

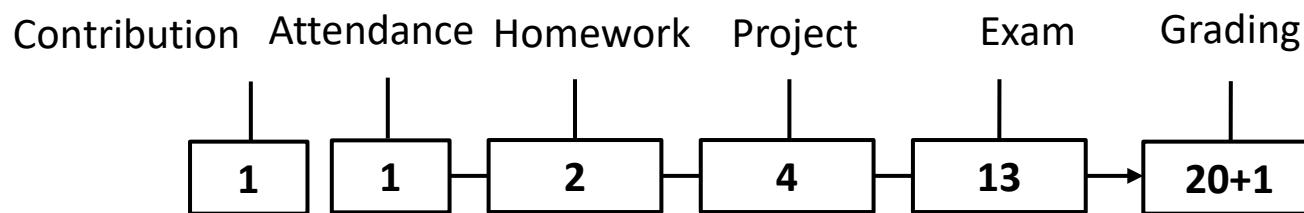
به نام خداوند جان و خرد

زبان مدل سازی پکارچه



مهندسی نرم افزار ۲

ارزیابی



زبان مدل سازی پکارچه (UML- Unified Modeling Language)

■ **UML** یک زبان گرافیکی است که اهداف آن :

■ مشخص کردن (specifying)

■ ساختن (constructing)

■ مصور سازی (visualizing)

■ مستند سازی (documenting)

اجزای یک سیستم نرم افزاری است.

■ **UML** استانداری تحت حمایت گروه مدیریت شی (Object Management Group – OMG) است

■ **UML** بهترین ابزار برای مدل سازی شی گرا است.

مورد کاربرد (Use-Cases)

- رفتار داینامیک یک سیستم را مدل می کنند. برای موارد زیر به کار می روند:
- مدل کردن محتویات یک سیستم - شناسایی Actor ها(کاربرانی که از سیستم استفاده می کنند و سیستم های خارجی که با سیستم در ارتباط هستند). و نقش آنها
- مدل سازی خواسته های سیستم - تعیین اینکه سیستم چیست مستقل از اینکه چگونه به آن می رسد (توصیف اینکه سیستم چه کاری می کند اما نه چگونه؟)
- یک Use-case عملیاتی که برای کاربر قابل مشاهده هستند را ضبط می کند.
- هر use-case هدف خاصی از کاربر را نمایش می دهد.

مدل سازی Use case

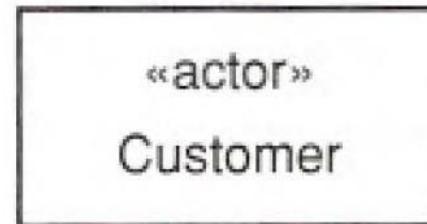
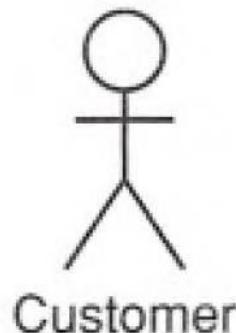
- مدل سازی use case شامل مراحل زیر است:
 - مرز سیستم را مشخص کنید.
 - Actor ها (بازیگران) را پیدا کنید.
- حداقل یک کاربر می تواند به عنوان Actor انتخاب شود.
- کمترین همپوشانی بین نقش ها وجود داشته باشد.
- Use case را پیدا کنید.
- کاری است که actor ها در سیستم انجام می دهد.
- از use case های خیلی بزرگ یا خیلی کوچک اجتناب کنید.
- تا زمانی که Actor ها و مرز سیستم به حالت ایستا برسد این عملیات را ادامه دهید.

The subject (system boundary)

- موضوع توسط کسی یا چیزی که از سیستم استفاده می کند (برای مثال : **Actor**ها) تعریف می شود و مشخص می کند که مزایای اصلی که سیستم برای **Actor**ها فراهم می کند چیست (**use case**)
- **Subject** به صورت جعبه ای همراه با نام سیستم مشخص می شود.
- **Actor**ها خارج از مزر و **use case**ها داخل جعبه قرار دارند.

Actor

- مشخص کننده نقش موجودیت های خارجی هنگام تعامل آنها با سیستم است.
- یک نقش می تواند بازی شود توسط :
 - یک کاربر
 - سیستم دیگر
 - یک قطعه سخت افزاری



Actor مدل سازی یک فروشگاه آنلاین مدرن

- بازیگران: (Actors)
 - .1 مشتری: (Customer) کاربری که به فروشگاه وارد شده، به جستجو، انتخاب، خرید محصولات و دریافت پیشنهادات هوشمندانه مشغول می‌شود.
 - .2 مدیر فروشگاه: (Store Manager) مسئول مدیریت محصولات، سفارش‌ها، موجودی کالا و نظارت بر عملکرد کلی سیستم.
 - .3 سیستم هوش مصنوعی: (AI System) این سیستم وظیفه تحلیل داده‌های مشتری، ارائه پیشنهادات شخصی‌سازی شده، پیش‌بینی روندهای خرید و بهبود تعامل با کاربر را دارد.
 - .4 سیستم پرداخت: (Payment Gateway) مسئول پردازش تراکنش‌های مالی به صورت امن و تأیید پرداخت‌ها.
 - .5 سیستم مدیریت موجودی: (Inventory Management System) سامانه‌ای برای کنترل و به روزرسانی موجودی کالاهای دریافت هشدارها در مورد کاهش موجودی و هماهنگی با بخش سفارش‌ها.

شناسایی actor

- بررسی اینکه چه کسی یا چه چیزی از سیستم استفاده می کند و آنها در تعامل با سیستم چه نقشی بازی می کنند.
- پرسش سوالات زیر برای شناسایی Actor ها :
 - چه کسی یا چه چیزی از سیستم استفاده می کند؟
 - نقش آنها در سیستم چیست؟
 - چه کسی سیستم را نصب می کند؟
 - چه کسی سیستم را شروع یا پایان می دهد؟
 - چه کسی از سیستم نگهداری میکند؟
 - چه سیستم هایی با این سیستم در ارتباطند؟
 - چه کسی یا چه چیزی اطلاعات دریافت می کند یا اطلاعات برای سیستم فراهم میکند؟

مشخصه ها Actor

- هر Actor به یک نام کوتاه نیاز دارد
- هر Actor باید یک توصیف اجمالی داشته باشد.

Actor name: Order Processing Clerk

Description: The Order Processing Clerk is responsible for processing sales orders, submitting reorder requests, requesting necessary deposits from members and scheduling the delivery of the goods to members.

Use case

- مشخصه ای از عملیات، شامل مجموعه ای از عملیات و خطاهایی که یک سیستم، زیر سیستم یا کلاس در تعامل با خارجی انجام میدهد.
- همیشه توسط یک Actor شروع می شود.
- همیشه از دیدگاه actor نوشته می شود.

PlaceOrder

GetStatus
OnOrder

شناസایی use case ها

موارد استفاده (Use Cases):

جستجو و نمایش محصولات:

- توضیح: مشتری از طریق موتور جستجو و فیلترهای پیشرفته، محصولات مورد نظر خود را جستجو می‌کند.
- نقش هوش مصنوعی: ارائه پیشنهادات اولیه بر اساس سابقه جستجو و خرید مشتری.
- پیشنهاد محصولات شخصی‌سازی شده:

توضیح: پس از ثبت‌نام یا ورود به سیستم، مشتری براساس تاریخچه فعالیت‌هایش، پیشنهادات ویژه دریافت می‌کند.

نقش هوش مصنوعی: تحلیل داده‌های رفتاری مشتریان و استفاده از الگوریتم‌های یادگیری ماشین برای پیشنهاد محصولات مطابق با سلایق فردی.

خرید و پرداخت آنلاین:

- توضیح: مشتری محصولات مورد نظر خود را به سبد خرید اضافه کرده و از طریق سیستم پرداخت، تراکنش را نهایی می‌کند.
- نقش سیستم پرداخت: تأمین امنیت تراکنش‌ها و پردازش پرداخت‌ها به صورت آنلاین.

مدیریت سفارشات و پیگیری وضعیت سفارش:

- توضیح: مشتری پس از خرید قادر به پیگیری وضعیت سفارش است؛ از ثبت سفارش تا تحویل نهایی.
- نقش مدیر فروشگاه: نظارت بر روند پردازش سفارش‌ها و هماهنگی با بخش‌های مختلف جهت تحویل به موقع.

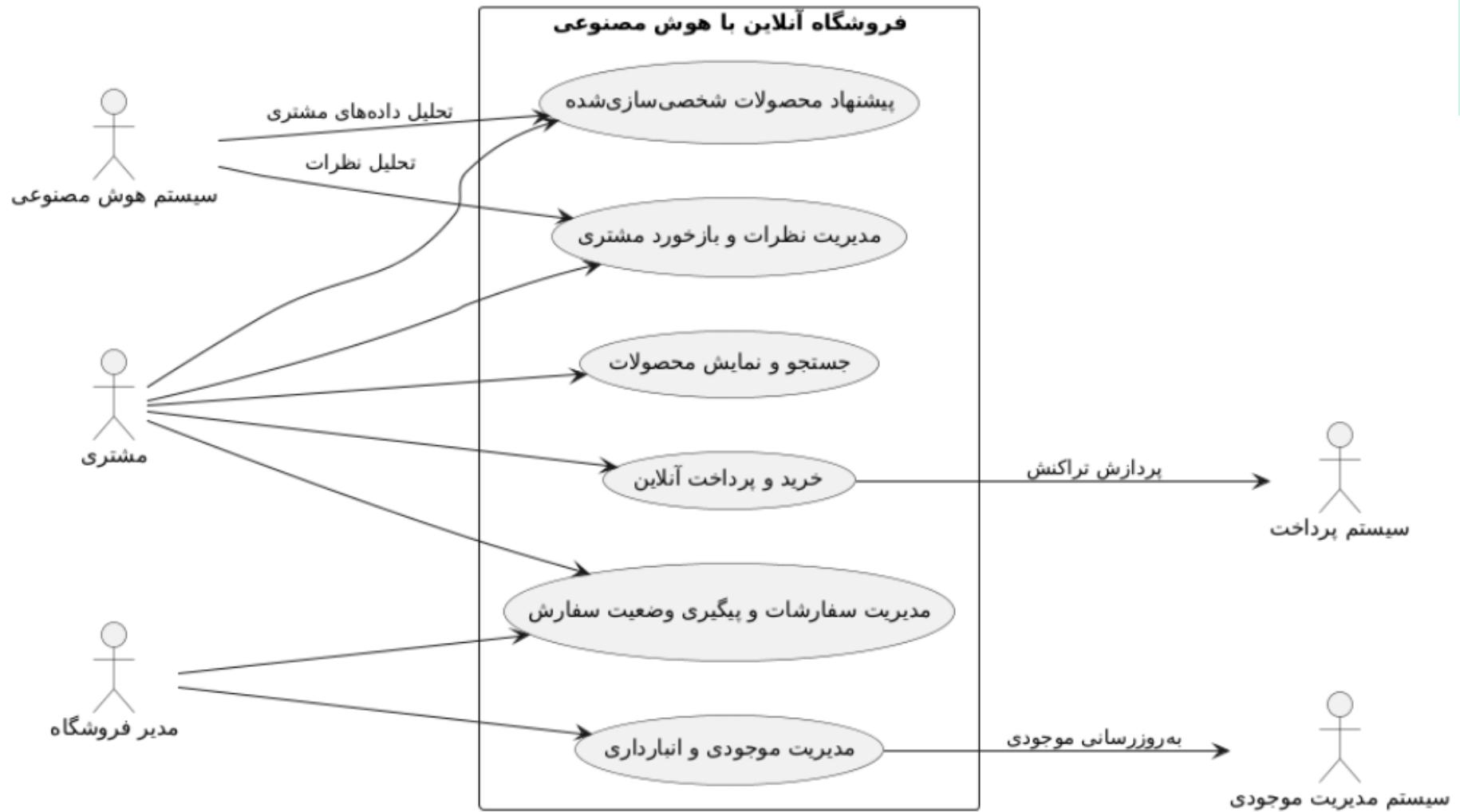
مدیریت موجودی و انبارداری:

توضیح: سیستم به صورت خودکار موجودی کالاها را به روزرسانی کرده و در صورت کاهش موجودی، هشدارهای لازم را به مدیر فروشگاه ارسال می‌کند.

نقش سیستم مدیریت موجودی: کاهش خطاهای انسانی و بهبود روند سفارش‌دهی مجدد کالاها.

مدیریت نظرات و بازخورد مشتری:

- توضیح: مشتریان پس از خرید می‌توانند نظرات و امتیازات خود را ثبت کنند که به بهبود کیفیت محصولات و خدمات کمک می‌کند.
- نقش هوش مصنوعی: تحلیل بازخوردها برای شناسایی نقاط قوت و ضعف و ارائه گزارش‌های تحلیلی به مدیر فروشگاه.



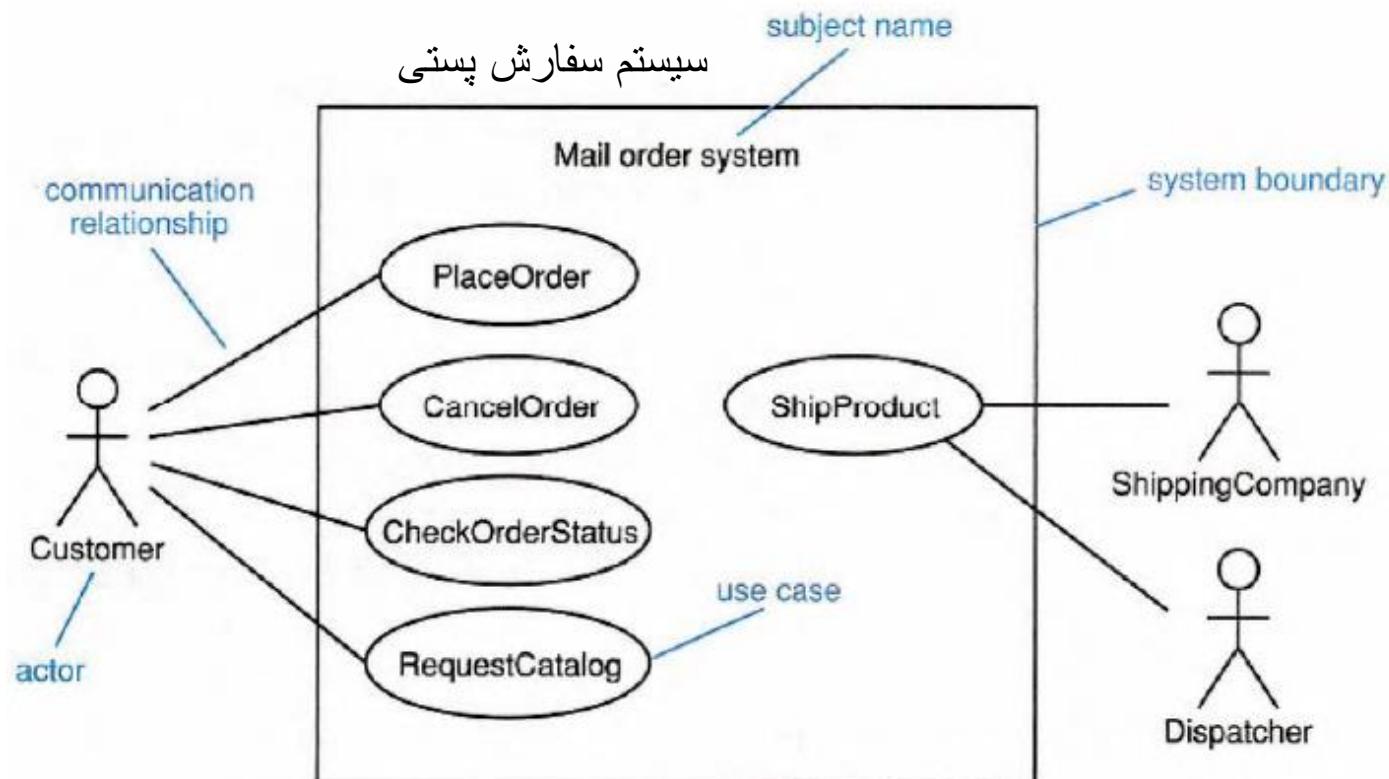
شناسایی use case ها

- بهترین راه برای شناسایی use case ها شروع با لیستی از Actor ها است و سپس بررسی اینکه چگونه هر Actor از سیستم استفاده می کند.
- هر use case با یک عبارت فعلی مشخص می شود.
- شناسایی use case ها گاهی منجر به پیدا کردن Actor های جدید می شود.

شناسایی use case ها

- سوالاتی که در شناسایی use case ها به ما کمک می کنند:
 - چه عملکردی یک Actor از سیستم انتظار دارد؟
 - آیا سیستم اطلاعات را ذخیره و بازیابی می کند؟ اگر اینچنان است کدام Actor ها این عملیات را انجام می دهند؟
 - هنگامی که حالت سیستم تغییر می کند چه چیزی اتفاق می افتد؟ (برای مثال شروع و پایان سیستم) آیا هیچ Actor ی نقش دارد؟
 - آیا رخدادهای خارجی هم روی سیستم تاثیر می گذارند؟
 - آیا سیستم با سیستم خارجی دیگری در تعامل است؟
 - آیا سیستم گزارشی تهیه می کند؟

Use case diagram



مشخصه use case

- نام use case: عبارت فعلی توصیفی و کوتاه
- شناسه use case
- توصیف اجمالی: یک پارگراف که مشخص کننده هدف use case است.
- Actor هایی که در use case هستند:
 - actor :Primary actors: use case که راه اندازی می کند.
 - Secondary actors: بعد از اینکه use case شروع به کار کرد با آن در تعامل است.
- پیش شرط ها: چیزهایی که باید قبل از اینکه use case اجرا شود برقرار باشند - آنها محدودیت های روی حالت سیستم هستند.
- روند اصلی : مراحل use case
- پس شرط ها: چیزهایی که باید در پایان use case درست باشند.
- روند جایگزین : لیست جایگزین ها برای روند اصلی

مثال : use case مشخصه

| | |
|---|---|
| use case name | Use case: PaySalesTax |
| use case identifier | ID: 1 |
| brief description | Brief description: Pay Sales Tax to the Tax Authority at the end of the business quarter. |
| the actors involved in the use case | Primary actors: Time |
| the system state before the use case can begin | Secondary actors: TaxAuthority |
| the actual steps of the use case | Preconditions: 1. It is the end of the business quarter. Main flow: implicit time actor 1. The use case starts when it is the end of the business quarter. 2. The system determines the amount of Sales Tax owed to the Tax Authority. 3. The system sends an electronic payment to the Tax Authority. |
| the system state when the use case has finished | Postconditions: 1. The Tax Authority receives the correct amount of Sales Tax. |
| alternative flows | Alternative flows: None. |

مشخصه use case : مثال

| |
|--|
| مورد کاربرد: فروش کتاب |
| شماره: ۶ |
| توصیف اجمالی: فروشنده کتاب اطلاعات فروش انجام شده را وارد می کند و سامانه موجودی کتاب ها را بهنگام می کند. |
| عامل اصلی: فروشنده |
| عامل فرعی: ندارد |
| شرایط اولیه: فروشنده باید وارد سامانه شده باشد. |
| روند اصلی: |
| ۱. این مورد کاربرد وقتی آغاز می شود که فروشنده بخواهد، اطلاعات یک فروش را در سامانه وارد کند. ۲. شامل: تهیه لیست ۳. سامانه تاریخ فروش انجام شده را از فروشنده می خواهد. ۴. فروشنده تاریخ فروش انجام شده را وارد می کند. ۵. فروشنده اطلاعات فروش را تائید می کند. ۶. سامانه تعداد درخواست شده کتاب ها را با موجودی کتابفروشی مقایسه می کند. ۷. سامانه اطلاعات خرید را ثبت می کند و موجودی را به هنگام می کند. |
| شرایط نهایی: موجودی تعدادی از کتاب ها کاهش یافه اند. |
| روند جایگزین: نداشتن موجودی کافی |

Use case : جریان ها

- مراحل در use case به صورت جریانی از رخدادها لیست می شود
- هر use case یک روند اصلی (main flow) دارد، که مراحل یک use case را وقتی همه چیز همانطور که انتظار داریم اتفاق می افتد و هیچ خطأ، وقفه یا مشکلی وجود ندارد، نشان می دهد.
- روند فرعی (alternative flow): انحراف ها از جریان اصلی که منجر به خطأ، شاخه های دیگر یا وقفه در روند اصلی می شوند.
- روند اصلی همیشه با یک عامل اصلی (primary actor) اتفاق می افتد

Use case : جریان های فرعی

- معمولاً به روند اصلی برنمی گردند زیرا اغلب به خطاهای استثنای روند اصلی رسیدگی کرده و پس شرط‌های متفاوتی دارند.
- بهتر است به طور جداگانه مستند شوند.
- ممکن است به سه حالت مختلف باشند :
 - بجای روند اصلی : توسط Actor اصلی صدازده شده و جایگزین use case می شوند.
 - بعد از یک مرحله خاص در روند اصلی اتفاق می افتد.
 - در هر زمانی در روند اصلی اتفاق می افتد.

Use case : مثال جریان های فرعی

روند جایگزین: فروش کتاب : نداشتن موجودی کافی کتاب

شماره: ۶.۱

توصیف اجمالی: سامانه به فروشنده اطلاع می دهد که تعدادی از کتاب های لیست مورد نظر به اندازه تقاضا شده موجودی ندارند و همچنین آن کتاب ها را به لیست کتاب های تقاضا شده اضافه می کند.

عامل اصلی: فروشنده

عامل فرعی: ندارد

شرایط اولیه: تعداد تقاضا شده حداقل یک کتاب در لیست از موجودی آن کتاب کمتر باشد.

روند جایگزین:

۱. روند جایگزین بعد از اتمام مرحله ۶ روند اصلی ، امکان وقوع دارد.
۲. سامانه به فروشنده کتاب هایی را که از آنها به تعداد مورد نیاز نداریم را اعلام می کند.
۳. سامانه کتاب های مورد تقاضا را ثبت می کند.

شرایط نهایی: ندارد

Use case : پیدا کردن جریان های فرعی

- جریان های فرعی را با بررسی جریان اصلی پیدا کنید. در هر مرحله در جریان اصلی موارد زیر را جستجو کنید:
 - انتخاب های ممکن برای روند اصلی
 - خطاهایی که ممکن است در روند اصلی اتفاق افتد.
 - وقفه هایی که ممکن است در یک نقطه خاص اتفاق افتد.
 - وقفه هایی که ممکن است هر لحظه اتفاق افتد.

Use case : مثال جریان های فرعی

روند جایگزین: فروش کتاب : نداشتن موجودی کافی کتاب

شماره: ۶.۱

توصیف اجمالی: سامانه به فروشنده اطلاع می دهد که تعدادی از کتاب های لیست مورد نظر به اندازه تقاضا شده موجودی ندارند و همچنین آن کتاب ها را به لیست کتاب های تقاضا شده اضافه می کند.

عامل اصلی: فروشنده

عامل فرعی: ندارد

شرایط اولیه: تعداد تقاضا شده حداقل یک کتاب در لیست از موجودی آن کتاب کمتر باشد.

روند جایگزین:

۱. روند جایگزین بعد از اتمام مرحله ۶ روند اصلی ، امکان وقوع دارد.
۲. سامانه به فروشنده کتاب هایی را که از آنها به تعداد مورد نیاز نداریم را اعلام می کند.
۳. سامانه کتاب های مورد تقاضا را ثبت می کند.

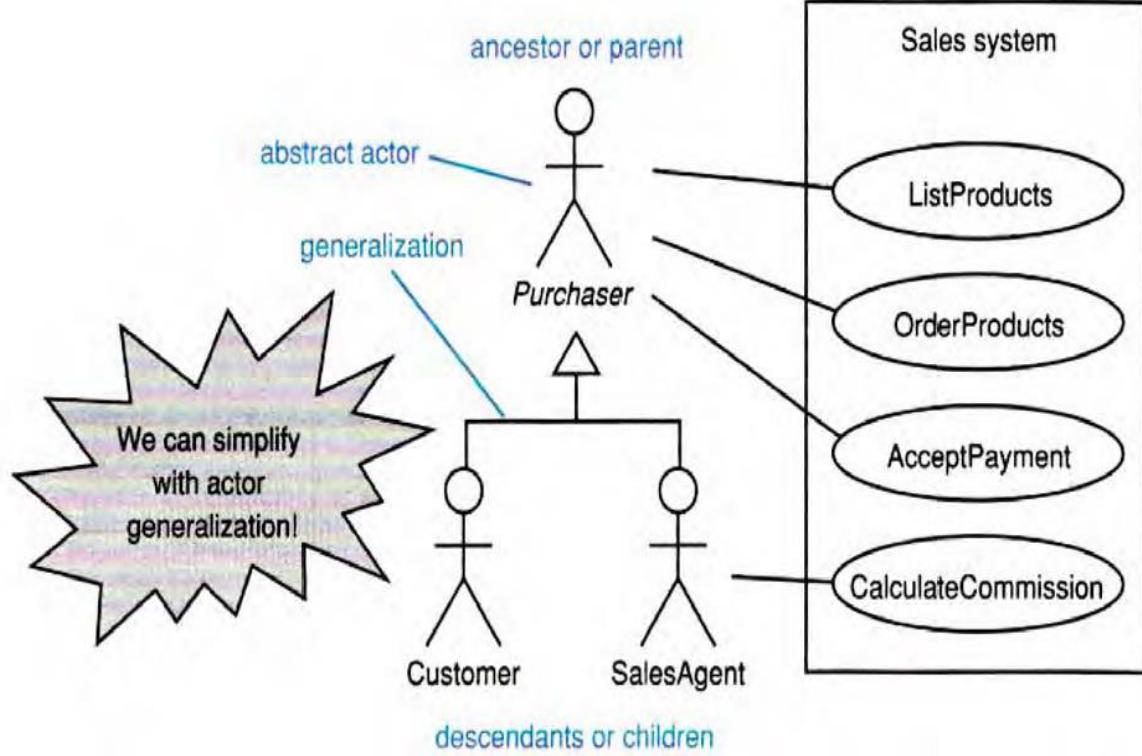
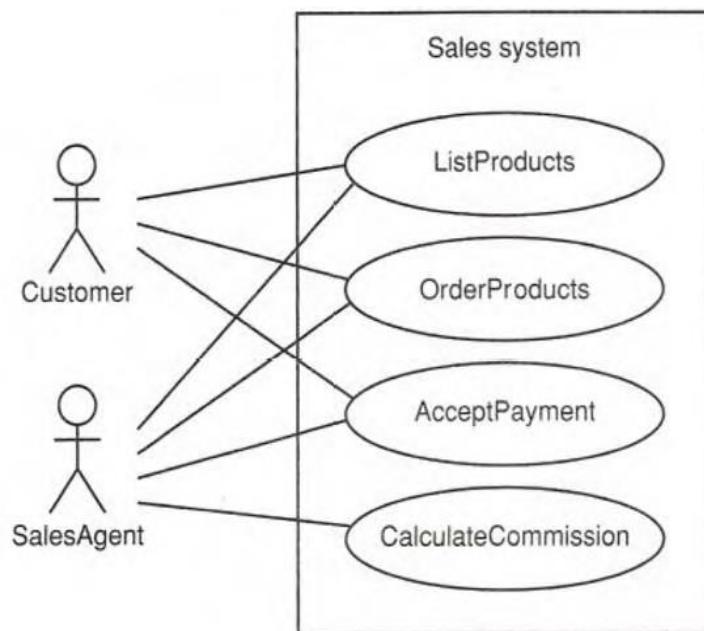
شرایط نهایی: ندارد

Relationships

- رابطه بین یک Actor کلی و یک Actor جزئی: **actor generalization**
- رابطه بین یک use case کلی و یک use case جزئی: **use case generalization**
 - فرزندهای use-case از use case پدر را به ارث می‌برد.
- رابطه بین use case‌ها که یک use case دیگر استفاده می‌کند: **<<include>>**
 - رفتار use case دیگر ضمیمه use case اصلی است.
- رابطه بین use case‌ها که یک use-case خاصی از use-case اصلی است: **<<extend>>**
 - use-case خاصی ممکن است مستقل باشد، اما تحت شرایط خاص ممکن است گسترش داده شود.

