به نام خدا

آزمایشگاه مهندسی نرم افزار

فهرست مطالب

- یادآوری از مفاهیم اولیه در مهندسی نرم افزار
- تاریخچه UML
- معرفی مفاهیم شئ گرا (Object Oriented)
- معرفی نمودارهای UML
- Class Diagram
- Use Case Diagram
- Activity Diagram
- Sequence & Collaboration Diagrams

❑ یادآوری از مفاهیم اولیه در مهندسی نرم افزار

- ❑ Software
- ❑ Software Engineering
- ❑ Process
- ❑ Process Models (Waterfall, Spiral, ..)
- ❑ Methodology
  - ■ Structured
  - ■ Object Oriented
  - ■ Component Oriented

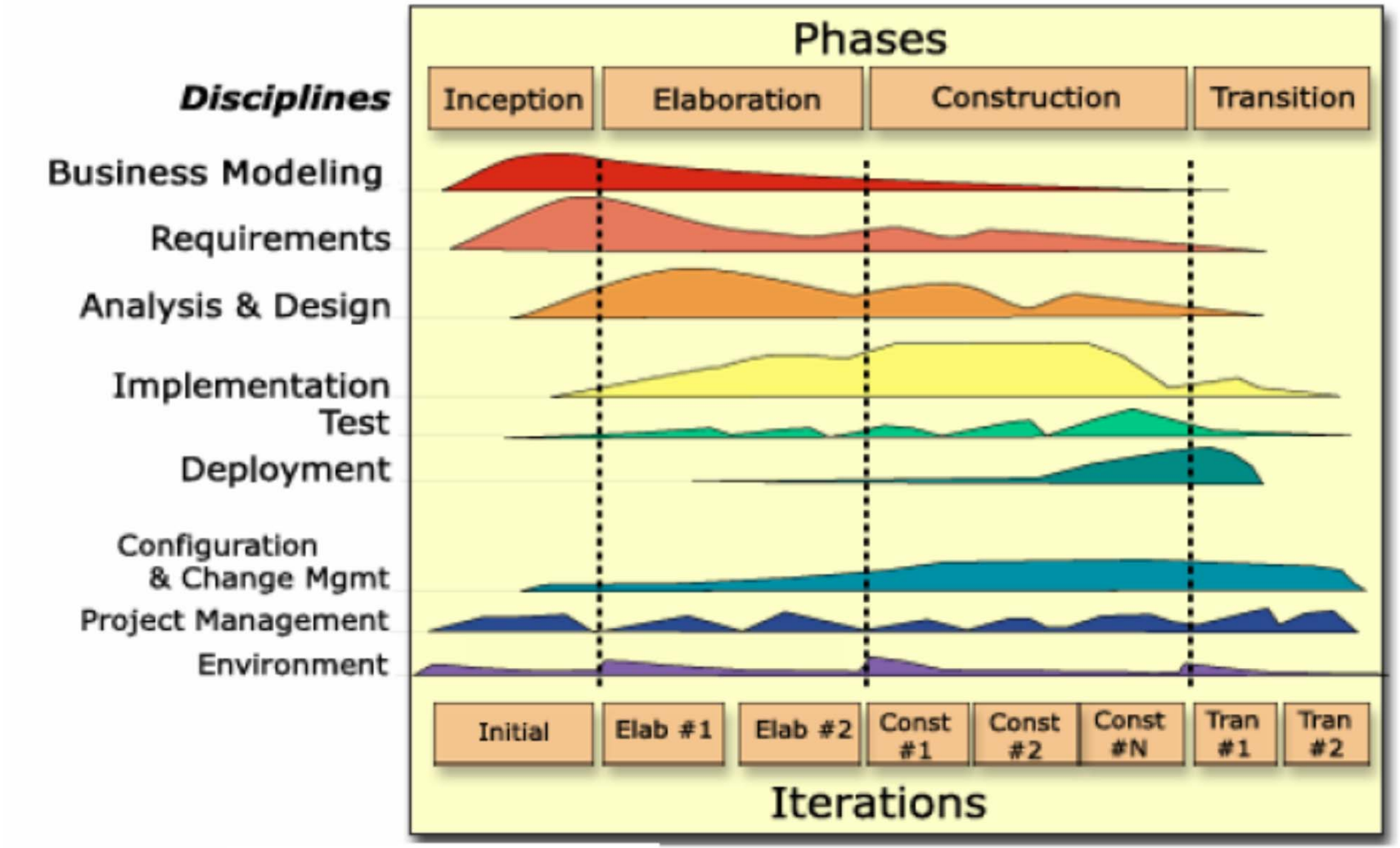
❑ تاریخچه UML

- ❑ اولین نسل از متدولوژی های شئ گرا در اواخر دهه 80 و اوایل دهه 90 میلادی مطرح شد. جنگ متدها (90 تا 95 میلادی)
- ❑ فراخوان شرکت OMG(Object Management Group) در سال 96 میلادی به منظور خاتمه بخشی به جنگ متدها
- ❑ سه متدولوژی معروف :

✓ Booch:
✓ OMT(Object Model Technique)
✓ OOSE(Object Oriented Software Engineering)

- ❑ برتری شرکت Rational در فراخوان با ارائه متدولوژی RUP

❑ نمایی از متدولوژی RUP



❑ تاریخچه UML (ادامه ..)

