

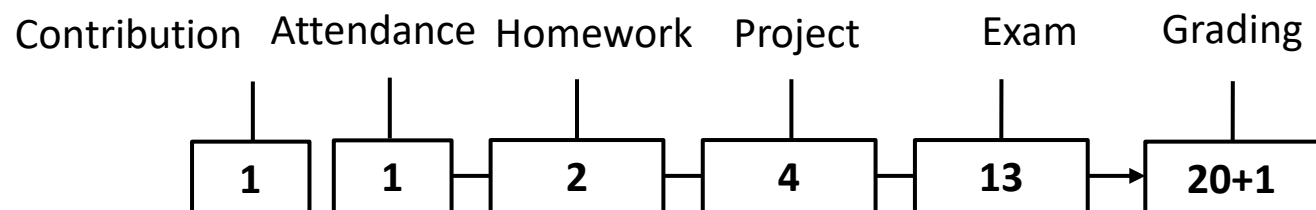
به نام خداوند جان و خرد

زبان مدل سازی یکپارچه



مهندسی نرم افزار ۲

ارزیابی



زبان مدل سازی یکپارچه (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 ها را پیدا کنید.

■ Use case کاری است که actor ها در سیستم انجام می دهد.

■ از use case های خیلی بزرگ یا خیلی کوچک اجتناب کنید.

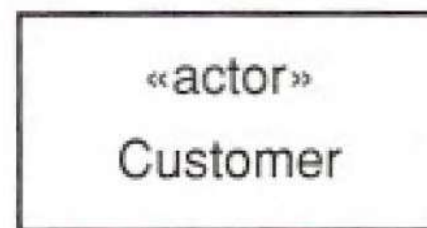
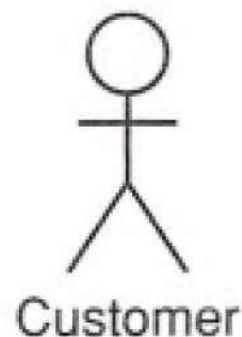
■ تا زمانی که use case ها، Actor ها و مرز سیستم به حالت ایستا برسد این عملیات را ادامه دهید.

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** ها)
- **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 می شود.

2. مدیر فروشگاه: (Store Manager)

مسئول مدیریت محصولات، سفارش ها، موجودی کالا و نظارت بر عملکرد کلی سیستم.

3. سیستم هوش مصنوعی: (AI System)

این سیستم وظیفه تحلیل داده های مشتری، ارائه پیشنهادات شخصی سازی شده، پیش بینی روندهای خرید و بهبود تعامل با کاربر را دارد.

4. سیستم پرداخت: (Payment Gateway)

مسئول پردازش تراکنش های مالی به صورت امن و تأیید پرداخت ها.

5. سیستم مدیریت موجودی: (Inventory Management System)

سامانه ای برای کنترل و به روز رسانی موجودی کالاها، دریافت هشدارها در مورد کاهش موجودی و هماهنگی با بخش سفارش ها.

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 Actorها :
 - 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 Actorها

- 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 **Actor** خارجی انجام میدهد.
- همیشه توسط یک **Actor** شروع می شود.
- همیشه از دیدگاه **actor** نوشته می شود.



شناسایی use case ها

موارد استفاده: (Use Cases)

1. جستجو و نمایش محصولات:

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

2. پیشنهاد محصولات شخصی سازی شده:

- توضیح: پس از ثبت نام یا ورود به سیستم، مشتری براساس تاریخچه فعالیت هایش، پیشنهادات ویژه دریافت می کند.
- نقش هوش مصنوعی: تحلیل داده های رفتاری مشتریان و استفاده از الگوریتم های یادگیری ماشین برای پیشنهاد محصولات مطابق با سلیقه فردی.

3. خرید و پرداخت آنلاین:

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

4. مدیریت سفارشات و پیگیری وضعیت سفارش:

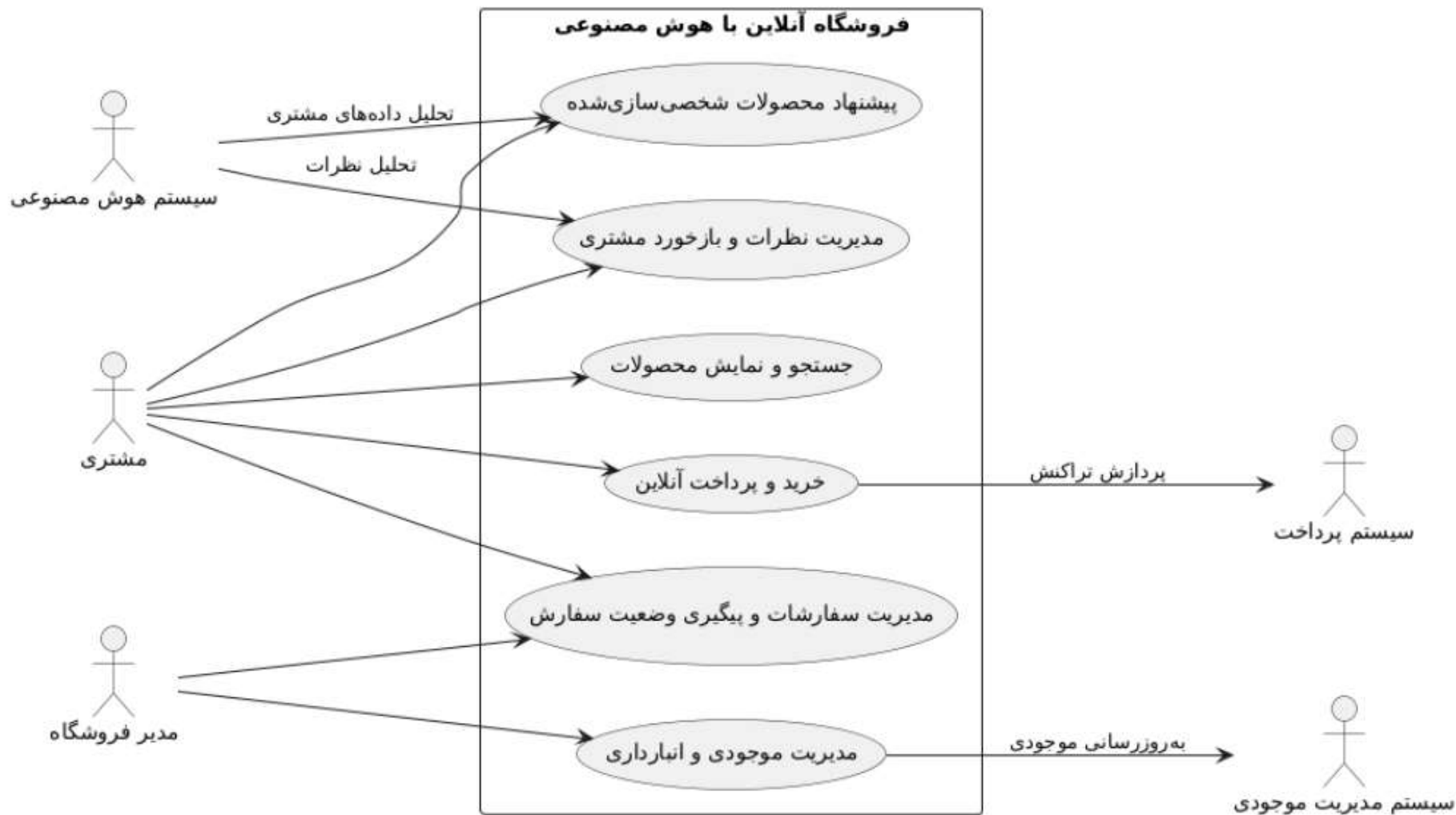
- توضیح: مشتری پس از خرید قادر به پیگیری وضعیت سفارش است؛ از ثبت سفارش تا تحویل نهایی.
- نقش مدیر فروشگاه: نظارت بر روند پردازش سفارش ها و هماهنگی با بخش های مختلف جهت تحویل به موقع.

5. 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 می کند.
- نقش سیستم مدیریت موجودی: کاهش خطاهای انسانی و بهبود روند سفارش دهی مجدد کالاها.

6. مدیریت نظرات و بازخورد مشتری:

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



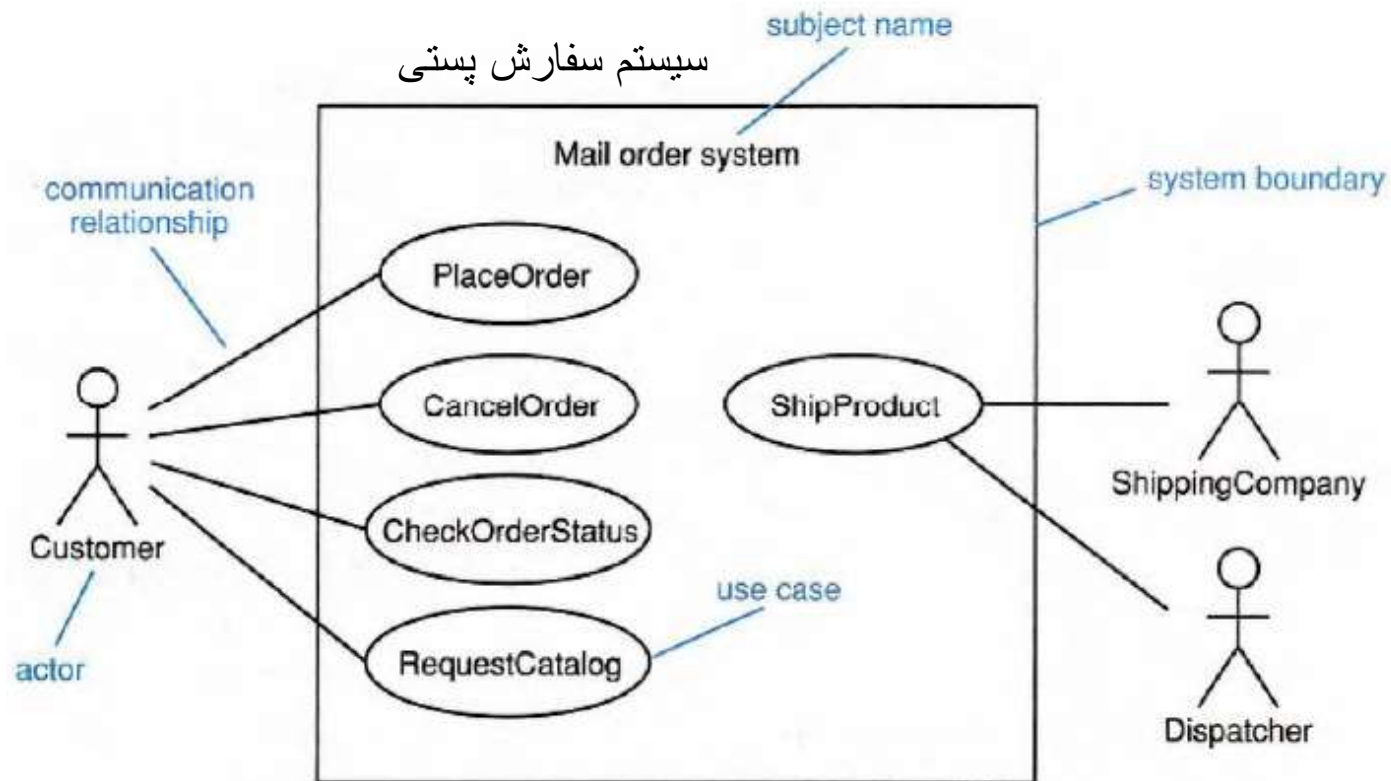
شناسایی use case ها

- 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 فعل ی 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 اینچنین 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 رخداد های 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.
- Actorهایی که در use case هستند:
 - Primary actors: actorی که use case را 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

مشخصه use case : مثال

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
	Secondary actors: TaxAuthority
the system state before the use case can begin	Preconditions: 1. It is the end of the business quarter.
the actual steps of the use case	Main flow: <i>implicit time actor</i> 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 تأیید 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 یافته 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 خطا، وقفه یا مشکلی وجود ندارد، نشان می دهد.
- روند فرعی (alternative flow): انحراف ها از جریان اصلی که منجر به خطا، 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 خطاها و استثنای روند اصلی رسیدگی کرده و پس شرط های متفاوتی دارند.
- بهتر است به طور جداگانه مستند شوند.
- ممکن است به سه حالت متخلف 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

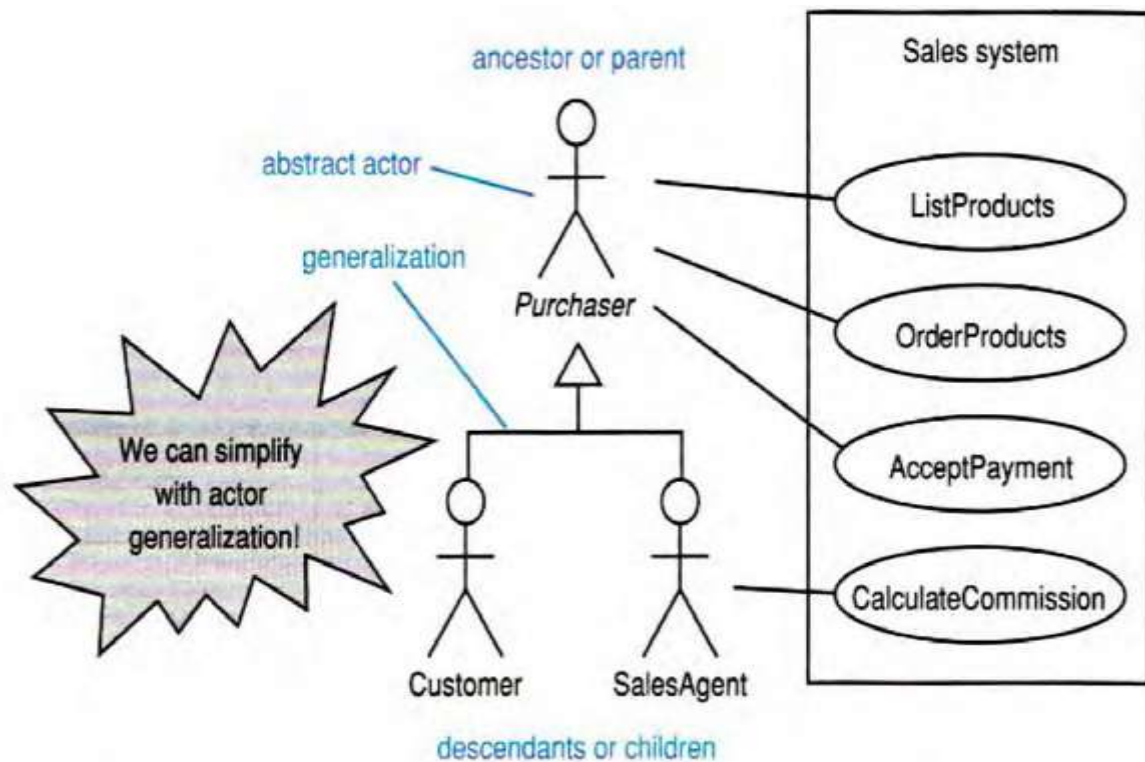
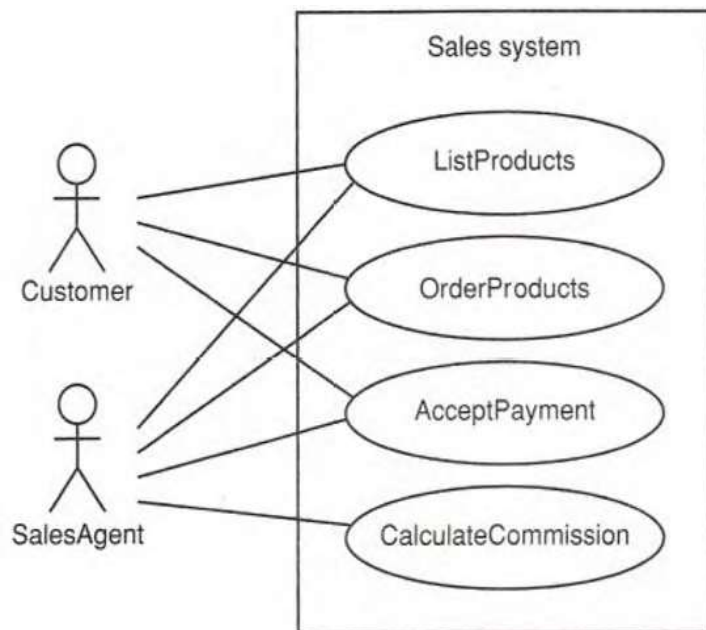
روند جایگزین: فروش کتاب : نداشتن موجودی کافی کتاب
شماره: ۶.۱
توصیف اجمالی: سامانه به فروشنده اطلاع می دهد که تعدادی از کتاب های لیست مورد نظر به اندازه تقاضا شده موجودی ندارند و همچنین آن کتاب ها را به لیست کتاب های تقاضا شده اضافه می کند.
عامل اصلی: فروشنده
عامل فرعی: ندارد
شرایط اولیه: تعداد تقاضا 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

- **actor generalization**: رابطه بین یک **Actor** کلی و یک **Actor** جزئی
- **Use case generalization**: رابطه بین یک **use case** کلی و یک **use case** جزئی – **use-case** فرزند رفتار پدر را به ارث می برد.
- **<<include>>**: رابطه بین **use case** ها که یک **use case** از رفتار **use case** دیگر استفاده می کند.
- **<<extend>>**: رابطه بین **use case** ها که یک **use-case** (**base use-case**) به طور ضمنی از رفتار **use-case** دیگر استفاده می کند. **Base use-case** ممکن است مستقل باشد، اما تحت شرایط خاص ممکن است گسترش داده شود.

Actor generalization

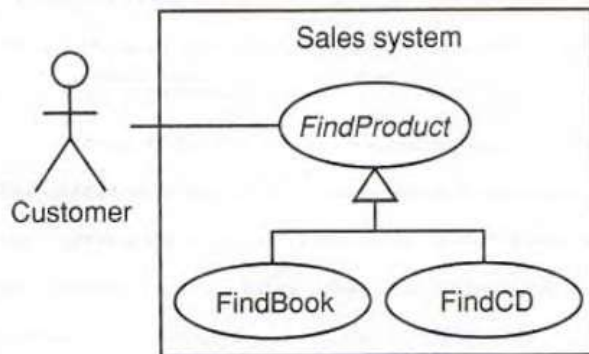
■ اگر دو Actor با هم از طریق مجموعه یکسانی از use case ها در ارتباط باشند



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

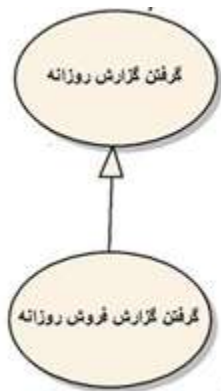


Use case: <i>FindProduct</i>
ID: 6
Brief description: The Customer searches for a product.
Primary actors: Customer
Secondary actors: None.
Preconditions: None.
Main flow: 1. The Customer selects "find product". 2. The system asks the Customer for search criteria. 3. The Customer enters the requested criteria. 4. The system searches for products that match the Customer's criteria. 5. If the system finds some matching products 5.1 The system displays a list of the matching products. 6. Else 6.1 The system tells the Customer that no matching products could be found.
Postconditions: None.
Alternative flows: None.

overridden
 overridden
 inherited without change
 overridden
 overridden
 added
 overridden and renumbered
 added
 added
 inherited without change
 added
 renumbered

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: 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 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. (5.) Else 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.
Postconditions: None.
Alternative flows: None.