Actor generalization

اگر دو Actor با هم از طریق مجموعه یکسانی از use case ها در ارتباط باشند



Use case generalization

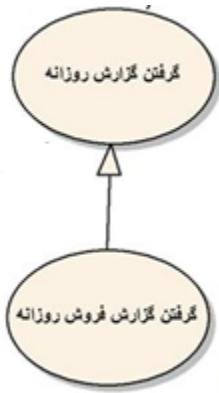
- هنگامی استفاده می شود که یک یا چند **use case** حالت خاص یک نمونه کلی هستند.
- **Use case** های فرزند ممکن است:
 - ویژگی هایی را از **use case** پدر به ارث ببرند.
 - ویژگی های جدیدی اضافه کنند
 - ویژگی های ارث برده را تغییر دهند.

use case generalization مثال

| Sales system | |
|---|---|
| Customer | FindProduct |
| | FindBook |
| | FindCD |
| Use case: FindProduct | |
| ID: 6 | |
| Brief description: | |
| The Customer searches for a product. | |
| Primary actors: | |
| Customer | |
| Secondary actors: | |
| None. | |
| Preconditions: | |
| None. | |
| Main flow: | |
| <ol style="list-style-type: none"> 1. The Customer selects "find product". 2. The system asks the Customer for search criteria. 3. The Customer enters the requested criteria. 4. The system searches for products that match the Customer's criteria. 5. If the system finds some matching products <ol style="list-style-type: none"> 5.1 The system displays a list of the matching products. 6. Else <ol style="list-style-type: none"> 6.1 The system tells the Customer that no matching products could be found. | overridden overridden inherited without change overridden overridden added overridden and renumbered added added inherited without change added renumbered |
| Postconditions: | |
| None. | |
| Alternative flows: | |
| None. | |

| | |
|--|---|
| | Use case: FindBook |
| ID: 7 | |
| Parent ID: 6 | |
| Brief description: | |
| The Customer searches for a book. | |
| Primary actors: | |
| Customer | |
| Secondary actors: | |
| None. | |
| Preconditions: | |
| None. | |
| Main flow: | |
| <ol style="list-style-type: none"> 1. (o1.) The Customer selects "find book". 2. (o2.) The system asks the Customer for book search criteria comprising author, title, ISBN, or topic. 3. (3.) The Customer enters the requested criteria. 4. (o4.) The system searches for books that match the Customer's criteria. 5. (o5.) If the system finds some matching books <ol style="list-style-type: none"> 5.1 The system displays the current best seller. 5.2 (o5.1) The system displays details of a maximum of five books. 5.3 For each book on the page the system displays the title, author, price, and ISBN. 5.4 While there are more books, the system gives the Customer the option to display the next page of books. 6. (6.) Else <ol style="list-style-type: none"> 6.1 The system displays the current best seller. 6.2 (6.1) The system tells the Customer that no matching products could be found. | inherited without change overridden overridden added overridden and renumbered added added inherited without change added renumbered |
| Postconditions: | |
| None. | |
| Alternative flows: | |
| None. | |

مثال use case generalization



مورد کاربرد: گرفتن گزارش روزانه

شماره: ۷

توضیف اجمالی: مدیر می تواند بر روی اطلاعات مشخصی در سامانه گزارش روزانه تهیه کند.

عامل اصلی: مدیر

عامل فرعی: ندارد

شرایط اولیه: مدیر باید وارد سامانه شده باشد.

رونده اصلی:

۱. این مورد کاربرد وقتی آغاز می شود که مدیر بخواهد از سامانه گزارش روزانه بگیرد.

۲. سامانه از مدیر، تاریخ جهت گزارش گیری را سوال می کند.

۳. مدیر تاریخ مورد نظر خود را وارد می کند.

۴. مدیر درخواست مشاهده گزارش را می دهد.

۵. سامانه گزارش مورد نظر مدیر را به او نشان می دهد.

شرایط نهایی: ندارد

روندهایگزین: ندارد

مورد کاربرد: گرفتن گزارش روزانه فروش

شماره: ۸

شماره پدر: ۷

توضیف اجمالی: مدیر می تواند بر روی اطلاعات فروش ثبت شده در سامانه گزارش روزانه تهیه کند.

عامل اصلی: مدیر

عامل فرعی: ندارد

شرایط اولیه: مدیر باید وارد سامانه شده باشد.

رونده اصلی:

۱. (۰۱) این مورد کاربرد وقتی آغاز می شود که مدیر بخواهد، از سامانه گزارش روزانه فروش بگیرد.

۲. (۰۲) سامانه از مدیر، تاریخ جهت گزارش گیری را سوال می کند.

۳. مدیر تاریخ مورد نظر خود را وارد می کند.

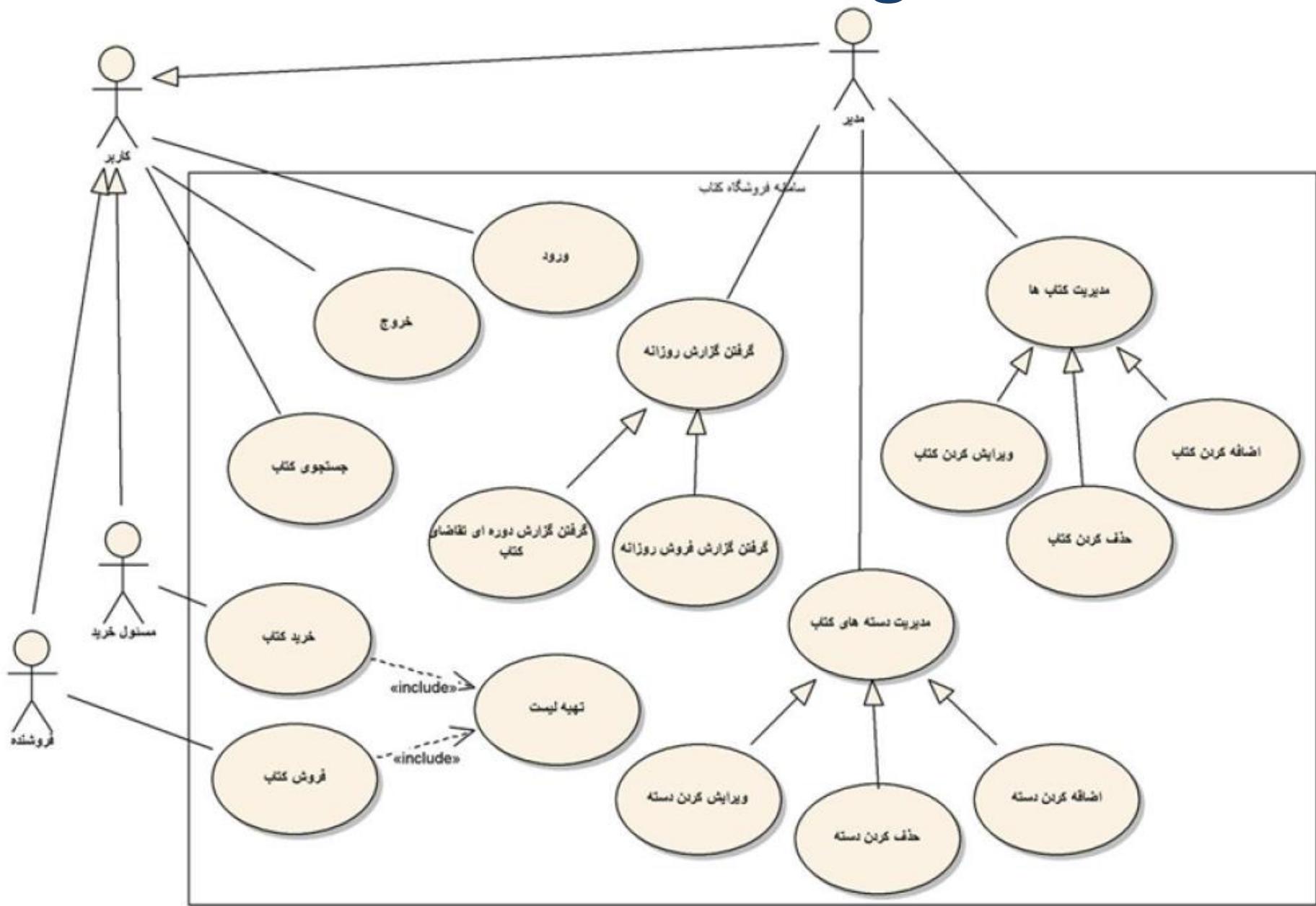
۴. مدیر درخواست مشاهده گزارش را می دهد.

۵. (۰۵) سامانه گزارش فروش روزانه را به مدیر نشان می دهد.

شرایط نهایی: ندارد

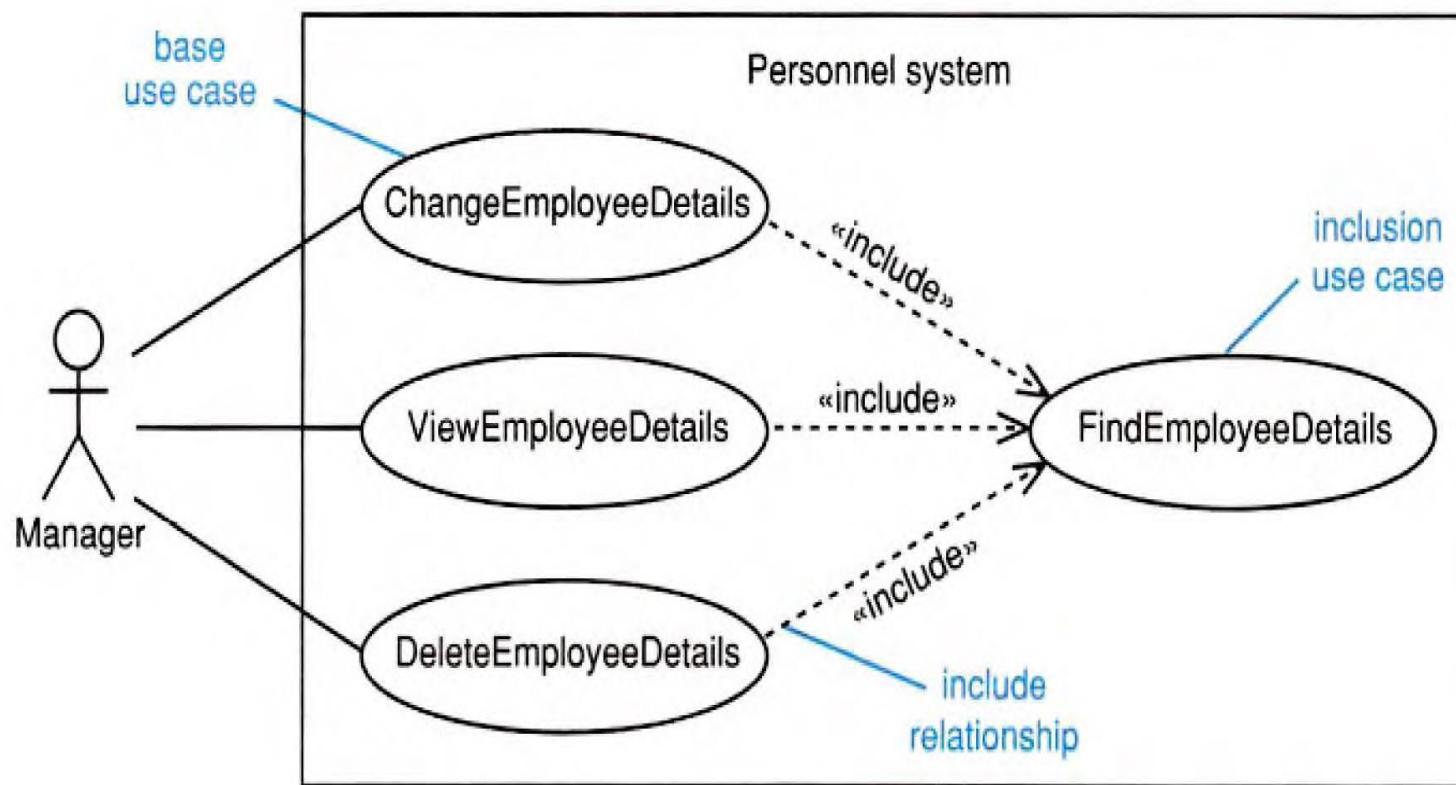
روندهایگزین: ندارد

use case diagram مثال



<<INCLUDE>> رابطه

- رابطه <<include>> بین use case ها به شما اجازه می دهد تا رفتار یک use case را به جریان use case دیگر اضافه نمایید.



رابطه <<INCLUDE>> (مشخصه)

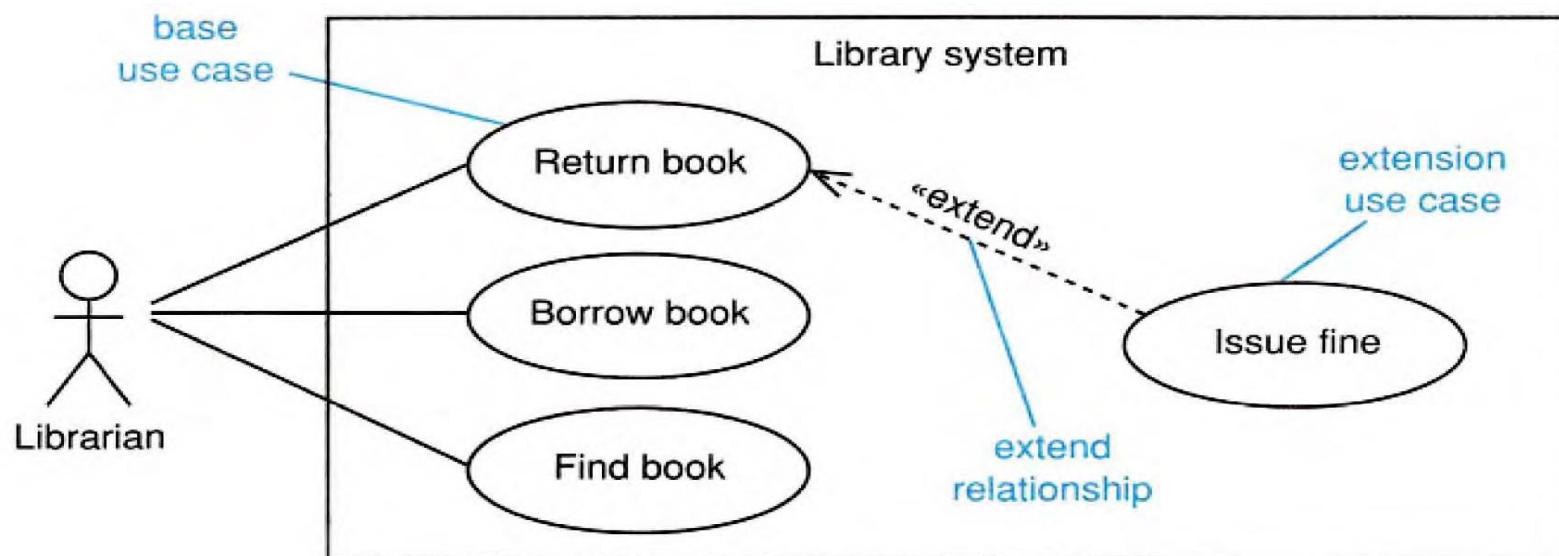
پایه بدون اضافه کردن **use case** دیگر کامل **Use case** نیست. **include** شده می تواند کامل باشد یا نباشد.

| Use case: ChangeEmployeeDetails | |
|---------------------------------|---|
| ID: 1 | |
| Brief description: | The Manager changes the employee details. |
| Primary actors: | Manager |
| Secondary actors: | None. |
| Preconditions: | 1. The Manager is logged on to the system. |
| Main flow: | 1. include(FindEmployeeDetails). 2. The system displays the employee details. 3. The Manager changes the employee details. ... |
| Postconditions: | 1. The employee details have been changed. |
| Alternative flows: | None. |

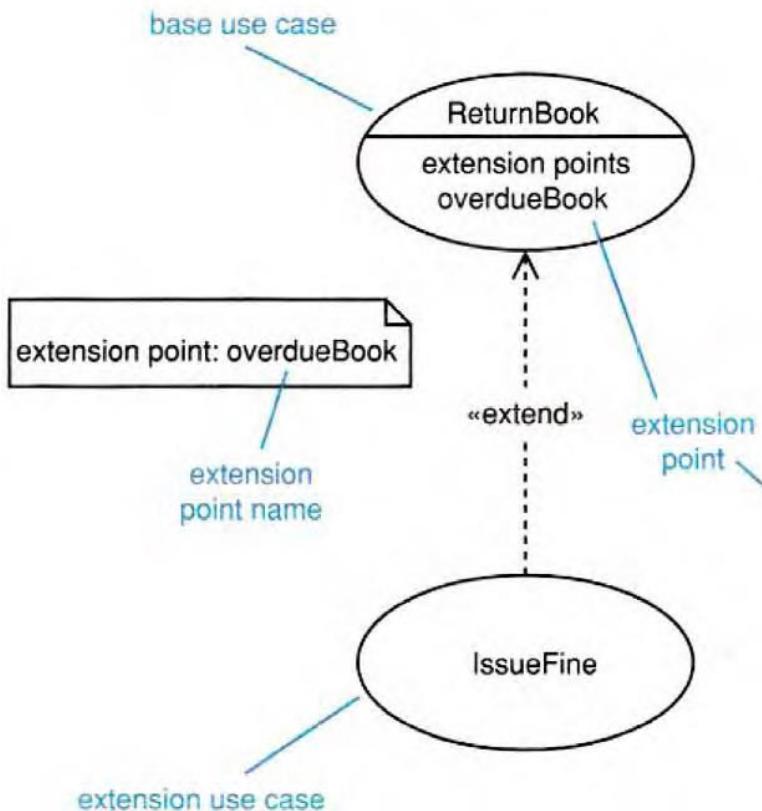
| Use case: FindEmployeeDetails | |
|-------------------------------|---|
| ID: 4 | |
| Brief description: | The Manager finds the employee details. |
| Primary actors: | Manager |
| Secondary actors: | None. |
| Preconditions: | 1. The Manager is logged on to the system. |
| Main flow: | 1. The Manager enters the employee's ID. 2. The system finds the employee details. |
| Postconditions: | 1. The system has found the employee details. |
| Alternative flows: | None. |

<<extend>> رابطه

- راهی برای اضافه کردن رفتار جدید به **use case** موجود فراهم می کند.
- **use case** پایه ، کامل است و **use case** اضافه شده مجموعه ای از قطعات اضافی است که می توانند به **use case** اضافه شوند.



رابطه <<extend>> (مثال)



| |
|---|
| Use case: ReturnBook |
| ID: 9 |
| Brief description: The Librarian returns a borrowed book. |
| Primary actors: Librarian |
| Secondary actors: None. |
| Preconditions: 1. The Librarian is logged on to the system. |
| Main flow: 1. The Librarian enters the borrower's ID number. 2. The system displays the borrower's details including the list of borrowed books. 3. The Librarian finds the book to be returned in the list of books. extension point: overdueBook 4. The Librarian returns the book. ... |
| Postconditions: 1. The book has been returned. |
| Alternative flows: None. |

تفاوت‌های کلیدی پین روابط <<include>> و <<extend>>

| Extending Use Case | Included Use Case | |
|--------------------|-------------------|--|
| خیر | بله | این use case اجباری است؟ |
| بله | خیر | آیا use case اولیه بدون این use case کامل است؟ |
| بله | خیر | آیا اجرای این use case شرطی است؟ |
| بله | خیر | آیا این use case رفتار اولیه را تغییر می دهد؟ |

Class diagram

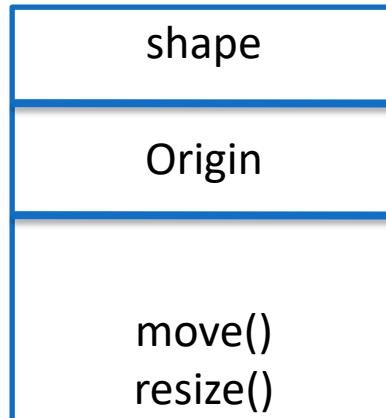
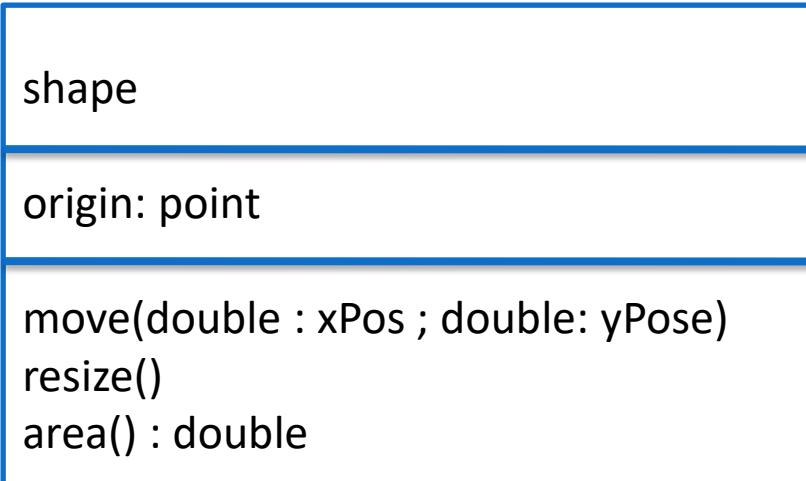
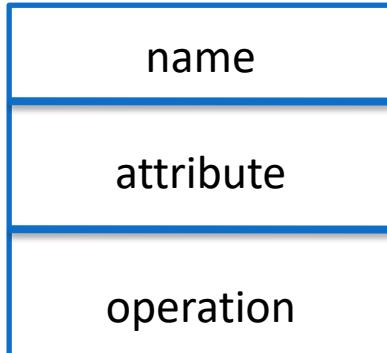
کلاس چیست؟

- اشیائی که ساختار و رفتار مشترک دارند در یک کلاس قرار می گیرند.
- کلاس ها انتزاعی از اشیائی هستند که در زمان و فضا وجود دارند. - کلاس ها و اشیا به یکدیگر متصل هستند.

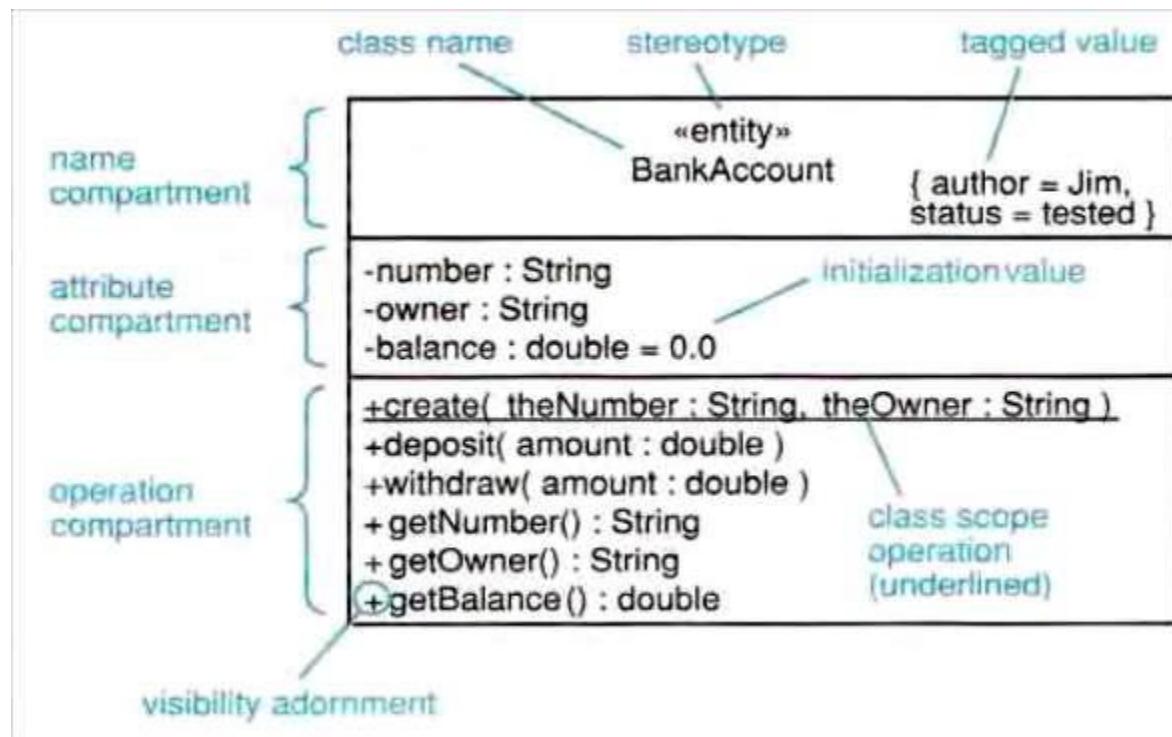
کلاس ها (ادامه...)

Semantic

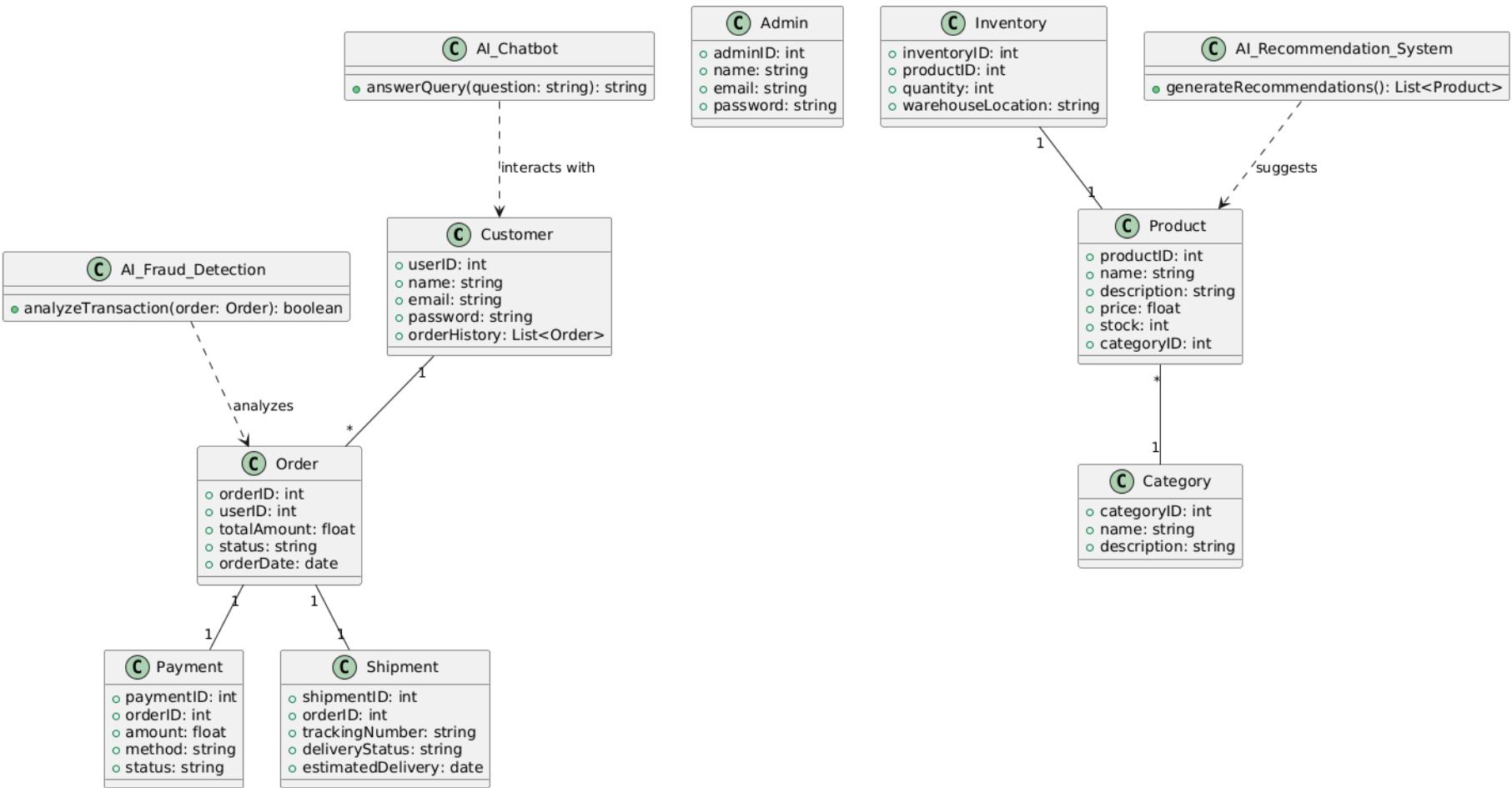
- Name
- Attributes
- operations
- نمادهای کلاس



نشانه گذاری کلاس در UML



online shop enhanced with AI features



روابط کلاس ها و اشیاء

▪ رابطه بین دو شی فرضیاتی است که هر کدام نسبت به هم دارند. عملیات و نتایج مورد انتظار بسیار مهم هستند.