❑ ورژن های UML:

- ✓ UML 1.0 (1997)
- ✓ UML 1.1 (1998)
- ✓ UML 1.2 (1999)
- ✓ UML 1.3 (2000)
- ✓ UML 1.4 (2001)
- ✓ UML 1.5 (2002)
- ✓ UML 2.0 (2005)

معرفی مفاهیم شئ گرایی (Object Oriented)

❑ معرفی مفاهیم شئ گرا (Object Oriented)

- ✓ Class
- ✓ Object
- ✓ Attribute
- ✓ Method
- ✓ Message
- ✓ Encapsulation
- ✓ Inheritance
- ✓ Polymorphism

UML 2 – The Diagrams

Structural	Behavioural
Class	Activity
Object	Use Case
Package	Behavioural State
Deployment	Protocol State
Component	Interaction Overview
Composite Structure	Sequence
	Timing
	Communication
	Information Flow

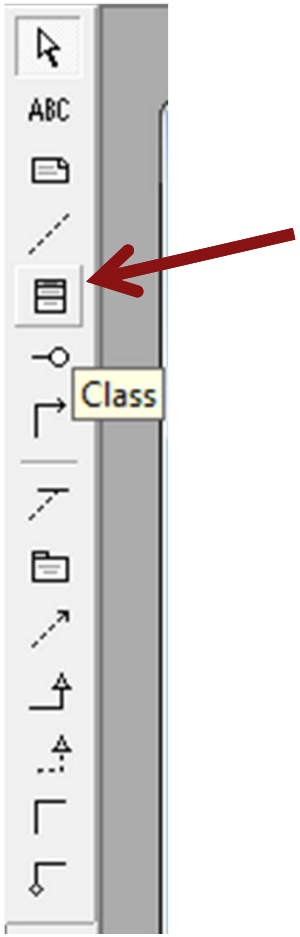
Class Diagram

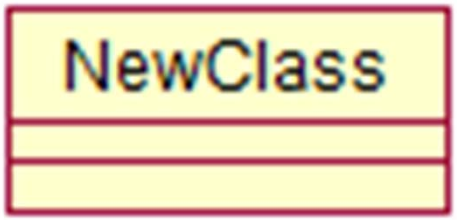
Class Diagram

- نمایش Class در UML
- روابط میان کلاسها
  - Association
  - Generalization
  - Aggregation ,Composition
  - Realization

نمایش Class در Rational Rose

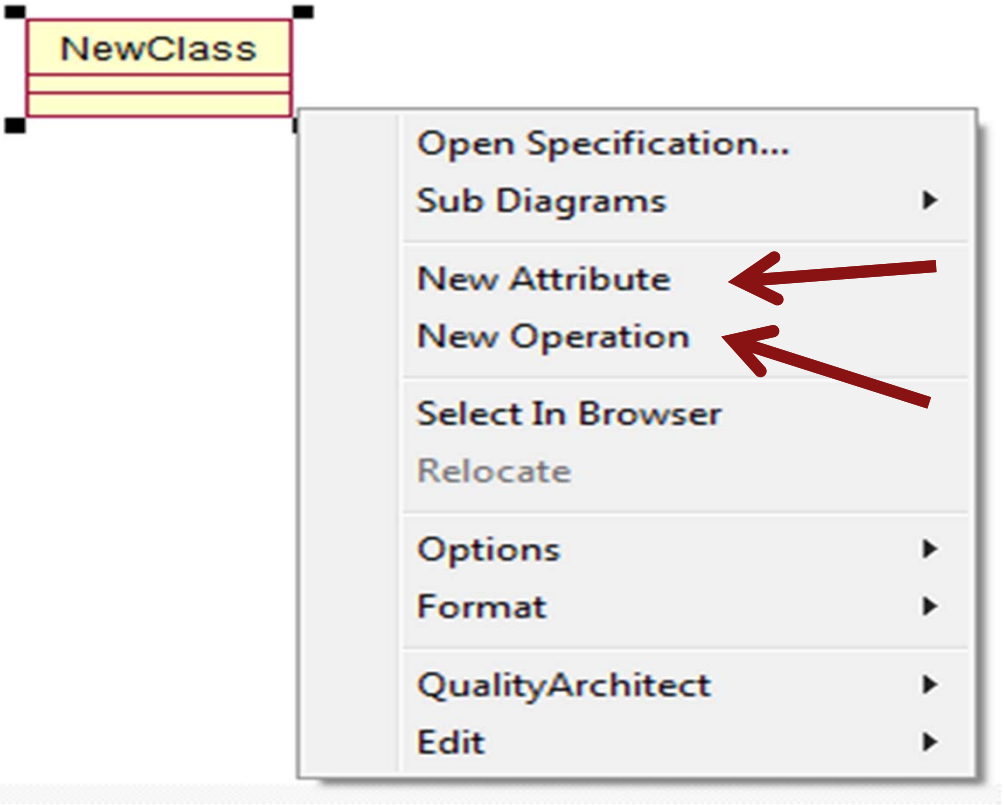
- در UML کلاس را با مربع نشان می دهند که شامل سه بخش است. در این بخشها به ترتیب از بالا به پایین: نام کلاس، صفات کلاس و متدهای کلاس قرار می گیرد.





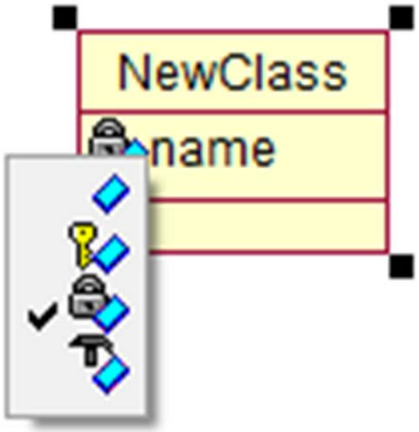
نمایش Class در Rational Rose (..)