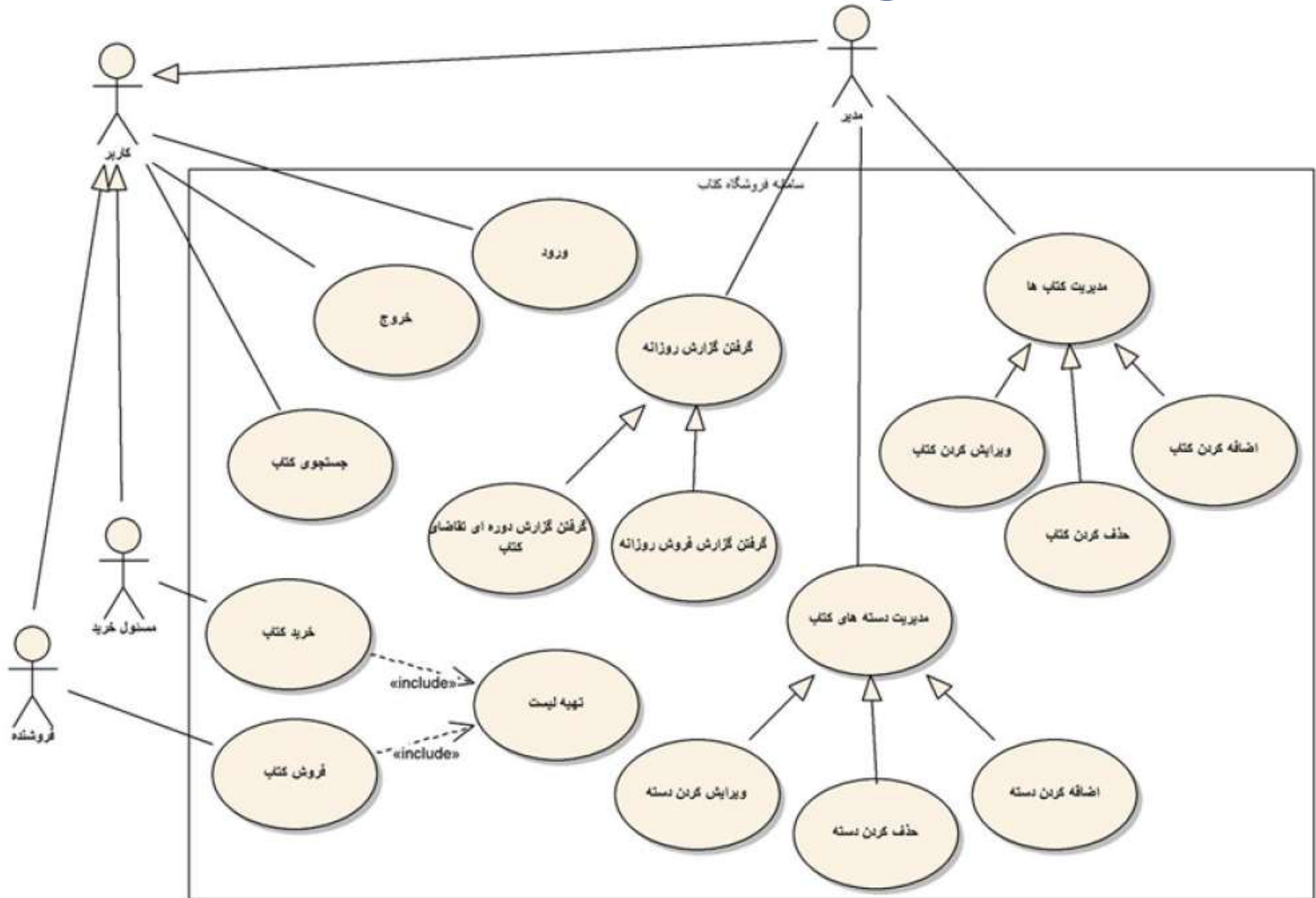
مثال use case generalization



مورد کاربرد: گرفتن گزارش روزانه
شماره: ۷
توصیف اجمالی: مدیر می تواند بر روی اطلاعات مشخصی در سامانه گزارش روزانه تهیه کند.
عامل اصلی: مدیر

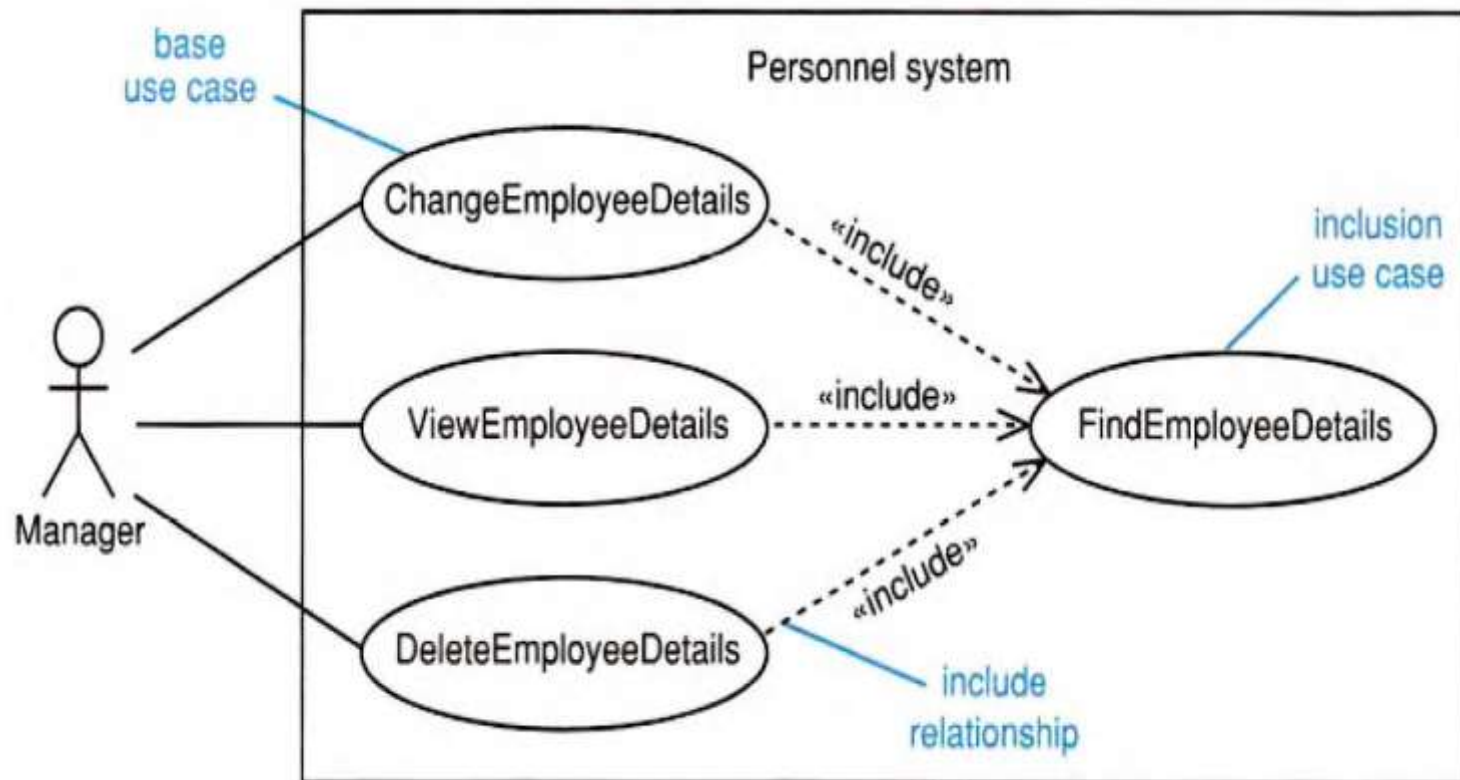
مورد کاربرد: گرفتن گزارش روزانه فروش	عامل فرعی: ندارد
شماره: ۸	شرایط اولیه: مدیر باید وارد سامانه شده باشد.
شماره پدر: ۷	روند اصلی:
توصیف اجمالی: مدیر می تواند بر روی اطلاعات فروش ثبت شده در سامانه گزارش روزانه تهیه کند.	۱. این مورد کاربرد وقتی آغاز می شود که مدیر بخواهد از سامانه گزارش روزانه بگیرد.
عامل اصلی: مدیر	۲. سامانه از مدیر، تاریخ جهت گزارش گیری را سوال می کند.
عامل فرعی: ندارد	۳. مدیر تاریخ مورد نظر خود را وارد می کند.
شرایط اولیه: مدیر باید وارد سامانه شده باشد.	۴. مدیر درخواست مشاهده گزارش را می دهد.
روند اصلی:	۵. سامانه گزارش مورد نظر مدیر را به او نشان می دهد.
۱. (01) این مورد کاربرد وقتی آغاز می شود که مدیر بخواهد، از سامانه گزارش روزانه فروش بگیرد.	شرایط نهایی: ندارد
۲. (۲) سامانه از مدیر، تاریخ جهت گزارش گیری را سوال می کند.	روند جایگزین: ندارد
۳. (۳) مدیر تاریخ مورد نظر خود را وارد می کند.	
۴. (۴) مدیر درخواست مشاهده گزارش را می دهد.	
۵. (05) سامانه گزارش فروش روزانه را به مدیر نشان می دهد.	
شرایط نهایی: ندارد	
روند جایگزین: ندارد	

مثال use case diagram



رابطه <<INCLUDE>>

- 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)

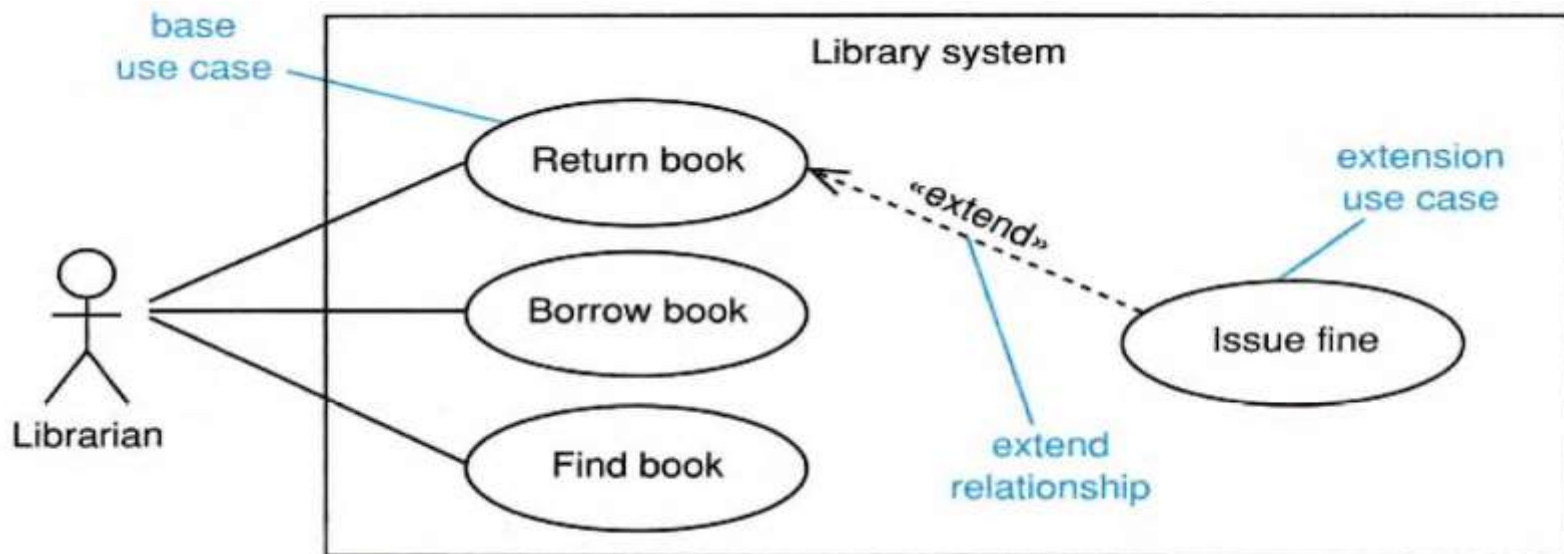
- Use case پایه بدون اضافه کردن use case دیگر کامل نیست. Use case include شده می تواند کامل باشد یا نباشد.

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.

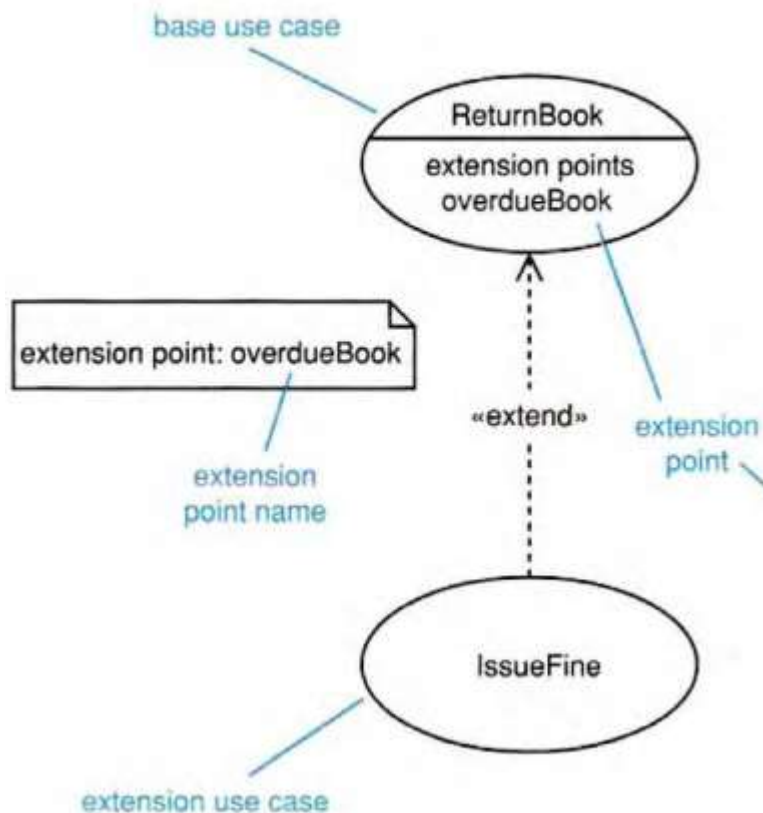
Use case: FindEmployeeDetails
ID: 4
Brief description: The Manager finds the employee details.
Primary actors: Manager
Secondary actors: None.
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. The Manager enters the employee's ID. 2. The system finds the employee details.
Postconditions: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

رابطه <<extend>>

- 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 بین 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 use case اولیه را تغییر می دهد؟



Class diagram ■

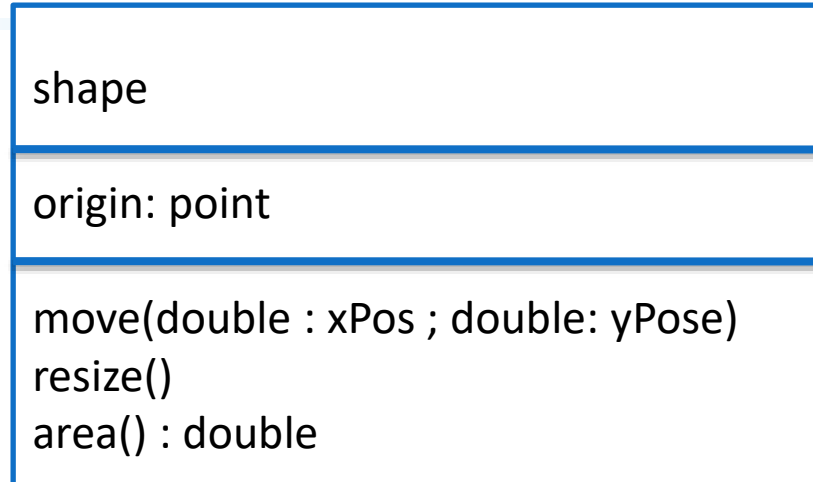
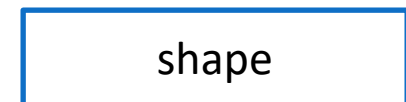
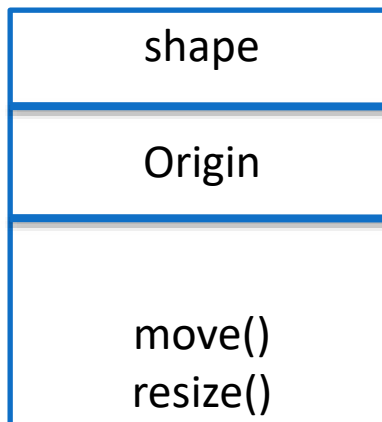
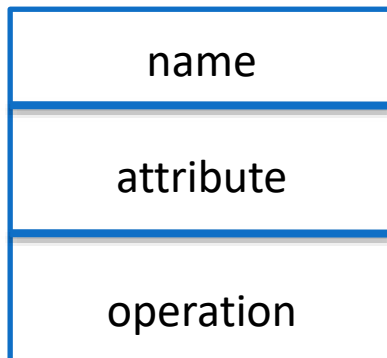
کلاس چیست؟

- 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.

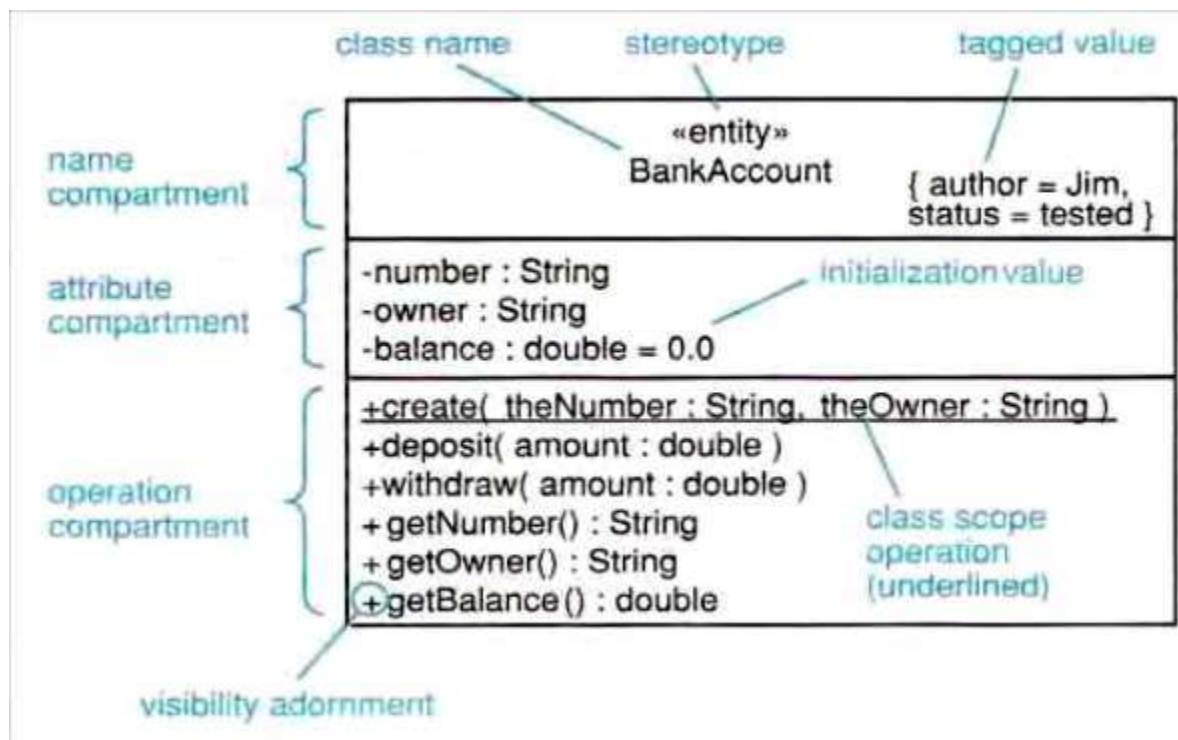
کلاس ها (ادامه...)

Semantic

- Name
- Attributes
- operations
- نمادهای کلاس



نشانه گذاری کلاس در UML



روابط کلاس ها و اشیاء

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

■ روابط :

■ انجمنی (association)

■ وابستگی (dependency)

■ تعمیم / تخصصی کردن (generalization/specialization)

■ 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 Association یا Dependency بین کلاسهای این اشیا وجود داشته باشد.

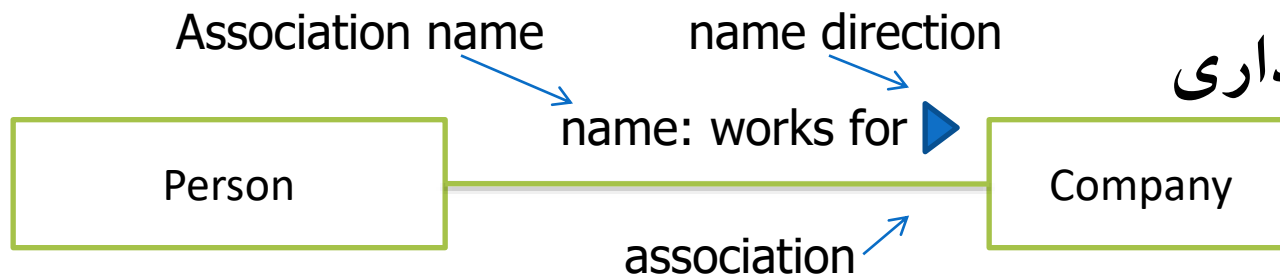
■ نام (name)

■ نقش (role)

■ چندگانگی (multiplicity): چندی رابطه بین دو شی را نشان می دهد. یک به یک ، یک به چند، چند به چند

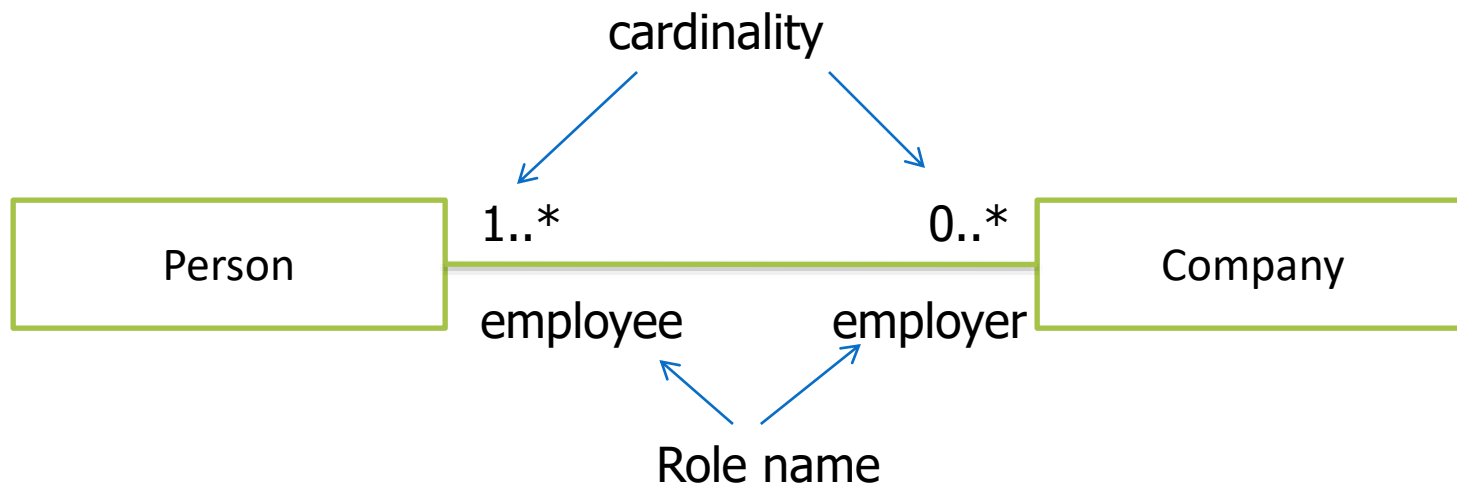
■ جهت (direction)

■ نمادگذاری



رابطه انجمنی (ادامه...)

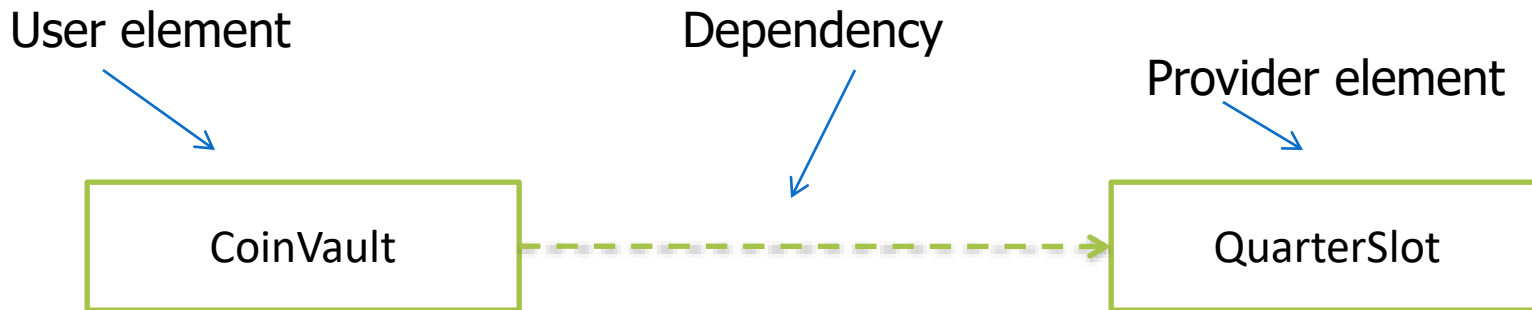
- نام **Association**: باید یک فعل یا عبارت باشد
- نام نقش :
- باید یک اسم یا عبارت اسمی باشد که توصیف کننده نقش است.
- چندی: نشان دهنده تعداد اشیائی است که می توانند در رابطه وجود داشته باشند.