▪ **روابط :**

▪ **انجمنی (association)**

▪ **وابستگی (dependency)**

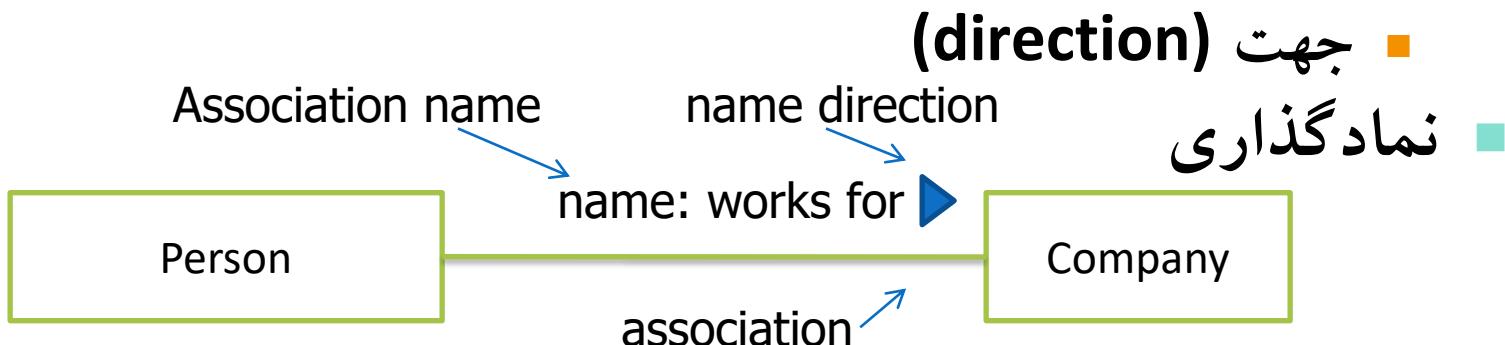
▪ **(generalization/specialization) تعمیم / تخصصی کردن**

▪ **تجمعی (Aggregation)**

رابطه انجمنی (association relationship)

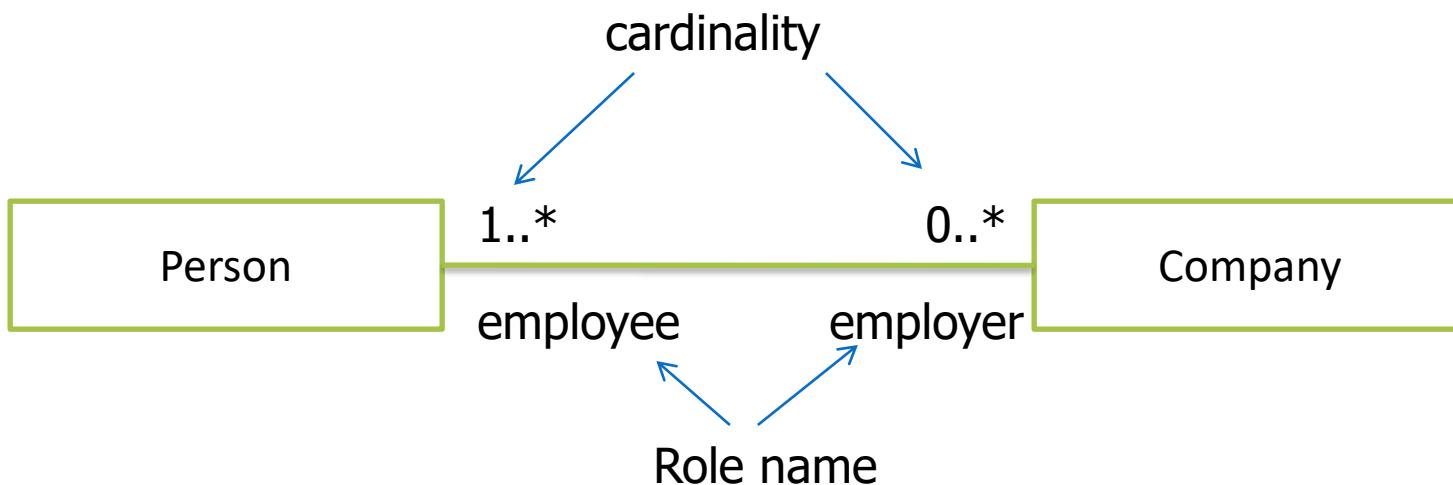
- رابطه ای که در آن یک شی به شی دیگر متصل می شود.
- اگر بین دو شی لینکی وجود دارد باید یک رابطه وجود داشته باشد.

- Dependency** یا **Association**
- نام (name) ■
 - نقش (role) ■
 - چندگانگی (multiplicity): چندی رابطه بین دو شی را نشان می دهد. یک به یک ، یک به چند، چند به چند



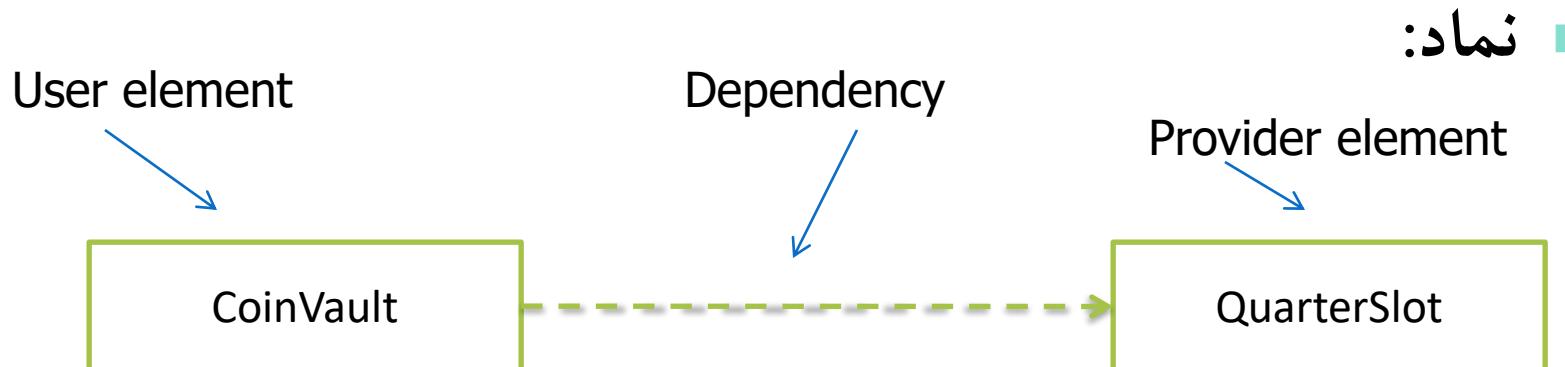
رابطه انجمنی (ادامه...)

- نام Association: باید یک فعل یا عبارت باشد
- نام نقش :
- باید یک اسم یا عبارت اسمی باشد که توصیف کننده نقش است.
- چندی: نشان دهنده تعداد اشیائی است که می توانند در رابطه وجود داشته باشند.



رابطه وابستگی (Dependency Relationship)

- این رابطه نشان می دهد که یک تغییر در **provider** نیاز به تغییری در **user element** را باعث می کند. معمولاً رابطه وابستگی نشان می دهد که یک کلاس (**user**) از یک کلاس دیگر (**provider**) به عنوان آرگومانی در امضای یکی از توابع اش استفاده می کند.



QuarterSlot از CoinVault استفاده می کند

رابطه وابستگی (Dependency Relationship)

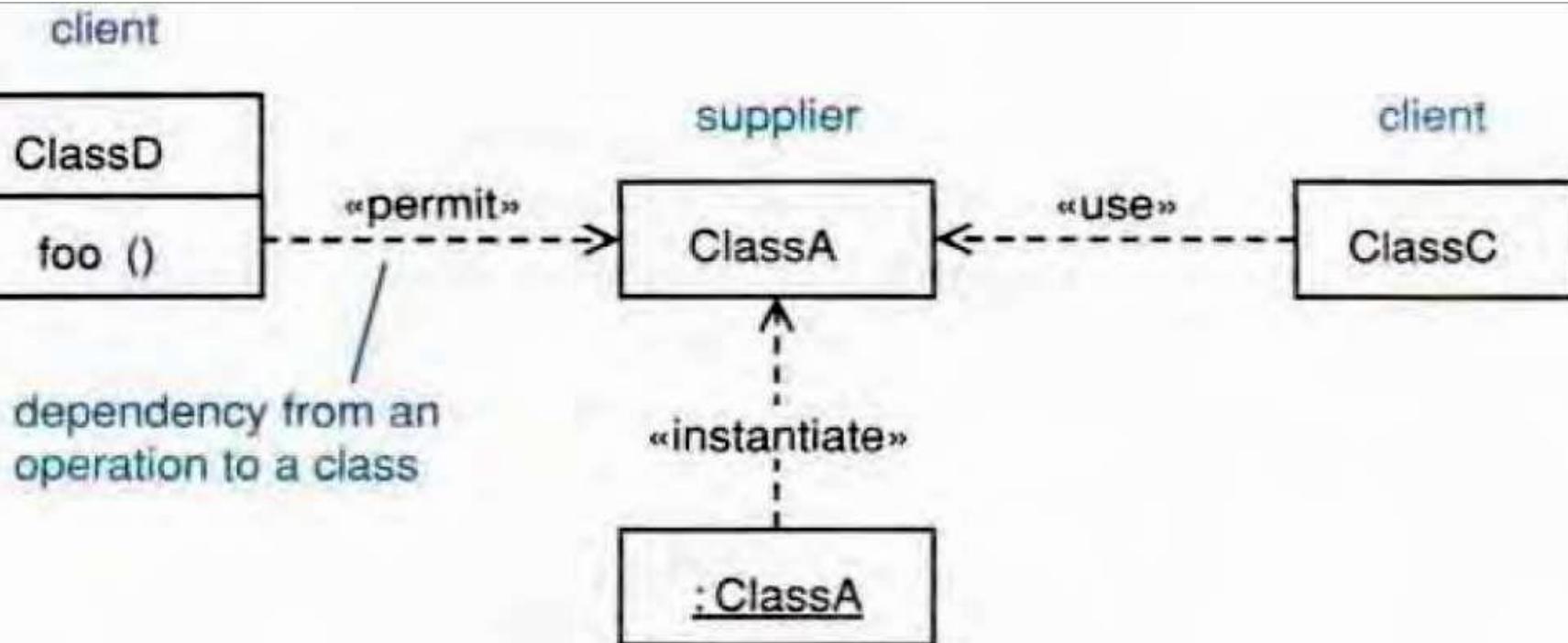
- این رابطه نشان می دهد که یک تغییر در **provider** نیاز به تغییری در **user element** را می خواهد. معمولاً رابطه وابستگی نشان می دهد که یک کلاس (**user**) از یک کلاس دیگر (**provider**) به عنوان آرگومانی در امضای یکی از توابع اش استفاده می کند.
- نماد:

رابطہ وابستگی (Dependency Relationship)

```
class Document {  
    String content;  
  
    Document(String content) {  
        this.content = content;  
    }  
  
    String getContent() {  
        return content;  
    }  
}  
  
class Printer {  
    void print(Document doc) {  
        System.out.println("Printing: " + doc.getContent());  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Document myDoc = new Document("Hello, UML!");  
        Printer myPrinter = new Printer();  
        myPrinter.print(myDoc);  
    }  
}
```

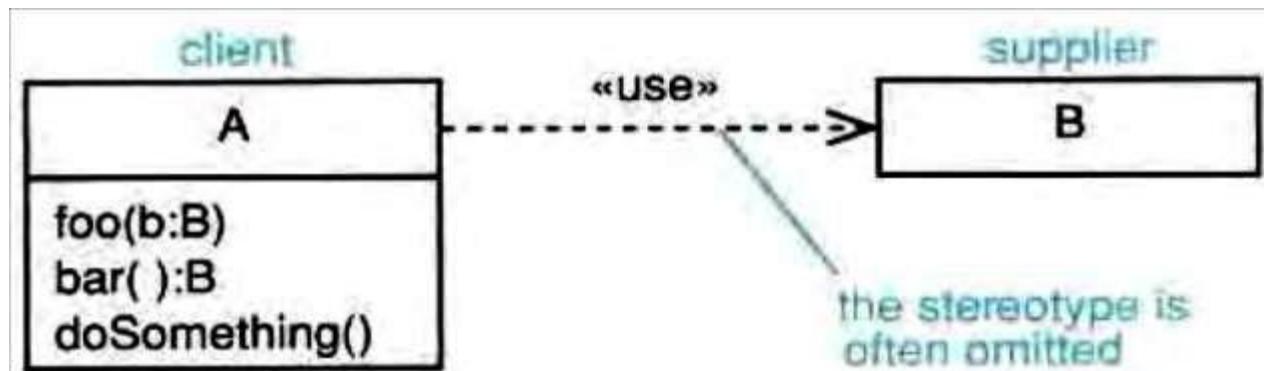
رابطہ وابستگی

(Dependency Relationship)

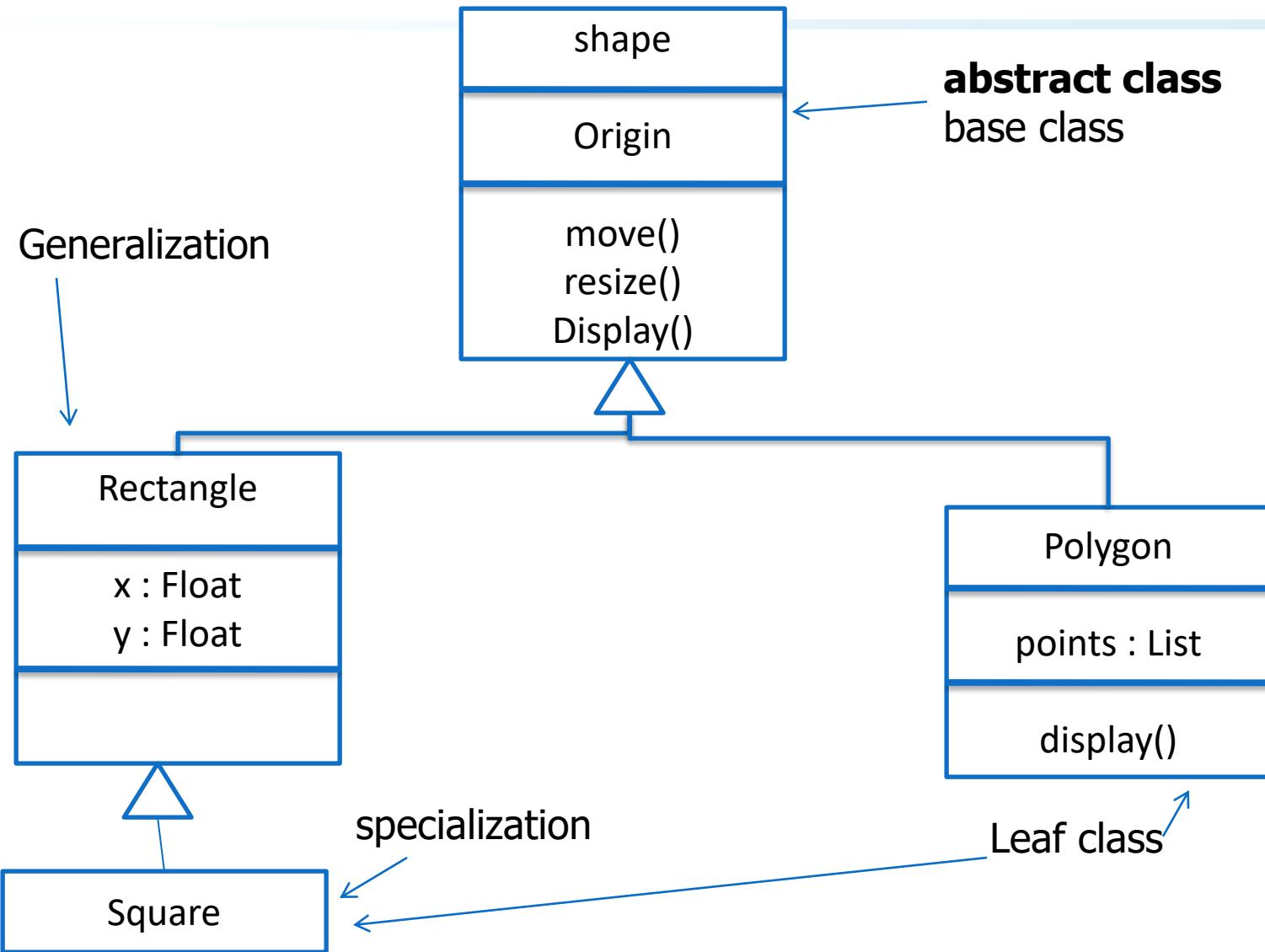


کاربرد و ابستگی

- کلاس کلاینت از **supplier** استفاده می کند. ■
- تابع کلاینت ، تابع **supplier** را صدا میزنند. ■
- یک پارامتر یا مقدار برگشتی **supplier** - <<parameter>> ■
یکی از توابع کلاینت است. ■
- کلاینت یک سیگنال به **supplier** می فرستد. ■
- کلاینت نمونه ای از **Supplier** است. ■



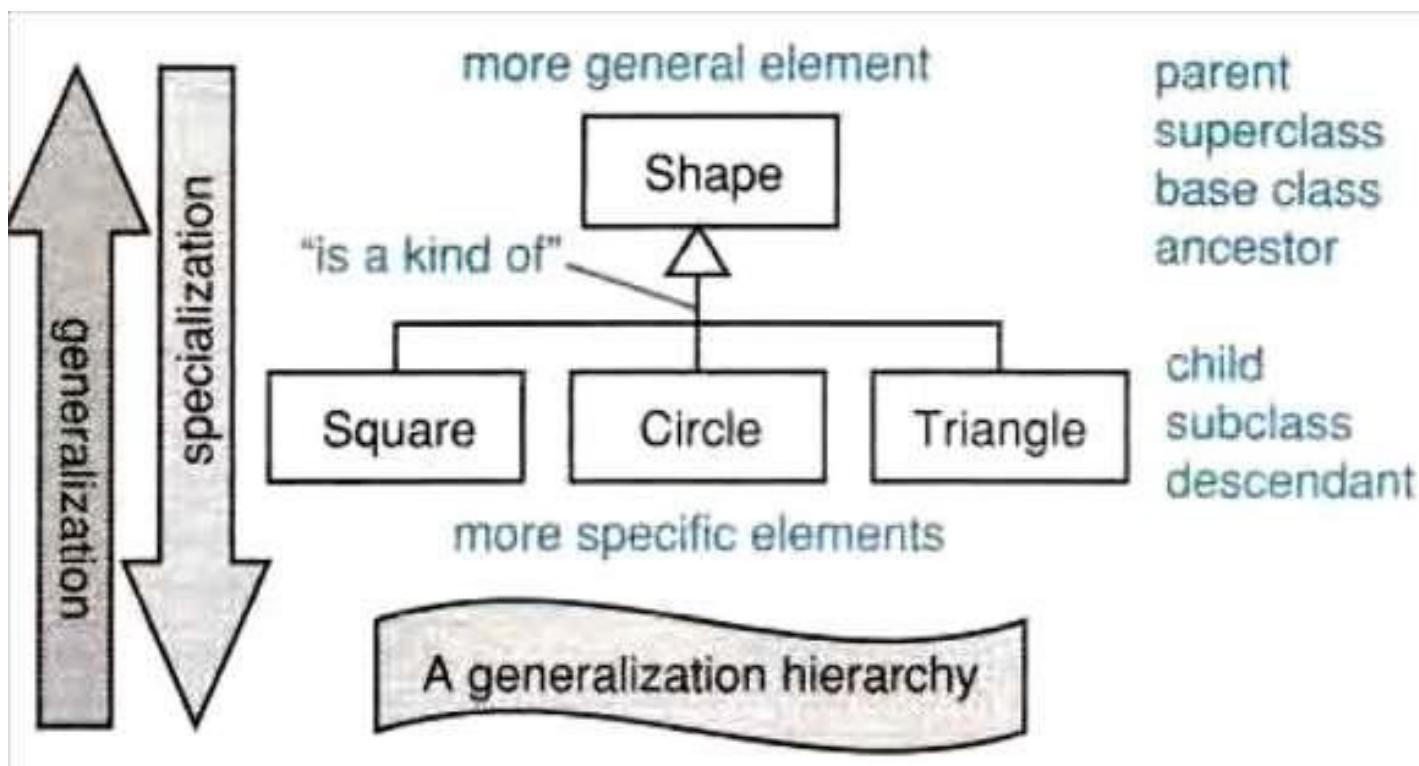
رابطه وارثت یگانه (single inheritance)



Generalization/specialization

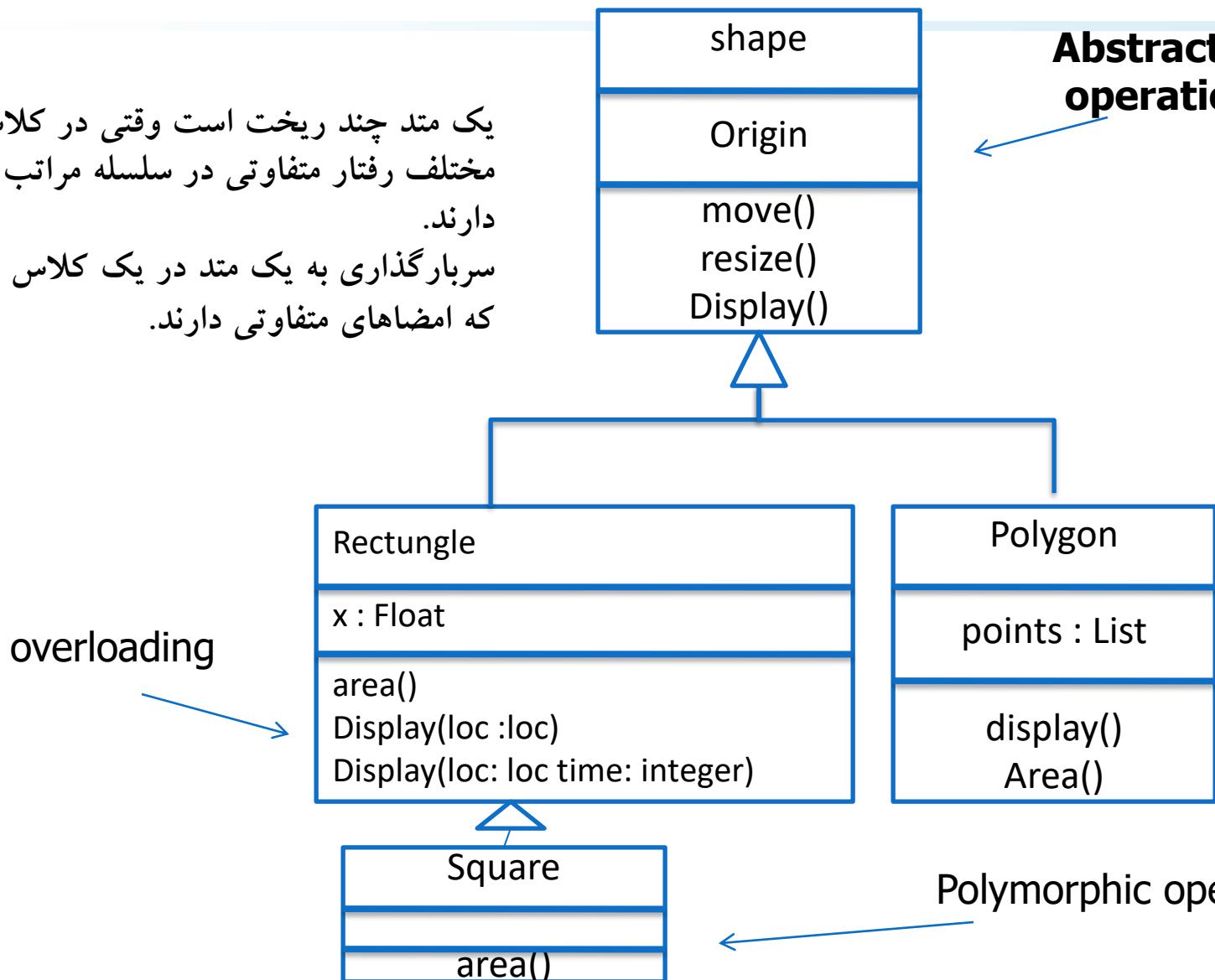
تعمیم (حرکت به سمت بالا) : فرآیندی است که در آن ویژگی‌های مشترک دو یا چند کلاس استخراج شده و یک کلاس فوق العاده عمومی‌شده ایجاد می‌شود.

تخصص‌سازی (حرکت به سمت پایین) : فرآیندی است که در آن زیرکلاس‌های جدید از یک کلاس موجود با افزودن ویژگی‌ها یا رفتارهای خاص ایجاد می‌شوند.

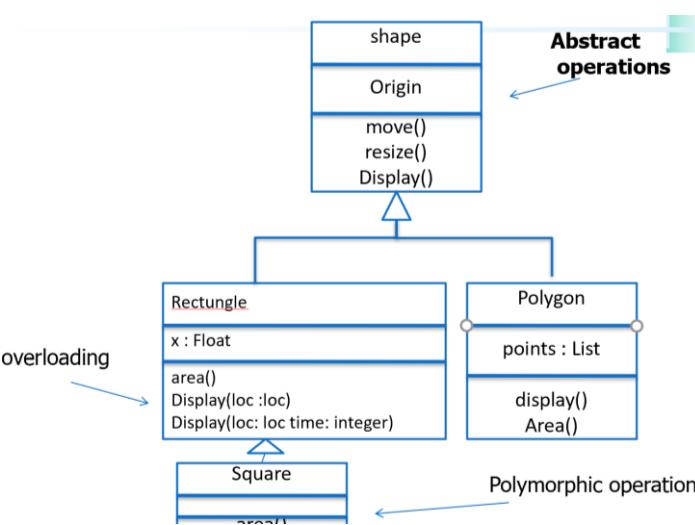


Polymorphism and Overloading

یک متدهای مختلف ریخت است وقتی در کلاسها دارند.
متدهای مختلف رفتار متفاوتی در سلسله مراتب کلاسها دارند.
سربارگذاری به یک متدهای اشاره میکند که امضاها متفاوتی دارند.



Polymorphism and Overloading



تفاوت چندریختی (Overloading) و تابع چندبارگی (Polymorphism)

1. چندریختی (Polymorphism):

وقتی کلاس‌های فرزند متدهای انتزاعی کلاس والد را بازنویسی (Override) می‌کنند و هر کدام رفتار مخصوص به خودشان را پیاده‌سازی می‌کنند. برای مثال، `area()` در مستطیل، چندضلعی یا مربع به شکل‌های متفاوتی پیاده‌سازی می‌شود؛ در نتیجه هنگام صدا زدن `area()` روی یک آبجکت از جنس **Shape**، بسته به نوع واقعی آن (Rectangle, Polygon, Square) پیاده‌سازی متفاوتی اجرا می‌شود.

2. چندبارگی توابع (Overloading):

وقتی در یک کلاس واحد چند متدهای انتزاعی با نام یکسان اما امضای متفاوت تعریف می‌شوند. در این شکل، `Display(loc : loc, time : integer)` و `Display(loc : loc)` نمونه‌هایی از Overloading هستند. هر دو نامشان در کلاس **Rectangle** است، اما پارامترهای متفاوتی دارند.

پلی مورفیسم – چندریختی (Polymorphism)

تعریف:

چندریختی به این معناست که یک تابع، متدها یا شیء می‌تواند چند شکل مختلف داشته باشد. به عبارت دیگر، یک اینترفیس واحد می‌تواند رفتارهای مختلفی بسته به نوع شیء واقعی ارائه دهد.

```
class Animal:
```

```
    def speak(self):
```

```
        print("Animal sound")
```

```
class Dog(Animal):
```

```
    def speak(self):
```

```
        print("Woof!")
```

```
class Cat(Animal):
```

```
    def speak(self):
```

```
        print("Meow!")
```

```
def make_sound(animal: Animal):
```

```
    animal.speak()
```

```
make_sound(Dog()) # خروجی: Woof!
```

```
make_sound(Cat()) # خروجی: Meow!
```

تابع Overriding بازنویسی

- بازنویسی یعنی یک کلاس فرزند متده را که از کلاس پدر به ارث برده باز تعریف کند تا رفتار خاص خود را داشته باشد. این کار فقط زمانی انجام می‌شود که متده در کلاس پایه موجود باشد و در کلاس مشتق با همان نام و امضا دوباره تعریف شود.