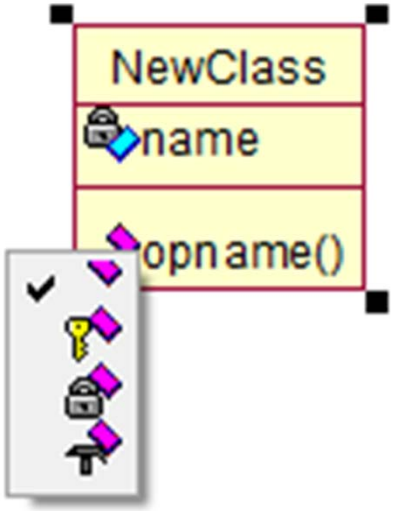
- برای ایجاد صفات و یا متدهای یک کلاس، روی کلاس کلیلک راست نموده و به ترتیب گزینه های New Attribute و یا New Operation را انتخاب می نماییم.



سطوح دسترسی به اجزای کلاس

- سه سطح دسترسی وجود دارد:
  - **Public یا عمومی :** قابل استفاده توسط تمامی کلاس ها
  - **Protected یا محافظت شده:** قابل استفاده توسط کلاسهای فرزند
  - **Private یا خصوصی :** قابل استفاده تنها توسط همان کلاس



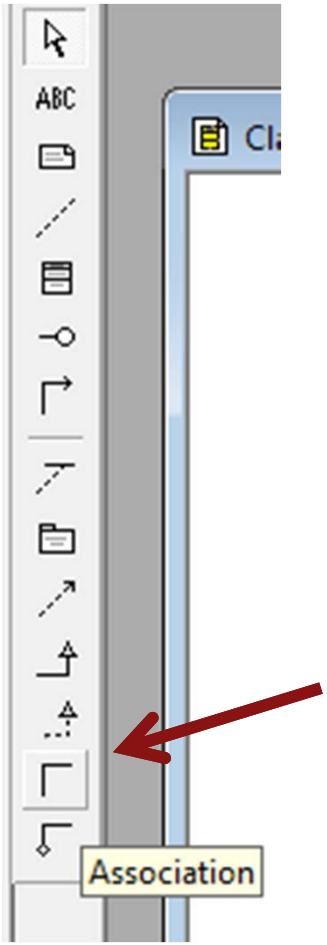


روابط میان کلاسها

- کلاسهای سیستم از روی مذاکرات و صحبتهایی که مشتری مطرح می کند استخراج می شود. اسامی که شنیده می شود کاندیدی برای کلاس محسوب می شود به شرط اینکه بتوان چندین نمونه از آنرا در سیستم مشاهده کرد.
- اگر میان کلاسهای تعیین شده، رابطه ایی برقرار نباشد نمودار کلاس به فرهنگ لغات سیستم تبدیل می شود. بنابراین نیاز است روابط میان کلاسها را تعیین کرد.

رابطه تناظر (Association)

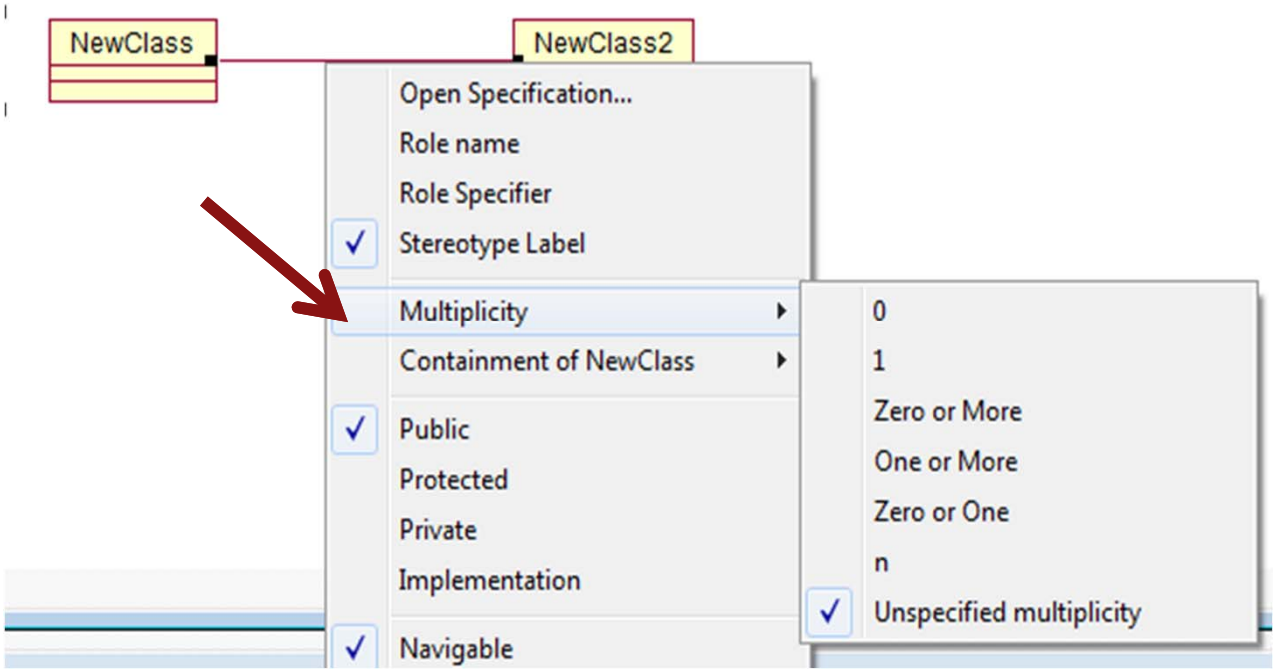
- یک رابطه ی ساده میان کلاسها را بیان می کند و با یک خط مستقیم نمایش داده می شود.