رابطه وابستگی (Dependency Relationship)

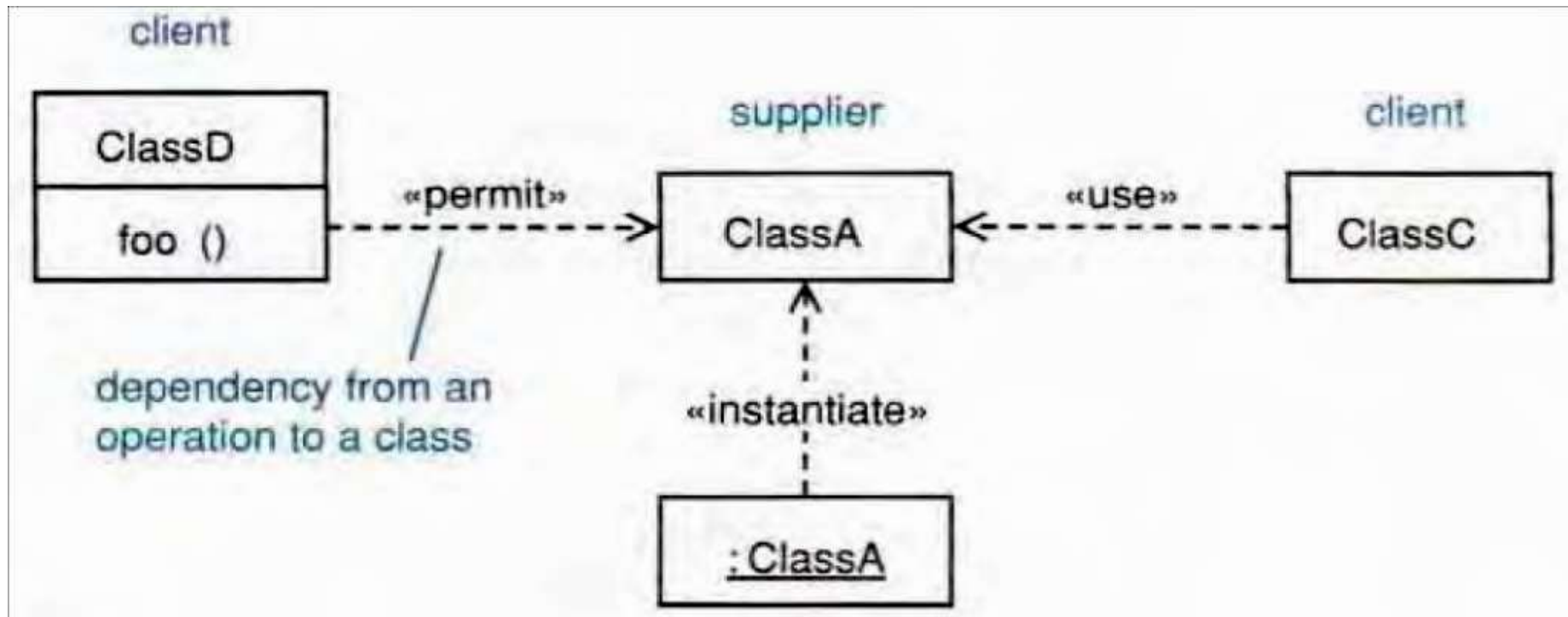
- این رابطه نشان می دهد که یک تغییر در **provider** **element** نیاز به تغییری در **user element** دارد. معمولاً رابطه وابستگی نشان می دهد که یک کلاس (**user**) از یک کلاس دیگر (**provider**) به عنوان آرگومانی در امضای یکی از توابع اش استفاده می کند.

■ نماد:



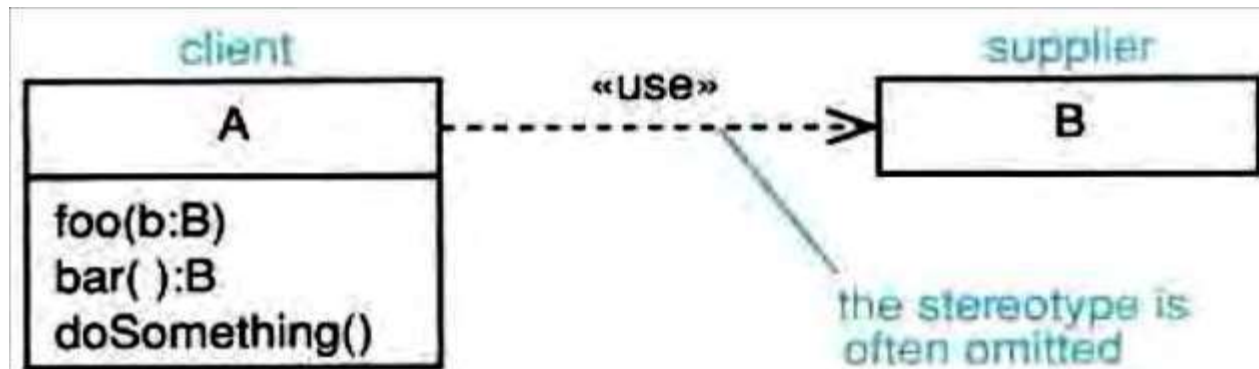
CoinVault از QuarterSlot استفاده می کند

رابطه وابستگی (Dependency Relationship)

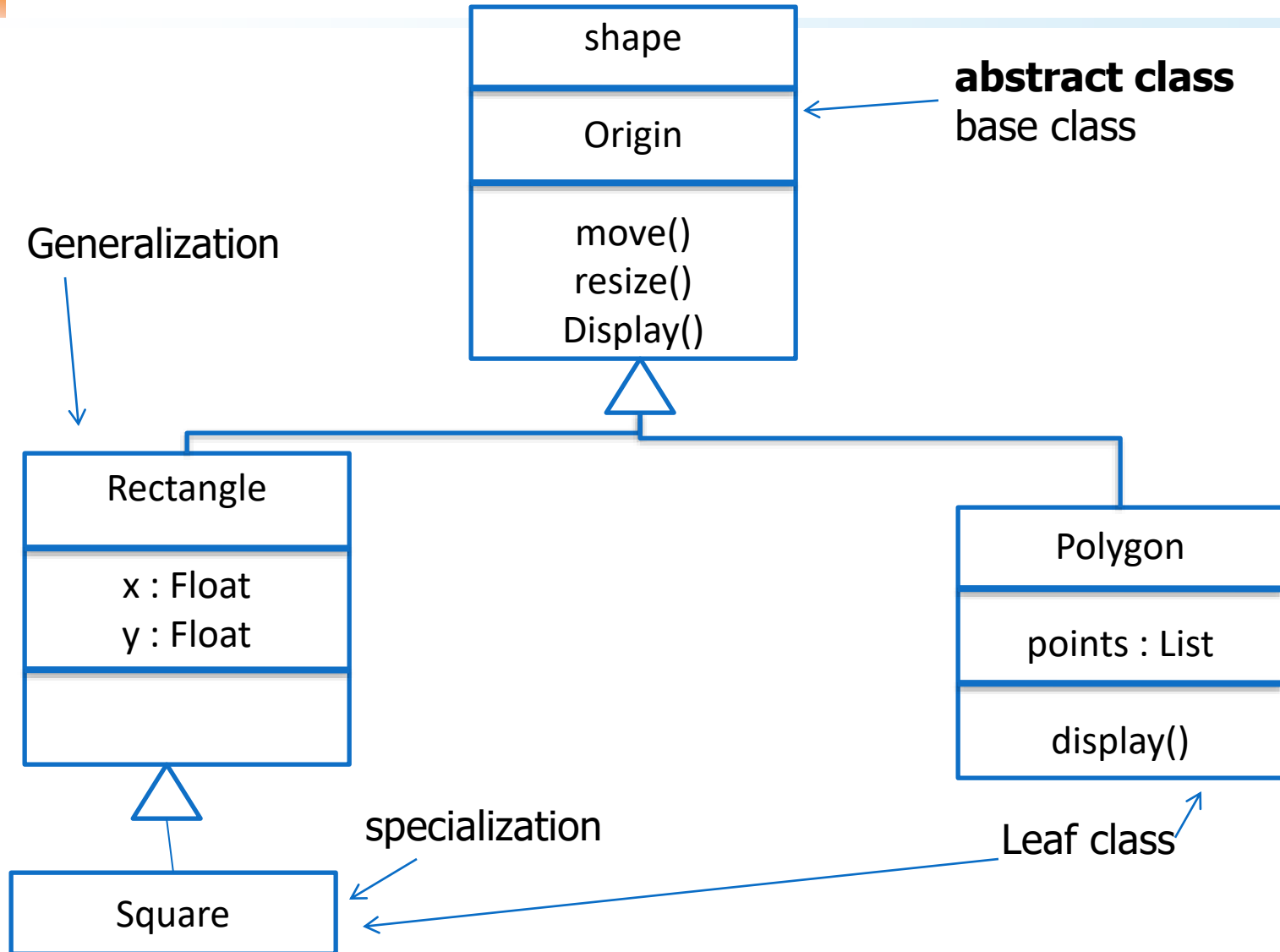


کاربرد وابستگی

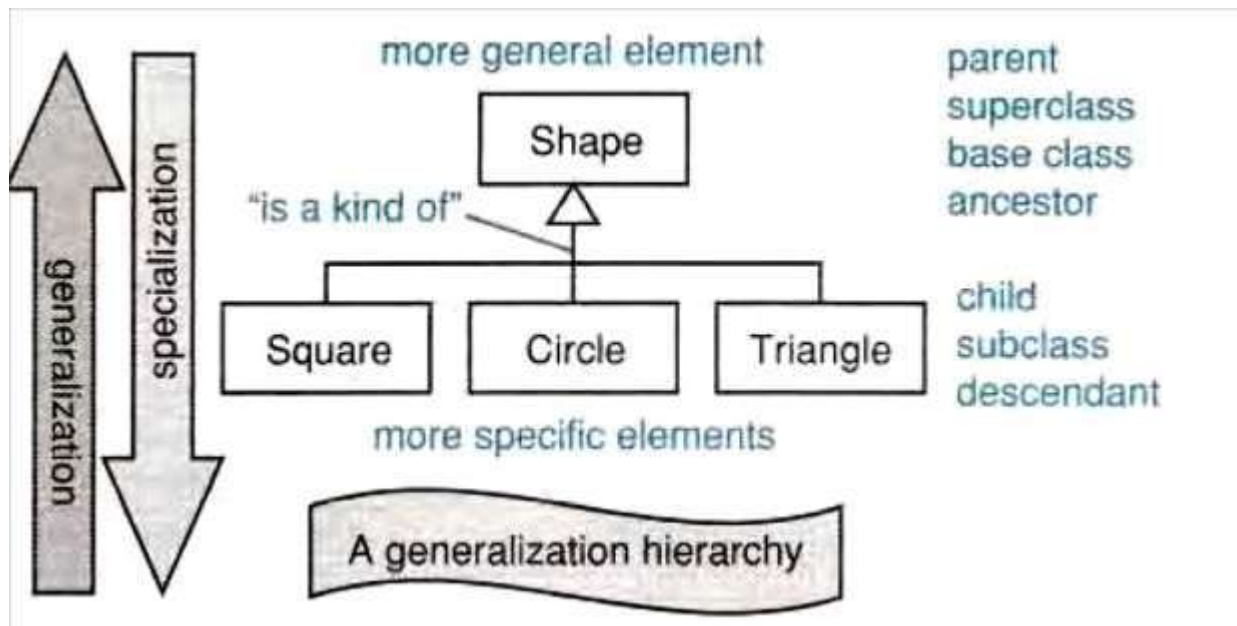
- **<<use>>** - کلاس کلاینت از **supplier** استفاده می کند.
- **<<call>>** - تابع کلاینت ، تابع **supplier** را صدا میزند.
- **<<parameter>>** - **supplier** یک پارامتر یا مقدار برگشتی یکی از توابع کلاینت است.
- **<<send>>** - کلاینت یک سیگنال به **supplier** می فرستد.
- **<<instantiate>>** - کلاینت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



رابطه وارث یگانه (single inheritance)

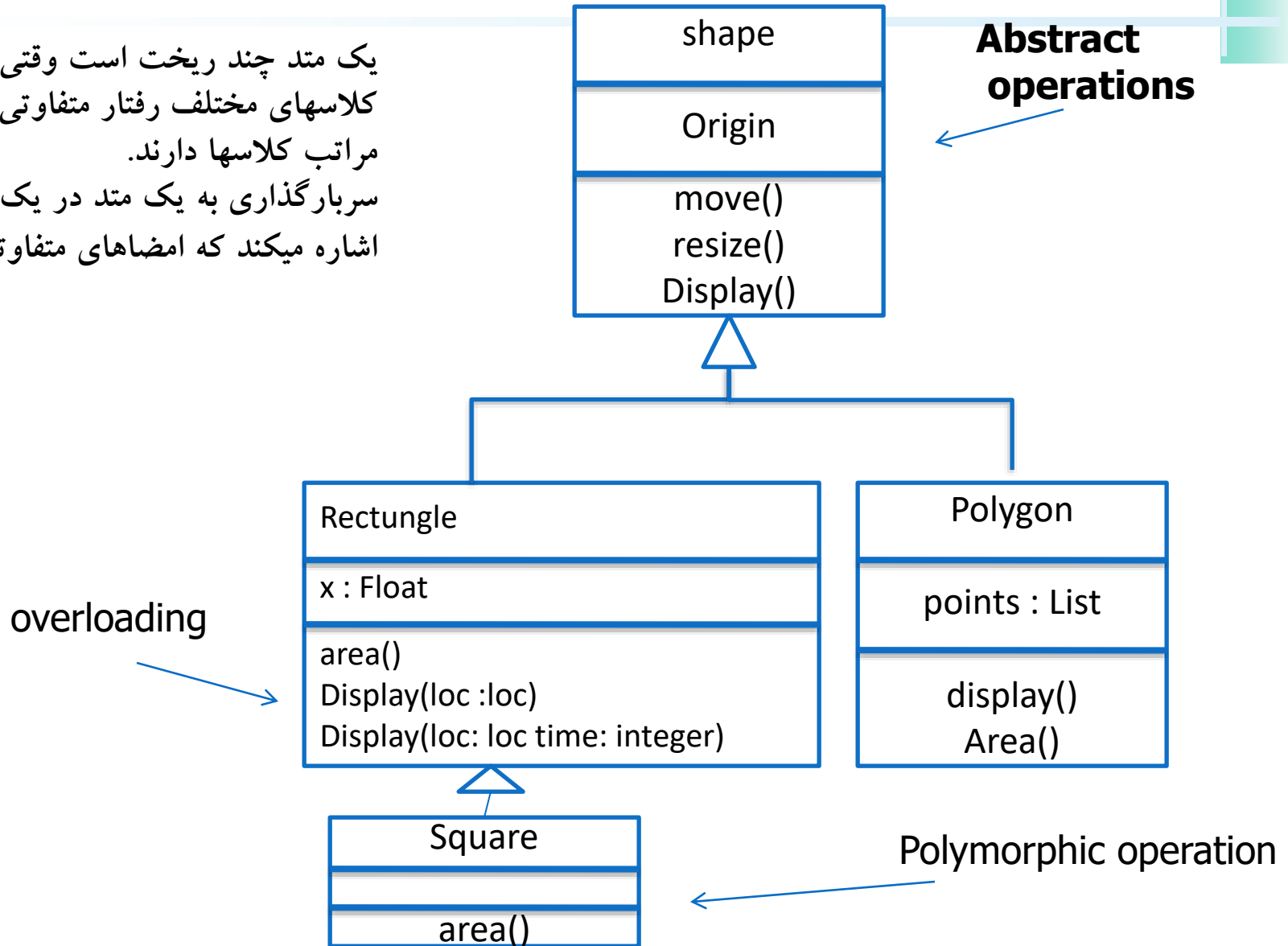


Generalization/specialization



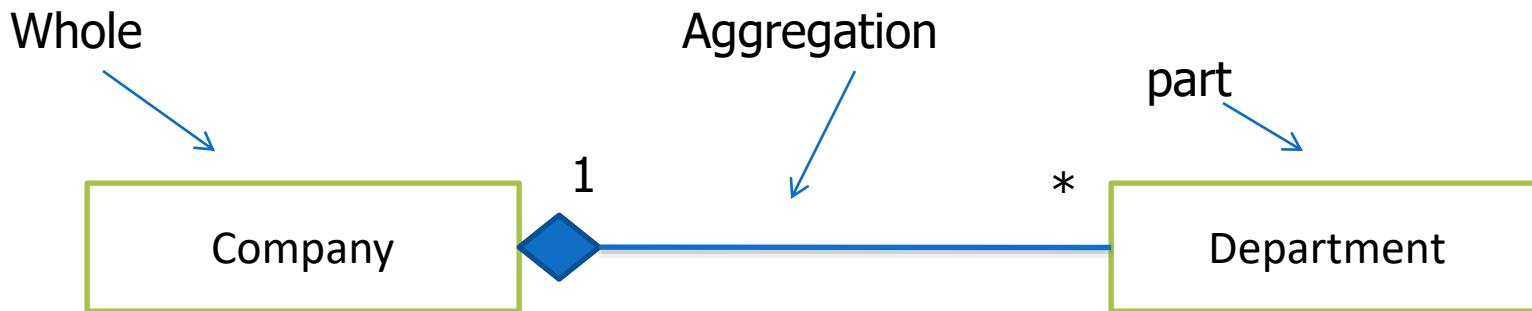
Polymorphism and Overloading

یک متد چند ریخت است وقتی در کلاسهای مختلف رفتار متفاوتی در سلسله مراتب کلاسها دارند. سربارگذاری به یک متد در یک کلاس اشاره میکند که امضاهای متفاوتی دارند.



رابطه تجمعی (Aggregation Relationship)

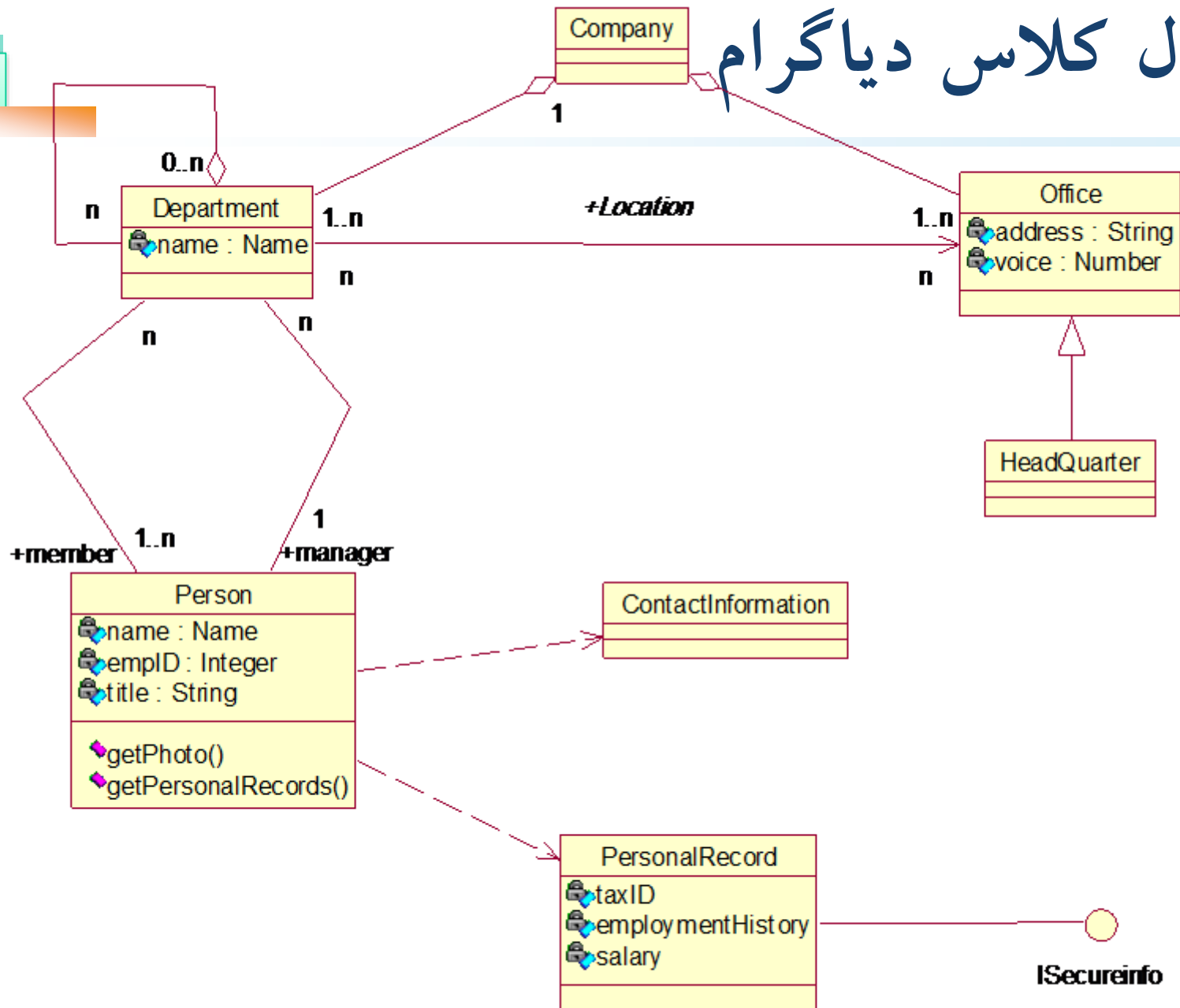
- معنایی : این رابطه کل (whole) را از بخش (part) جدا می کند. aggregation یک رابطه "has - a" است - یک شی از کل شامل اشیائی از بخش است.
- نماد:



نمودار کلاس (class diagram)

- گرافی از کلاس ها که توسط روابط مختلف به هم متصل شده اند، کلاس دیاگرام نامیده می شود.
- نمودار کلاس جهت نمایش دید استاتیک سیستم به کار می رود - مدل سازی **vocabulary** و **collaborations**
- نمودارهای کلاس توصیفی هستند - این نمودارها برای مصورسازی، تعیین، مستند سازی و ساخت سیستم های قابل اجرا از طریق **forward engineering** مفید هستند.
- ممکن است بیش از یک کلاس دیاگرام وجود داشته باشد
- یک کلاس دیاگرام ممکن است به چند **sub-class diagram** تجزیه شود.
- کلاس دیاگرام ها برای ایجاد **component diagram** ها و **deployment diagram** ها مفید هستند.