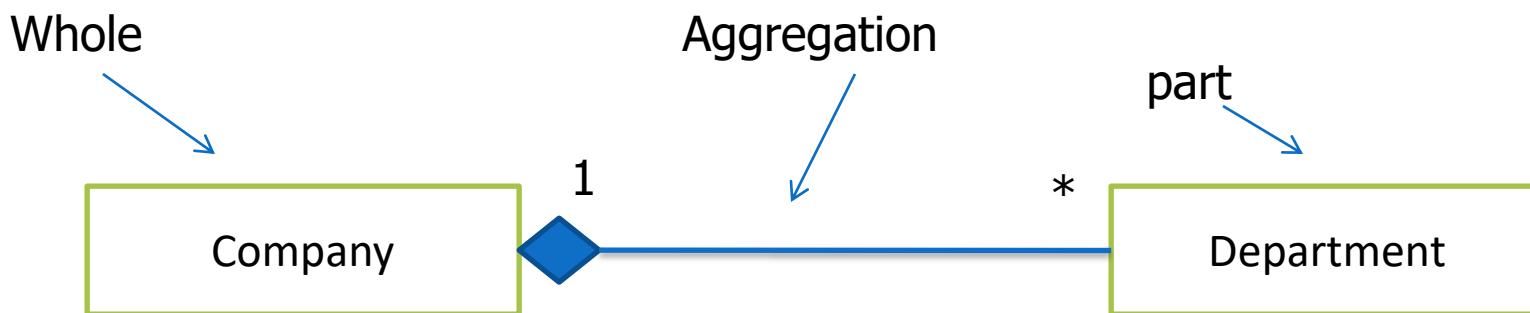
```
class Parent:  
    def greet(self):  
        print("Hello from Parent")
```

```
class Child(Parent):  
    def greet(self):  
        print("Hello from Child")
```

```
obj = Child()  
obj.greet() # خروجی: Hello from Child
```

رابطه تجمعی (Aggregation Relationship)

- معنایی: این رابطه کل (whole) را از بخش (part) جدا می کند. یک رابطه aggregation یک رابطه "has - a" است - یک شی از کل شامل اشیائی از بخش است.
- نماد:



نمودار کلاس (class diagram)

مدل سازی واژگان (Vocabulary) و همکاری‌ها (Collaborations): این بخش به تعریف موجودیت‌ها و نحوه همکاری یا ارتباط آن‌ها می‌پردازد. به کمک نمودار کلاس می‌توانیم مفاهیم کلیدی (واژگان) و روابط متقابل (همکاری‌ها) را در سیستم شناسایی کنیم.

ماهیت توصیفی نمودارهای کلاس: این نمودارها برای مصورسازی (Specification)، مشخص‌سازی (Visualization) و حتی ساخت سیستم‌های اجرایی از طریق فرایند مهندسی پیشرو (Forward Engineering) سودمند هستند. در مهندسی پیشرو، می‌توان بخشی از کد سیستم را مستقیماً از روی مدل‌ها تولید کرد.

یک کلاس دیاگرام ممکن است به چند sub-class diagram تجزیه شود. کمک به سایر نمودارها:

نمودارهای کلاس پایه‌ای برای تهیه نمودارهای مؤلفه (Component Diagrams) و نمودارهای استقرار (Deployment Diagrams) نیز هستند و نقش مهمی در تکمیل دید کلی از سیستم ایفا می‌کنند.

کاربردهای نمودار کلاس (class diagram)

طراحی سیستم‌های نرم‌افزاری:

- دیاگرام کلاس به توسعه‌دهندگان کمک می‌کند تا ساختار سیستم را قبل از شروع کدنویسی طراحی کنند.

◦ مثلاً در طراحی یک سیستم مدیریت دانشگاه، کلاس‌هایی مانند "دانشجو"، "استاد"، و "درس" با روابط مشخص ایجاد می‌شوند.

درک روابط بین اجزا:

- با استفاده از دیاگرام کلاس، می‌توان روابط بین اجزای سیستم را به وضوح مشاهده کرد.
- مثلاً در یک سیستم بانکی، رابطه بین کلاس‌های "حساب بانکی" و "مشتری" مشخص می‌شود.

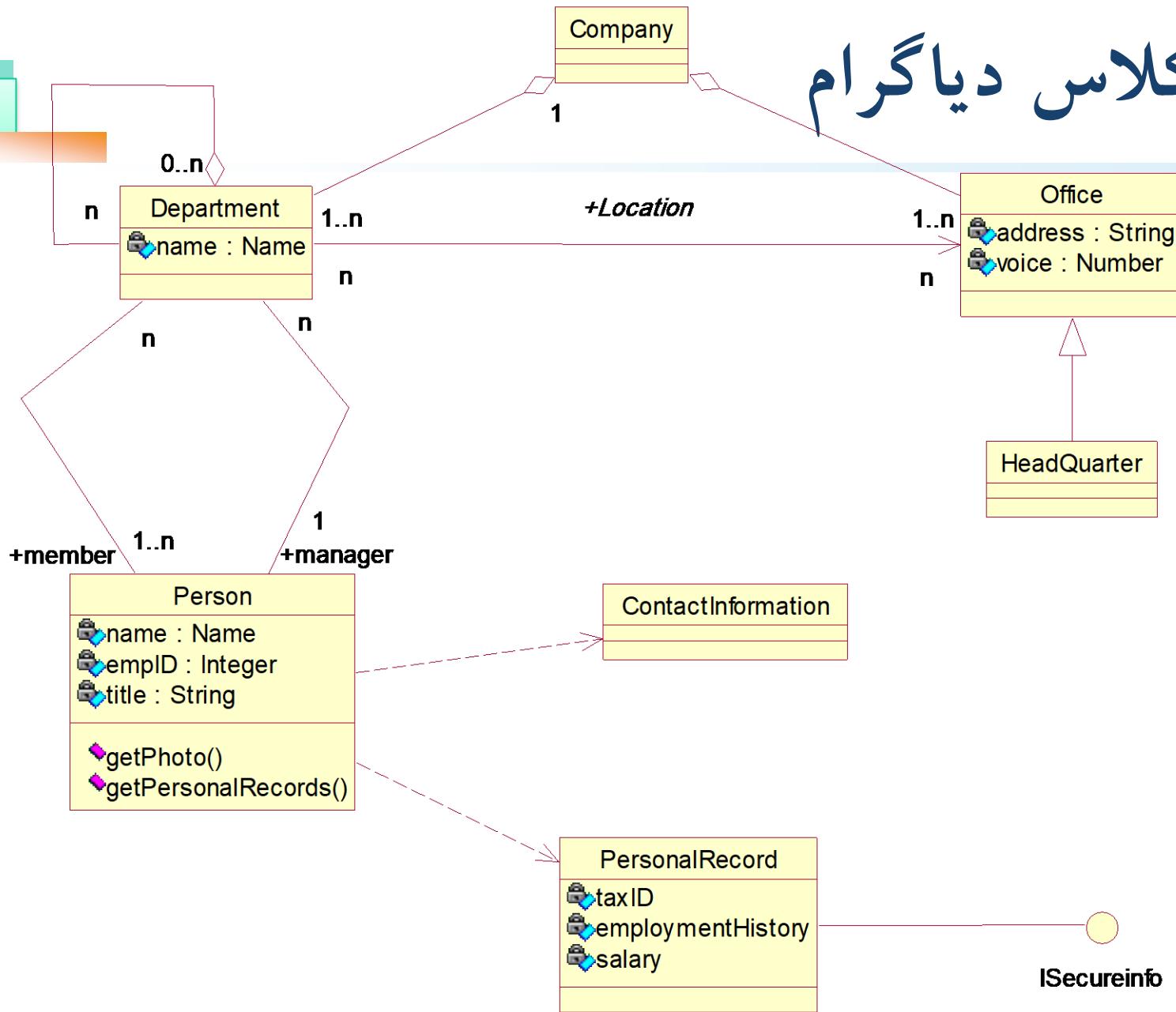
مدیریت پیچیدگی:

- در سیستم‌های بزرگ، دیاگرام کلاس به سازماندهی کد و کاهش پیچیدگی کمک می‌کند.
- مثلاً در یک سیستم تجارت الکترونیک، کلاس‌های "محصول"، "سفارش"، و "پرداخت" به طور جداگانه طراحی می‌شوند.

برقراری ارتباط بین تیم‌ها:

- دیاگرام کلاس به عنوان یک زبان مشترک بین توسعه‌دهندگان، طراحان، و ذینفعان عمل می‌کند.
- مثلاً در یک پروژه تیمی، همه اعضا می‌توانند با نگاه کردن به دیاگرام کلاس، ساختار سیستم را درک کنند.

مثال کلاس دیاگرام



نمودار کلاس (ادامه...)

هر نمودار کلاس شامل :

Classes ■

Abstract class ■

Interfaces ■

Packages and subsystems ■

Association, dependency, generalization and aggregation relationship ■

Notes ■

کلاس مجرد (abstract class) یک کلاس معمولی است با این تفاوت که نمی توان از آن کلاس نمونه ای ایجاد کرد.

نماد کلاس

■ آیکن کلاس شامل سه بخش است:

■ نام: نام باید منحصر به فرد باشد. به طور قراردادی، نام با حروف بزرگ شروع می شود و فاصله بین کلمات حذف می شود. به طور قراردادی، حرف اول صفت و نام های عملیات با حروف کوچک شروع می شوند.

■ صفات : فرمت صفت :

Visibility attributeName : Type [multiplicity] = ■
DefaultValue {property string}

■ عملیات: فرمت عملیات

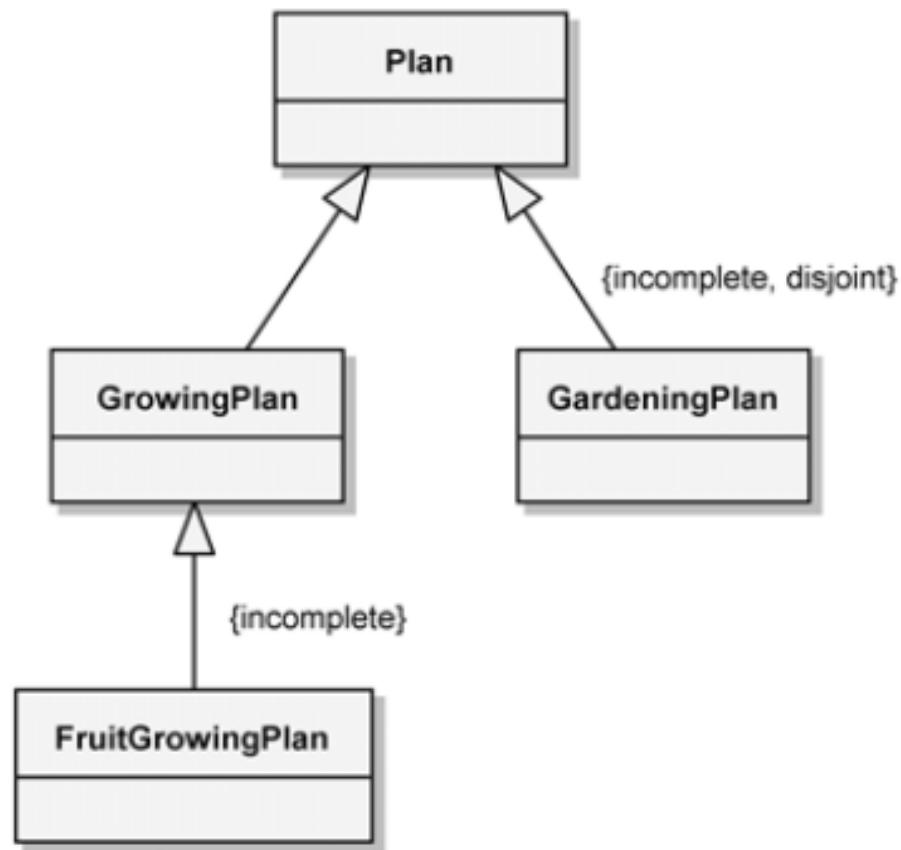
Visibility operationName (parameterName : Type) : ■
ReturnType {property string}

روابط کلاس ها

ASSOCIATION

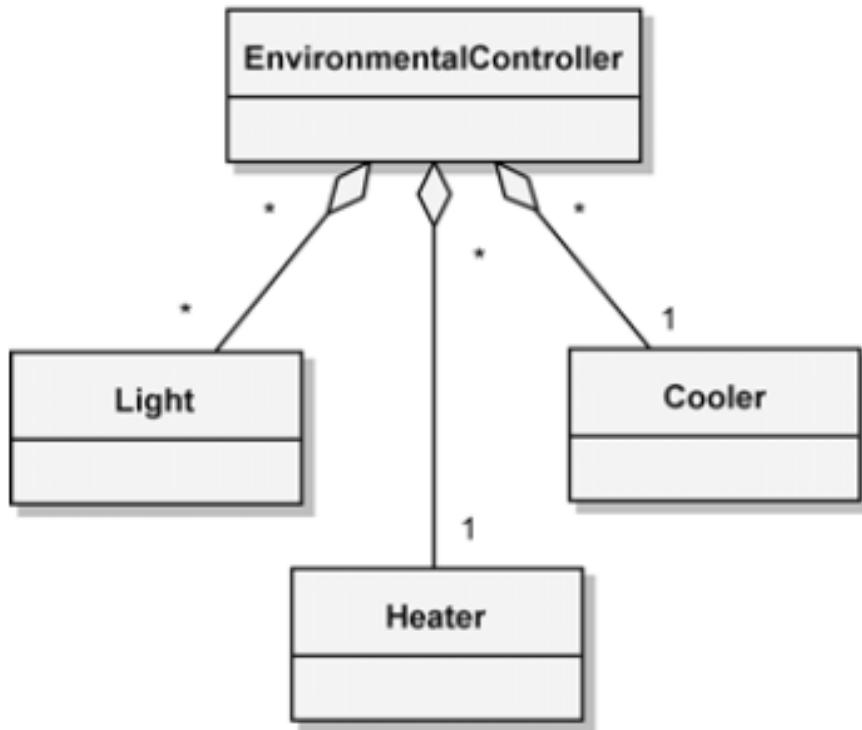


GENERALIZATION

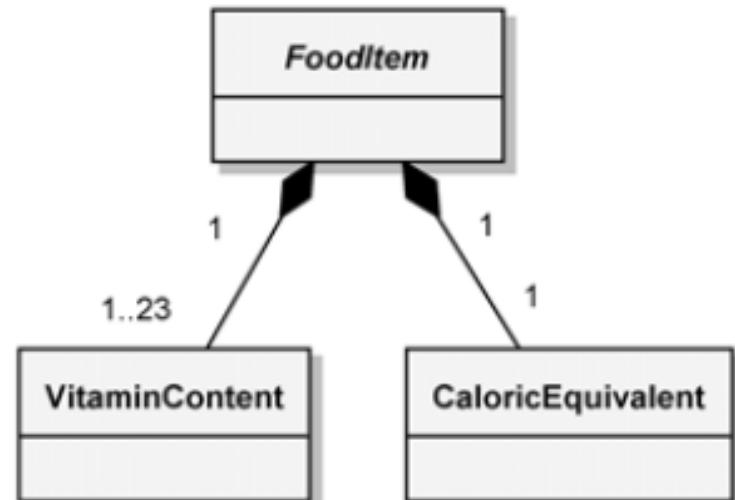


روابط کلاس ها (ادامه...)

AGGREGATION



COMPOSITION



تفاوت Composition و Aggregation .

- (خط توخالی با الماس): رابطه "جزء-کل" ضعیف. جزء میتواند مستقل از کل وجود داشته باشد مثال: رابطه Fooditem و CaloricEquivalent.
- (خط توپر با الماس پر): رابطه "جزء-کل" قوی. جزء بدون کل نمیتواند وجود داشته باشد مثال: رابطه EnvironmentalController با Heater یا Cooler

قابل رویت بودن کلاس ها

- در C++ اعضا می توانند:
 - **Public** : توسط همه قابل دسترس باشند.
 - **Protected** : فقط توسط زیرکلاس ها، **friend** ها یا خود کلاس قابل دسترس هستند.
 - **Private** : فقط توسط خود کلاس یا **friend** هایش قابل دسترس باشند.
 - قابل دسترس بودن :
 - **Public (+)** : قابل مشاهده توسط هر جزئی که کلاس را می بیند.
 - **Protected (#)** : قابل مشاهده توسط اجزای دیگر داخل کلاس و زیرکلاسها
 - **Private (-)** : قابل مشاهده توسط اجزای دیگر در داخل کلاس
 - **Package (~)** : قابل مشاهده توسط اجزای داخل همان package

دیاگرام شی (object diagram)

- نمودار شی یک گراف است - مجموعه ای از گره ها (اشیاء) و یال ها (لینک ها)
- نمودار شی مجموعه ای از نمونه های کلاس ها و روابط آنها را در یک واحد زمانی مدل می کند - یک نمونه از کلاس دیاگرام است
- چند نمودار شی می تواند برای یک کلاس دیاگرام وجود داشته باشد.
- نمودار شی دید استاتیک یک سیستم را نشان می دهد - این نمودار شامل مجموعه ای از اشیا ، حالت های آنها و روابطشان است.

دیاگرام شی (object diagram)

- مشابه دیاگرام کلاس، یک خط افقی نام شی را از صفت‌ها و مقادیر صفت‌های شی مجزا می‌کند.
- اشیاء از طریق **link**‌ها با یکدیگر در ارتباطند.
- یک **link**، نمونه‌ای از یک **association** است.

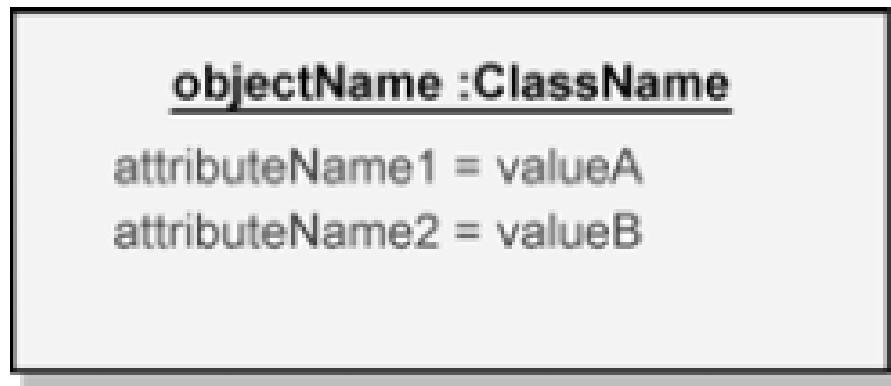
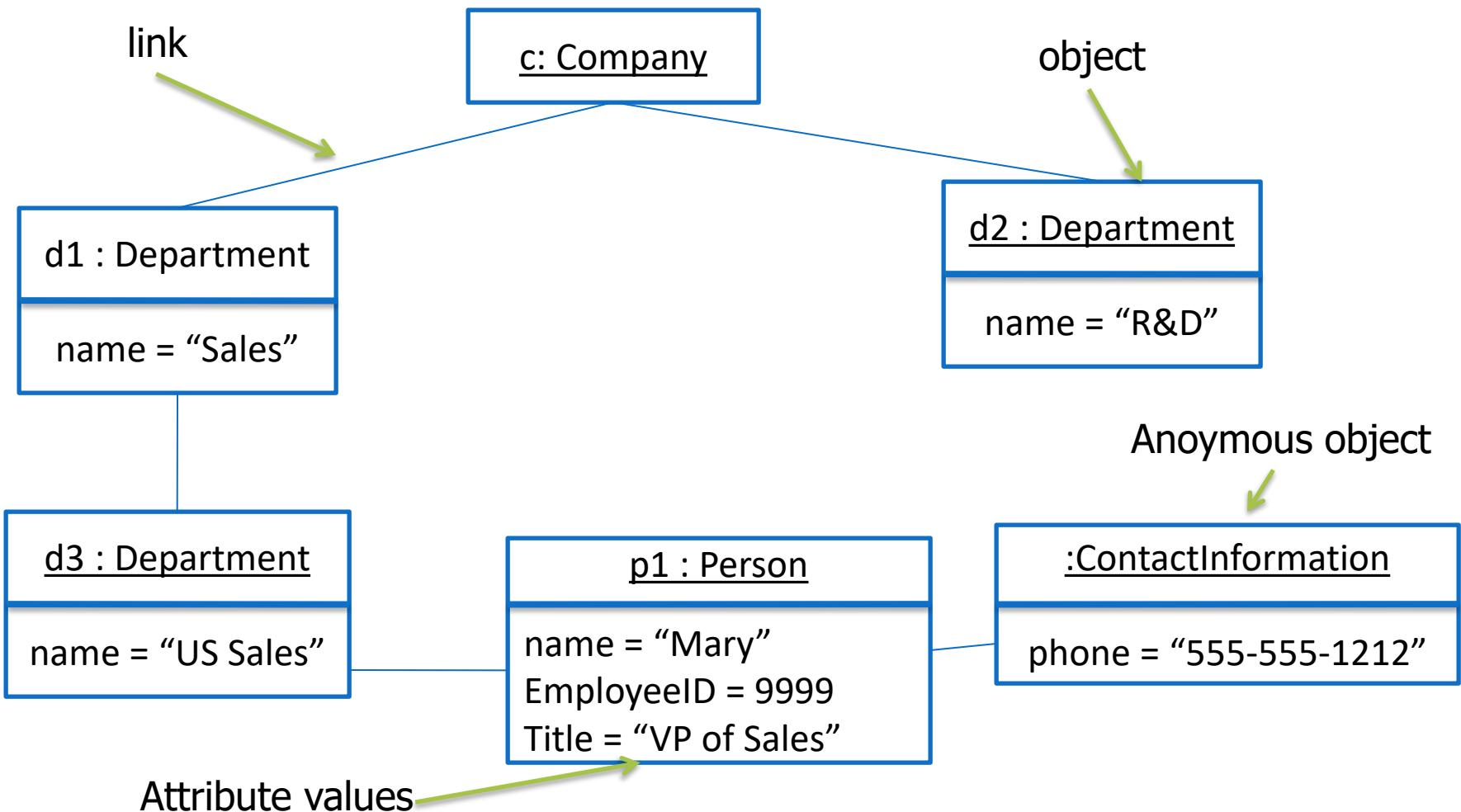


Figure 5–75 A Generic Object Icon

مثال نمودار شی



Package diagram

- دیاگرام بسته (Package Diagram) چیست؟
- دیاگرام بسته یک ابزار UML برای گروه‌بندی و سازماندهی عناصر سیستم (مانند کلاس‌ها، رابط‌ها، یا حتی دیاگرام‌های دیگر) در قالب بسته‌ها (Packages) است. هر بسته مانند یک "پوشه" عمل می‌کند و وابستگی‌های بین بسته‌ها را نشان می‌دهد.
- مثال:
در یک سیستم دانشگاهی، بسته‌هایی مانند آموزش، مالی، و منابع انسانی می‌توانند وجود داشته باشند که هر کدام شامل کلاس‌ها و مازول‌های مرتبط هستند.

- UserManagement: User, Profile, LoginService
- ProductCatalog: Product, Category, SearchService
- OrderProcessing: Order, Payment, Invoice

Package diagram

| ملک | دیاگرام بسته | دیاگرام کلاس |
|------------|---|--|
| هدف اصلی | سازماندهی عناصر سیستم در سطح مازولار | نمایش ساختار داخلی کلاس‌ها و روابط بین آن‌ها |
| سطح جزئیات | سطح بالا (ماکرو) | سطح پایین (میکرو) |
| عناصر اصلی | بسته‌ها، وابستگی‌ها، رابط‌ها | کلاس‌ها، ویژگی‌ها، متدها، روابط (ارتبری، تجمعی، ...) |
| نمادها | (مستطیل با زبانه) + فلش‌های وابستگی  | □ (مستطیل برای کلاس) + خطوط و فلش‌های رابطه |
| وابستگی‌ها | نشان می‌دهد کدام کلاس به کلاس دیگر نشان می‌دهد کدام بسته به کدام بسته وابسته است مثلاً آموزش به مالی | وابسته است مثلاً دانشجو به درس |
| کاربرد | طراحی معماری نرم‌افزار و مدیریت پیچیدگی | طراحی جزئیات داخلی سیستم |

Package diagram

- مکانیزمی در UML برای گروه بندی است.
- اهداف **:package**
- گروه بندی معنایی اجزای به هم وابسته
- فراهم کننده فضای نام پنهان شده به طوری که نام‌ها باید منحصر به فرد باشند - برای دسترسی به جزئی در داخل یک فضای نام باید نام عنصر و نام فضای نام را مشخص کنید.
- مالک هر عنصر مدل یک **package** است
- عناصر **package** میدان دید دارند :
- (+) : عناصر برای پکیج‌های دیگر قابل رویت هستند.
- (-) : عناصر به طور کامل مخفی هستند.

Package diagram: Stereotypes

در نمودار بسته‌ها (Package Diagram) در UML، از استریوتایپ‌ها (Stereotypes) برای مشخص کردن نوع و نقش بسته‌ها (Packages) یا روابط بین آن‌ها استفاده می‌شود. استریوتایپ‌ها نوعی برچسب معنایی هستند که با علامت <><> نمایش داده می‌شوند.

◇ 1. <><>subsystem>>

توضیح: نشان‌دهنده‌ی یک زیربخش (Subsystem) از سیستم است. یک بسته که شامل کلاس‌ها، رابط‌ها یا بسته‌های دیگر است که رفتار یا کارکردی خاص را ارائه می‌دهد.

کاربرد:

در سیستم بانکداری، ممکن است یک بسته <><>subsystem>> Authentication داشته باشیم.

Package diagram: Stereotypes

◇ 2. <<framework>>

توضیح: نشان‌دهنده‌ی یک چارچوب نرم‌افزاری است که برای توسعه‌ی بخش‌های خاصی از سیستم استفاده می‌شود.

مثال کاربرد:

بسته‌ای به نام **UIFramework** برای کنترل ظاهر و رابط کاربری.

◇ 3. <<library>>

توضیح: بیانگر یک بسته‌ی کتابخانه‌ای است که شامل کدهای قابل استفاده مجدد می‌باشد.

مثال کاربرد:

که توابع ریاضی پایه را فراهم می‌کند.

Package diagram: Stereotypes

◊ 4. <<model>>

توضیح: این بسته شامل اجزای مدل سیستم است، مانند کلاس‌های دامنه (**domain classes**) یا نهادها.

مثال کاربرد:

Customer, **Order** و **OrderDomain** برای تعریف کلاس‌هایی مانند <<model>>

- بسته‌ای که مدل‌های منطقی یا مفهومی سیستم را در خود جای می‌دهد.

◊ 5. <<utility>>

توضیح: بسته‌ای شامل توابع کمکی یا ابزارهای کاربردی است که در بخش‌های مختلف سیستم استفاده می‌شوند.

