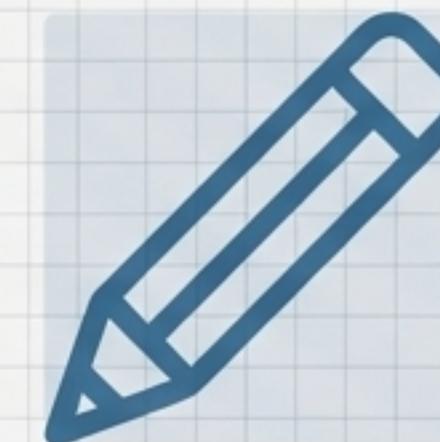
با توجه به قوانین کسب وکار زیر، یک دیاگرام کلاس UML برای یک فروشگاه اینترنتی طراحی کنید.

موجودیت‌ها:

- Customer (مشتری) 
- Order (سفارش) 
- Product (محصول) 
- ShoppingCart (سبد خرید) 

قوانین کسب وکار:

- هر مشتری دقیقاً یک سبد خرید دارد.
- سبد خرید لیستی از محصولات را نگه می‌دارد.
- مشتری سفارشات را ثبت می‌کند.
- هر سفارش شامل اقلام سفارش است که به یک محصول اشاره دارد.



دیاگرام را رسم کرده و نوع تمام روابط (Association, Aggregation) آنها را مشخص کنید. Multiplicity و Inter Regular

شما فقط کدنویس نیستید؛ شما یک معمار نرم افزار هستید

طراحی شیء‌گرا با UML فقط کشیدن شکل نیست؛ بلکه **مدل‌سازی واقعیت** است.
انتخاب رابطه درست، یک تصمیم معماری است که بر استحکام، انعطاف‌پذیری و
پایداری نرم افزار شما تأثیر می‌گذارد.

با تسلط بر این بخش، شما آماده‌اید تا وارد مباحث پیشرفته‌تری مثل ارث‌بری (Polymorphism) و چندریختی (Inheritance) شوید.