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 :

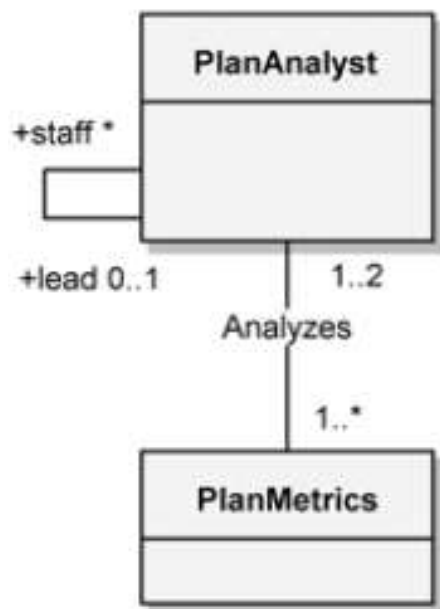
Visibility attributeName : Type [multiplicity] = ■
DefaultValue {property string}

■ عملیات: فرمت عملیات

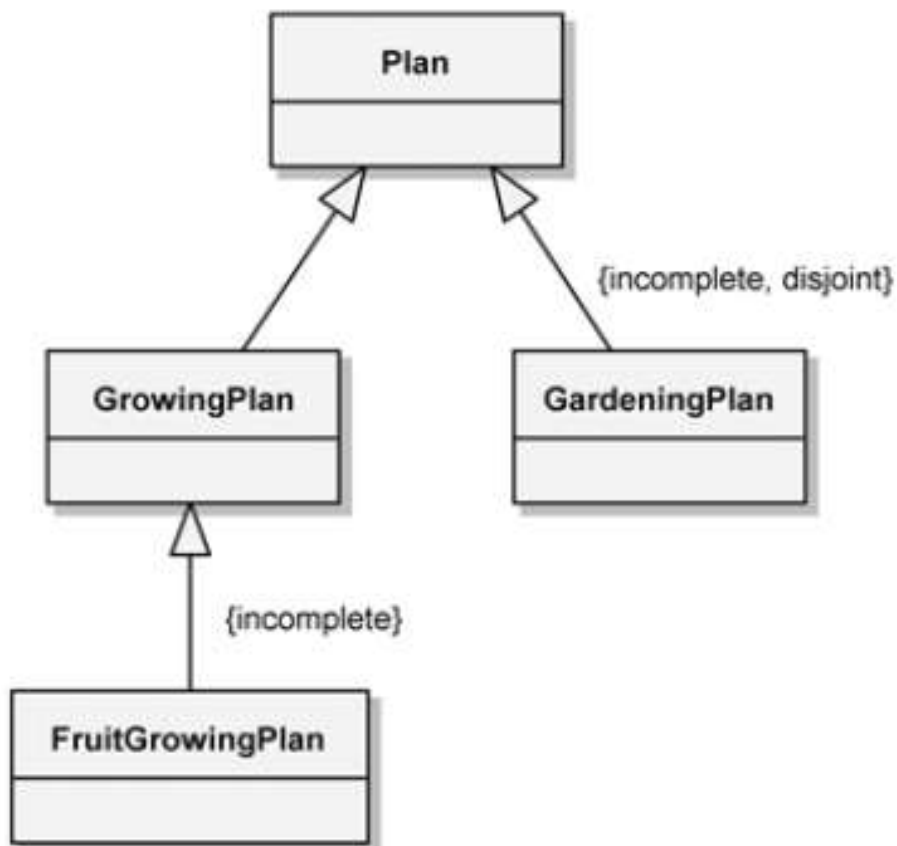
Visibility operationName (parameterName : Type) : ■
ReturnType {property string}

روابط کلاس ها

ASSOCIATION

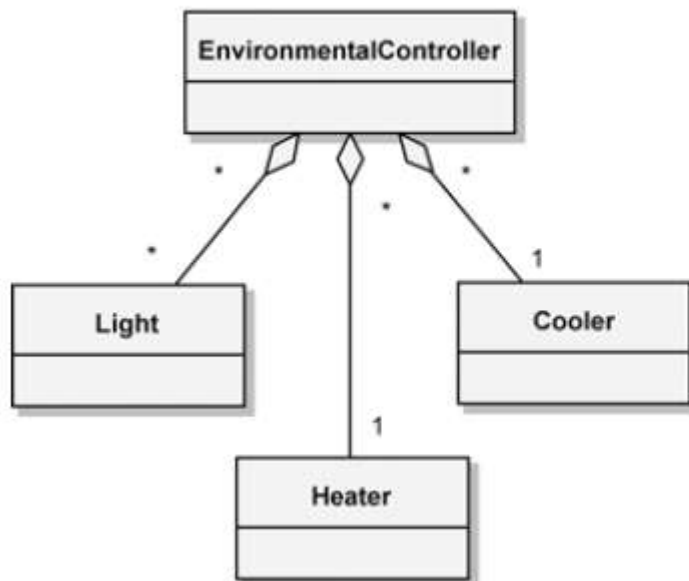


GENERALIZATION

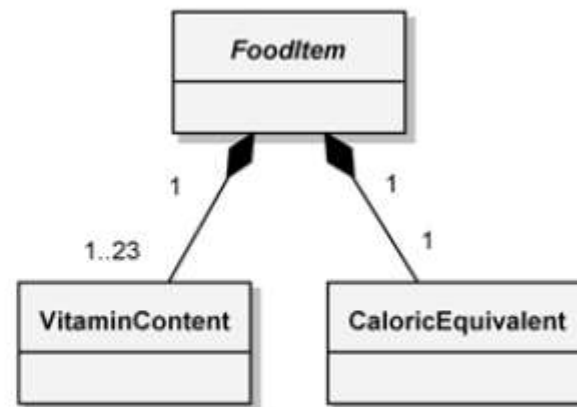


روابط کلاس ها (ادامه...)

AGGREGATION



COMPOSITION



قابل رویت بودن کلاس ها

■ در C++ اعضا می توانند:

■ **Public** : توسط همه قابل دسترس باشند.

■ **Protected** : فقط توسط زیرکلاس ها، friendها یا خود کلاس قابل دسترس هستند.

■ **Private**: فقط توسط خود کلاس یا friendهایش 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 صفت ها و مقادیر صفت های شی مجزا می کند.
- اشیاء از طریق **link** ها با یکدیگر در ارتباطند.
- یک **link**، نمونه ای از یک **association** است.

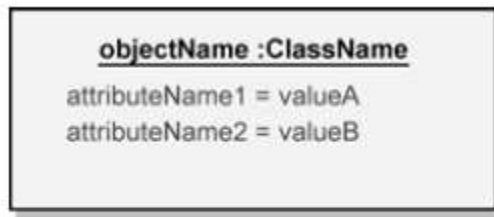
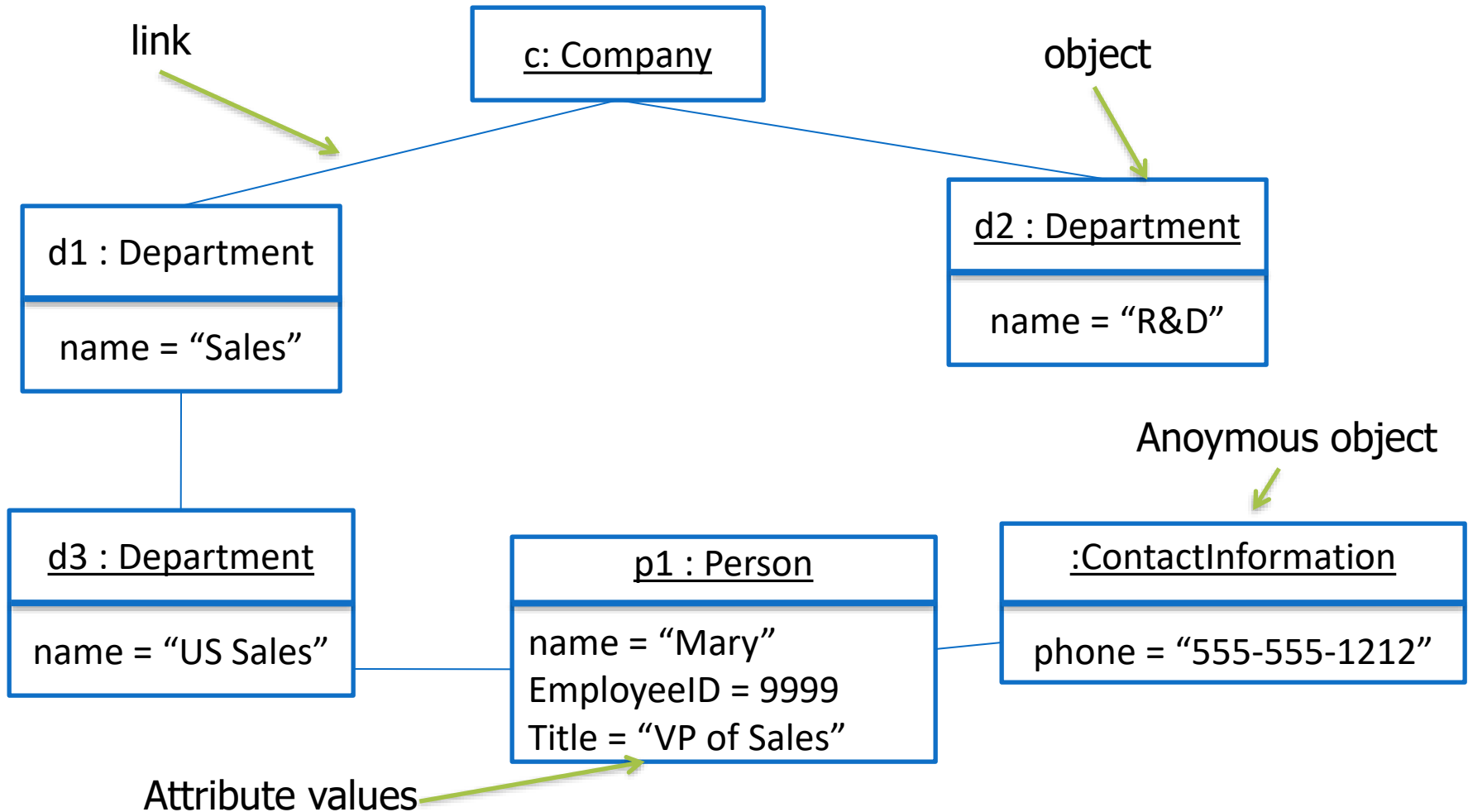


Figure 5–75 A Generic Object Icon

مثال نمودار شی



Package diagram

■ **Package** 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 نام ها 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 :

■ **Public (+)** : عناصر برای پکیج های دیگر قابل رویت هستند.

■ **Private (-)** : عناصر به طور کامل مخفی هستند.

Package diagram

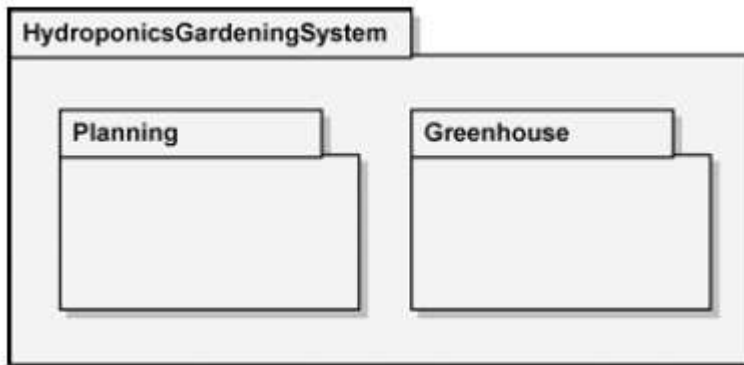
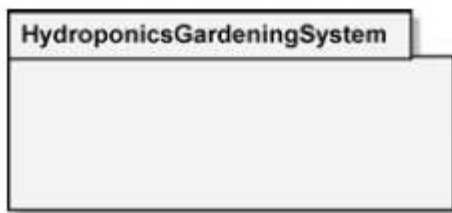
:Package stereotypes ■

- **<<framework>>**: پکیجی که شامل عناصر مدلی است که مشخص کننده معماری قابل استفاده مجدد هستند.
- **<<modelLibrary>>**: پکیجی که شامل عناصری است که توسط پکیج های دیگر استفاده می شوند.

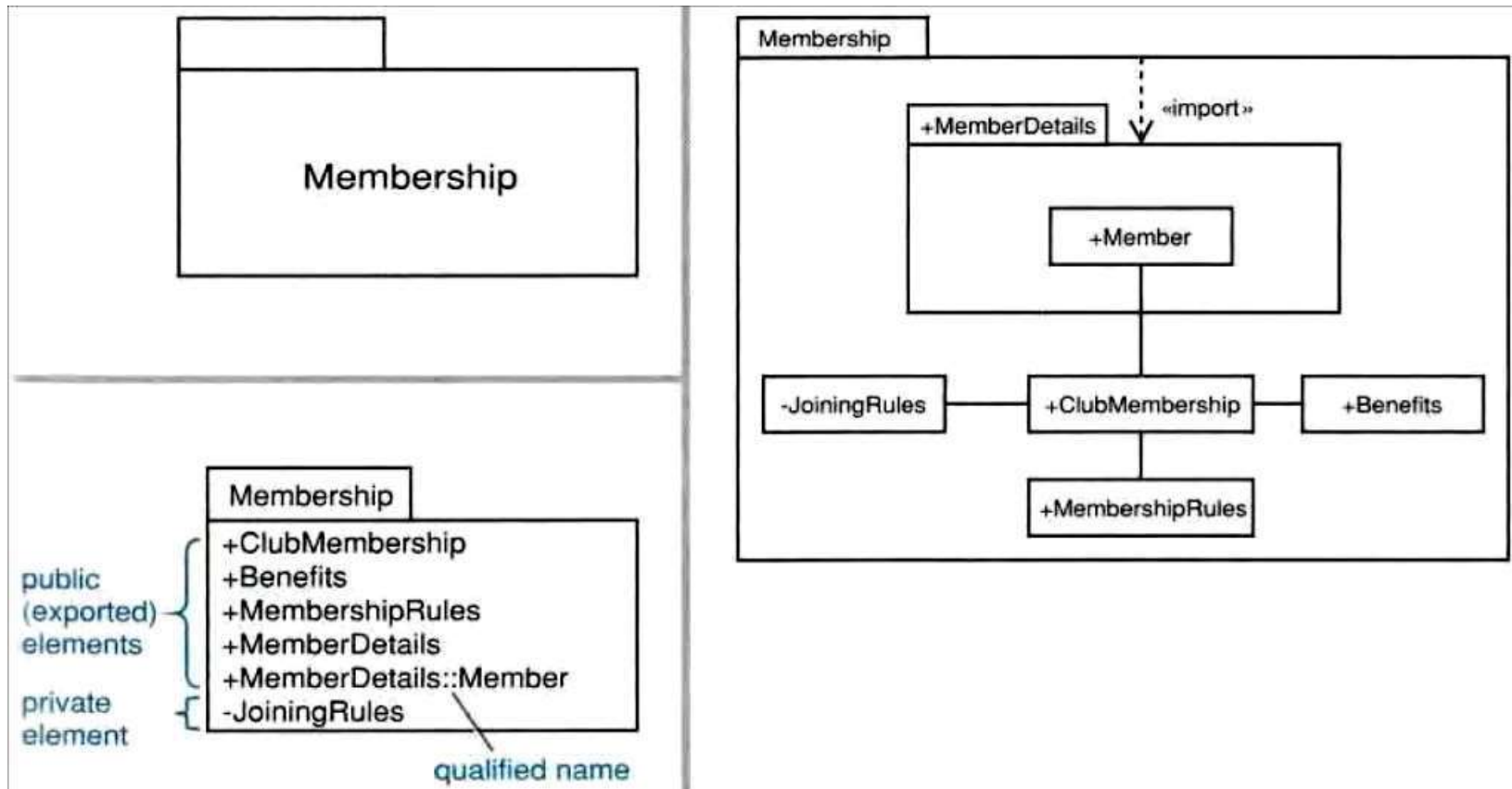
Library::Users::Librarian ■

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.



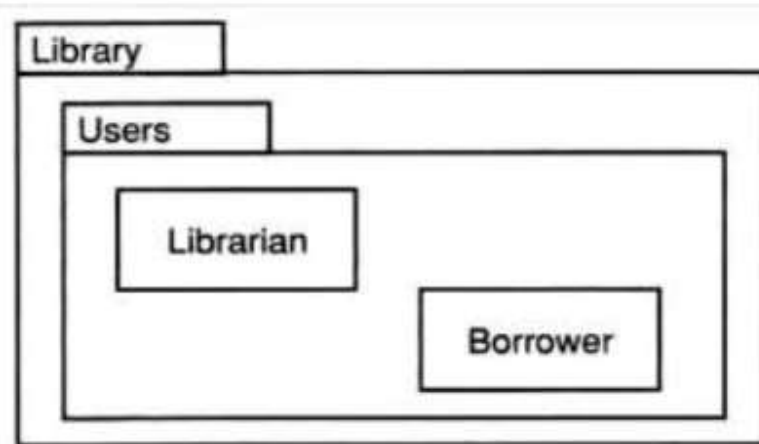
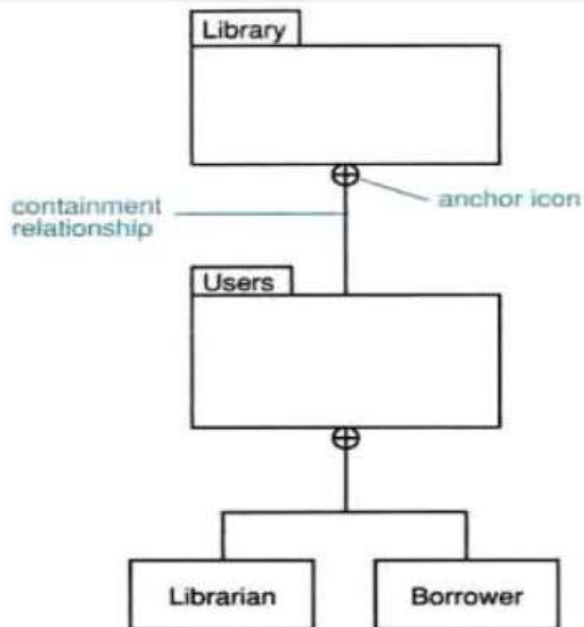
Package diagram (example)



Package های تو در تو

- پکیج داخلی می تواند تمام اعضای عمومی پکیج خارجی را ببیند.

- پکیج خارجی نمی تواند هیچکدام از اعضای پکیج داخلی را ببیند مگر اینکه یک وابستگی صریح با آنها داشته باشد (معمولاً `<<import>>` یا `<<access>>`)

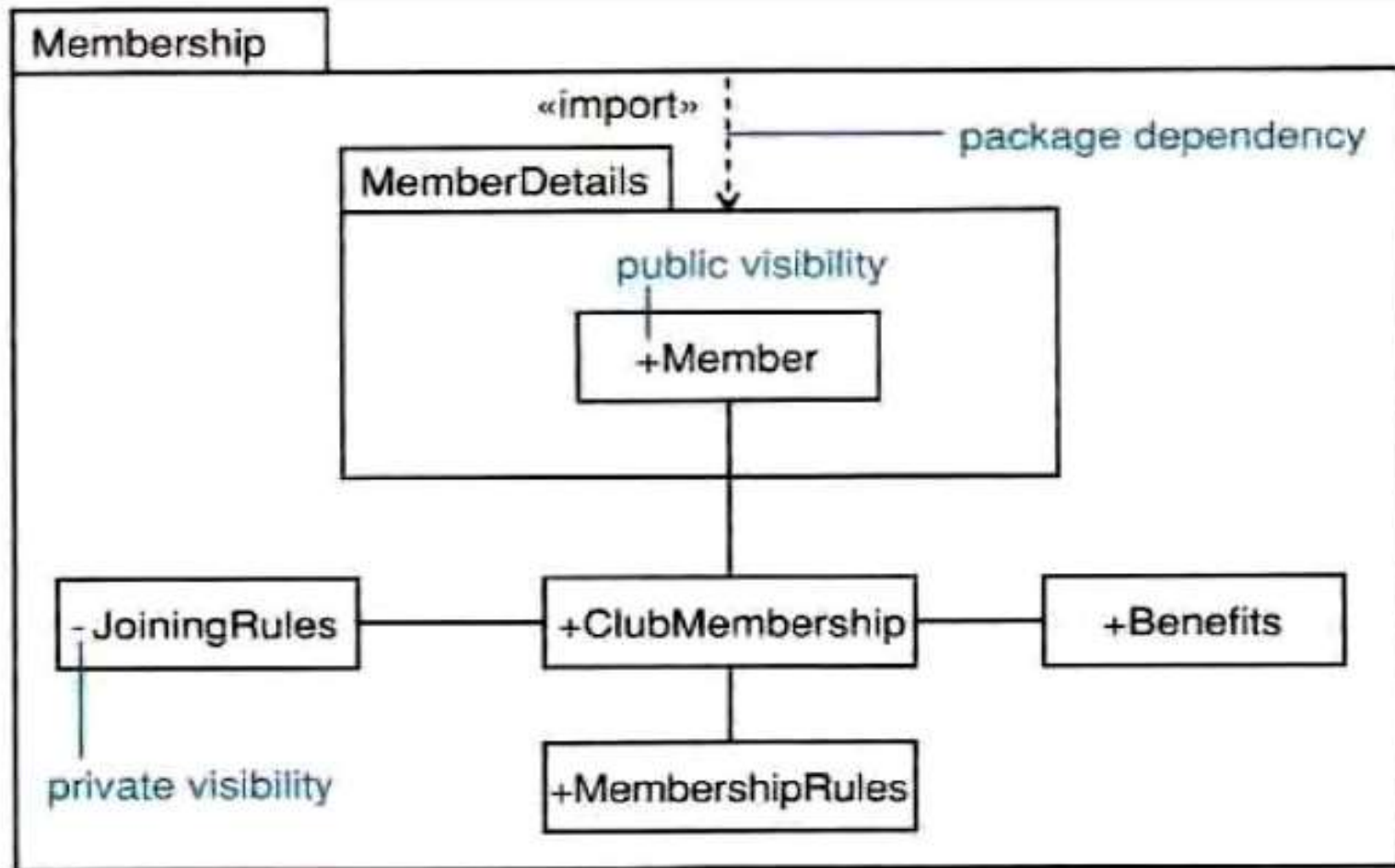


Library::Users::Librarian

وابستگی package ها

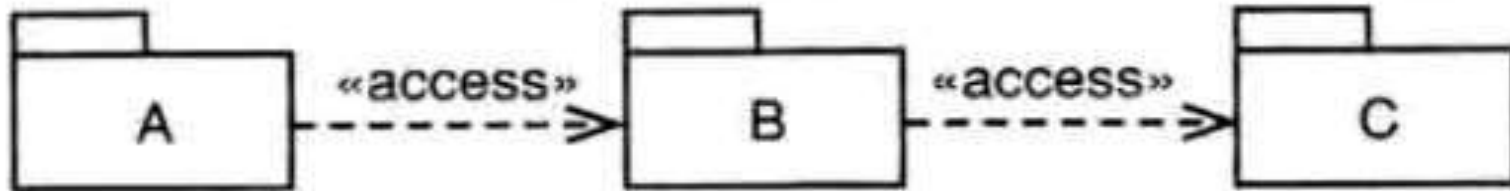
- رابطه وابستگی بین package ها نشان می دهد که client package به supplier package وابسته است.
- **<<use>> (default)** : یک عنصر در client package از یک عنصر عمومی در supplier package استفاده می کند.
- **<<import>>** : عناصر عمومی supplier namespace به عنوان عناصر عمومی client namespace اضافه می شوند.
عناصر در کلاینت می توانند به عناصر عمومی supplier دسترسی پیدا کنند.
- **<<access>>** : عناصر عمومی supplier namespace به عنوان عناصر خصوصی client namespace اضافه می شوند.
عناصر در کلاینت میتوانند به همه عناصر عمومی در supplier دسترسی داشته باشند.