مثال کاربرد <<utility>> **StringHelpers**

- ابزارهایی کمکی مانند توابع پردازش رشته، تاریخ و زمان، و ... که در کل سیستم قابل استفاده هستند

◊ 6. <<service>>

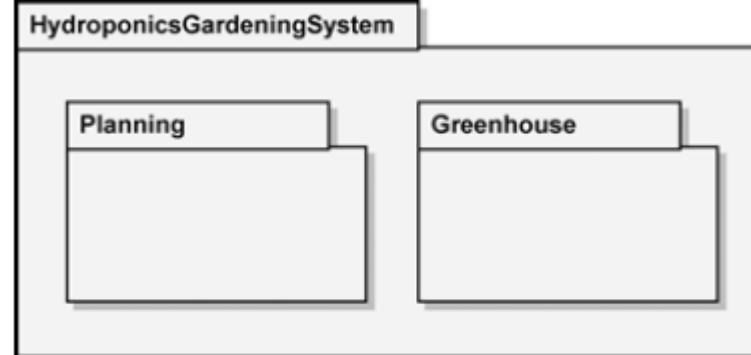
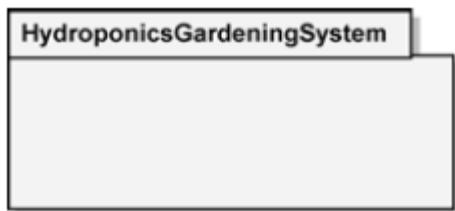
▪ توضیح: بسته‌ای که سرویس‌های مشخصی را ارائه می‌دهد، مانند API یا سرویس‌های منطق تجاری. مثال

کاربرد <<service>> **PaymentService**

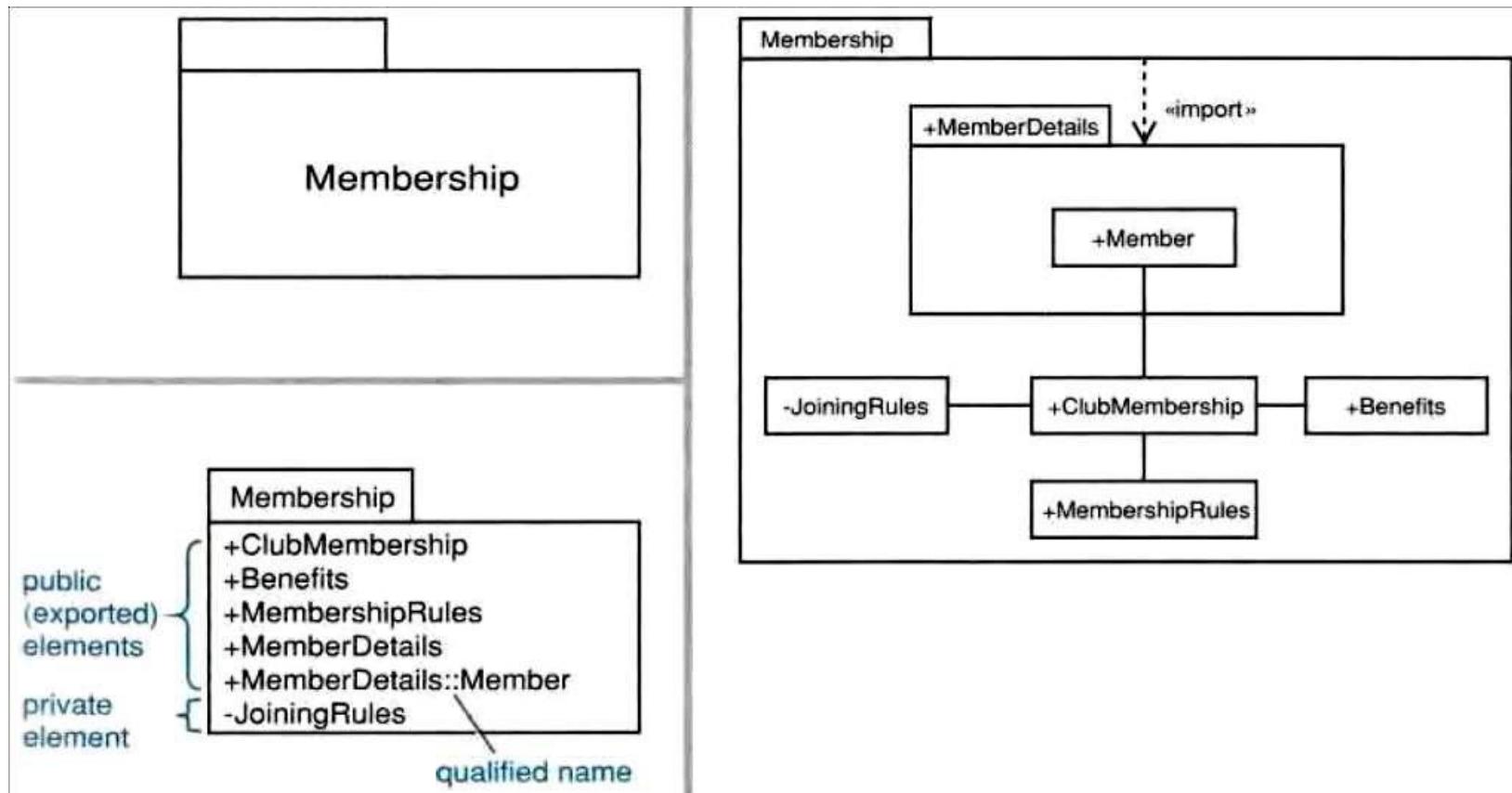
نمايانگر سرویس‌هایی است که عملیات خاصی را در سیستم انجام می‌دهند، مانند پرداخت، ورود، ارسال ایمیل و غیره.

Package diagram (cont.)

- نماد **Package** شامل یک چهارگوش به همراه نوار باریکی در بالای سمت چپ آن است.
- نماد **package** برای نمایش ساختار و محدوده اجزای مدل های مختلف مانند کلاس ها یا برای سازماندهی **use case**ها به کار میروند.
- آنها برای وضوح در یک سیستم بسیار بزرگ یا تقسیم کار استفاده میشوند.

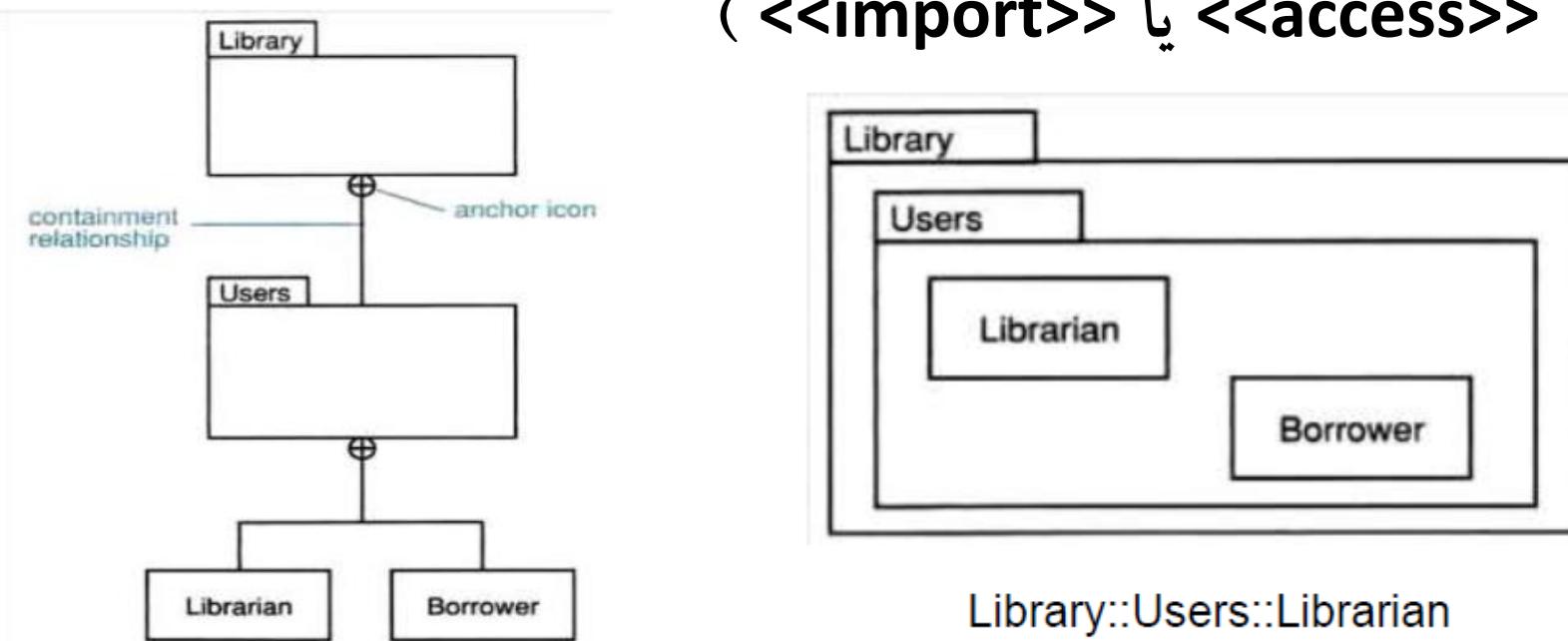


Package diagram (example)



تو در تو Package های

- پکیج داخلی می تواند تمام اعضای عمومی پکیج خارجی را بینند.
- پکیج خارجی نمی تواند هیچ کدام از اعضای پکیج داخلی را بیند مگر اینکه یک وابستگی صریح با آنها داشته باشد (معمولاً <<import>> یا <<access>>



وابستگی package ها

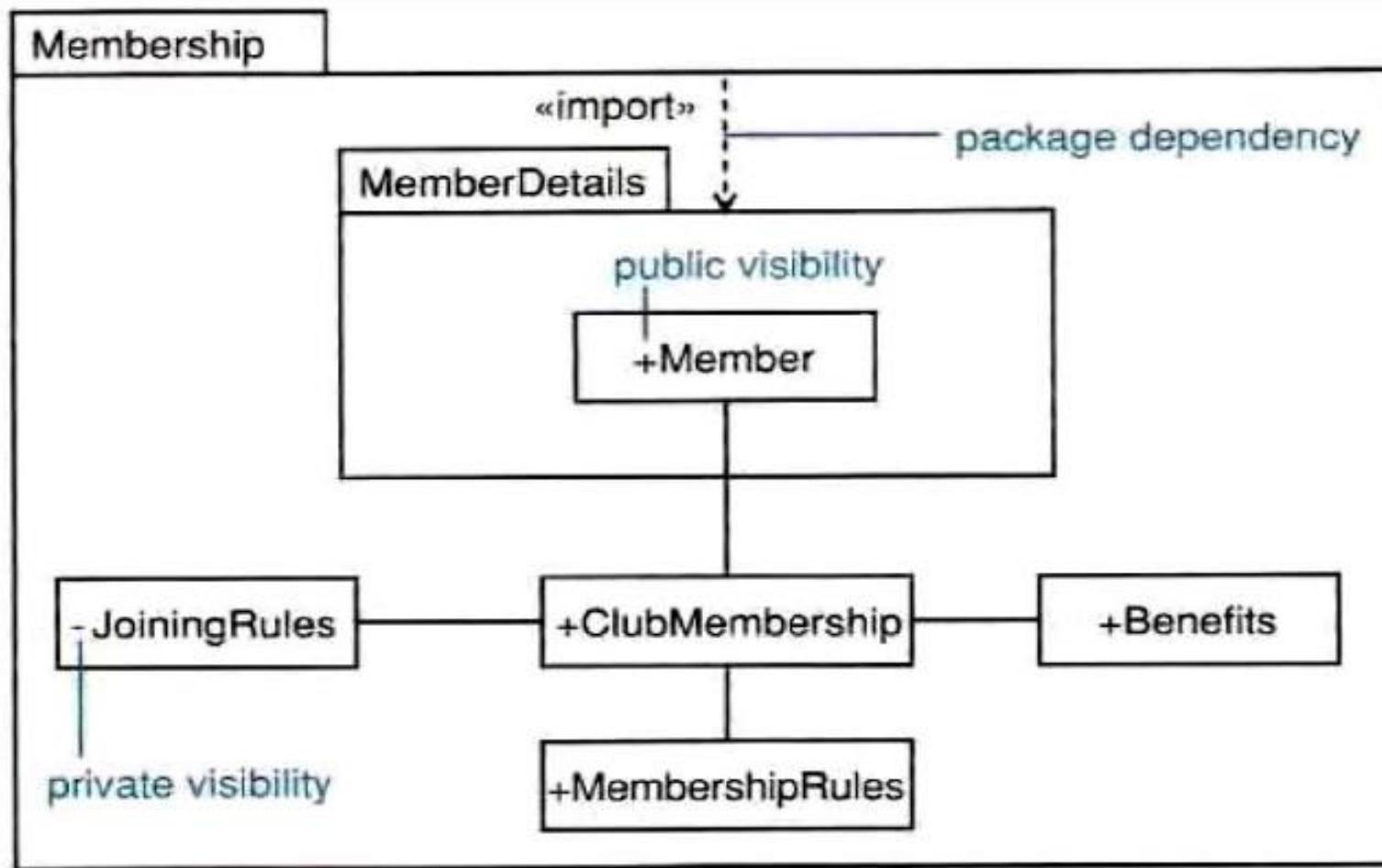
رابطه وابستگی بین package ها نشان می دهد که client package به supplier package وابسته است.

■ یک عنصر در (default) <<use>> از یک عنصر عمومی در supplier package استفاده می کند.

■ عناصر عمومی <<import>> اضافه می شوند. عناصر در client namespace کلاینت می توانند به عناصر عمومی supplier دسترسی پیدا کنند.

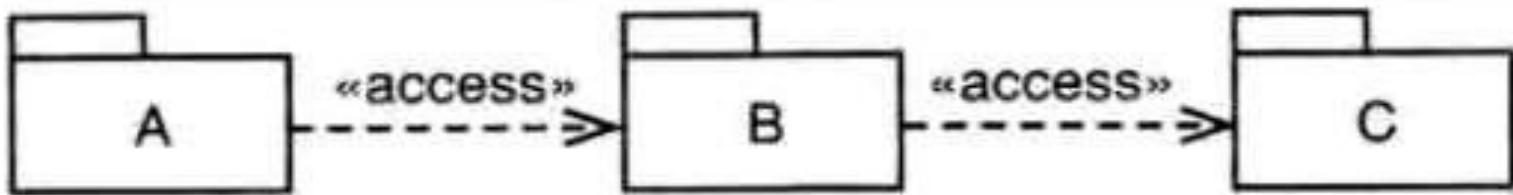
■ عناصر عمومی <<access>> اضافه می شوند. عناصر در client namespace کلاینت میتوانند به همه عناصر عمومی در supplier دسترسی داشته باشند.

واستگی package ها



رابطه تعدی

- تعدی : اگر A یک رابطه با B دارد و B هم با C در ارتباط است . بنابراین A با C در ارتباط است . تعدی اما <<access>> تعدی نیست .



Package Generalization

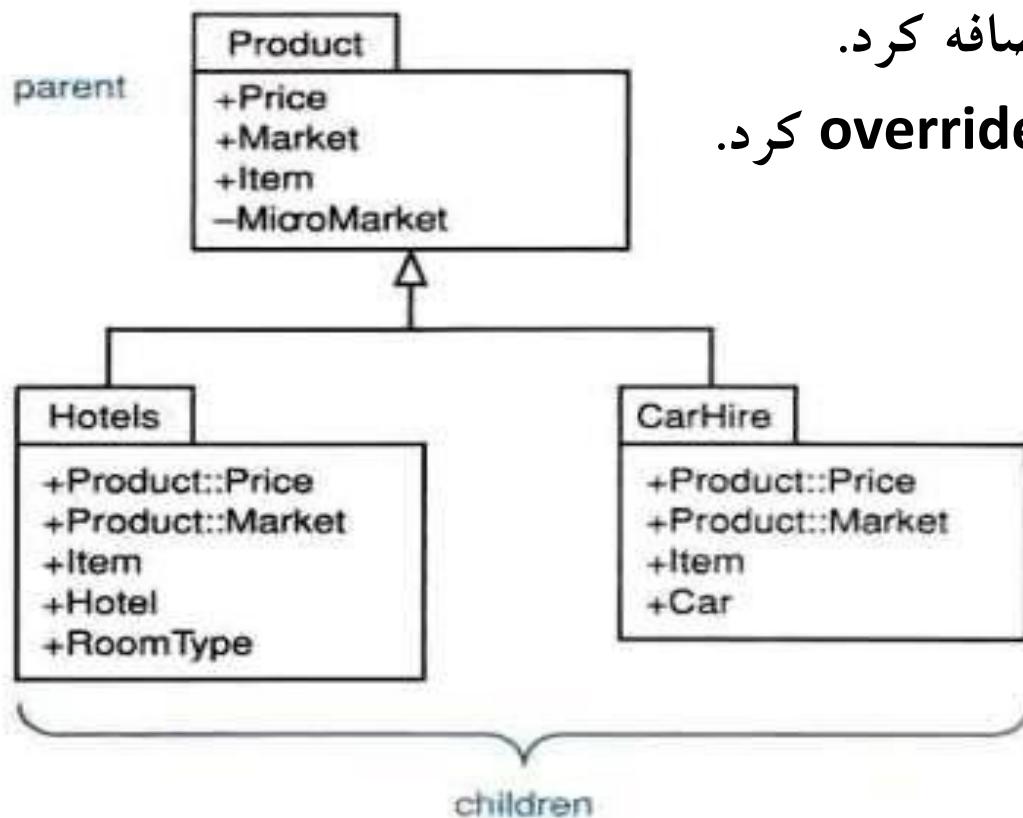
شبيه class generalization است.

هاي Package فرزند :

عناصر را از پکيج پدر به ارث می برند.

می توان عناصر جدید اضافه کرد.

می توان عناصر پدر را override کرد.



راه های انتخاب package ها

- راه های متفاوتی برای سازماندهی یک سیستم با package ها وجود دارد : توسط لایه های معماری، توسط زیرسیستم، توسط کاربران (برای usecase ها) و ...
- **highly coupled** و **loosely coupled** Package های خوب cohesive هستند.
- به این معنی که : باید تعاملات بیشتری بین اجزای یک package وجود داشته باشد و تعاملات بین package ها کمتر باشند.

راههای متفاوت سازماندهی سیستم با پکیج‌ها

سیستم‌های نرمافزاری را می‌توان با استفاده از پکیج‌ها به روش‌های مختلفی سازماندهی کرد. انتخاب روش مناسب به نیازهای سیستم و معماری آن بستگی دارد. برخی از رایج‌ترین روش‌ها عبارتند از:

سازماندهی بر اساس لایه‌های معماری (Layered Architecture):

- سیستم به لایه‌های مجزا تقسیم می‌شود که هر لایه مسئولیت خاصی دارد.
- مزیت: جداسازی وظایف و کاهش وابستگی بین لایه‌ها.
- سازماندهی بر اساس زیرسیستم‌ها (Subsystems):
 - سیستم به مأژول‌های مستقل تقسیم می‌شود که هر کدام یک قابلیت خاص را ارائه می‌دهند.
 - مزیت: توسعه و تست آسان هر زیرسیستم به صورت جداگانه.

راههای متفاوت سازماندهی سیستم با پکیج‌ها

سازماندهی بر اساس موارد استفاده (Use Cases):

- پکیج‌ها بر اساس نیازهای کاربران یا سناریوهای استفاده (Use Cases) گروه‌بندی می‌شوند.
- مزیت: تمرکز بر نیازهای کاربر نهایی.

سازماندهی بر اساس فناوری (Technology):

- پکیج‌ها بر اساس تکنولوژی مورد استفاده مانند API‌ها، دیتابیس، یا کتابخانه‌ها گروه‌بندی می‌شوند.

راههای متفاوت سازماندهی سیستم با پکیج‌ها

برای طراحی پکیج‌های کارآمد، دو اصل اساسی باید رعایت شود:
۱. وابستگی کم: **Loose Coupling**

- معنی: پکیج‌ها تا حد امکان کمتر به یکدیگر وابسته باشند.
- مزایا:
 - تغییرات در یک پکیج، تأثیر کمی بر پکیج‌های دیگر دارد.
 - قابلیت نگهداری و توسعه سیستم افزایش می‌یابد.
- مثال:
 - اگر پکیج Payment مستقل از پکیج Inventory باشد، تغییر در موجودی کالا، پرداخت‌ها را مختل نمی‌کند.

۲. انسجام بالا **High Cohesion**

راههای متفاوت سازماندهی سیستم با پکیج‌ها

برای طراحی پکیج‌های کارآمد، دو اصل اساسی باید رعایت شود:
۲. انسجام بالا **High Cohesion**.

- معنی: عناصر داخل یک پکیج باید به شدت مرتبط و حول یک هدف مشترک متمرکز باشند.
- مزايا:
 - خوانایی و درک کد بهبود می‌یابد.
 - احتمال خطا کاهش می‌یابد.
- مثال:
 - تمام کlassen‌های مرتبط با مدیریت کاربران مانند User، Role، Permission در یک پکیج UserManagement قرار می‌گیرند.

پیدا کردن package ها

زیر سیستم ها را بررسی کنید.

بررسی کلاس ها برای :

■ کلاسترهای منسجمی از کلاس های به هم وابسته

■ کلاستر (Cluster) به مجموعه ای از کلاس هایی گفته می شود که تعامل زیادی با هم دارند. این کلاس ها باید در یک package قرار گیرند.

■ در بخش سفارش : کلاس های Order, OrderItem, Invoice, ShippingDetail چون به هم مرتبط هستند، باید در یک package با عنوان OrderModule قرار گیرند.

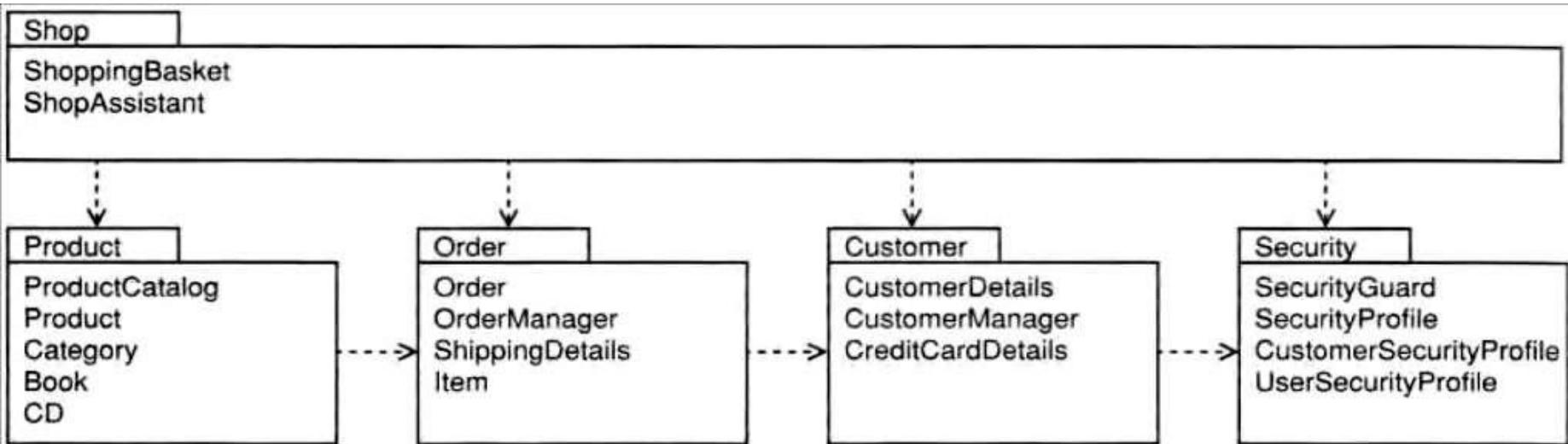
■ ارث بری : کلاس هایی که در یک ساختار ارث بری (سلسله مراتبی) قرار دارند معمولاً با هم وابستگی مفهومی دارند و می توانند در یک package قرار گیرند.

■ کلاس های به هم وابسته به ترتیب از طریق وراثت (inheritance)، ترکیب (dependency)، تجمع (Aggregation) و وابستگی (composition)

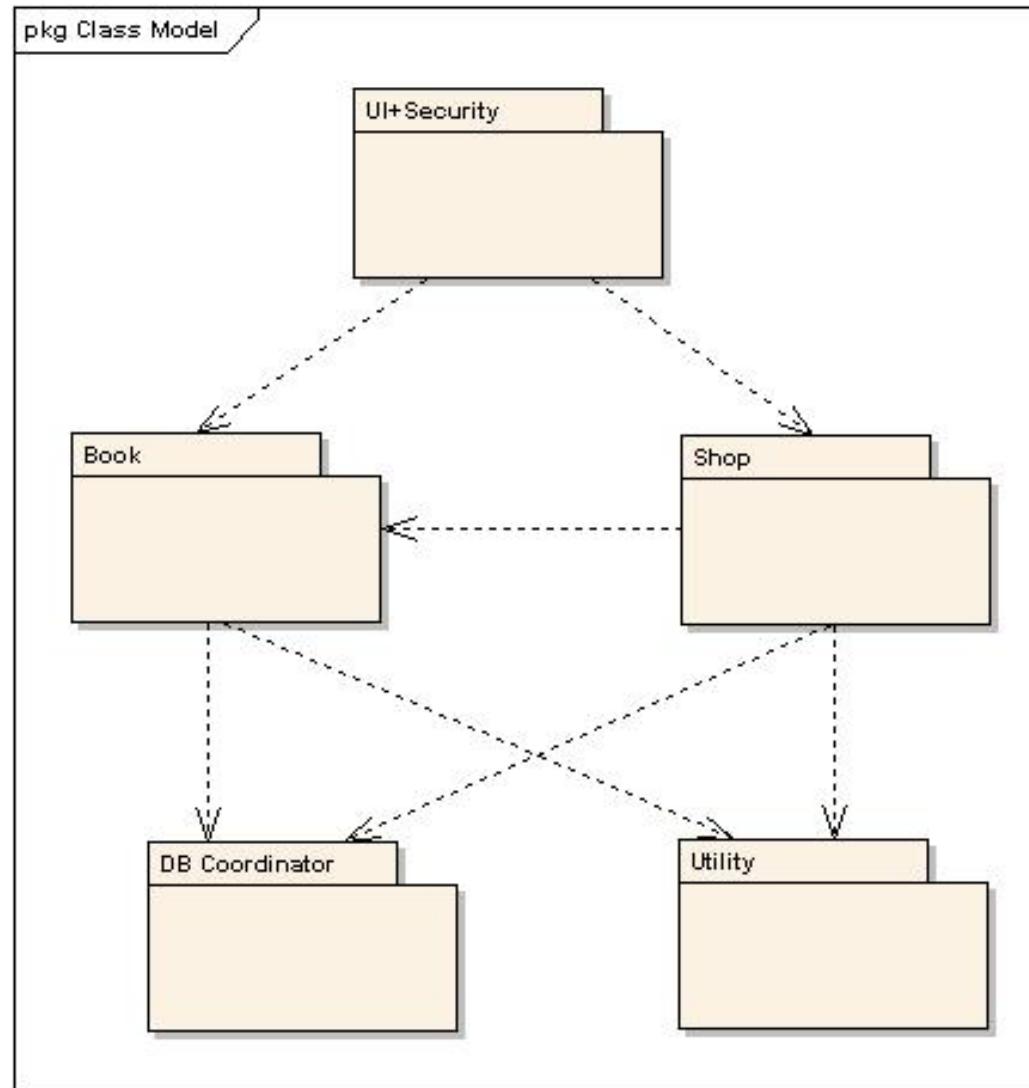
بررسی use case ها :

■ کلاسترهایی از use case ها که یک Actor یا پروسه را ساپورت می کنند باید در یک package قرار گیرند.

Package diagram (cont.)



Package diagram (cont.)