ویژگی های رابطه تناظر

- **Multiplicity یا چند به چندی رابطه:**

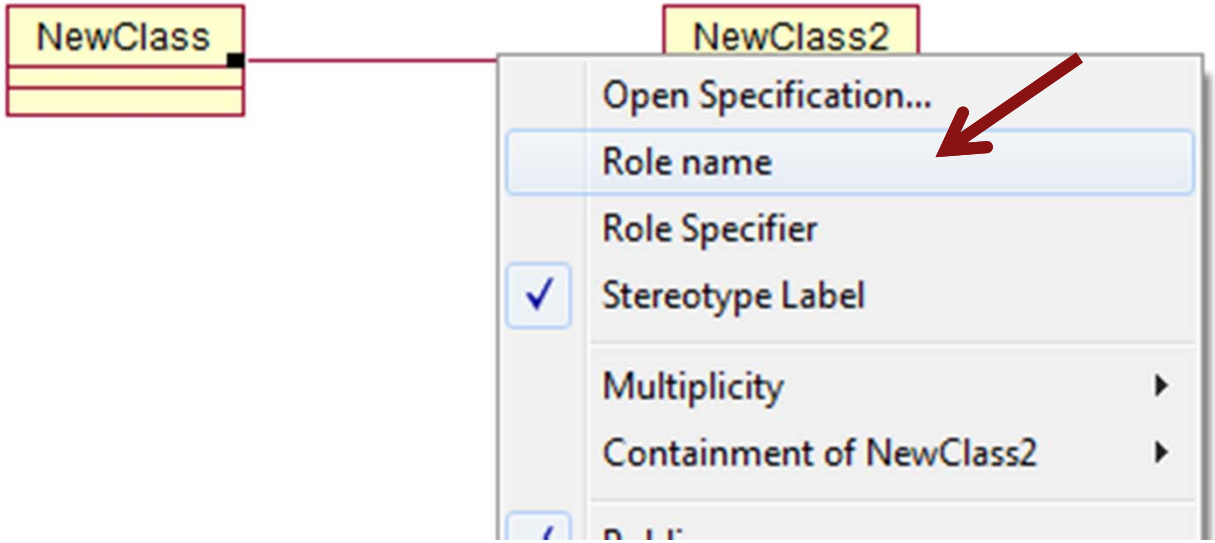
نشان می دهد چند نمونه از یک کلاس با چند نمونه از کلاس دیگر در رابطه است. برای ایجاد چند به چندی یک رابطه روی رابطه کلیلک راست کرده و گزینه Multiplicity را انتخاب می کنیم.



ویژگی های رابطه تناظر (ادامه..)

- **Role Name یا نام نقش:**

در برخی از مواقع نقشهایی که هر یک از کلاسها در سیستم ایفا می کنند، در رابطه ی تناظر مشخص می شود. برای مشخص کردن نام نقش روی رابطه و در سمت کلاس مورد نظر، کلیلک راست کرده و گزینه Role Name را انتخاب می کنیم.

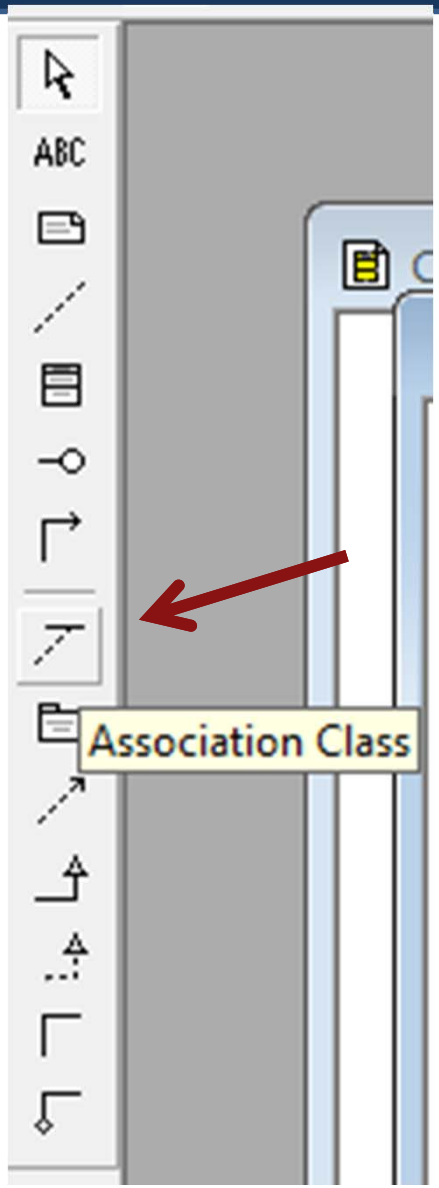


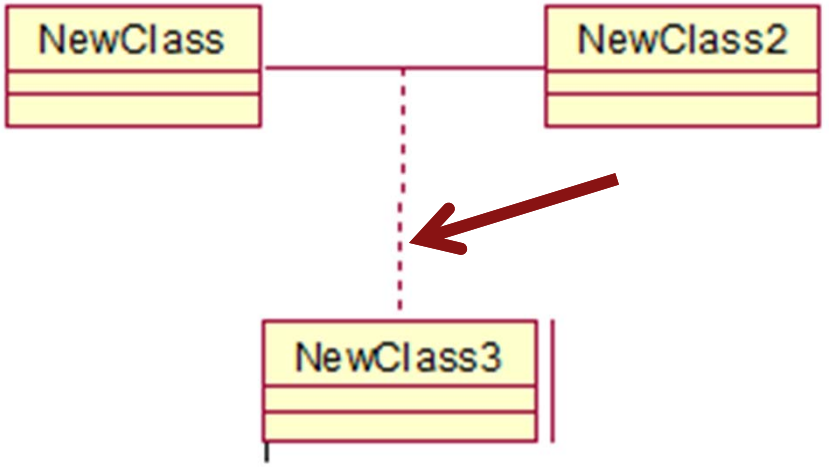
نکته 1

- **Association Class یا کلاس تناظر:**

در برخی از موارد رابطه ی تناظر خود در بر دارنده ی صفات و عملیات متعددی می باشد. در چنین مواردی می توان برای رابطه کلاسی در نظر گرفت و صفات و عملیات رابطه را در کلاس قرار داد. به چنین کلاسی، کلاس تناظر گفته می شود.با استفاده از ابزار Association Class می توان این کلاس را به رابطه متصل نمود.