وابستگی 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.
- `<<import>>` تعدی است اما `<<access>>` تعدی نیست.



Package Generalization

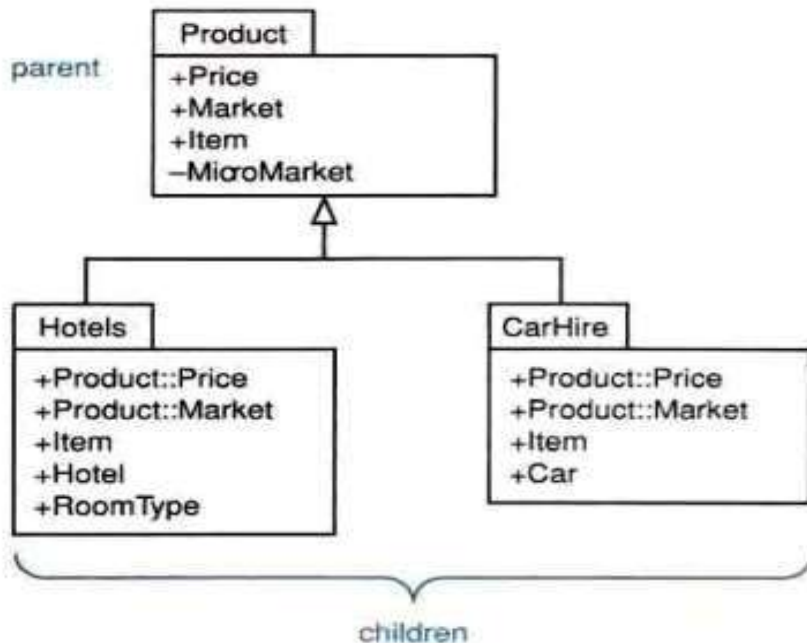
■ شبیه class generalization است.

■ Package های فرزند :

■ عناصر را از پکیج 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 ...
- Package های خوب highly و loosely coupled و cohesive هستند.
- به این معنی که : باید تعاملات بیشتری بین اجزای یک package وجود داشته باشد و تعاملات بین package ها کمتر باشند.

پیدا کردن package ها

- زیرسیستم ها را بررسی کنید.

- بررسی کلاس ها برای :

- کلاسترهای منسجمی از کلاسهای به هم وابسته

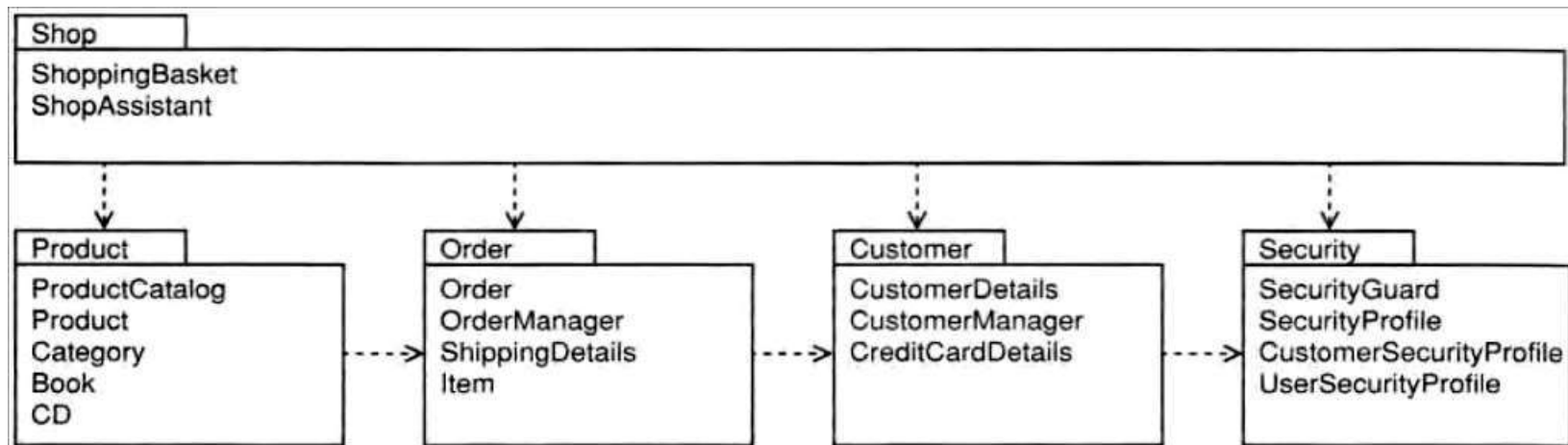
- ارث بری

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

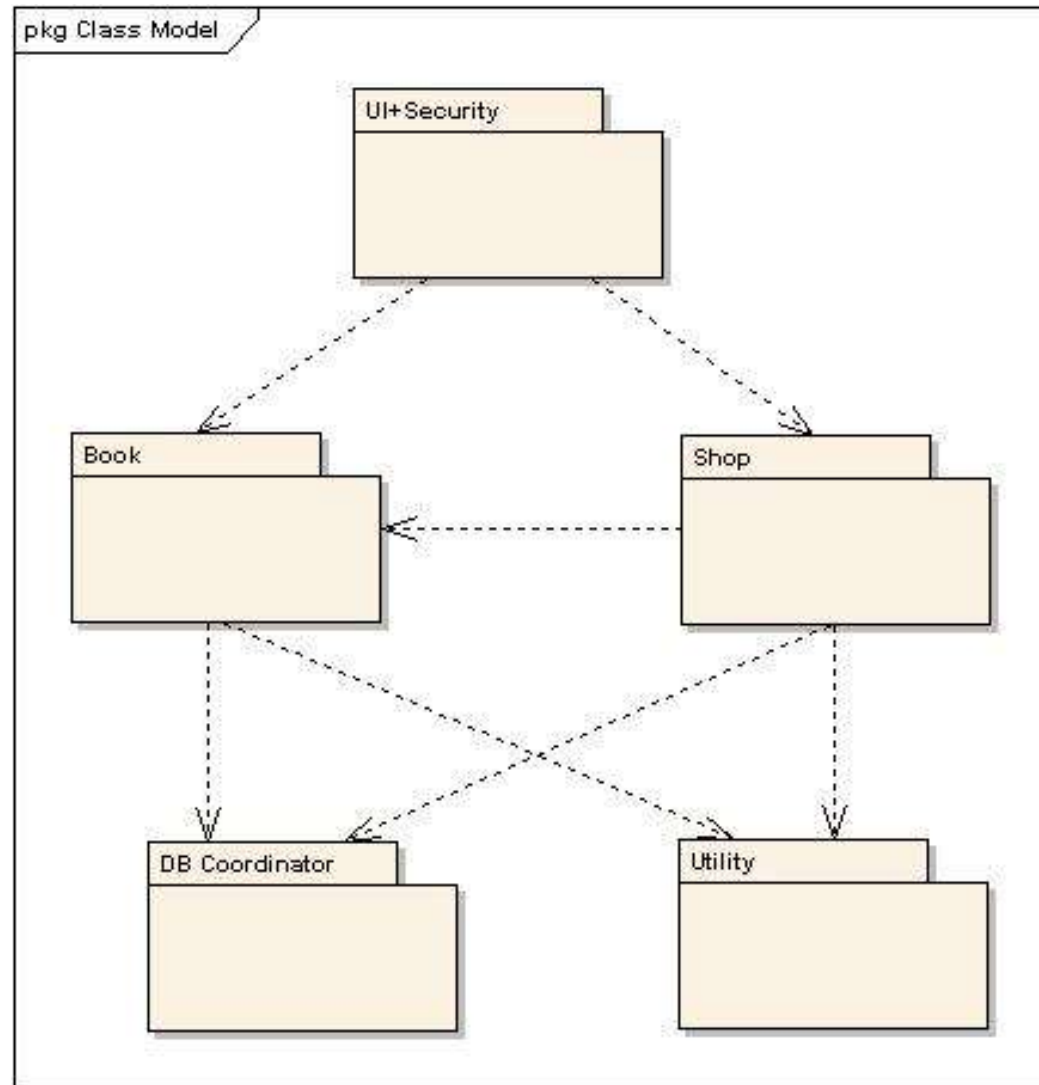
- بررسی use case ها :

- 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

- دیاگرام فعالیت فراهم کننده نمایش تصویری جریانی از فعالیت ها است. خواه آنها در یک سیستم، تجارت، جریان کاری یا فرایند دیگری باشند.
- این دیاگرام ها روی فعالیت ها و کسی (یا چیزی) که در انجام این فعالیت ها مسئول است تمرکز می کنند.
- اجزای یک دیاگرام فعالیت :
- ۱- گره ها

- **Action nodes** : واحدها در فعالیت
- **Control nodes** : جریان را از طریق فعالیت کنترل می کنند.
- **Object nodes**: نشان دهنده اشیایی هستند که در فعالیت استفاده می شوند.
- ۲- لبه ها

- **Object flows** : جریانی از اشیا در فعالیت را نشان می دهد.
- **Control flows** : جریانی از کنترل ها را در فعالیت نشان می دهد.
- سه نوع گره کنترلی (**control node**) وجود دارد:
- **Initial and final** (گره های پایانی دو نوع هستند : **activity final**, **flow final**)
- **Decision and merge**
- **Fork and join**

Activity diagrams (ادامه..)

Action



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

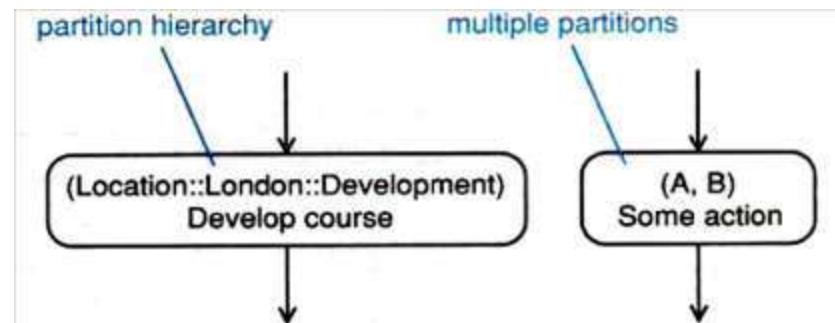
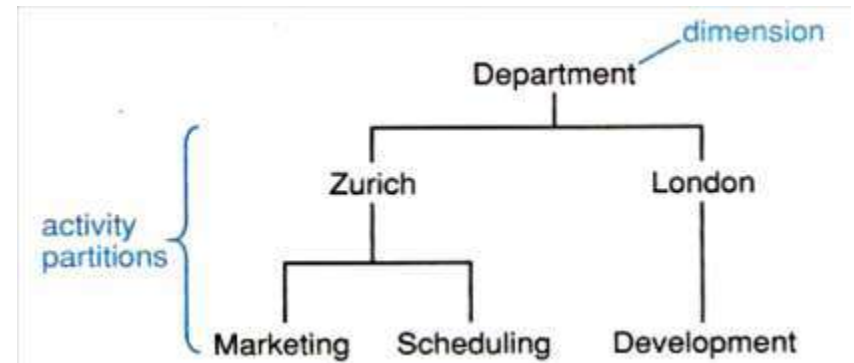
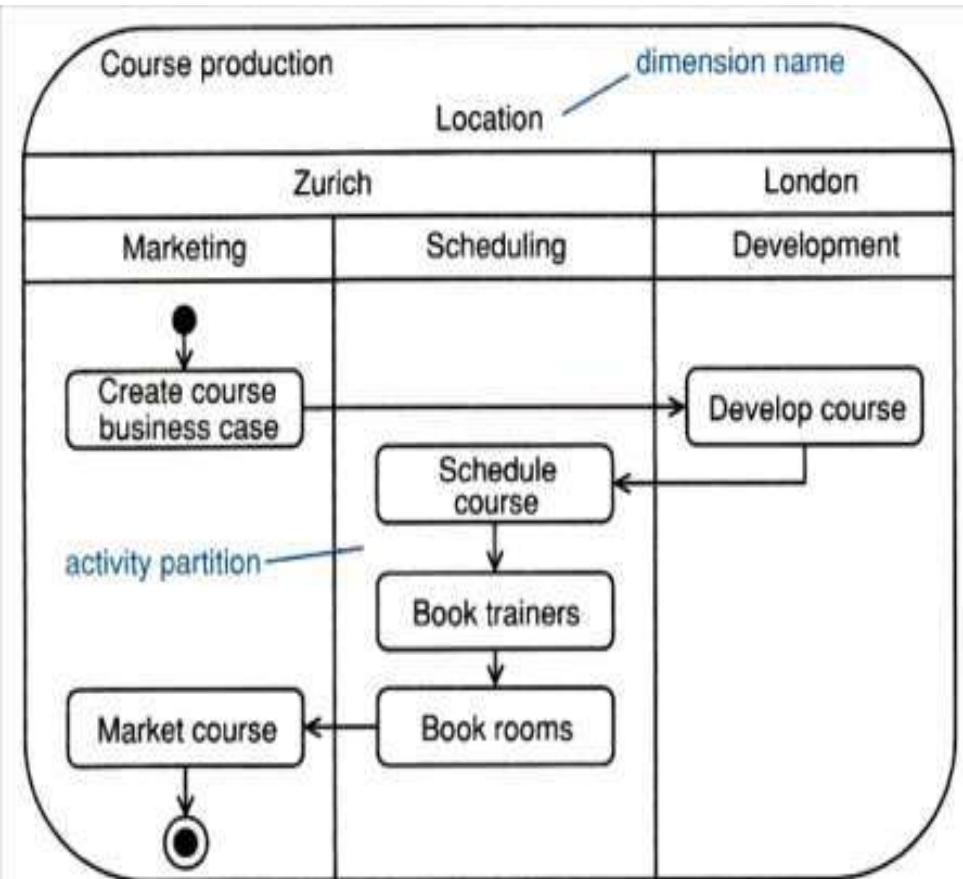
Starting and Stopping

- 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.



activity diagram در Call

■ گره عملیات :Call

■ Call an activity

■ Call a behavior

■ Call an operation

Create Order
rh

call an activity

Close Order

call a behavior

getBalance():double
(Account::)

operation name

class name
(optional)

Get balance
(Account::getBalance():double)

node name



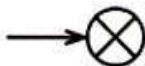
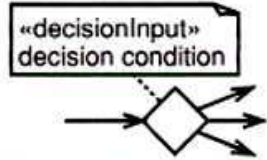

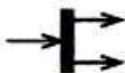
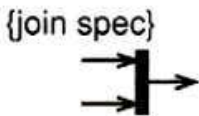
operation name
(optional)

call an
operation

```
if self.balance <= 0:  
    self.status = 'INCREDIT'  
else  
    self.status = 'OVERDRAWN'
```

programming
language
(e.g., Python)

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
	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 node	Synchronizes multiple concurrent flows May optionally have a join specification to modify its semantics

Final nodes

گره های merge و decision (ادامه...)