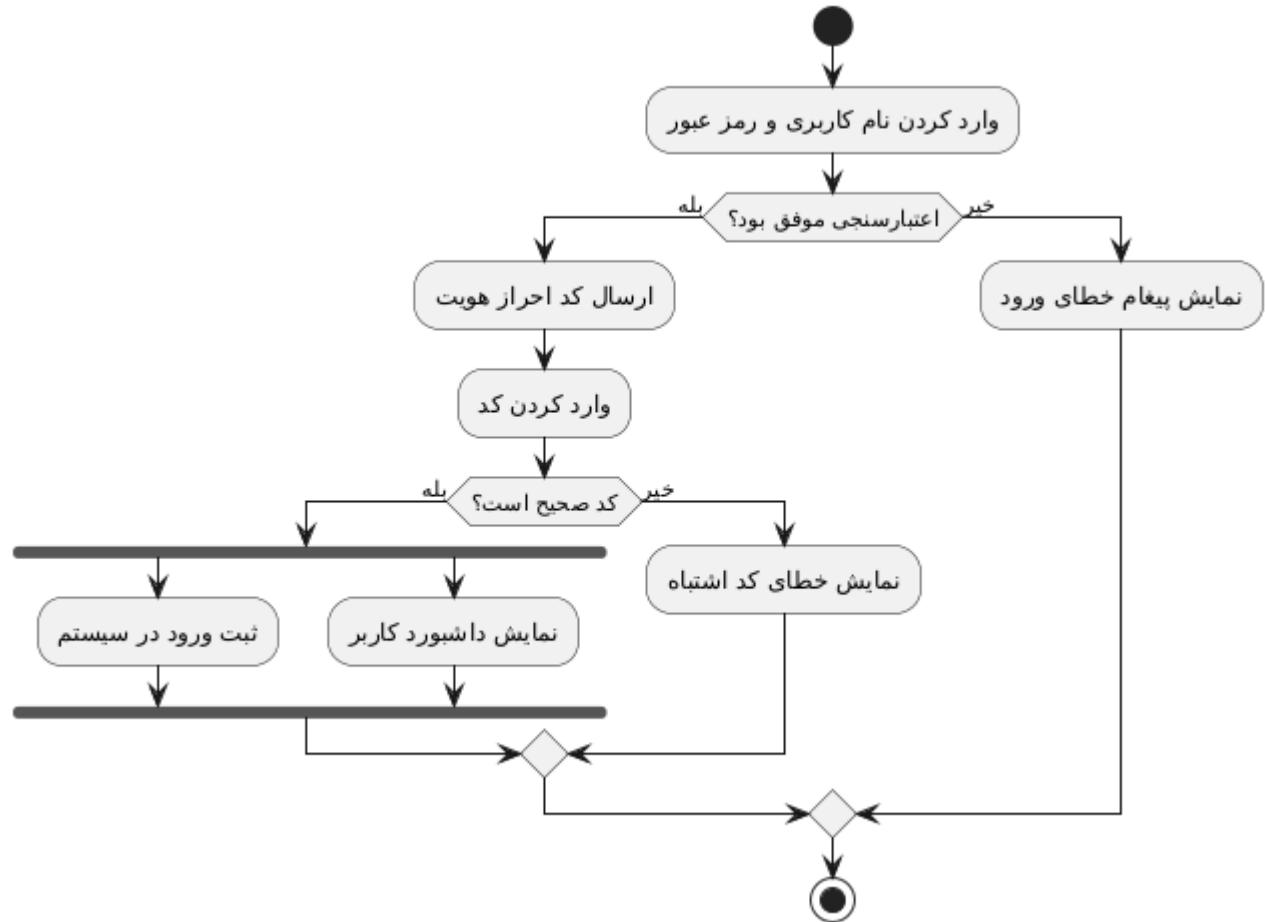


Activity diagrams

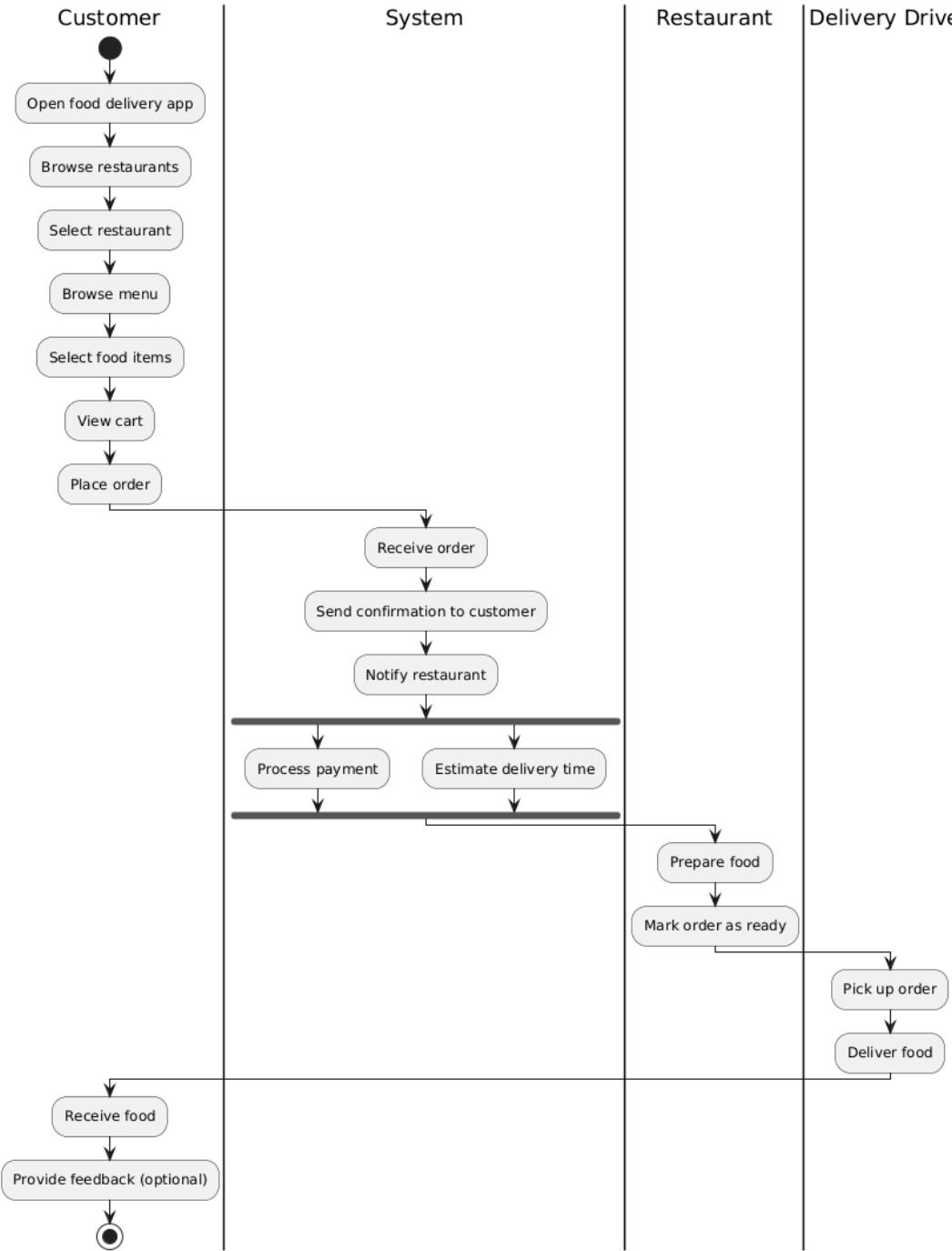
نمودار فعالیت یک نوع نمودار UML زبان مدل‌سازی یکپارچه است که جریان فعالیتها یا فرآیندهای یک سیستم را نشان می‌دهد. این نمودار شبیه به فلوچارت است، اما بیشتر بر گردش کار، فرآیندهای کسبوکار و رفتار سیستم تمرکز دارد تا صرفاً کنترل جریان.

- اجزای اصلی نمودار فعالیت
- .1 گره شروع (حالت اولیه) - آغاز فرآیند را مشخص می‌کند.
- .2 اقدامات/فعالیتها - نشان‌دهنده مراحل یا وظایف در فرآیند.
- .3 جریان کنترل (پیکان‌ها) - ترتیب فعالیتها را نمایش می‌دهد.
- .4 گره تصمیم (لوزی) - یک شرط انشعاب را نشان می‌دهد (مثلاً بله/خیر).
- .5 گره ادغام - چندین جریان را به یک مسیر برمی‌گرداند.
- .6 گره فورک و جوین - فعالیتهای موازی را مدل می‌کند.
- .7 گره پایان (حالت نهایی) - پایان فرآیند را مشخص می‌کند.

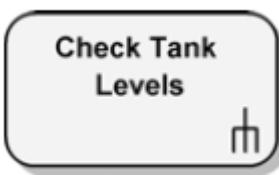
Activity diagrams



UML Sequence Diagrams



Activity diagrams (ادامه..)



: Action

- واحد اصلی رفتار در دیاگرام فعالیت هستند.
- فعالیت ها می توانند شامل چندین action باشند.

Starting and Stopping

- از آنجایی که دیاگرام فعالیت نشان دهنده جریان فرآیند است، جریان باید جایی شروع شده و جایی پایان یابد.

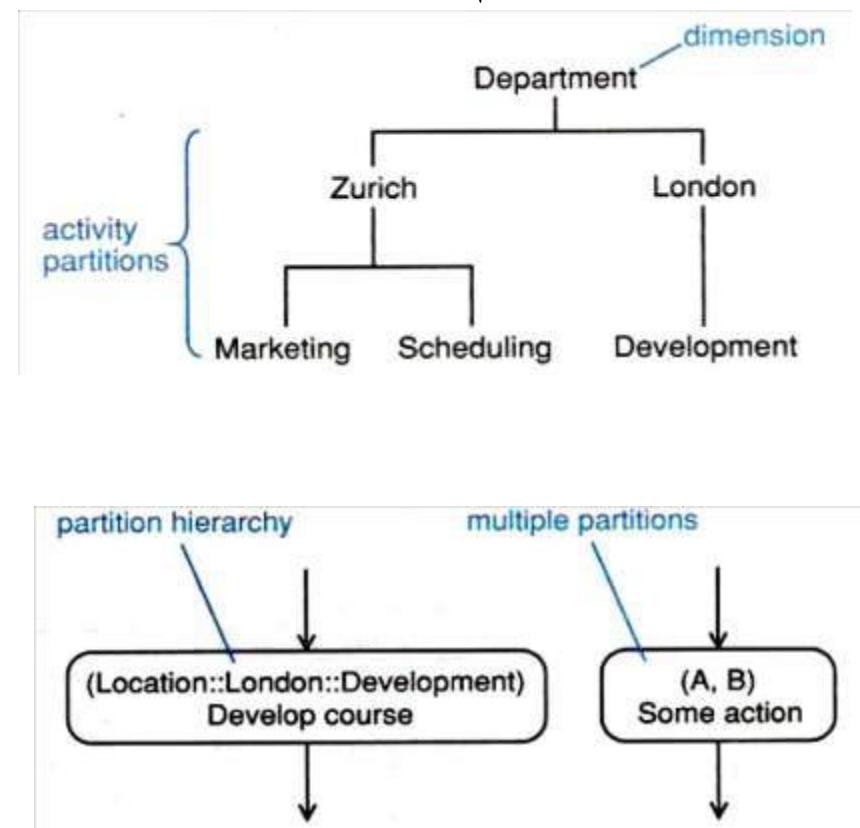
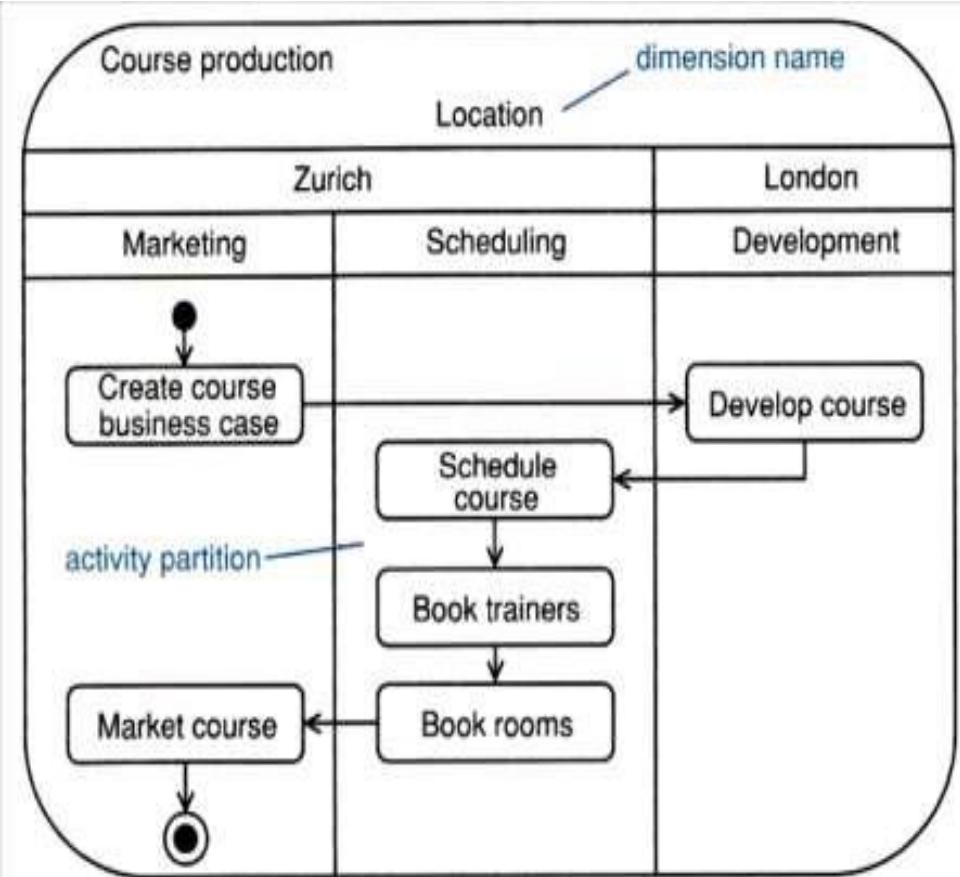
- نقطه شروع (starting point- Initial point) در جریان فعالیت با یک نقطه توپر نشان داده میشود.

- نقطه پایان (stopping point- final point)

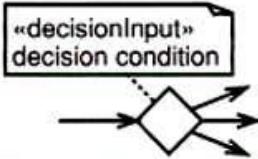
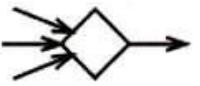
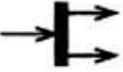


پارتیشن ها در activity diagram

- عناصر در دیاگرام فعالیت می توانند با استفاده از پارتیشن ها تقسیم بندی شوند.



گره های کنترلی

| Syntax | Name | Semantics | |
|---|---------------------|--|-------------|
| ● → | Initial node | Indicates where the flow starts when an activity is invoked | |
| → ● | Activity final node | Terminates an activity | |
| → ⊗ | Flow final node | Terminates a specific flow within an activity – the other flows are unaffected | Final nodes |
|  | Decision node | The output edge whose guard condition is true is traversed May optionally have a «decisionInput» | |
|  | Merge node | Copies input tokens to its single output edge | |
|  | Fork node | Splits the flow into multiple concurrent flows | |
| {join spec}  | Join node | Synchronizes multiple concurrent flows May optionally have a join specification to modify its semantics | |

گره های merge و decision (ادامه...)

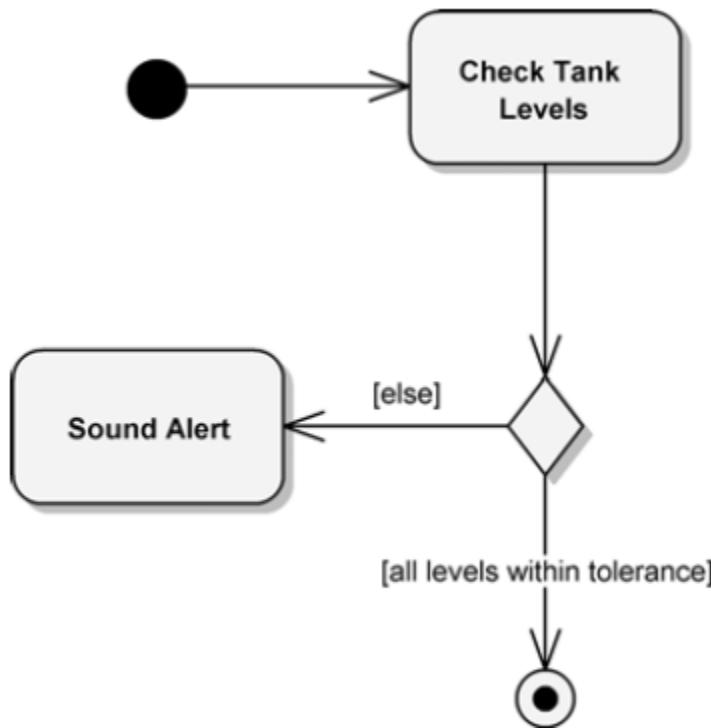


Figure 5–29 A Decision Node

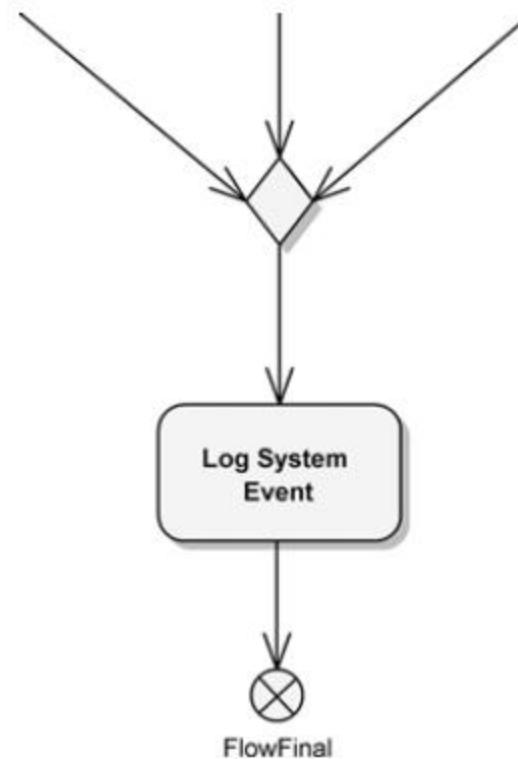
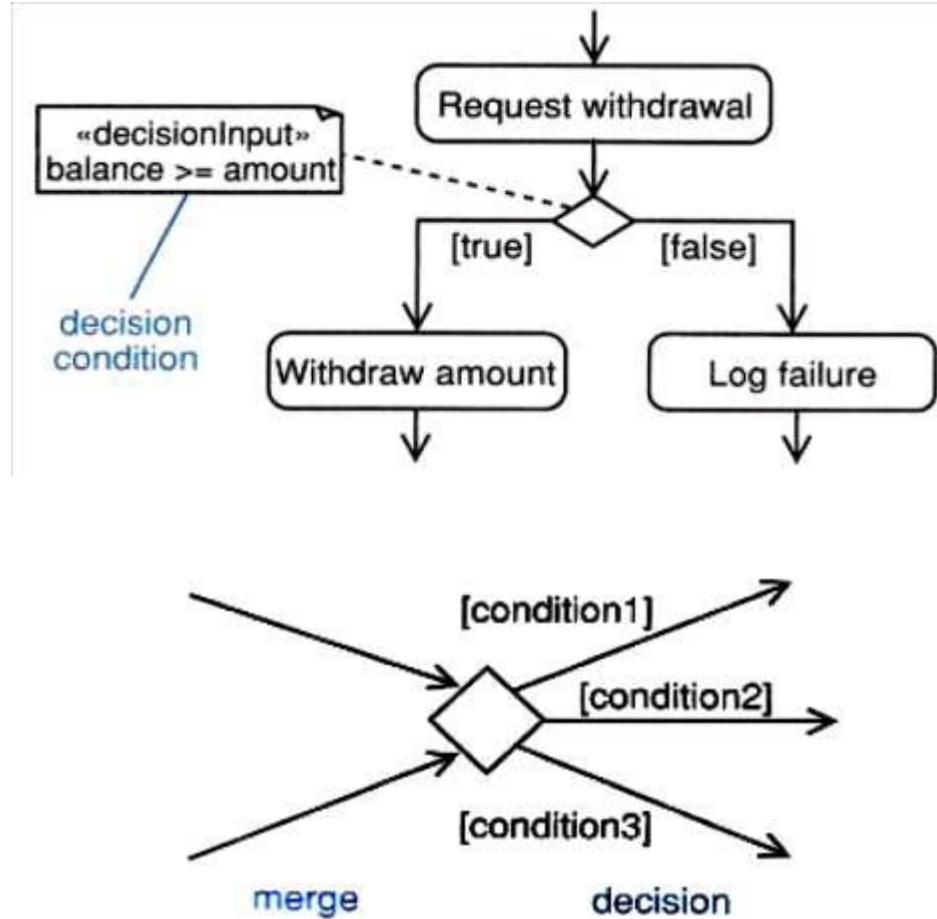
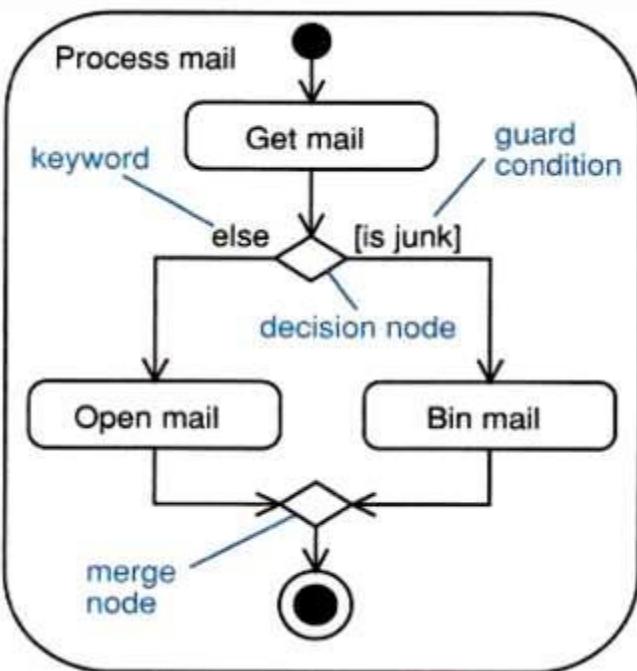


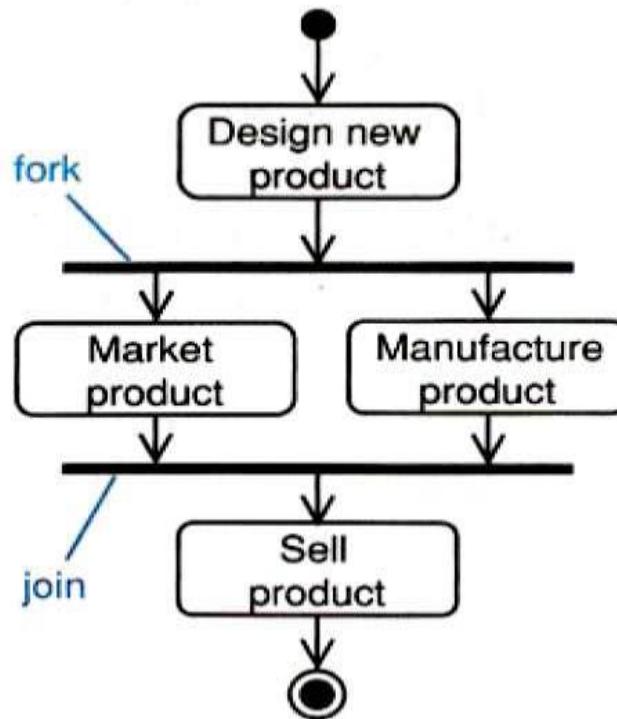
Figure 5–30 A Merge Node with a Flow Final Node

گره های merge و decision (ادامه...)