نمایش Association Class

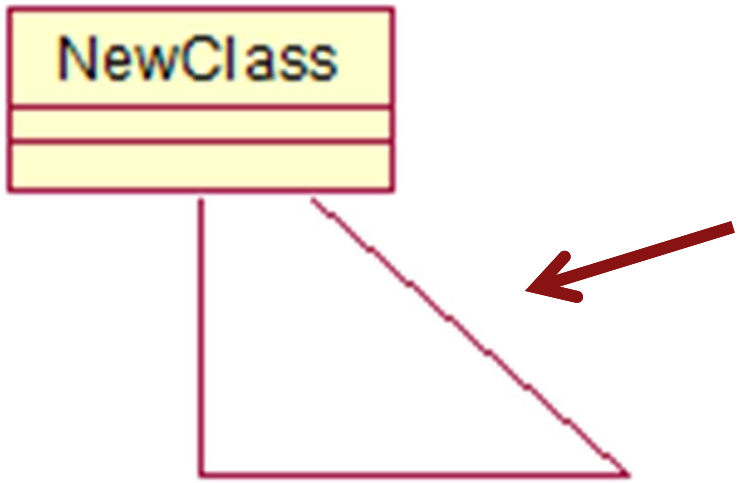




نکته 2

- **تناظر بازتابی**

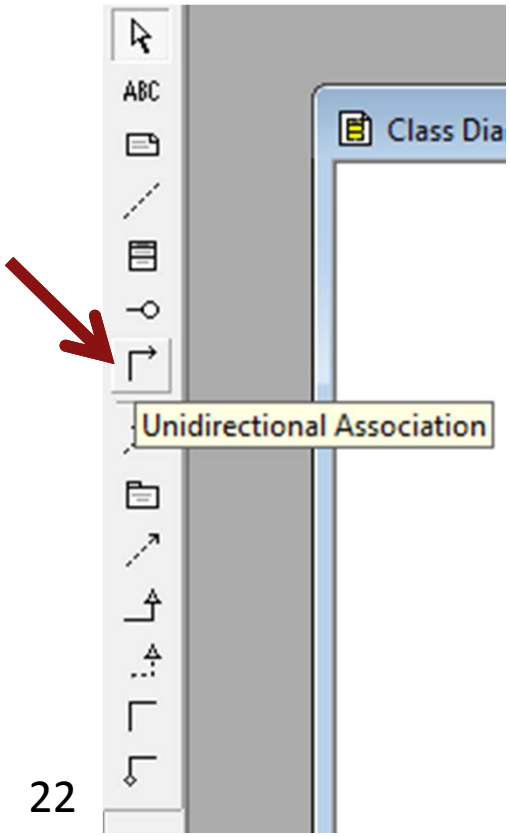
رابطه ی میان کلاس با خودش را تناظر بازتابی گویند. معمولا زمانی مطرح می شود که یک کلاس اشیایی دارد که می تواند نقشهای مختلفی در سیستم ایفا کند. برای ایجاد چنین رابطه ایی کافی است رابطه ی Association را از یک کلاس به همان کلاس رسم نمود.



نکته 3

- **رابطه Navigate**

رابطه ایی شبیه تناظر است با این تفاوت که صرفا یک طرفه و فقط از یک کلاس به کلاس دیگر رسم می شود. برای ایجاد آن از ابزار Unidirectional Association استفاده می شود.



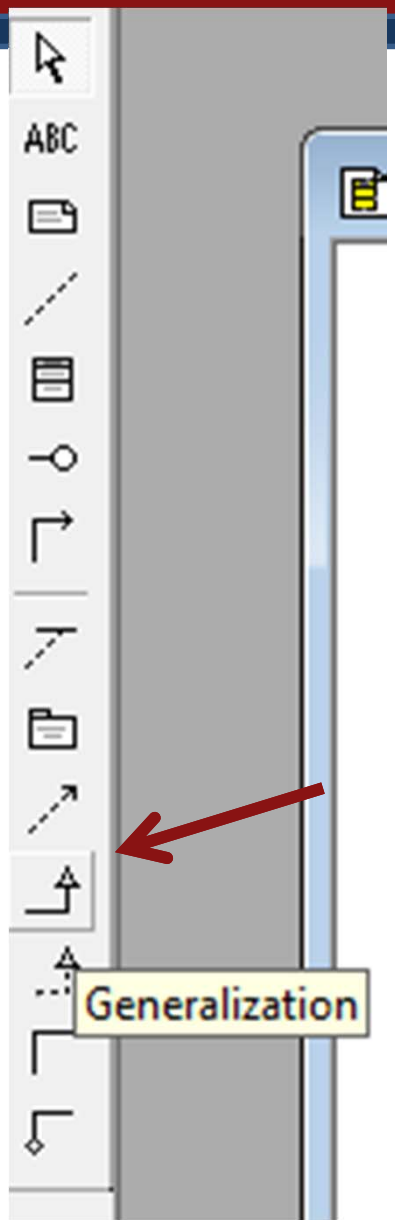
رابطه وراثت (Generalization)