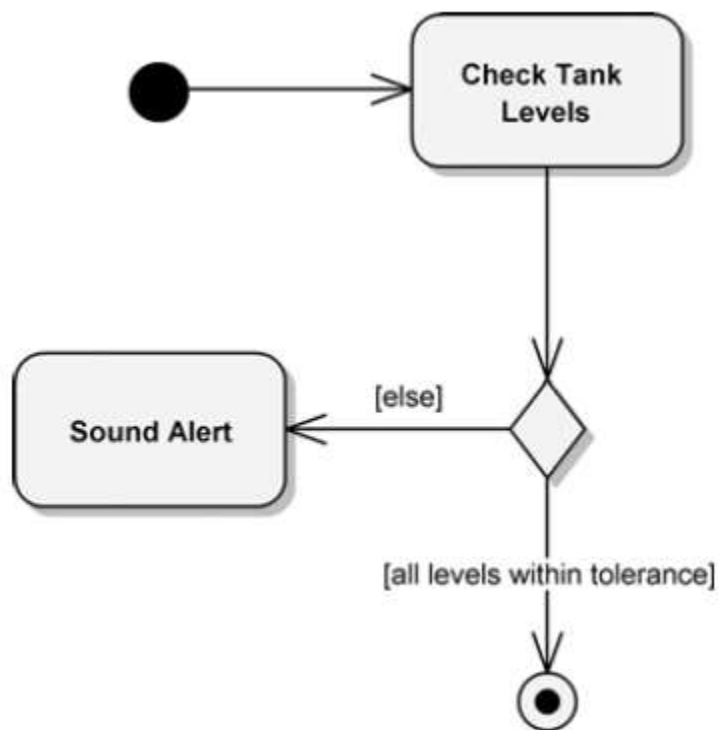


Figure 5–29 A Decision Node

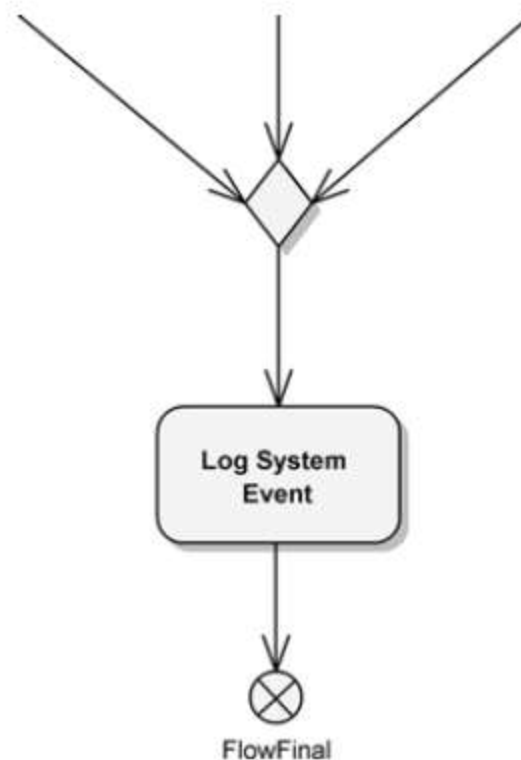
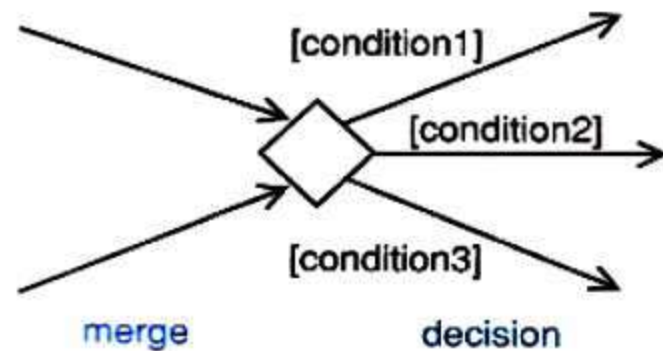
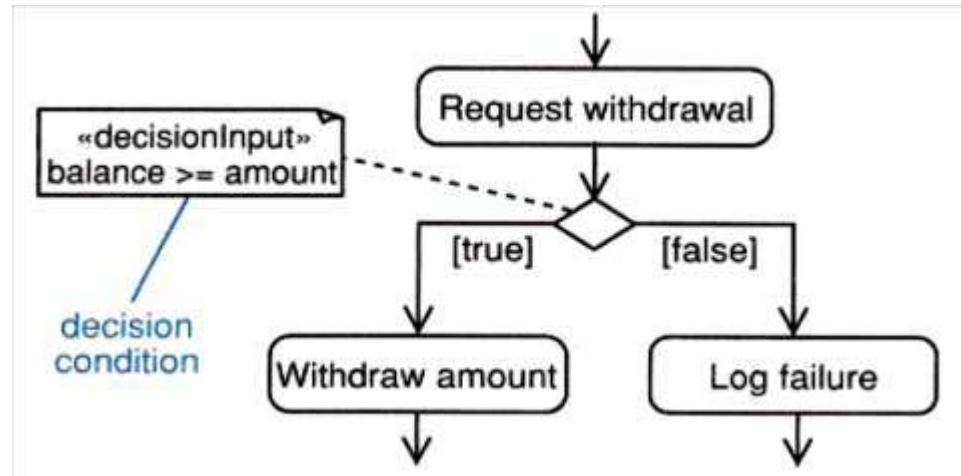
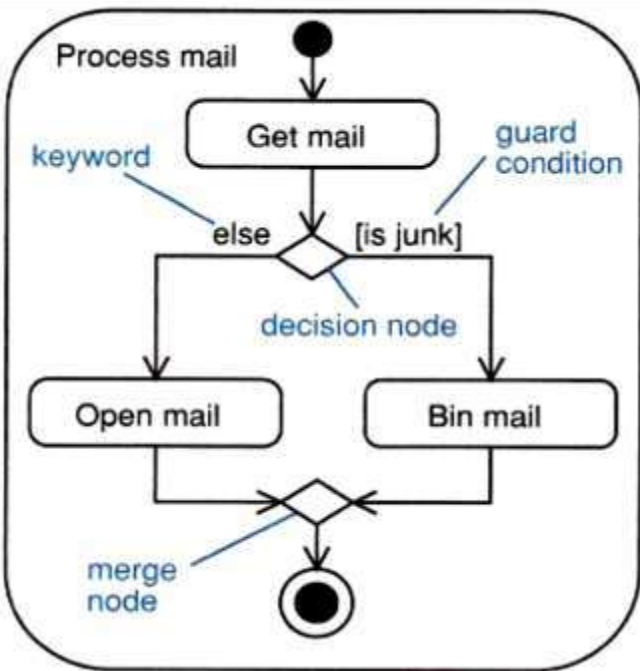
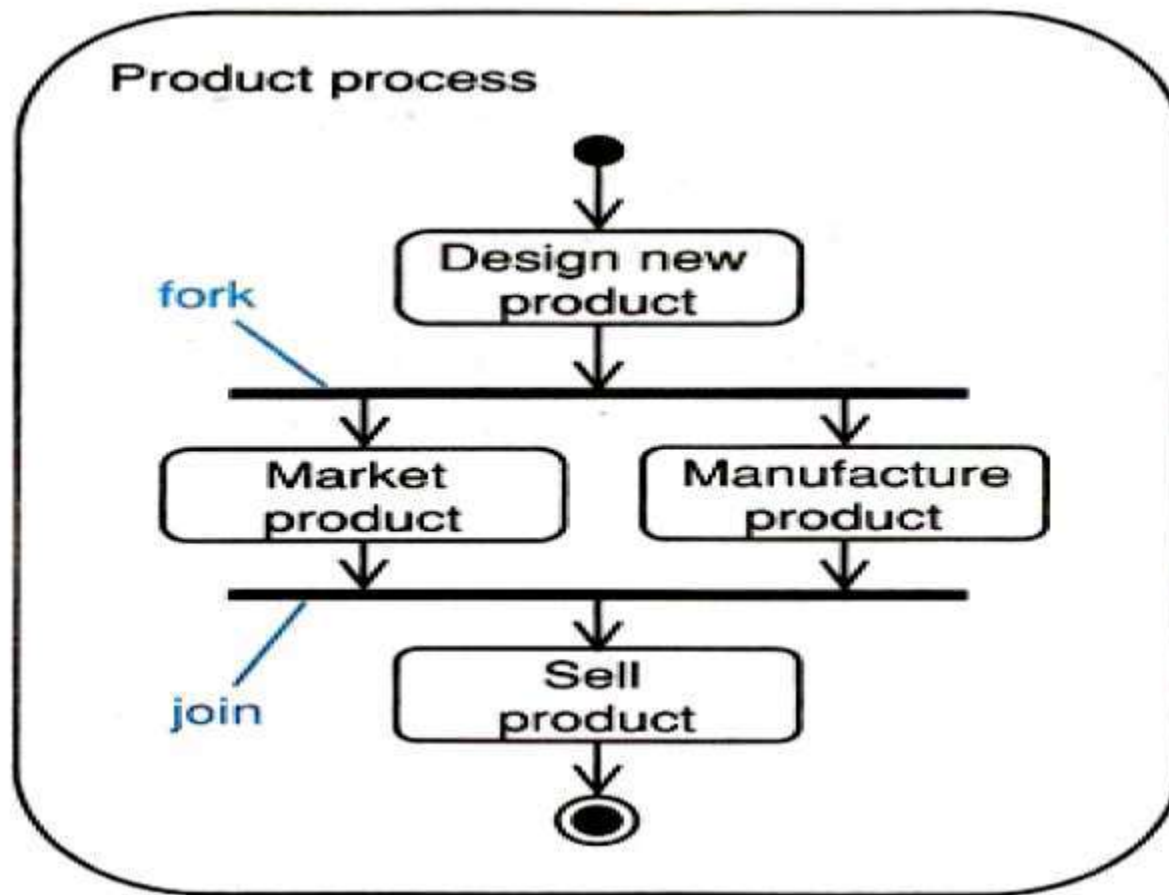


Figure 5–30 A Merge Node with a Flow Final Node

گره های decision و merge (ادامه...)

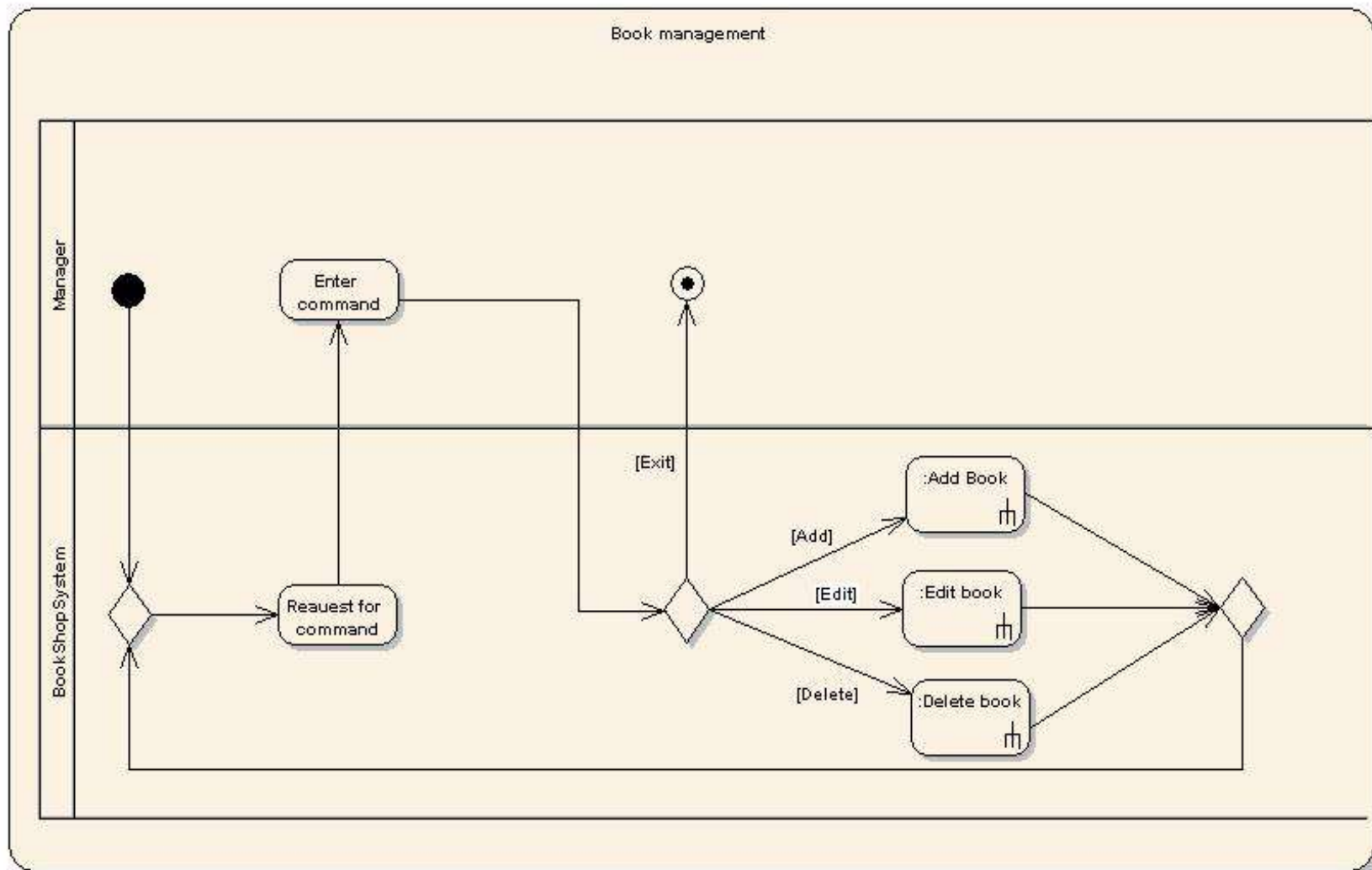


گره های کنترلی : Fork و Join



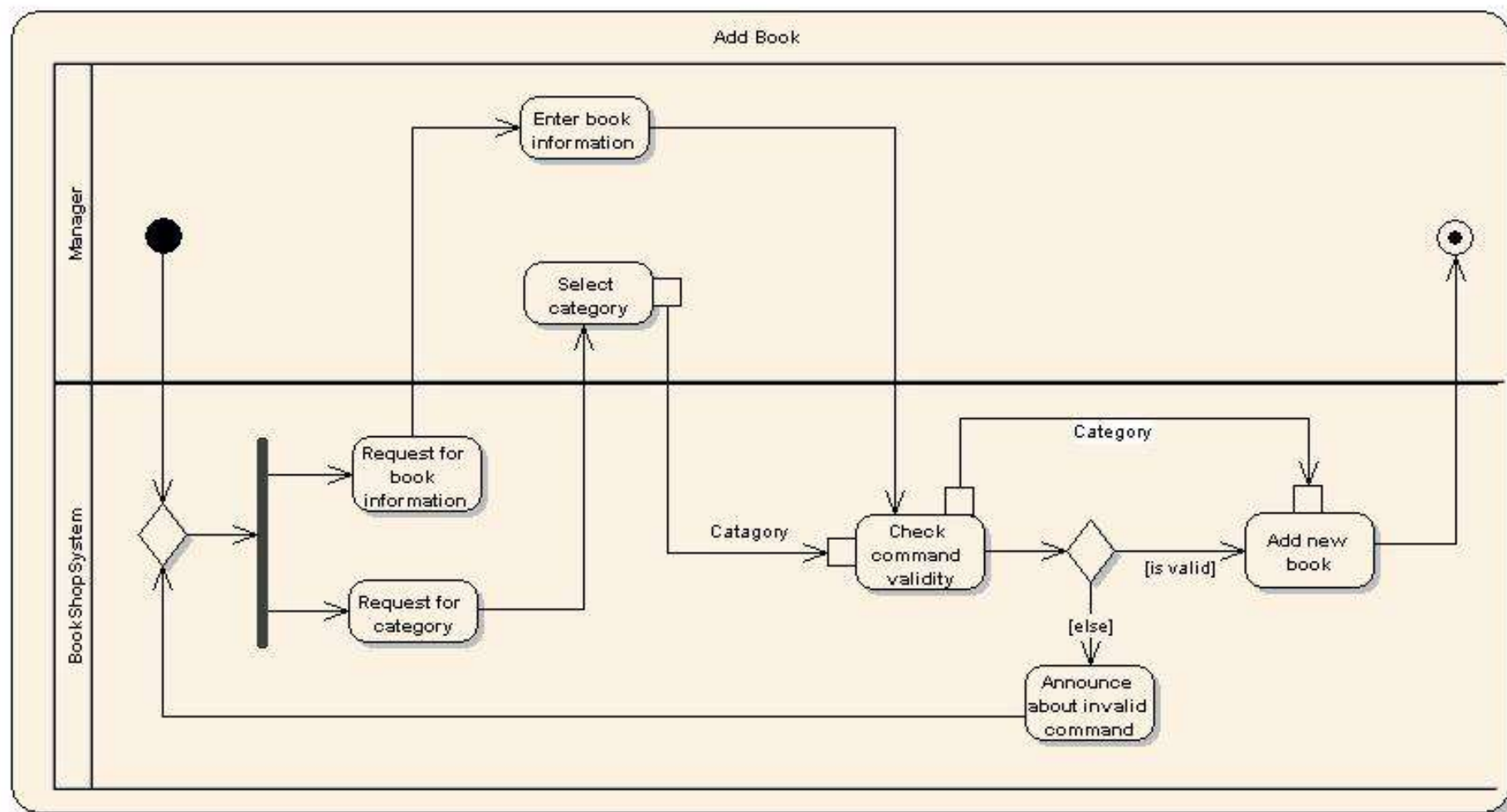
مثالی از activity diagram

act Book Management

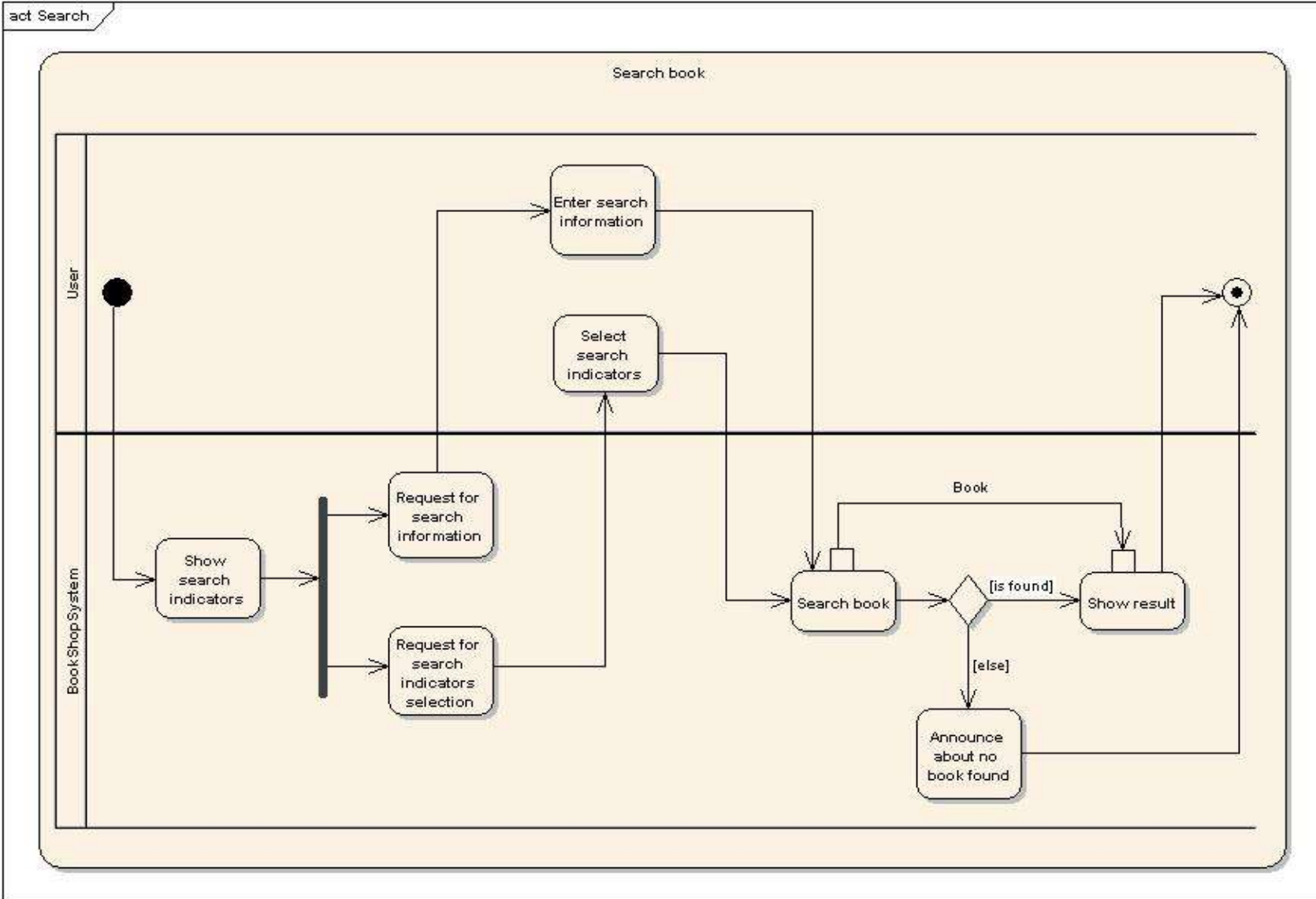


مثالی از activity diagram

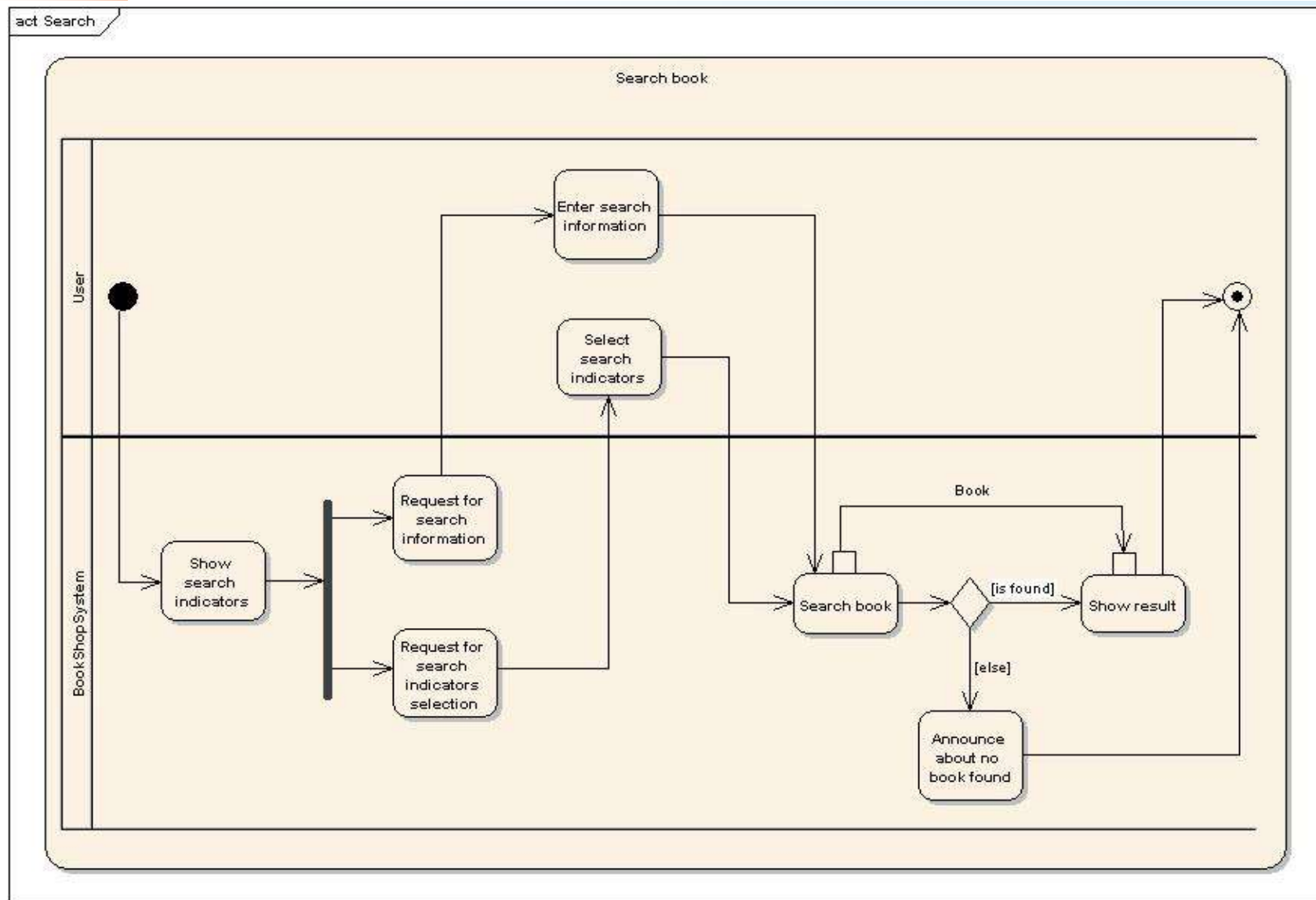
act Add Book



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

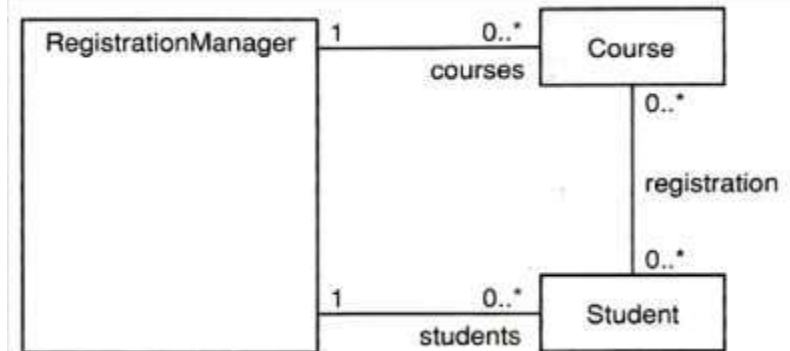
Secondary actors:
None.

Preconditions:
1. The Registrar has logged on to the system.

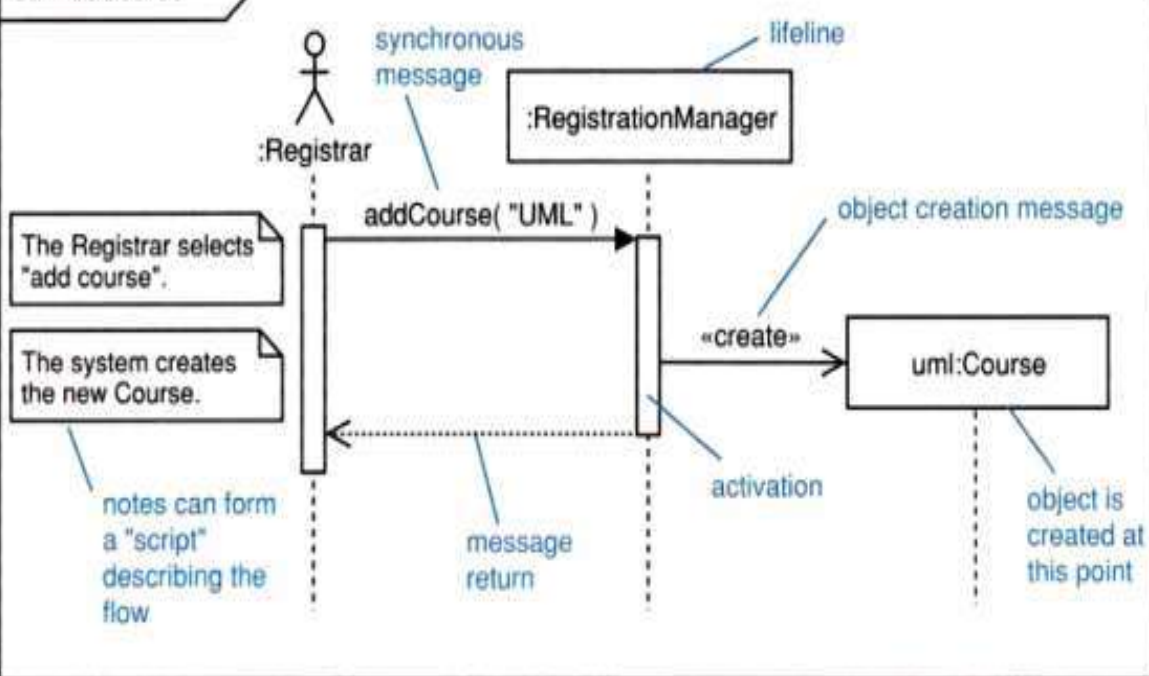
Main flow:
1. The Registrar selects "add course".
2. The Registrar enters the name of the new course.
3. The system creates the new course.