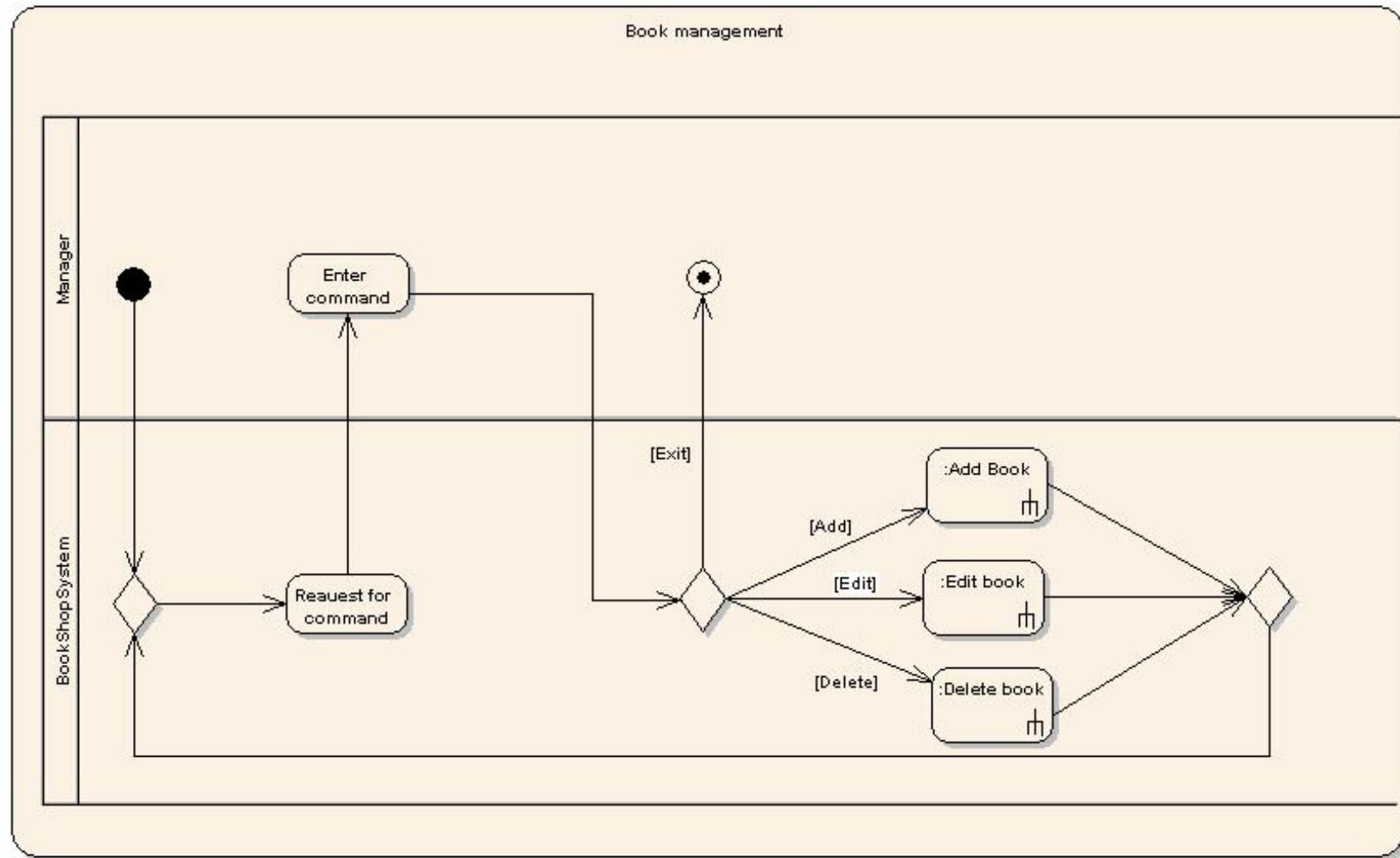
گره های کنترلی : Join و Fork

Product process

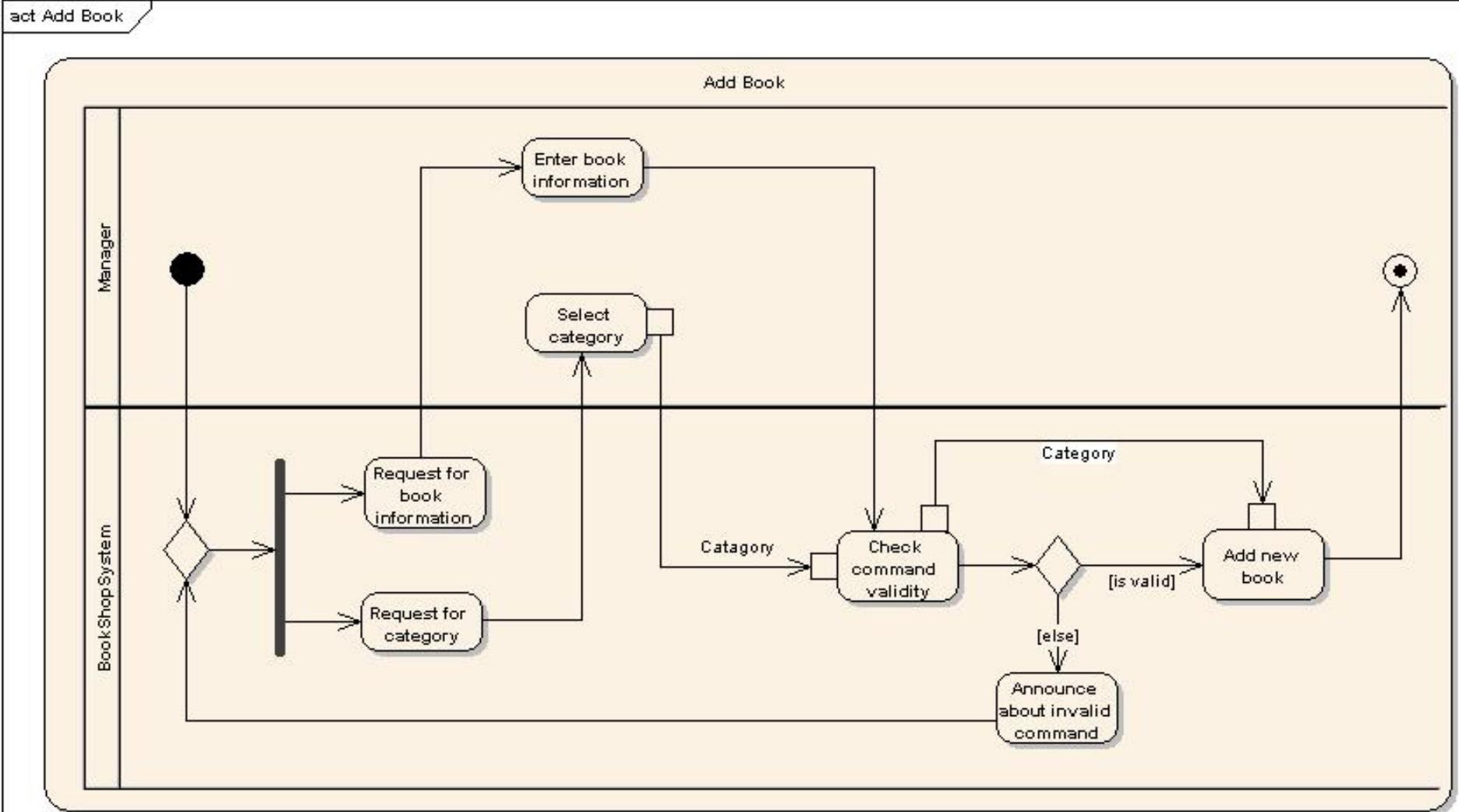


مثالی از activity diagram

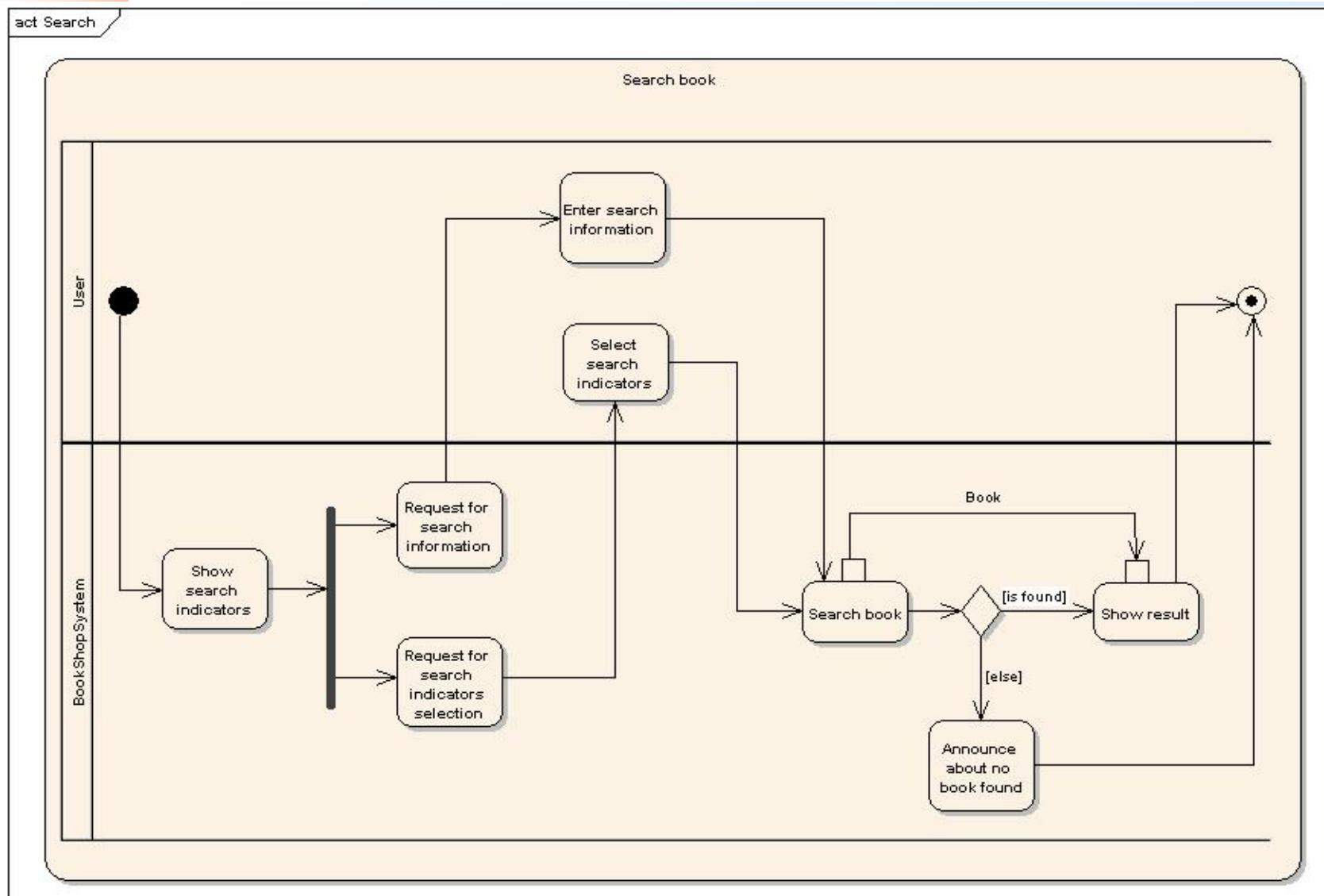
act Book Management



مثالی از activity diagram



مثالی از activity diagram



Sequence diagrams

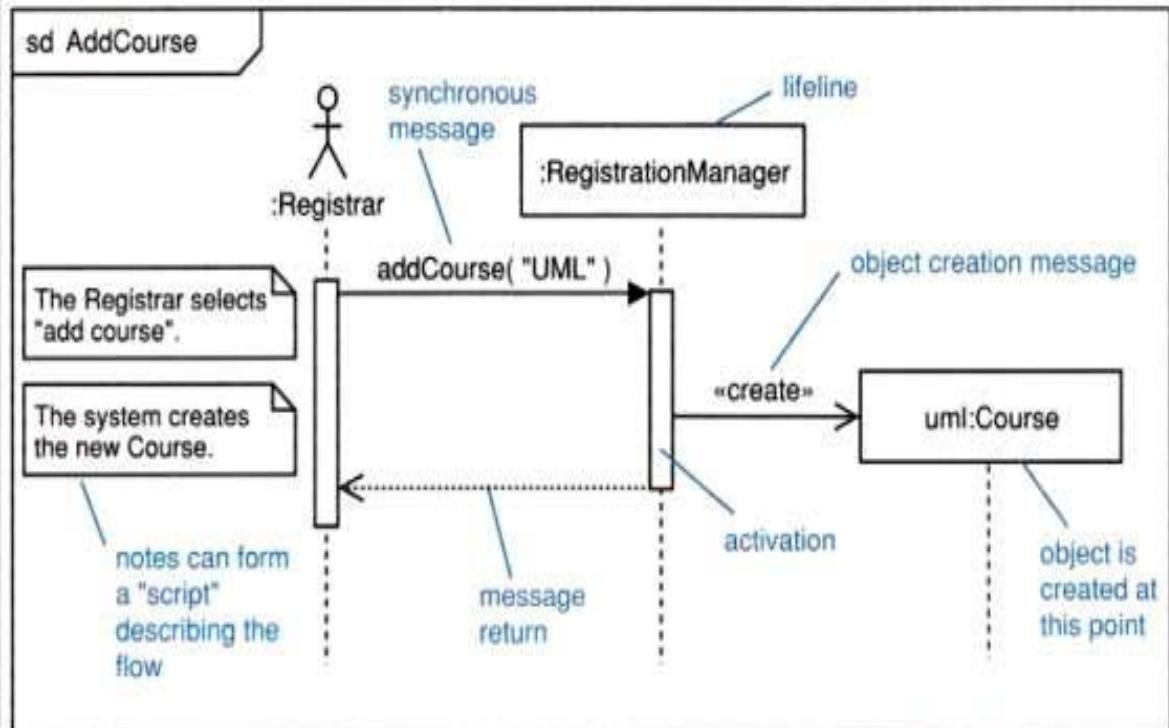
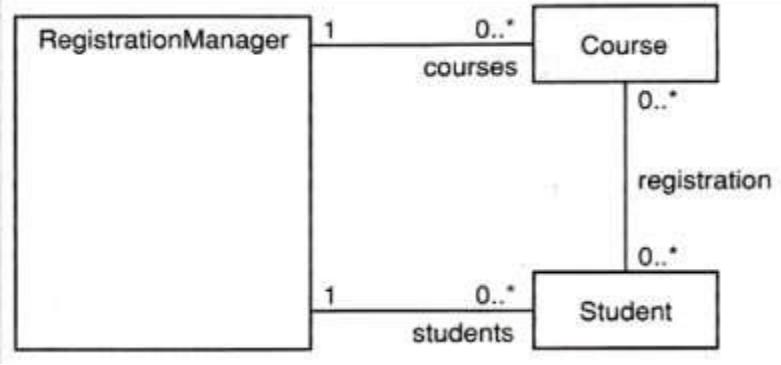
- یک دیاگرام توالی به منظور دنبال کردن یک سناریو به کار می رود.
- مزایا:
 - خواندن گذر پیام ها در یک ترتیب نسبی را ساده می کند.
 - معمولاً نسبت به **object diagram** ها در به دست آوردن مفاهیم سناریوها در مراحل اولیه توسعه نرم افزار مناسب ترند.

Sequence diagrams (cont.)

- در دیاگرام های توالی، موجودیت ها (اشیا) در قسمت بالای دیاگرام به صورت افقی قرار می گیرند.
- خط چین عمودی، خط عمر (**lifeline**) نامیده می شود که زیر هر شی کشیده می شود. این خطوط، وجود شی را نشان می دهند.
- پیام ها (**Messages**) که مشخص کننده رخدادها و احضارها هستند به صورت افقی نشان داده میشوند.
 - پیام ها از فرستنده به گیرنده فرستاده می شوند.
 - اولین پیام در بالای دیاگرام و آخرین پیام در انتهای دیاگرام قرار می گیرد.

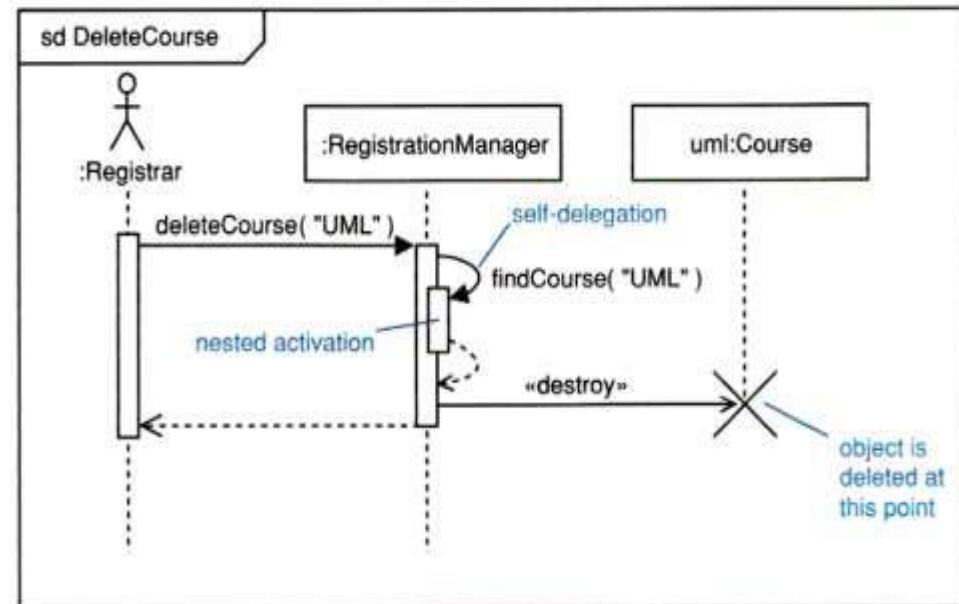
Sequence diagrams (cont.)

| Use case: AddCourse | |
|---------------------|--|
| ID: | 8 |
| Brief description: | Add details of a new course to the system. |
| Primary actors: | Registrar |
| Secondary actors: | None. |
| Preconditions: | 1. The Registrar has logged on to the system. |
| Main flow: | 1. The Registrar selects "add course". 2. The Registrar enters the name of the new course. 3. The system creates the new course. |
| Postconditions: | 1. A new course has been added to the system. |
| Alternative flows: | CourseAlreadyExists |



مثال ۲ - Sequence diagrams (cont.)

| |
|---|
| Use case: DeleteCourse |
| ID: 8 |
| Brief description: Remove a course from the system. |
| Primary actors: Registrar |
| Secondary actors: None. |
| Preconditions: 1. The Registrar has logged on to the system. |
| Main flow: 1. The Registrar selects "delete course". 2. The Registrar enters the name of the course. 3. The system deletes the course. |
| Postconditions: 1. A course has been removed from the system. |
| Alternative flows: CourseDoesNotExist |



مثال ۳

مورد کاربرد: جستجوی کتاب

شماره: ۳

توصیف اجمالی: کاربر اطلاعات جستجو را وارد می‌کند و سامانه کتاب‌هایی را که با این اطلاعات همخوانی دارند، به کاربر نشان می‌دهد.

عامل اصلی: کاربر

عامل فرعی: ندارد

شرایط اولیه: ندارد

روند اصلی:

۴. این مورد کاربرد وقتی آغاز می‌شود که کاربر از سامانه، درخواست جستجوی کتاب کند.

۵. سامانه شاخصه‌های جستجوی کتاب را به کاربر نشان می‌دهد و اطلاعات جستجو لازم را جهت جستجو را از کاربر می‌خواهد.

۶. کاربر اطلاعات کتاب مورد نظر خود را وارد می‌کند.

۷. سامانه اطلاعات وارد شده را با اطلاعات کتاب‌ها مقایسه می‌کند و موارد همخوانی را می‌یابد.

۸. اگر کتابی همخوان با اطلاعات وارد شده پیدا شد

۸.۱. سامانه کتاب‌های پیدا شده را به کاربر نشان می‌دهد.

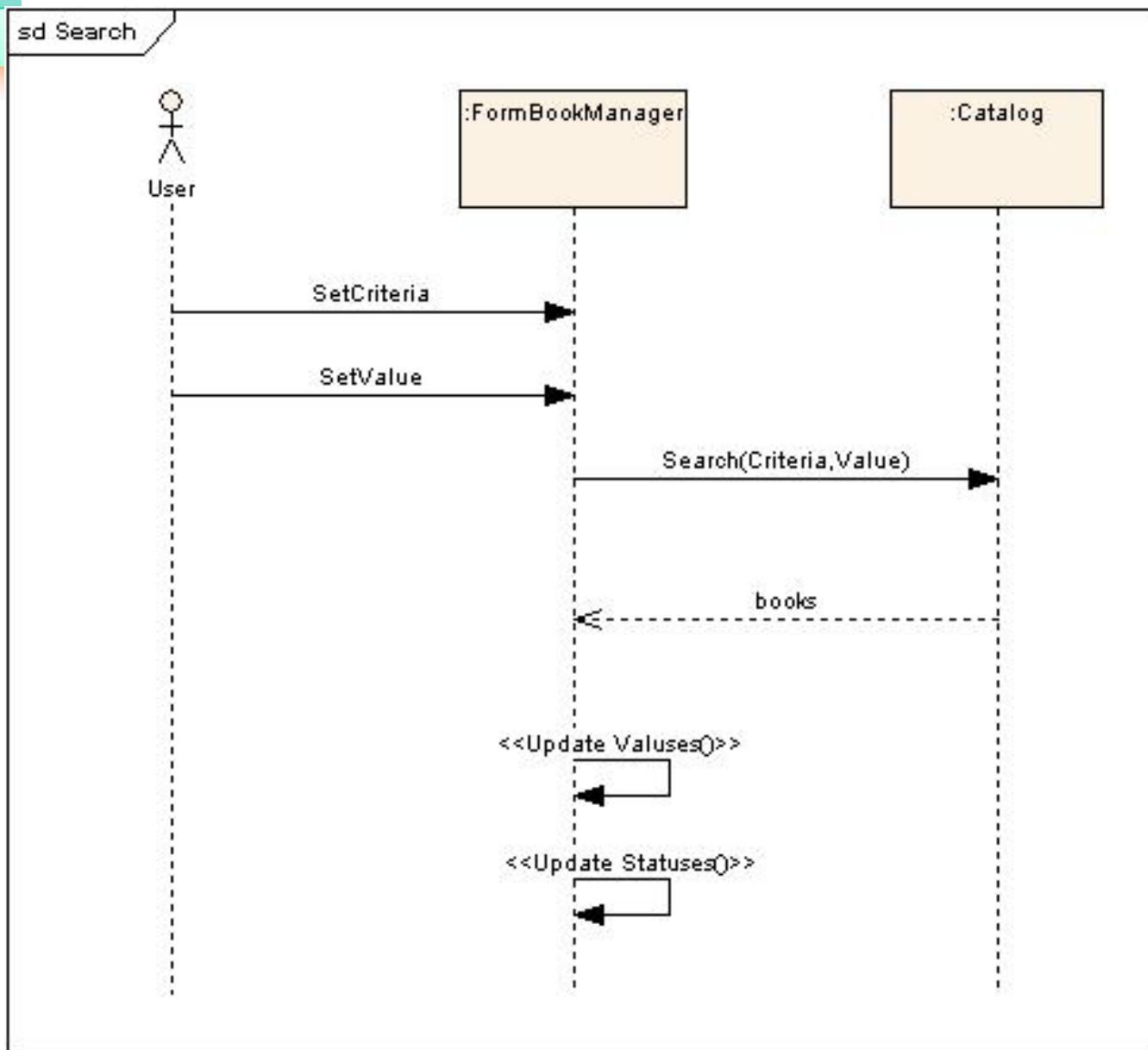
۹. و گرنه

۹.۱. سامانه به کاربر اطلاع می‌دهد که کتابی با اطلاعات وارد شده پیدا نشد.

شرایط نهایی: ندارد

روند جایگزین: ندارد

مثال ۳



انواع پیام ها

- پیام سنکرون (synchronous message) : با یک خط پر و یک پیکان توپر نشان داده می شود.
- پیام آسنکرون (asynchronous message) : یک خط پر به همراه یک پیکان باز.
- پیام بازگشت (return message) : یک خط چین به همراه یک پیکان باز
- پیام مفقود (lost message) : پیامی که به مقصد نمی رسد. یک پیام سنکرون است که در یک نقطه پایان(نقطه سیاه) تمام می شود.
- پیام پیدا شده (found message) : پیامی که فرستنده آن مشخص نیست. پیام سنکرونی که در یک نقطه پایان شروع می شود.

انواع پیام ها

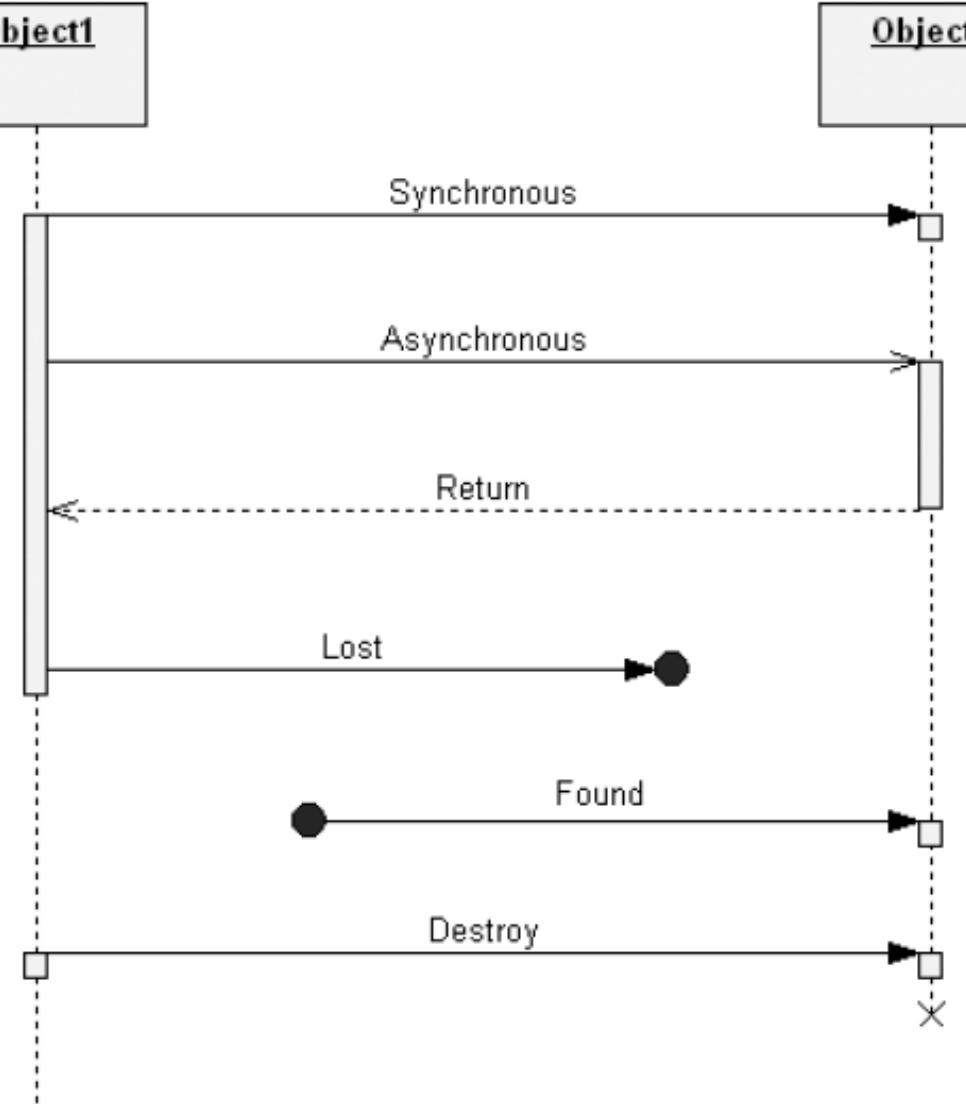
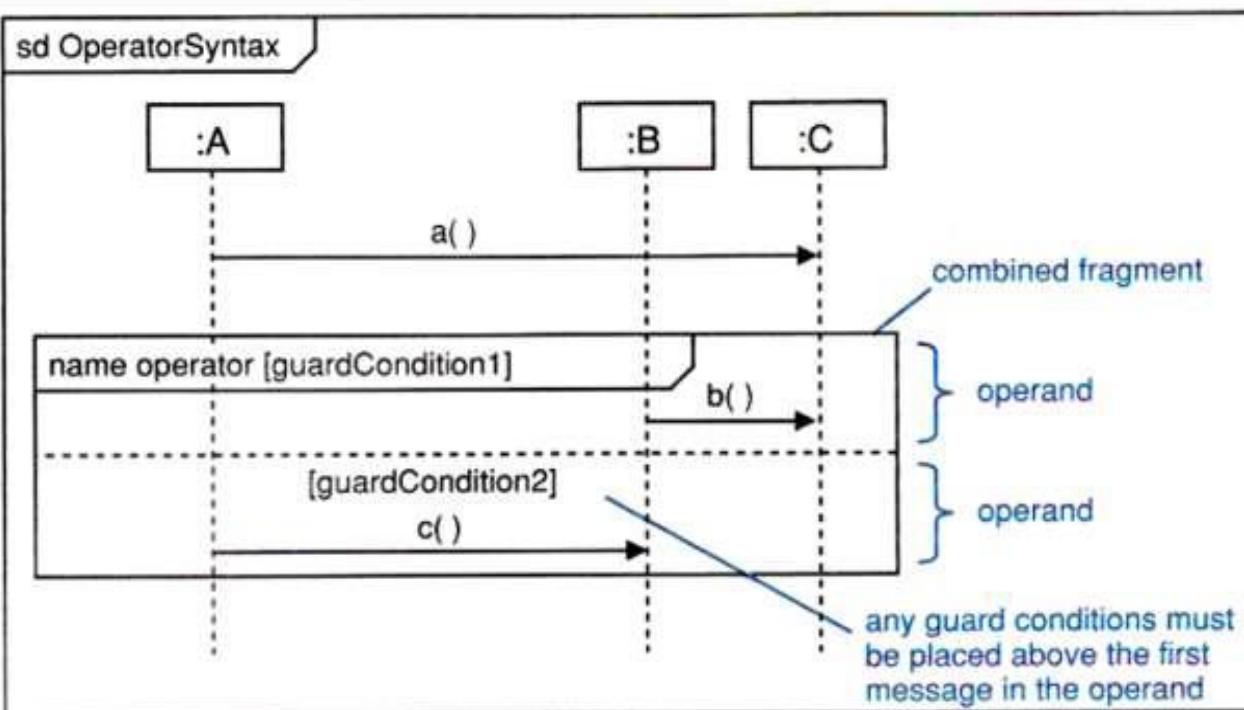


Figure 5–43 Notations for Types of Messages

قطعات ترکیبی

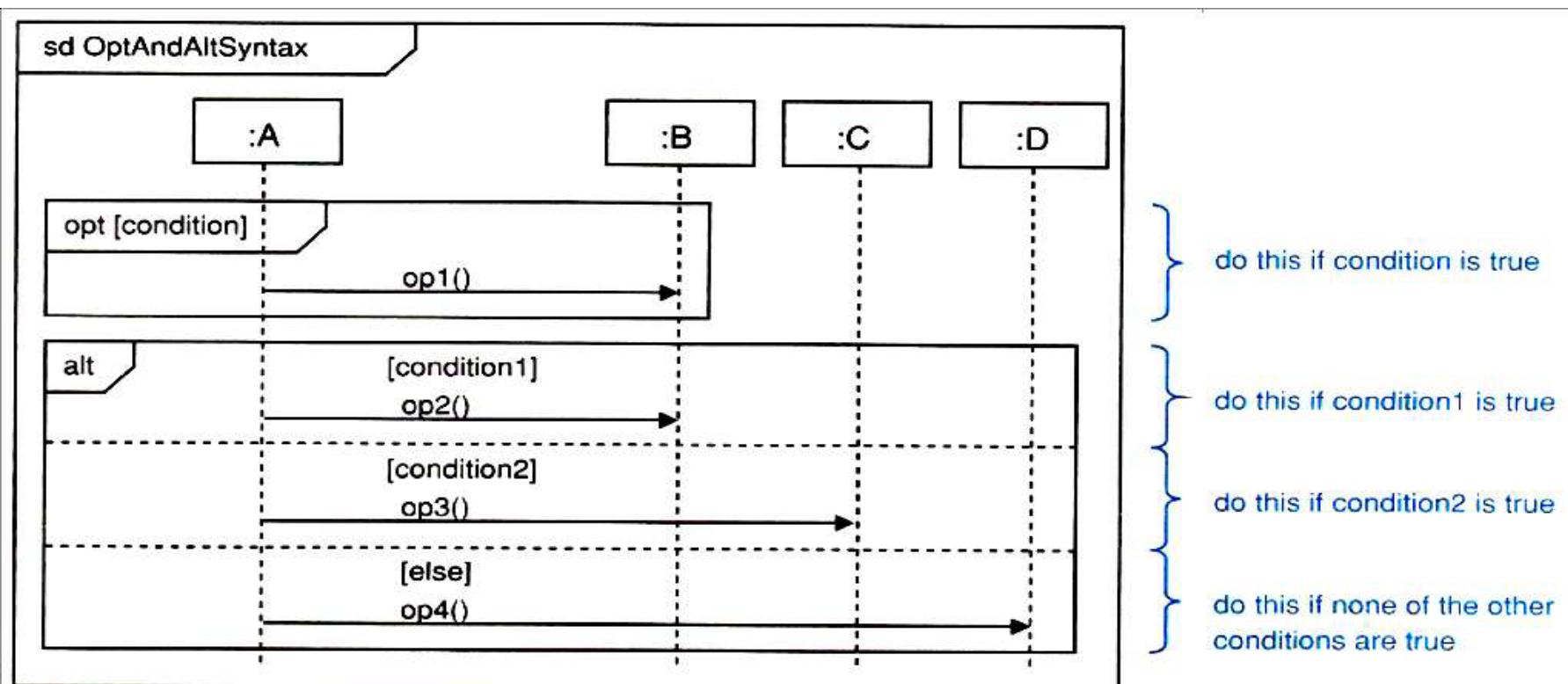
- نواحی داخل یک sequence diagram با رفتارهای مختلف مشخص می کند که چگونه Operatorها اجرا می شوند.
- مشخص می کند که operand Guard condition شود یا نه.
- شامل رفتار است.



قطعات ترکیبی: Operators – *opt* and *alt*:

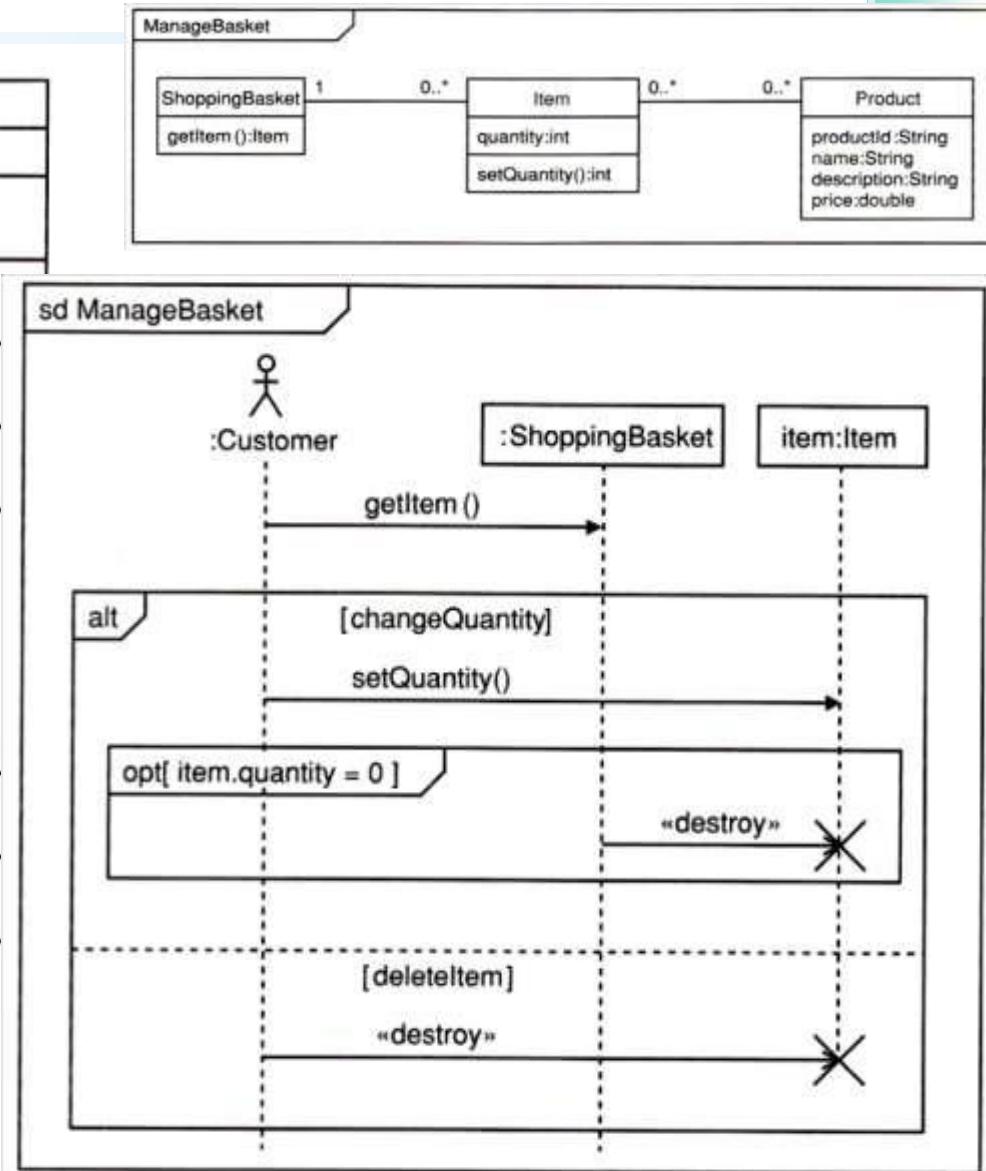
Operand : opt اگر شرط برقرار باشد اجرا می شود (مانند if ... then)

Operand : alt که شرط آن برقرار است اجرا می شود.