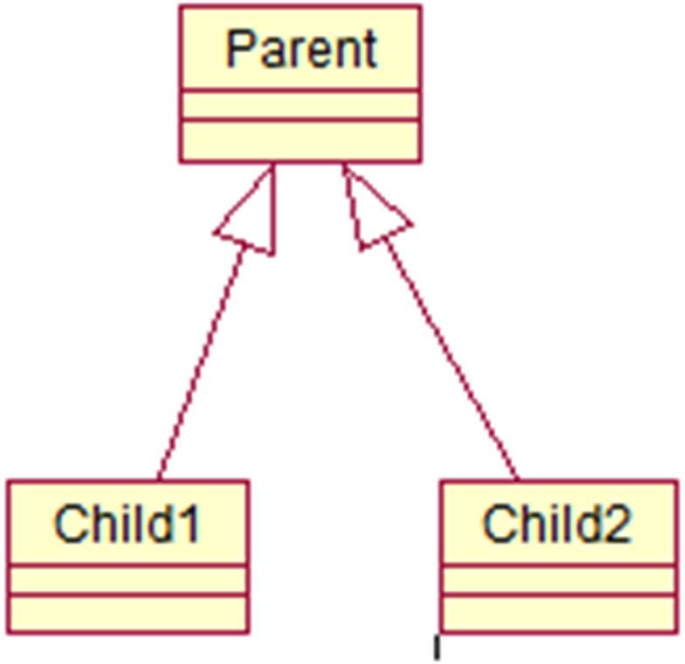
- این رابطه نوعی تناظر است که یک کلاس از صفات و متدهای کلاس دیگر استفاده می کند. این رابطه توسط تحلیل گر شناخته می شود. دو دیدگاه برای شناسایی این رابطه مطرح می شود:

  - تشخیص صفات و اعمال مشترک و جداسازی آن در کلاس پدر(تعمیم یا Generalization)

  - بازشناسی اعمال و صفات متفاوت بین نمونه های یک کلاس و جداسازی هریک از آنها در کلاس فرزند (تخصیص یا Specialization)

نمایش رابطه وراثت در Rational Rose



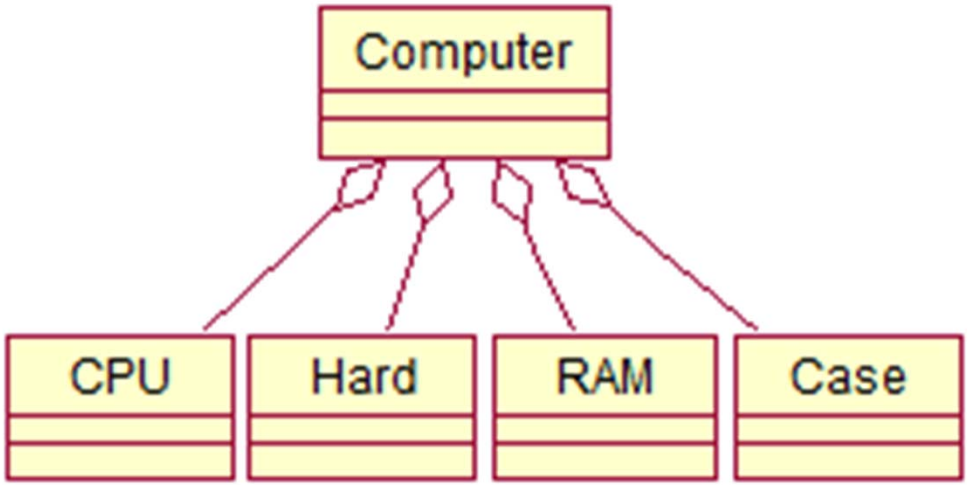


رابطه تجمع (Aggregation)

- این رابطه نوعی تناظر است که یک طرف رابطه از اهمیت بیشتری برخوردار است. به این رابطه، رابطه ی جزء – کل نیز گفته می شود. به این دلیل که رابطه تجمع بصورت سلسله مراتبی از کلاس کل در بالا و کلاسهای جزء در پایین نمودار می باشد.

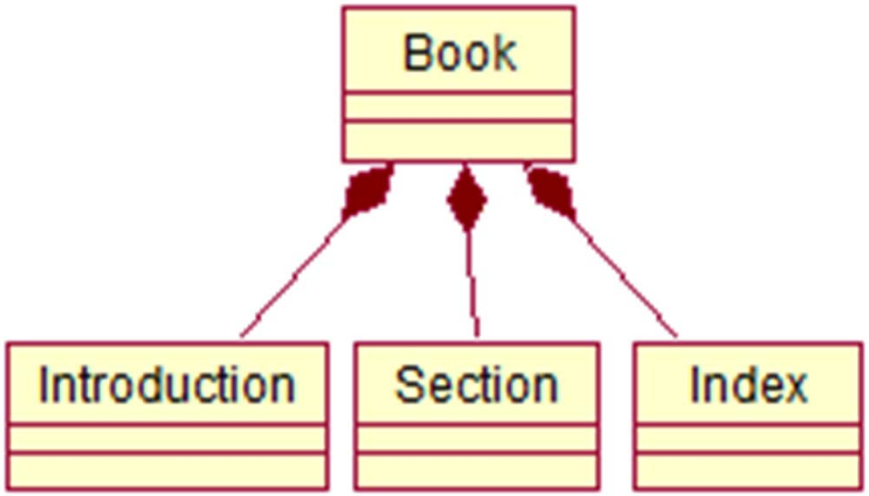
- مانند کلاس های CPU ، Hard، RAM و Case که جزعی از کلاس Computer هستند.

نمایش رابطه تجمع در Rational Rose



رابطه ترکیب (Composition)

- این رابطه نوعی رابطه تجمع قوی است بدین ترتیب که هر یک از کلاسهای جزء به تنهایی نمی تواند مطرح گردد بلکه همواره در ترکیب با کلاس کل در سیستم مطرح می گردند. به عبارت دیگر هر کلای جرء وابسته به کلاس کل است. بعنوان مثال کلاسهایی مثل مقدمه ، فهرست مطالب و فصل ترکیبی از کلاس کتاب هستند که به تنهایی مفهمومی ندارند.