Postconditions:
1. A new course has been added to the system.

Alternative flows:
CourseAlreadyExists



sd AddCourse



Sequence diagrams (cont.) – مثال ٢

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:

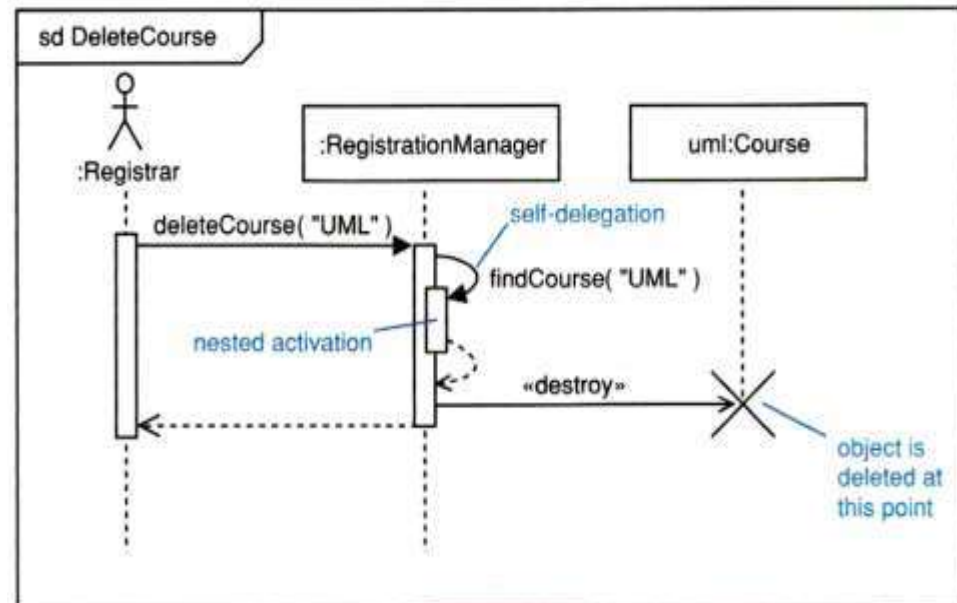
1. The Registrar selects "delete course".
2. The Registrar enters the name of the course.
3. The system deletes the course.

Postconditions:

1. A course has been removed from the system.

Alternative flows:

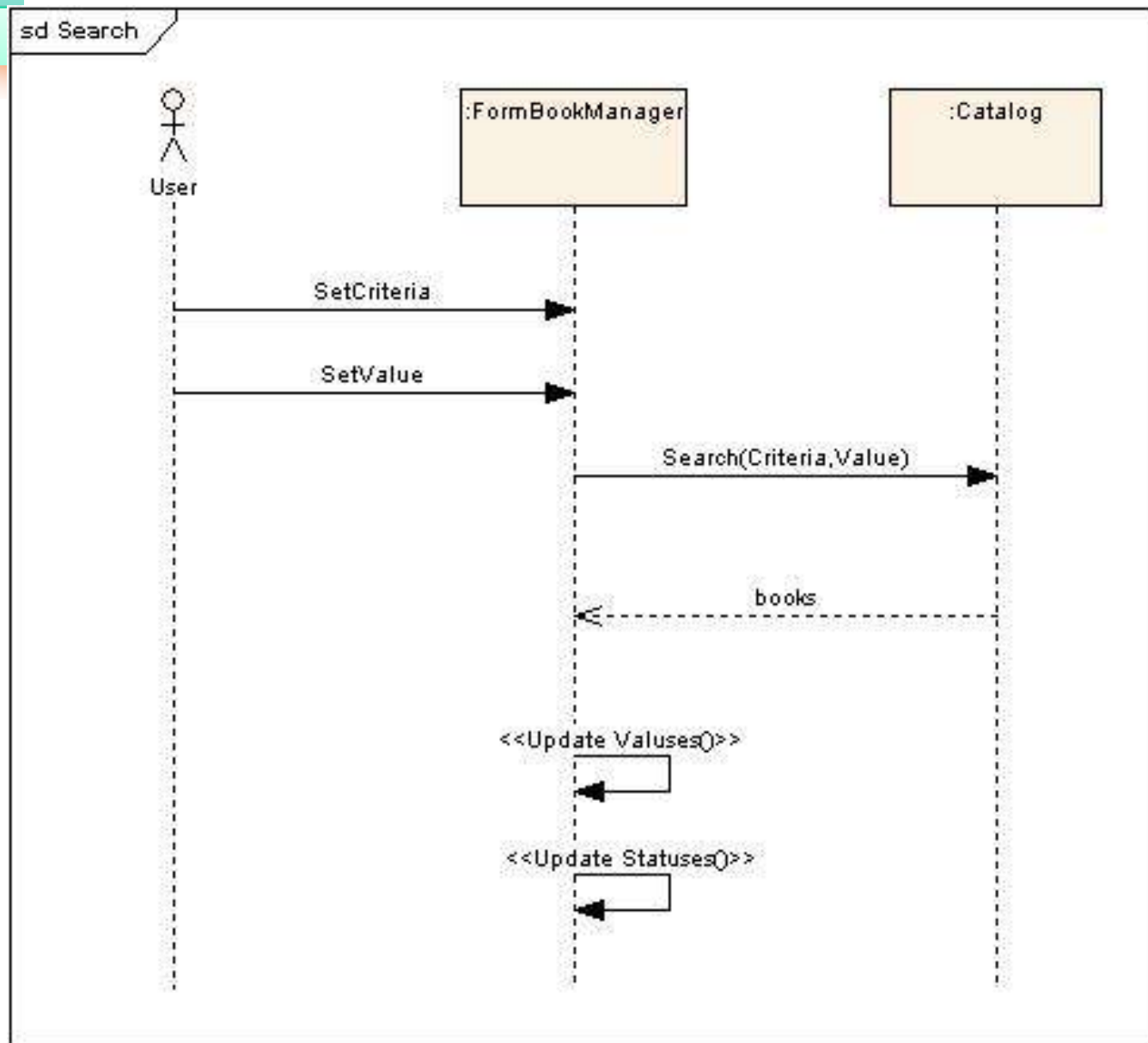
CourseDoesNotExist



مثال ۳

مورد کاربرد: جستجوی کتاب
شماره: ۳
توصیف اجمالی: کاربر اطلاعات جستجو را وارد می کند و سامانه کتاب هایی را که با این اطلاعات همخوانی دارند، به کاربر نشان می دهد.
عامل اصلی: کاربر
عامل فرعی: ندارد
شرایط اولیه: ندارد
روند اصلی: <ol style="list-style-type: none"> ۴. این مورد کاربرد وقتی آغاز می شود که کاربر از سامانه، درخواست جستجوی کتاب کند. ۵. سامانه شاخصه های جستجوی کتاب را به کاربر نشان می دهد و اطلاعات جستجو لازم را جهت جستجو را از کاربر می - خواهد. ۶. کاربر اطلاعات کتاب مورد نظر خود را وارد می کند. ۷. سامانه اطلاعات وارد شده را با اطلاعات کتاب ها مقایسه می کند و موارد همخوانی را می یابد. ۸. اگر کتابی همخوان با اطلاعات وارد شده پیدا شد ۸.۱. سامانه کتاب های پیدا شده را به کاربر نشان می دهد. ۹. و گرنه ۹.۱. سامانه به کاربر اطلاع می دهد که کتابی با اطلاعات وارد شده پیدا نشد.
شرایط نهایی: ندارد
روند جایگزین: ندارد

مثال ٣



انواع پیام ها

- پیام سنکرون (synchronous message) : با یک خط پر و یک پیکان توپر نشان داده می شود. (operation call)
- 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

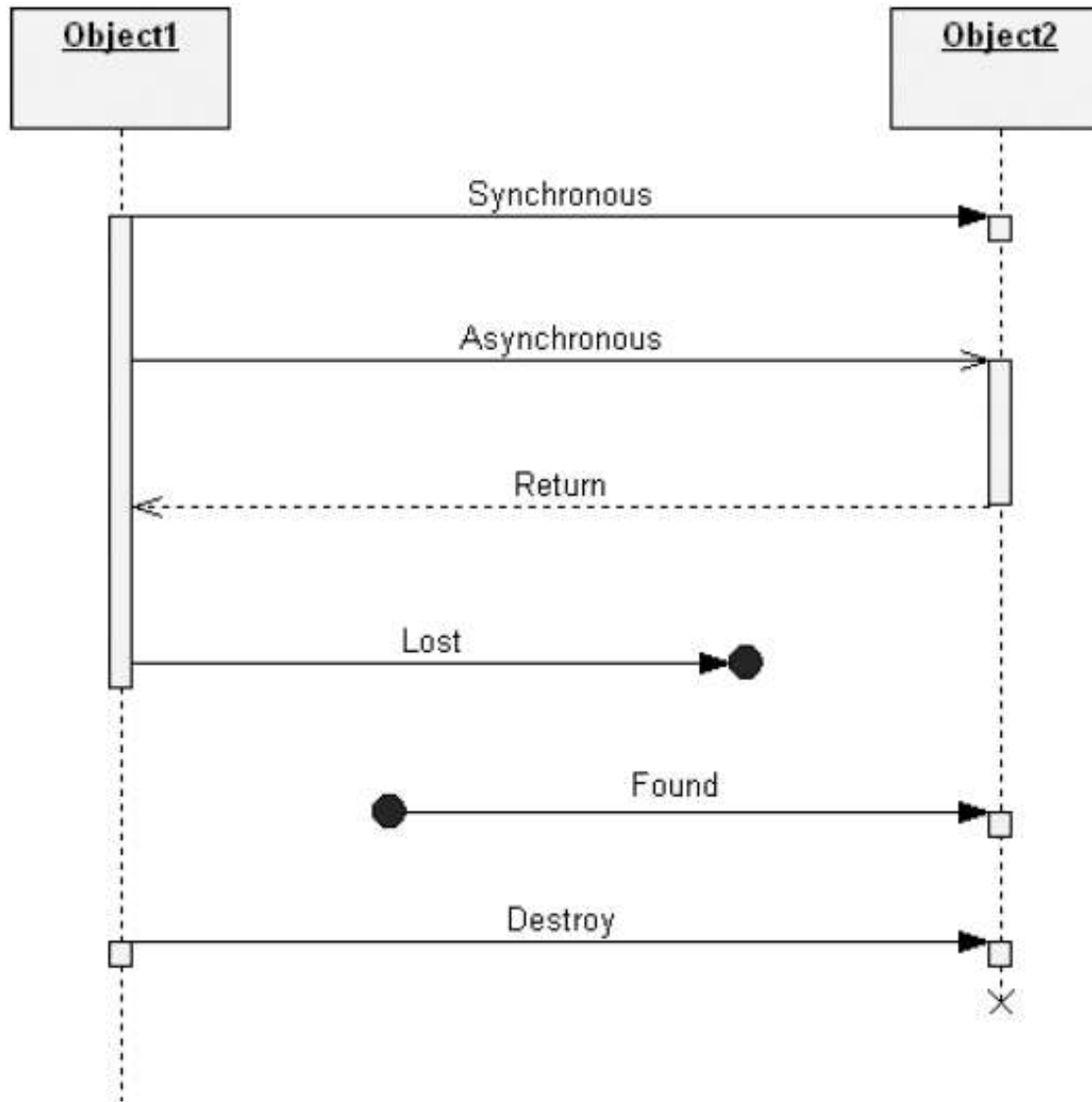
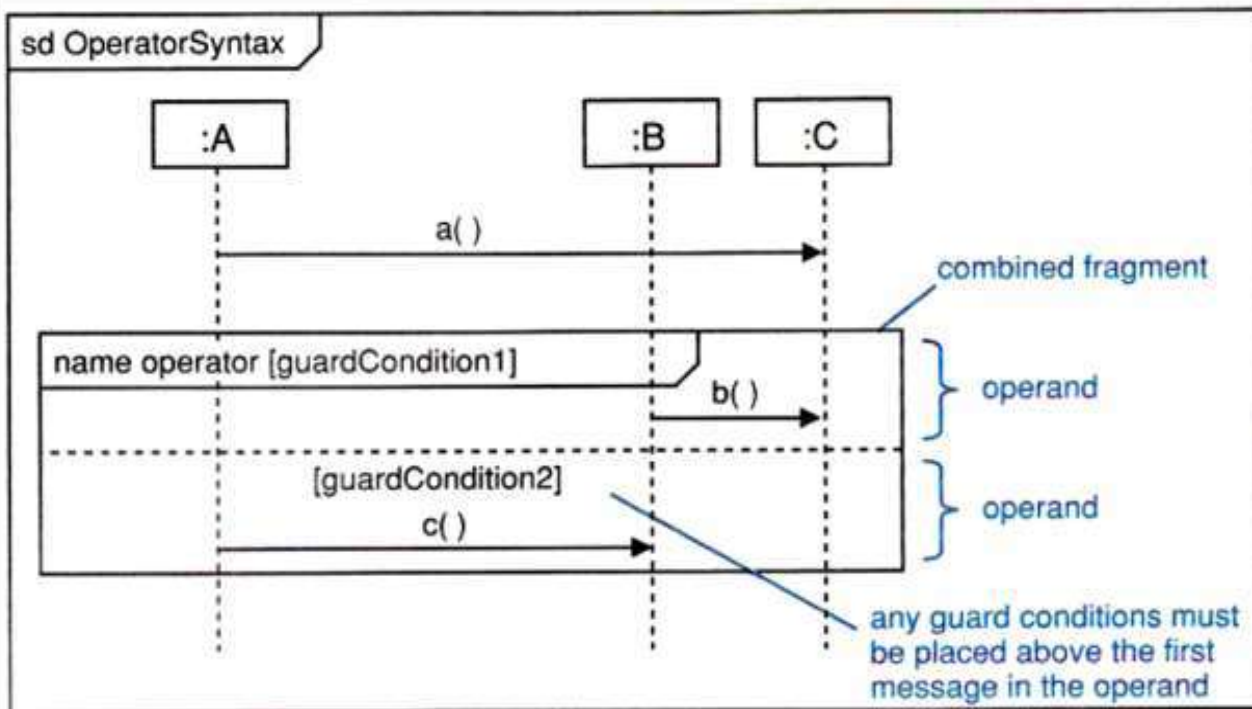


Figure 5-43 Notations for Types of Messages

قطعات ترکیبی

■ نواحی داخل یک sequence diagram با رفتارهای مختلف

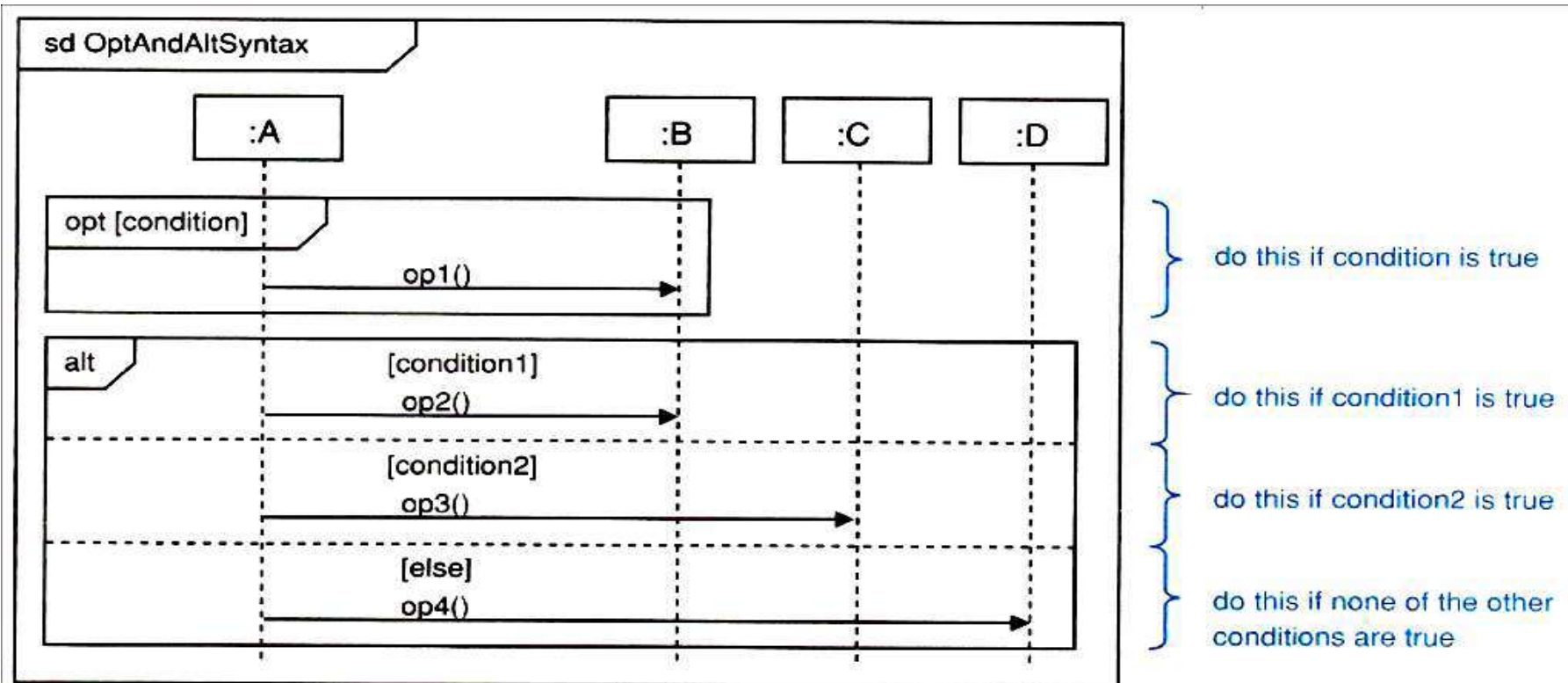
- Operator مشخص می کند که چگونه operand ها اجرا می شوند.
- Guard condition مشخص می کند که operand اجرا می شود یا نه.
- Operand شامل رفتار است.



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

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

■ **alt : operand** می شود که شرط آن برقرار است اجرا می شود.



قطعات ترکیبی: *opt* and *alt* Operators

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:

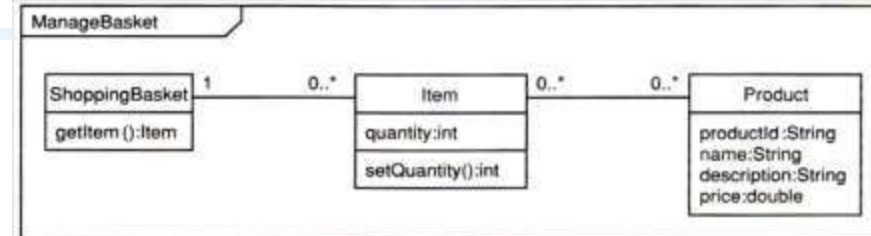
1. The use case starts when the Customer selects an item in the basket.
2. If the Customer selects "delete item"
 - 2.1 The system removes the item from the basket.
3. If the Customer types in a new quantity
 - 3.1 The system updates the quantity of the item in the basket.

Postconditions:

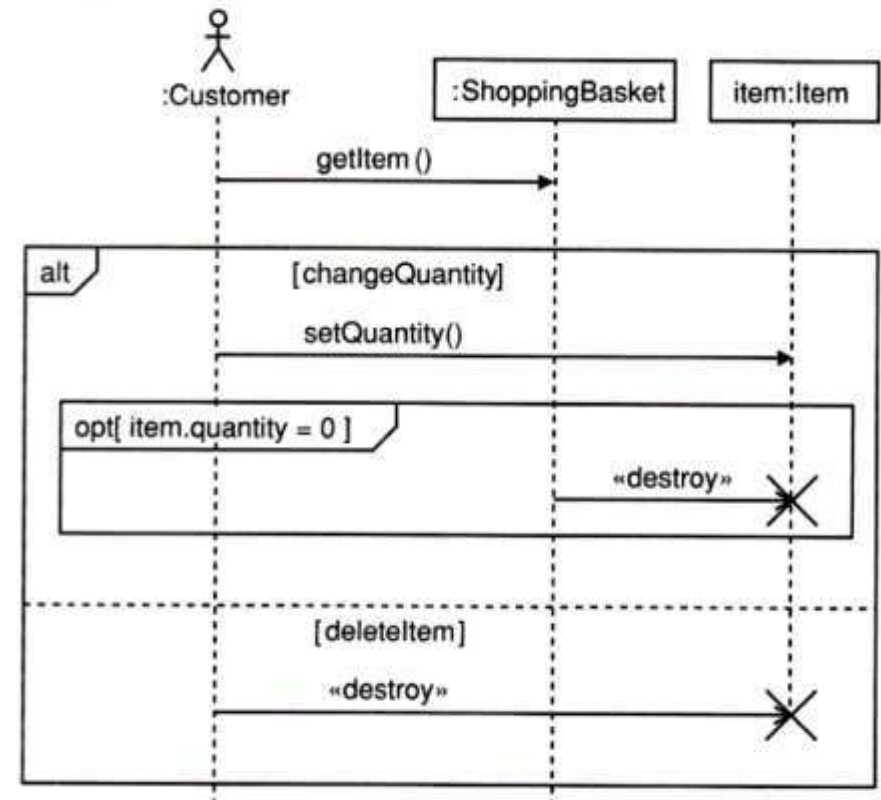
None.

Alternative flows:

None.