قطعات ترکیبی: *opt* and *alt*:

| | |
|---|--|
| Use case: ManageBasket | |
| ID: 2 | |
| Brief description: The Customer changes the quantity of an item in the basket. | |
| Primary actors: Customer | |
| Secondary actors: None. | |
| Preconditions: 1. The shopping basket contents are visible. | |
| Main flow: 1. The use case starts when the Customer selects an item in the basket. 2. If the Customer selects "delete item" 2.1 The system removes the item from the basket. 3. If the Customer types in a new quantity 3.1 The system updates the quantity of the item in the basket. | |
| Postconditions: None. | |
| Alternative flows: None. | |



مثال ۴

مورد کاربرد: اضافه کردن کتاب

شماره: ۱۵

شماره پدر: ۱۴

توصیف اجمالی: مدیر می تواند کتاب جدید تعریف کند.

عامل اصلی: مدیر

عامل فرعی: ندارد

شرایط اولیه: مدیر باید وارد سامانه شده باشد.

روند اصلی:

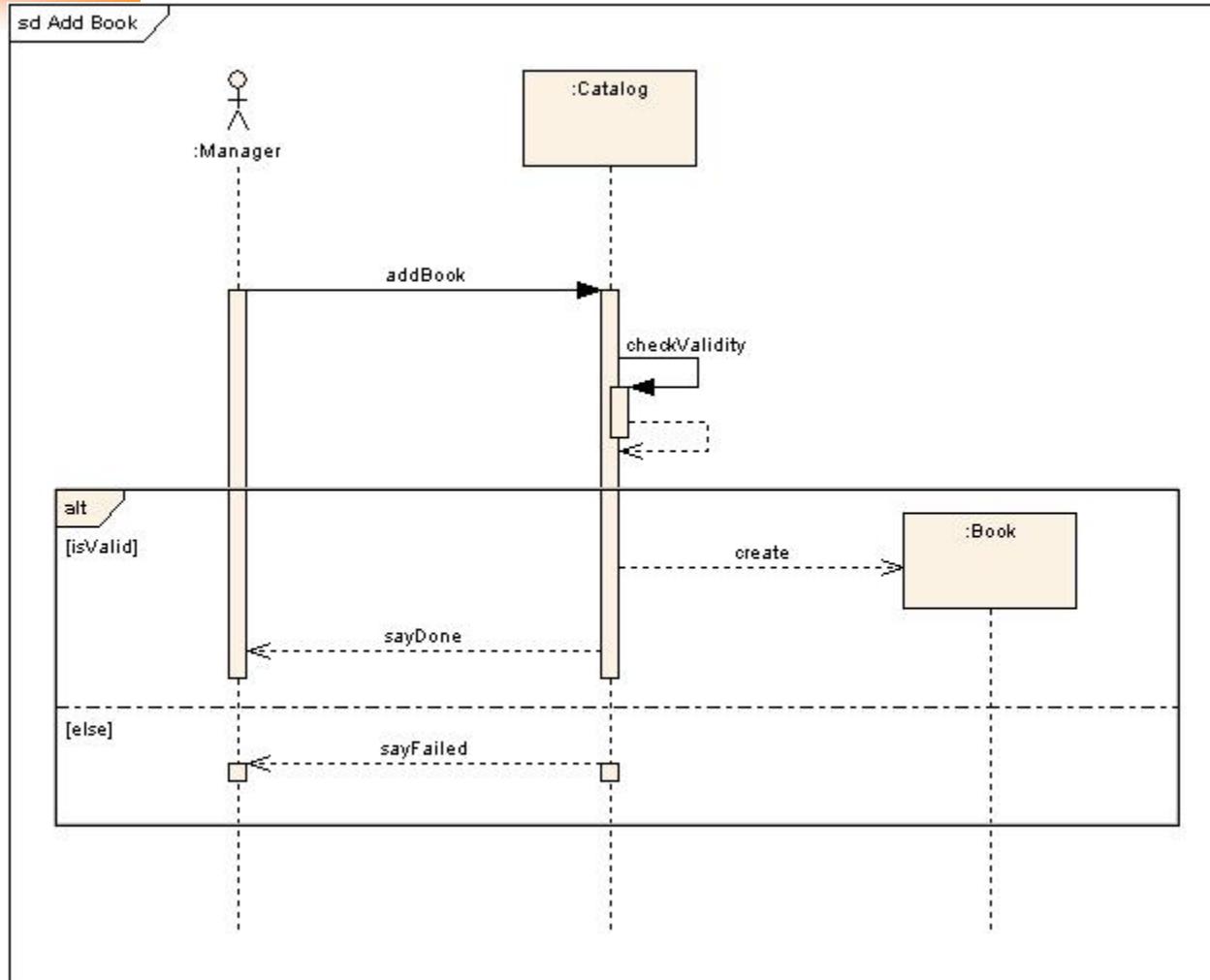
۱. این مورد کاربرد وقتی آغاز می شود که مدیر درخواست اضافه نمودن کتابی را اعلام کند.
۲. سامانه از مدیر درخواست می کند اطلاعات کتاب جدید را وارد کند.
۳. مدیر اطلاعات مورد نظر خود را وارد می کند.
۴. سامانه از مدیر می خواهد دسته مورد نظر جهت اضافه نمودن کتاب را انتخاب کند.
۵. مدیر دسته مورد نظر خود را مشخص می کند.
۶. مدیر اطلاعات وارد شده را تائید می کند.
۷. سامانه مجاز بودن اضافه نمودن کتاب را بررسی می کند.
۸. سامانه کتاب را اضافه می کند.

شرایط نهایی: کتابی به مجموعه کتاب ها اضافه می شود.

روند جایگزین:

مجاز نبودن اضافه کردن کتاب جدید

مثال ٤



قطعات ترکیبی: *loop* and *break*

loop – loop min, max [condition] ■

loop or loop * - loop forever; □

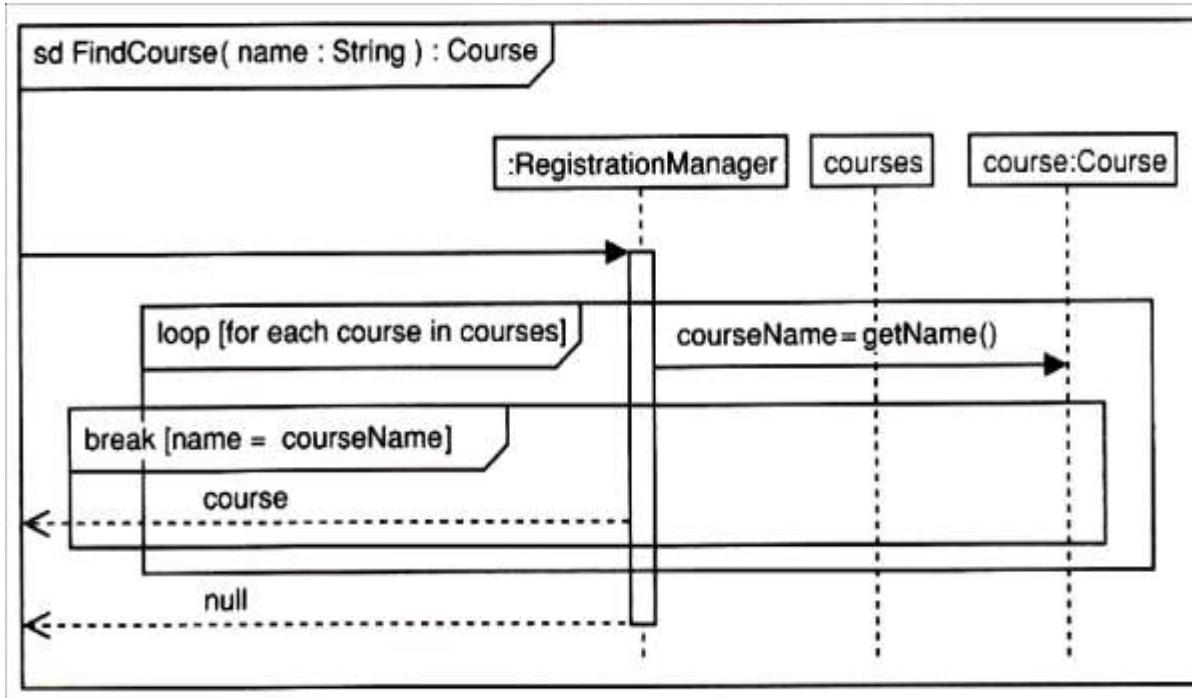
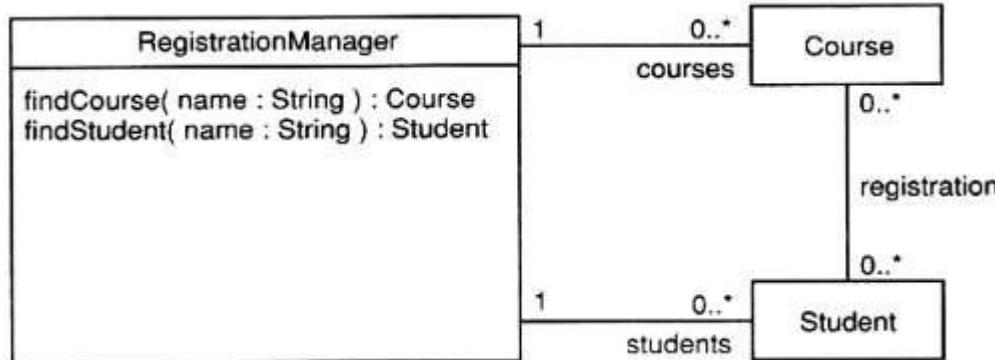
loop n, m – loop (m-n) times; □

loop [booleanExpression] – loop while
booleanExpression is true; □

loop 1, * [booleanExpression] – loop once then
loop while booleanExpression is true; □

operand guard condition : اگر درست باشد، *break* ■
اجرا می شود، نه بقیه حلقه تکرار

قطعات ترکیبی: *loop* and *break*



State Machine Diagrams

- ماشین های حالت در صنعت به دلیل استفاده در پردازش بلاذرنگ شهرت دارند.
- دیاگرام های ماشین حالت به منظور طراحی و فهم سیستم های مبتنی بر زمان استفاده می شوند.
- وسایل پزشکی، سیستم های مالی، سیستم های کنترل و فرمان ما هواره مثال هایی هستند که در آنها دیاگرام های ماشین حالت نقش مهمی در درک چگونگی عملکرد سیستم در مقابل اتفاقات کلیدی بازی می کنند.
- یک دیاگرام ماشین حالت، رفتار را به صورت توالی یک سری حالت ها، رخدادهای راه اندازی شده و عملیات وابسته ای که ممکن است اتفاق افتد.
- دیاگرام های ماشین حالت توصیف کننده رفتار اشیاء هستند. ولی می توانند اجزای بزرگتر هر سیستم را نشان دهند.

State Machine Diagrams (cont.)

- حالت شی نمایش دهنده نتایج متراکم رفتار آن است.
- برای مثال، وقتی یک تلفن راه اندازی می شود، در حالت **idle** قرار دارد و آماده شروع به کار کردن (**initiate**) یا پاسخگویی (**receive call**) است. هنگامیکه گوشی تلفن را بر میداریم، تلفن در حالت شماره گیری (**dialing**) قرار دارد. در این حالت، تلفن زنگ نخواهد خورد، اگر تلفن زنگ بخورد و گوشی برداشته شود، تلفن در حالت پاسخگویی قرار می گیرد و ما قادر به صحبت کردن با شخصی که تماس گرفته خواهیم بود.

State Machine Diagrams (cont.)

- هنگامی که یک شی در یک حالت مفروض قرار می‌گیرد، می‌تواند یکی از موارد زیر را انجام دهد:
 - اجرای یک فعالیت
 - انتظار برای یک رخداد
 - تکمیل یک شرط
 - انجام یک یا همه شرایط بالا
- در هر دیاگرام حالت، باید فقط یک حالت شروع (**initiate state**) وجود داشته باشد.

Initial State



State

Final State



State



State Machine Diagrams (cont.)

▪ حرکت بین حالت‌های مختلف transition نامیده می‌شود.

▪ هر transition دو حالت را به هم متصل می‌کند.

▪ یک حالت می‌تواند یک transition به خودش داشته باشد.

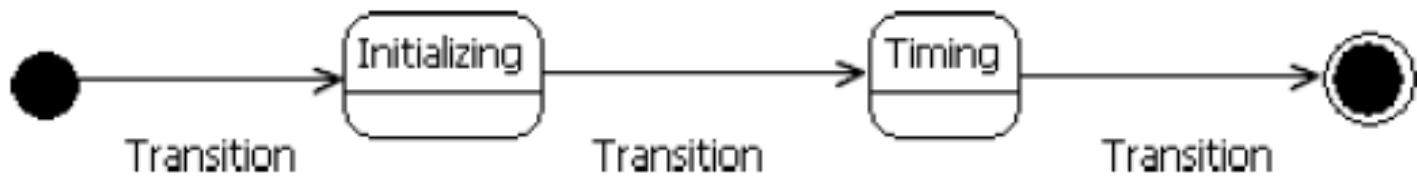


Figure 5–53 Transitions for the Duration Timer

State Machine Diagrams (cont.)

- رخدادهای خاصی باید اتفاق افتد تا یک **transition** انجام شود.
- این رخدادها روی **transition** نوشته می‌شوند.
- یک رخداد، اتفاقهایی است که ممکن است سبب تغییر حالت سیستم شود.

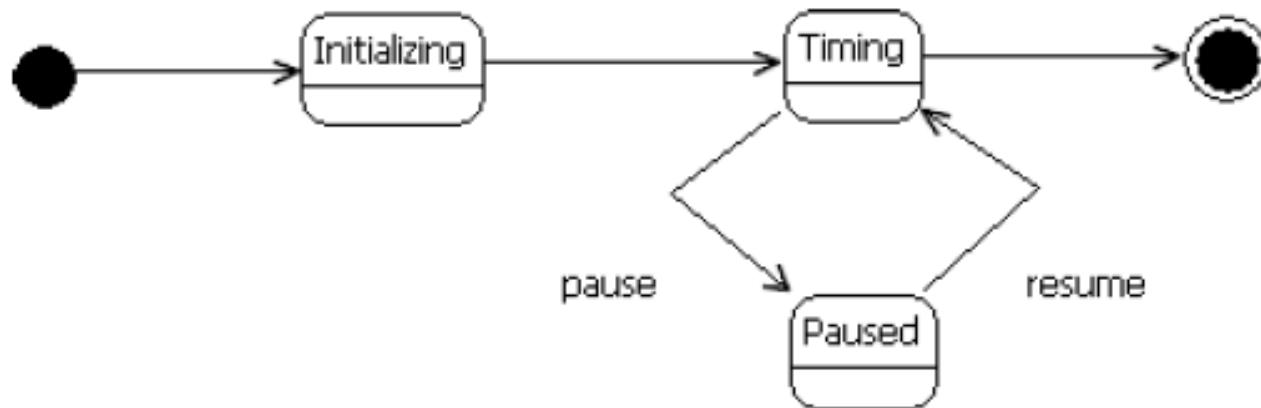


Figure 5–54 Additional States and Transition Events for the Duration Timer

State Machine Diagrams (cont.)

- فعالیت ها ممکن است با حالت ها همراه شوند.
- انجام یک فعالیت به محض ورود به حالت
- انجام یک فعالیت در حالی که در یک state هستیم.
- انجام یک فعالیت به محض خروج از یک حالت.

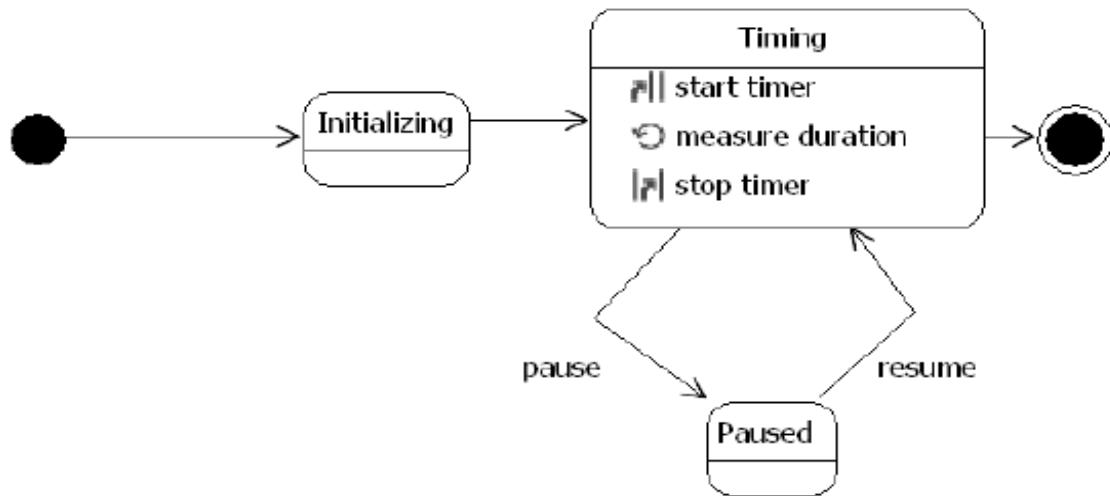


Figure 5–55 Entry, Do, and Exit Activities

State Machine Diagrams (cont.)

- شرط ها هم می توانند برای کنترل **transition** به کار روند.
- این شرایط به عنوان محافظ (guard) عمل می کنند، زیرا وقتی یک رخداد اتفاق می افتد، شرط می تواند اجازه دهنده **transition** باشد (اگر شرط درست باشد) یا به **transition** اجازه اجرا ندهد (اگر شرط نادرست باشد).
- راه دیگر کنترل رفتار **transition** استفاده از **effect** ها است.
- یک **effect** رفتاری(فعالیت یا عمل) است که وقتی یک رخداد اتفاق می افتد، رخ می دهد.
- بنابراین هنگامی که یک رخداد **transition** اتفاق می افتد، **effect** اجرا شده و **effect** هم اتفاق می افتد.

State Machine Diagrams (cont.)

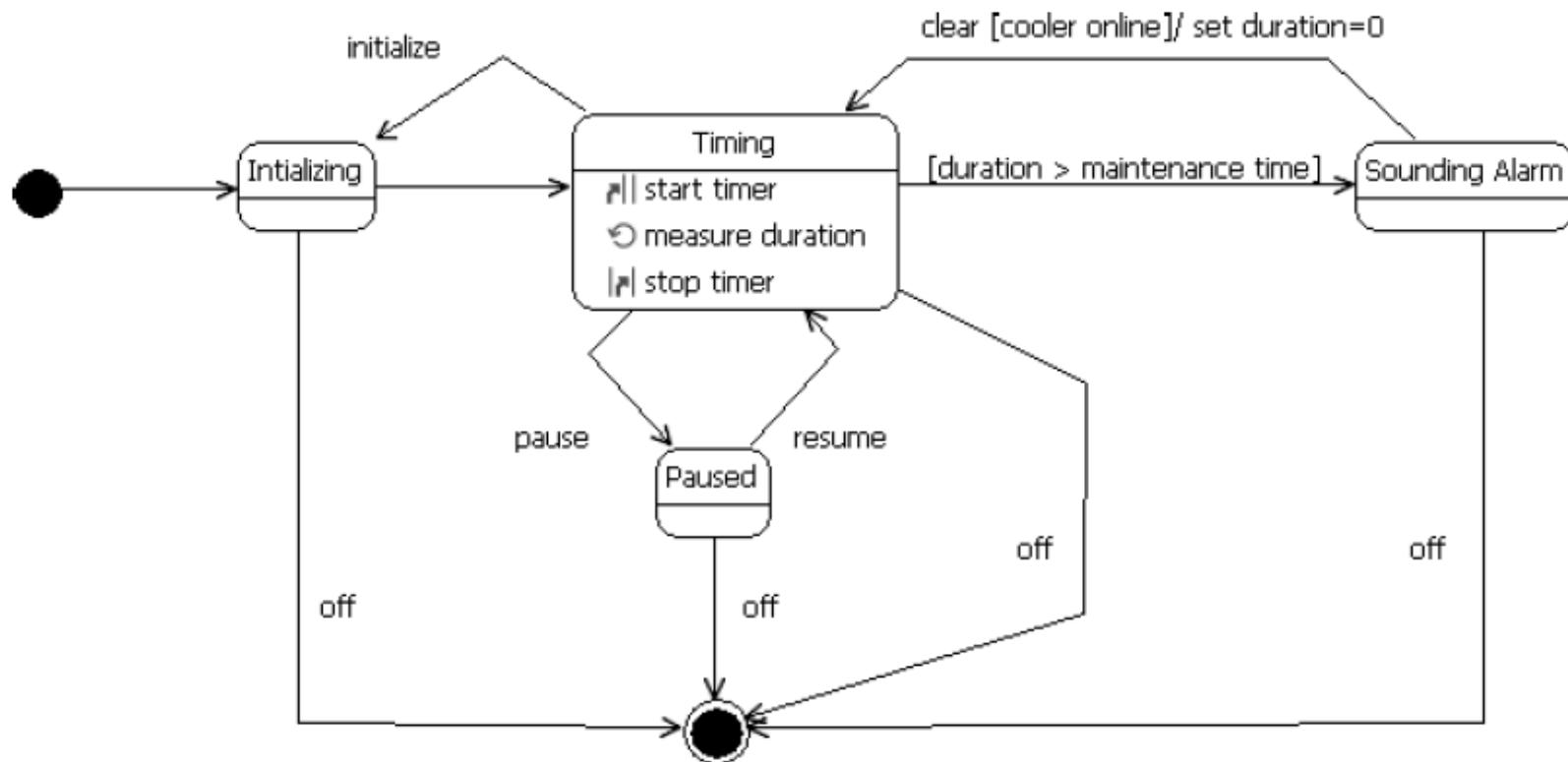


Figure 5–56 The Enhanced State Machine Diagram for the Duration Timer

State Machine Diagrams (cont.)

- ترتیب ارزیابی در حالت شرطی مهم است.
- حالت S با گذار T روی رخداد E با شرط C و A effect مفروض است. ترتیب زیر اعمال می شود:
 - .1 رخداد E اتفاق می افتد.
 - .2 شرط C ارزیابی می شود.
 - .3 اگر C درست باشد، T راه اندازی می شود و A effect احضار می گردد.
- به این معنی که اگر یک شرط به درستی ارزیابی نشود، حالت ممکن است راه اندازی نشود تا زمانیکه رخداد E دوباره اتفاق افتد و شرط دوباره ارزیابی شود.

State Machine Diagrams (cont.)

- در سیستم های پیچیده، دیاگرام های ماشین حالت، می توانند بسیار بزرگ باشند.
- امکان استفاده از حالت های تو در تو وجود دارد.
- در این حالت، دیاگرام ماشین حالت به صورت عمقی طراحی می شود.

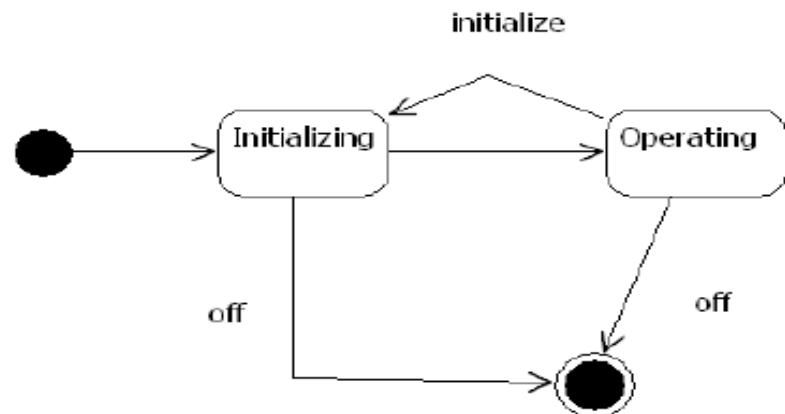


Figure 5–58 A Higher-Level View of the State Machine Diagram for the Duration Timer

State Machine Diagrams (cont.)

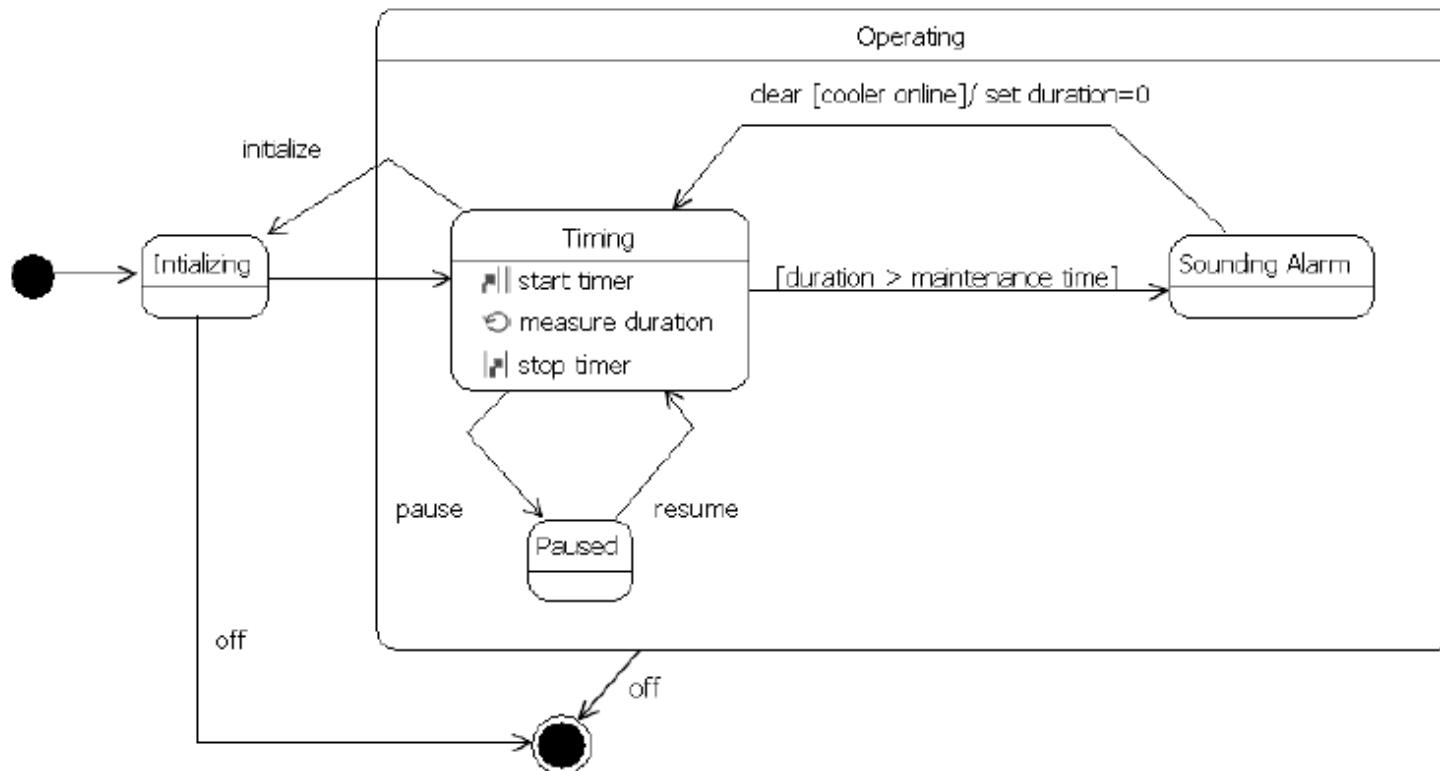


Figure 5–57 Composite and Nested States

پایان