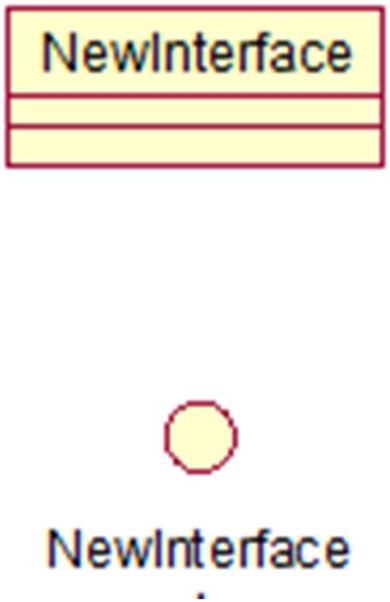


رابط (Interface) و محقق سازی (Realization)

- رابطها (واسط ها) مجموعه ایی از عملیات است که رفتار یک کلاس را مشخص می کنند و توسط آن کلاس به کلاسهای دیگر عرضه می شود.

- در Rational Rose رابط ها همانند کلاس مدلسازی می شوند (به شکل مستطیل) و هیچ صفتی ندارند و فقط شامل عملیات اند. روش دیگر مدلسازی رابط با استفاده از نماد Interface می باشد.

نماد رابط (Interface) در Rational Rose



رابط (Interface) و محقق سازی (Realization)

- به منظور بکارگیری مجدد عملیات کلاسها از واسط ها استفاده می شود. عملیات در رابط تعریف شده و توسط سایر کلاسها مجددا استفاده می شود.
- رابطه ی میان یک کلاس با کلاس رابط ، محقق سازی (Realization) نامیده می شود . در Rational Rose با نماد Realize مشخص می گردد.

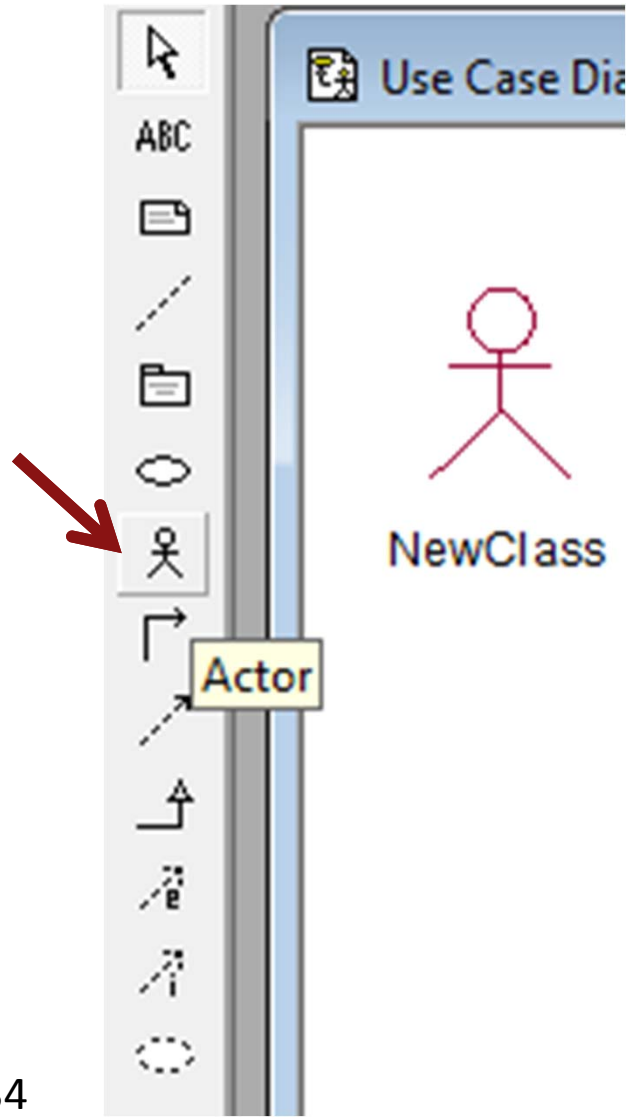
Use Case Diagram

- اجزای نمودار Use Case
  - Actor
  - Use Case
- ارتباط میان Actor و Use Case
- روابط میان Use Case ها
  - include
  - extend
  - generalization

Actor

- منظور از Actor یا بازیگر (کنشگر) عناصری در سیستم است که با سیستم در تعامل هستند و کارکردی از سیستم را اجرا می کند .
- سوالات زیر برای شناسایی Actor ها می تواند مفید باشد:
  - چه کسی از سیستم خواسته ایی دارد؟
  - چه کسی از بکارگیری سیستم سود می برد؟
  - چه کسی اطلاعات سیستم را وارد می کند، از گزارشات آن استفاده می کند و یا اطلاعات سیستم را تغییر می دهد؟
  - چه کسی سیستم را نگهداری و پشتیبانی می کند؟
  - ...