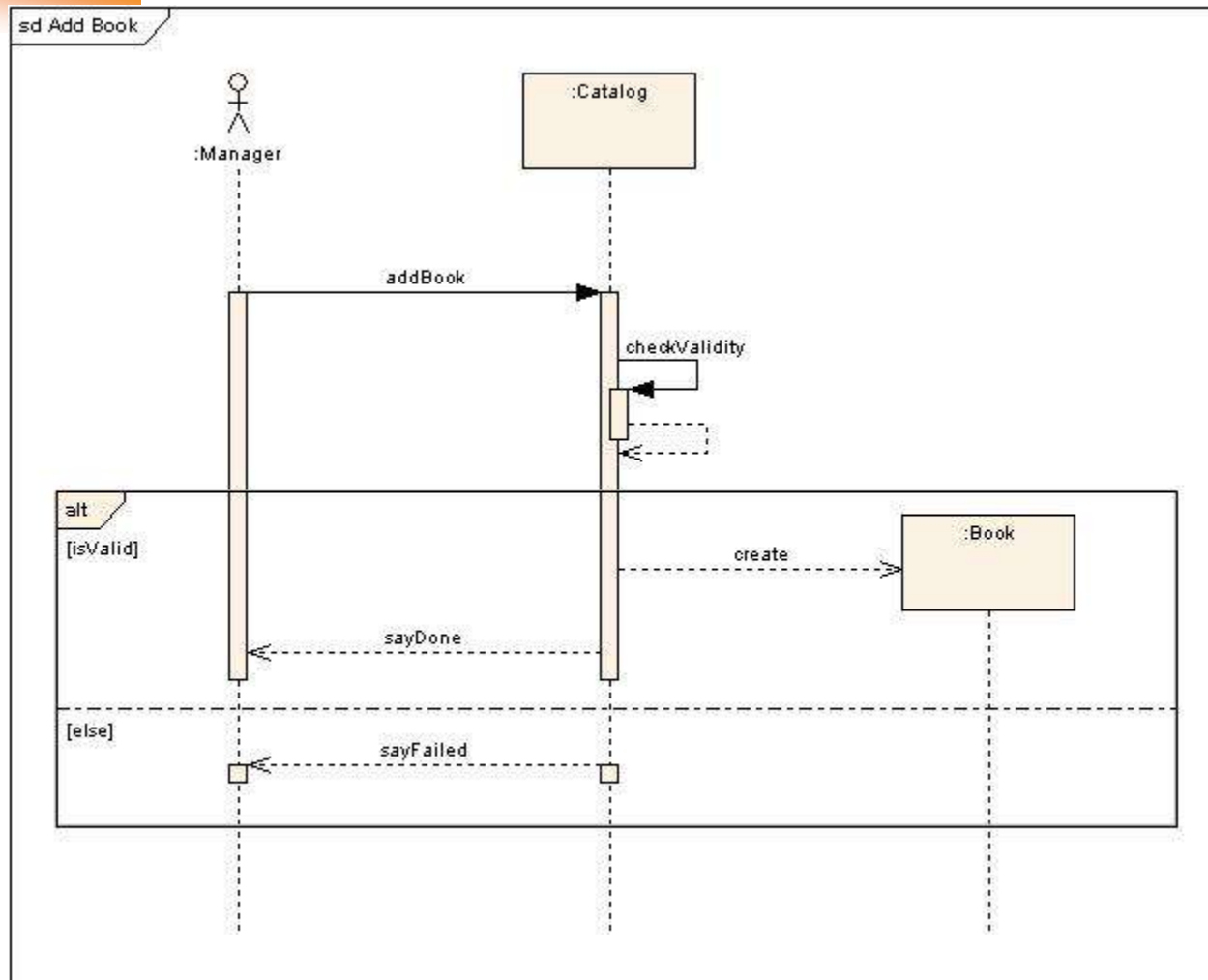
sd ManageBasket



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: <ol style="list-style-type: none"> این مورد کاربرد وقتی آغاز می شود که مدیر درخواست اضافه نمودن کتابی را اعلام کند. سامانه از مدیر درخواست می کند اطلاعات کتاب جدید را وارد کند. مدیر اطلاعات مورد نظر خود را وارد می کند. سامانه از مدیر می خواهد دسته مورد نظر جهت اضافه نمودن کتاب را انتخاب کند. مدیر دسته مورد نظر خود را مشخص می کند. مدیر اطلاعات وارد شده را تأیید می کند. سامانه مجاز بودن اضافه نمودن کتاب را بررسی می کند. سامانه کتاب را اضافه می کند.
شرایط نهایی: کتابی به مجموعه کتاب ها اضافه می شود.
روند جایگزین: <p>مجاز نبودن اضافه کردن کتاب جدید</p>

مثال ٤



Operators – *loop* and *break* : قطعات ترکیبی

loop – loop min, max [condition] ■

loop or loop * - loop forever; ■

loop n, m – loop (m-n) times; ■

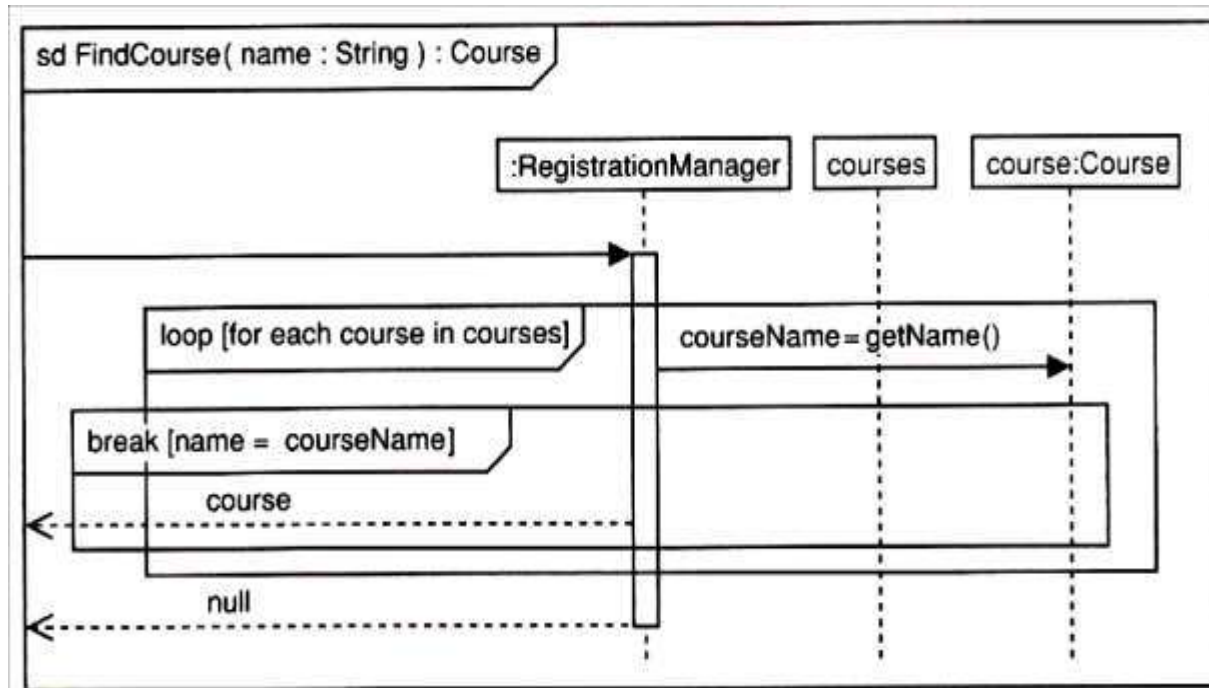
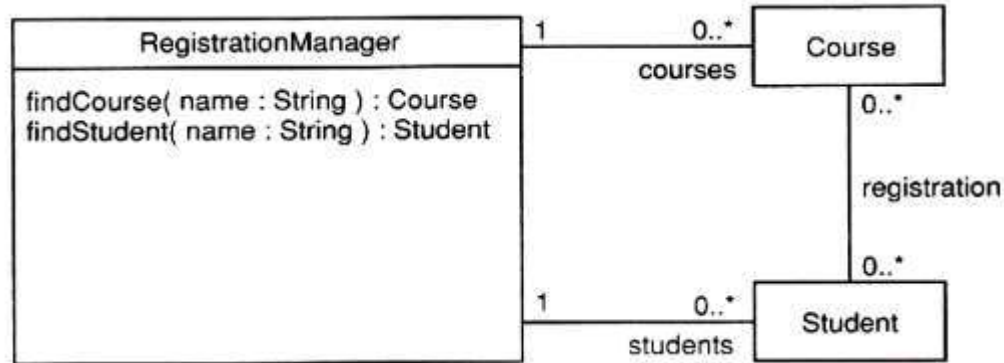
loop [booleanExpression] – loop while ■
booleanExpression is true;

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

break : اگر guard condition درست باشد، operand ■

اجرا می شود، نه بقیه حلقه تکرار

قطعرات ترکیبی : *loop and break*



State Machine Diagrams

- ماشین های حالت در صنعت به دلیل استفاده در پردازش بلادرنگ 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 ماهواره 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 می گیرد، می تواند 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



State

Final State



State Machine Diagrams (cont.)

- حرکت بین حالت‌های مختلف **transition** نامیده می شود.
- هر **transition** دو حالت را به هم متصل می کند.
- یک حالت می تواند یک **transition** به خودش داشته باشد.

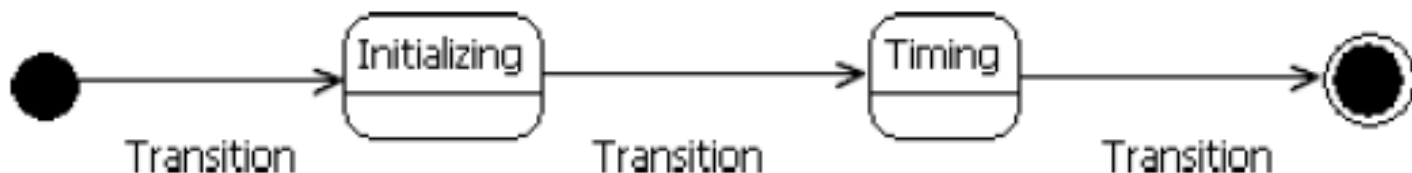


Figure 5–53 Transitions for the Duration Timer

State Machine Diagrams (cont.)

- رخدادهای خاصی باید اتفاق افتند تا یک **transition** انجام شود.
- این رخدادها روی **transition** نوشته می شوند.
- یک رخداد، اتفاقی است که ممکن است سبب تغییر حالت سیستم شود.

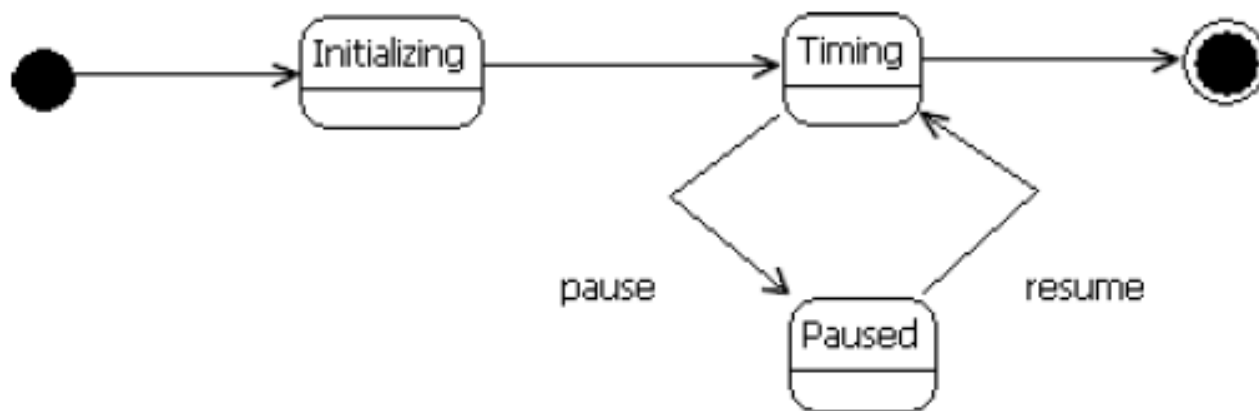


Figure 5–54 Additional States and Transition Events for the Duration Timer

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.

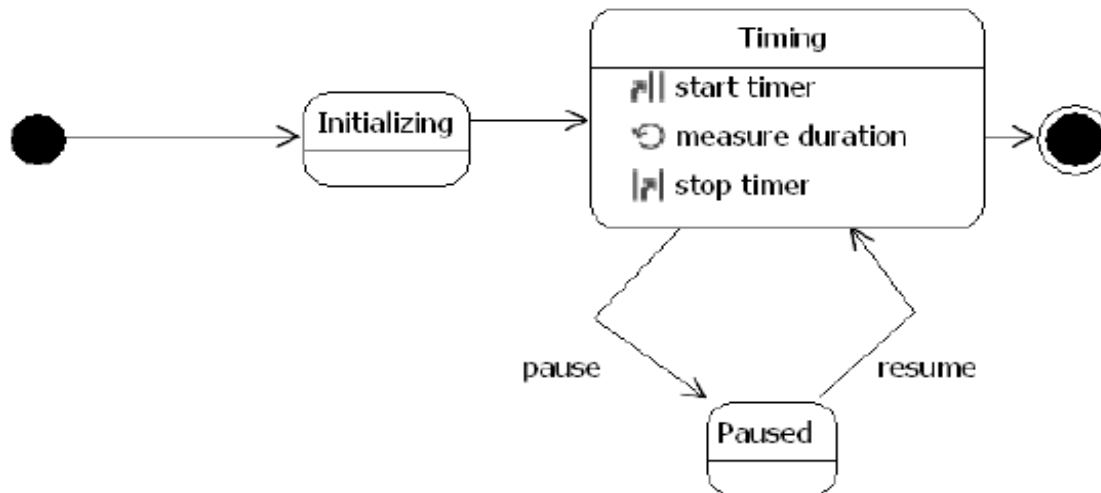


Figure 5–55 Entry, Do, and Exit Activities

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، **transition** اجرا شده و **effect** هم اتفاق می افتد.

State Machine Diagrams (cont.)

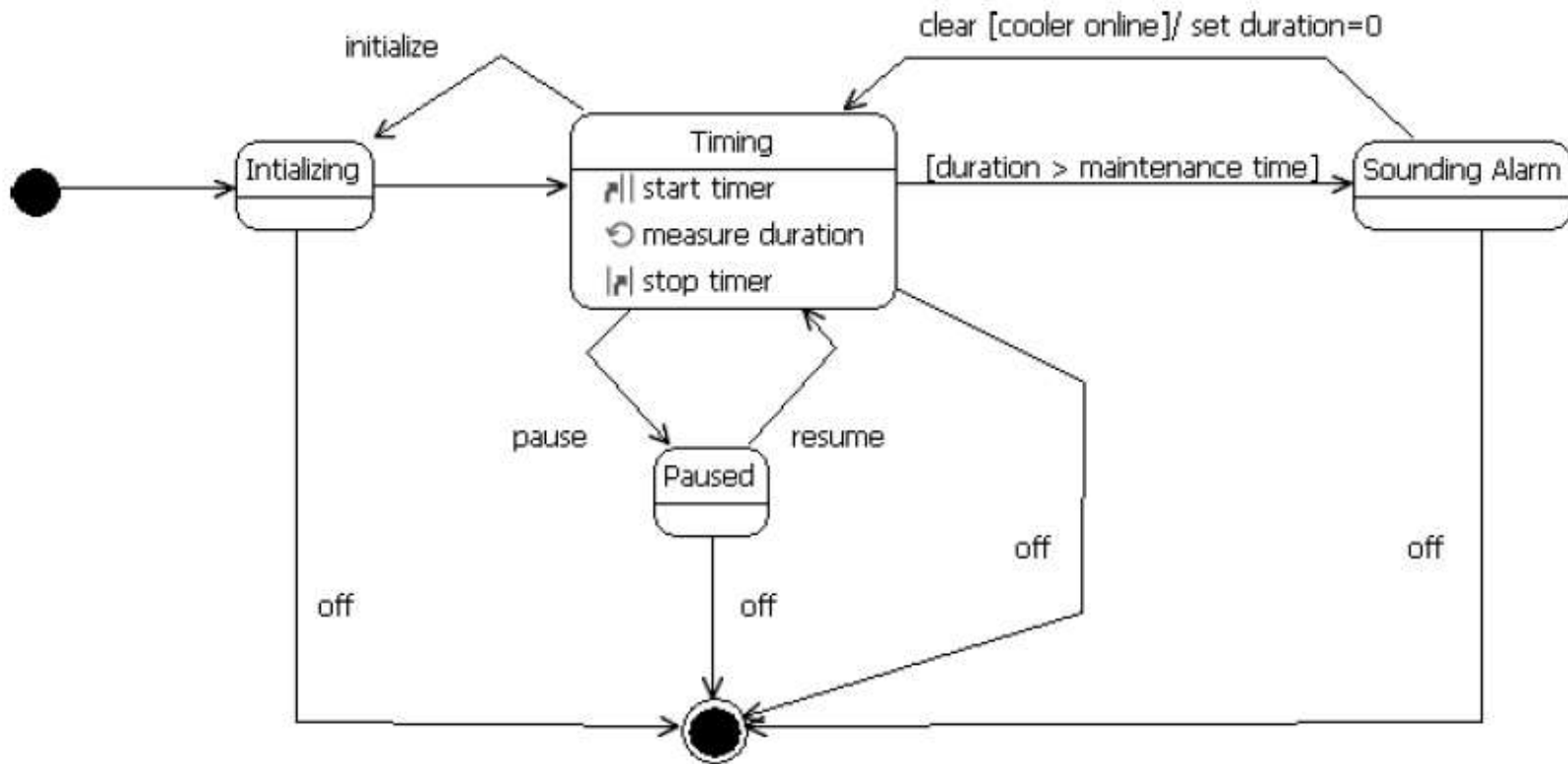


Figure 5–56 The Enhanced State Machine Diagram for the Duration Timer

State Machine Diagrams (cont.)

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

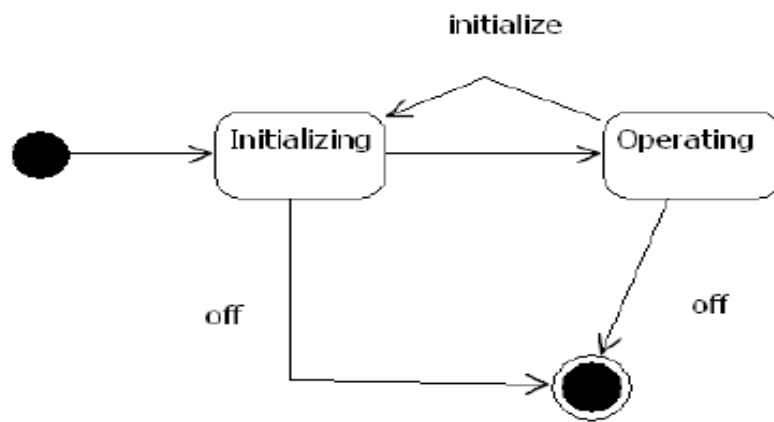


Figure 5–58 A Higher-Level View of the State Machine Diagram for the Duration Timer

State Machine Diagrams (cont.)

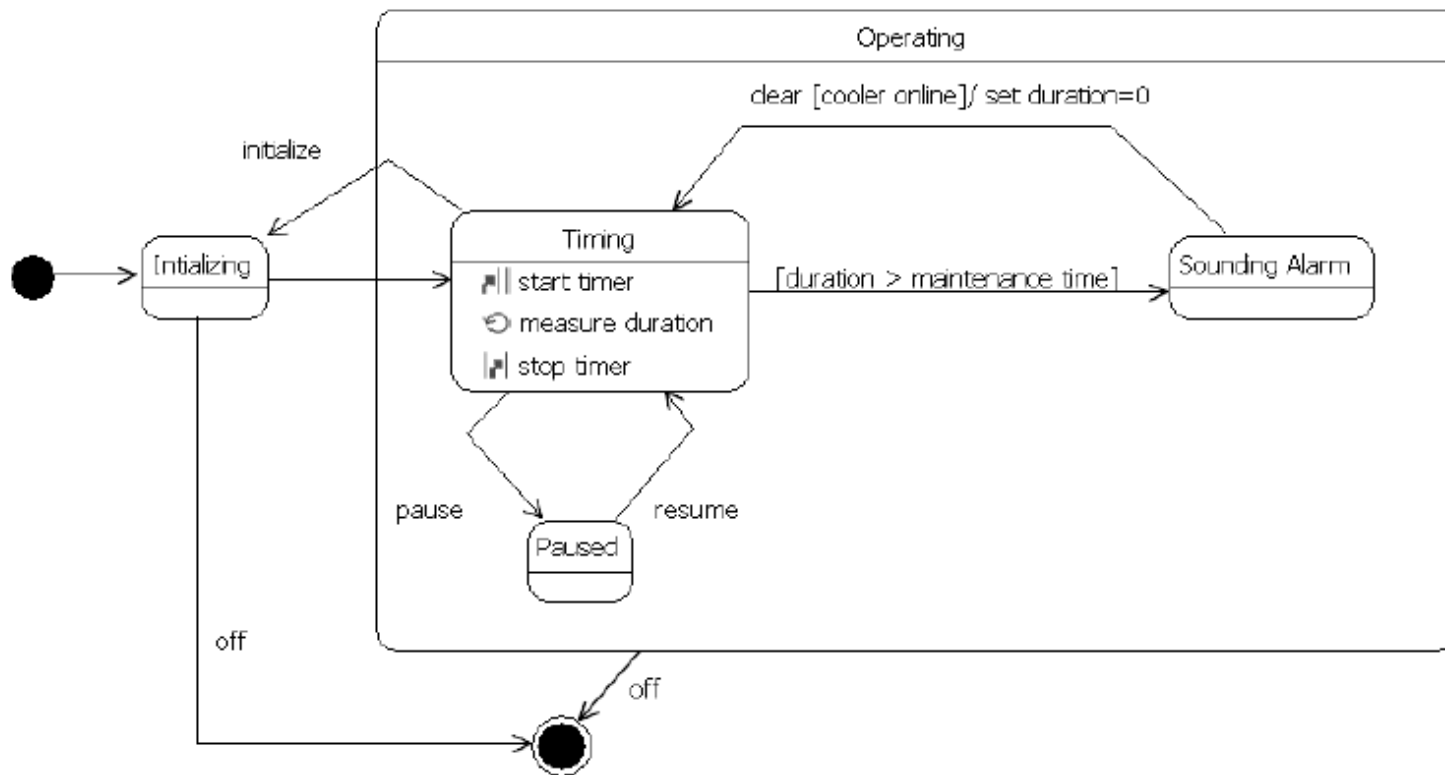


Figure 5–57 Composite and Nested States