نماد Actor در Rational Rose



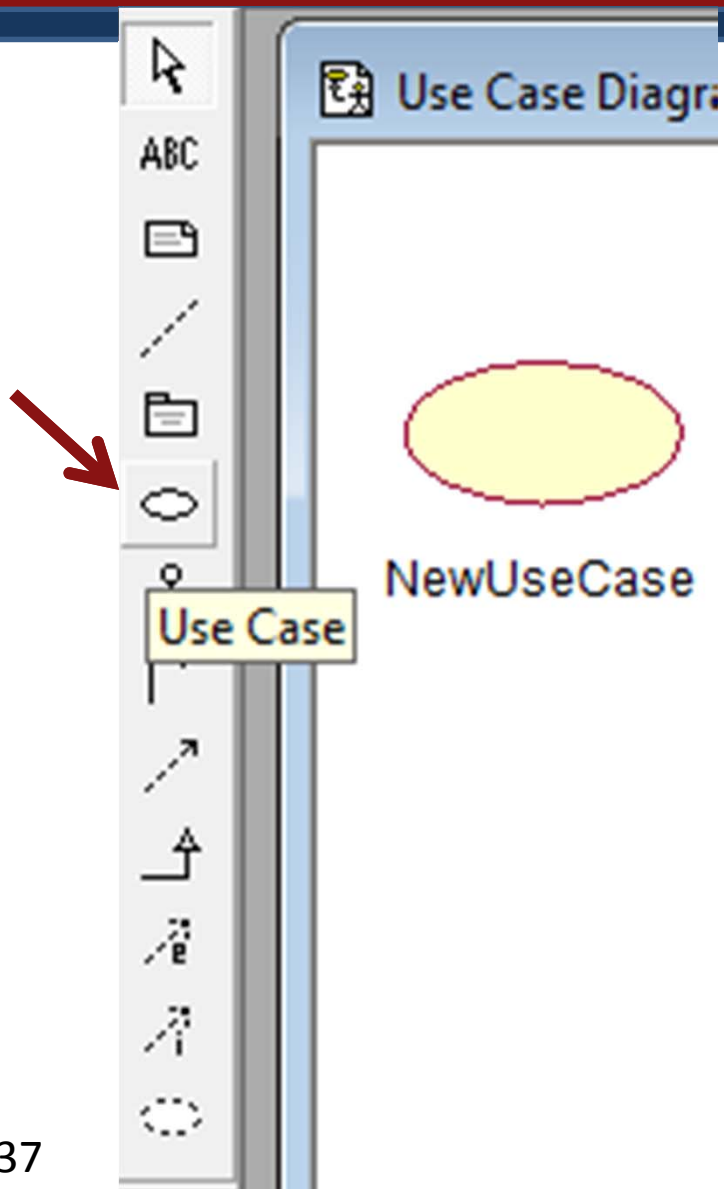
انواع Actor

- **اصلی (Principal):** عناصری که از امکانات اصلی سیستم استفاده می نمایند.
- **ثانوی (Secondary):** عناصری که مدیریت یا نگهداری وظایف سیستم را برعهده دارند.
- **سخت افزار خارجی (External Hardware):** شامل ابزارها و سخت افزارهایی که جزء دامنه ی سیستم بوده و سیستم روی آن ها نصب و اجرا می شود.
- **سیستم های دیگر (Other System):** کلیه ی سیستهایی که با سیستم مورد نظر در ارتباط هستند.

Use Case

- منظور ازUse Case یا مورد کاربرد، کارکردهایی است که در سیستم انجام می شود. هر مورد کاربرد مجموعه ایی از سناریوها (scenario) است. هر سناریو یک توالی از عملیاتی است که توسط یک Actor آغاز می گردد.
- در هر سناریوی یک Use Case موارد زیر مطرح می شود:
  - Actor های دخیل
  - پیش شرایط (عملیات و رخدادهایی که قبل از انجام یک مورد کاربرد انجام می شود)
  - پس شرایط (عملیات و رخدادهایی که پس از انجام یک مورد کاربرد انجام می شود)
  - ذینفعان پروژه (Stockholders)

نماد Use Case در Rational Rose

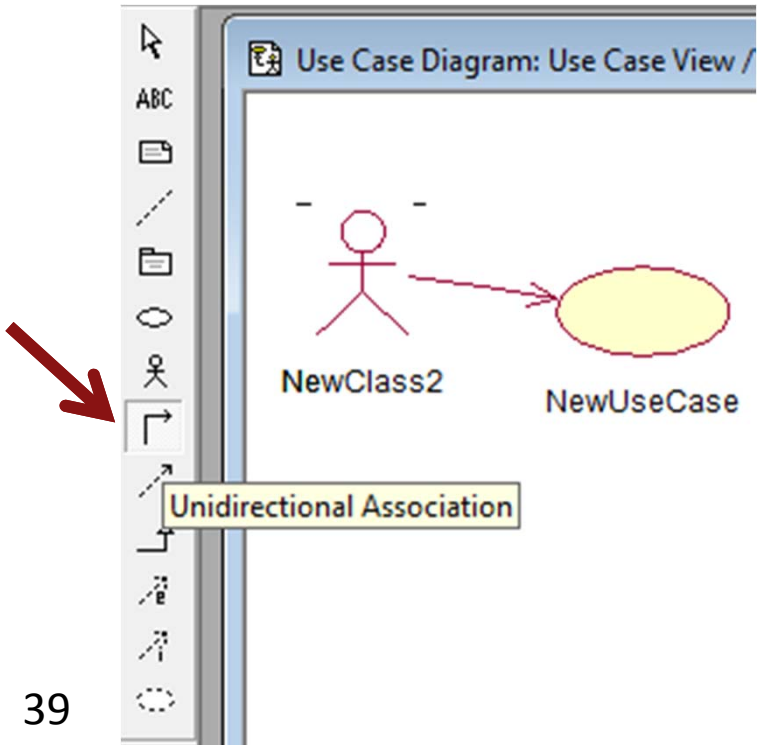


(..) Use Case

- سوالات زیر برای شناسایی می تواند مفید باشد:
  - چه کارکردهایی در سیستم مطرح می شود؟
  - وظایف هر Actor چیست؟
  - چه کارکردهایی ایجاد کننده، تغییردهنده ، استفاده کننده و یا حذف کننده اطلاعات محسوب می شوند؟
  - چه الزامات و خواسته هایی برای هر کارکرد مطرح می شود؟
  - ..

رابطه میان Actor و Use Case

- **رابطه تناظر:** یک ارتباط ساده میان Actor و Use Case را نشان می دهد. این رابطه با ابزار Unidirectional Association در Rational Rose نشان دهده می شود که از سمت Actor به Use Case رسم می گردد.



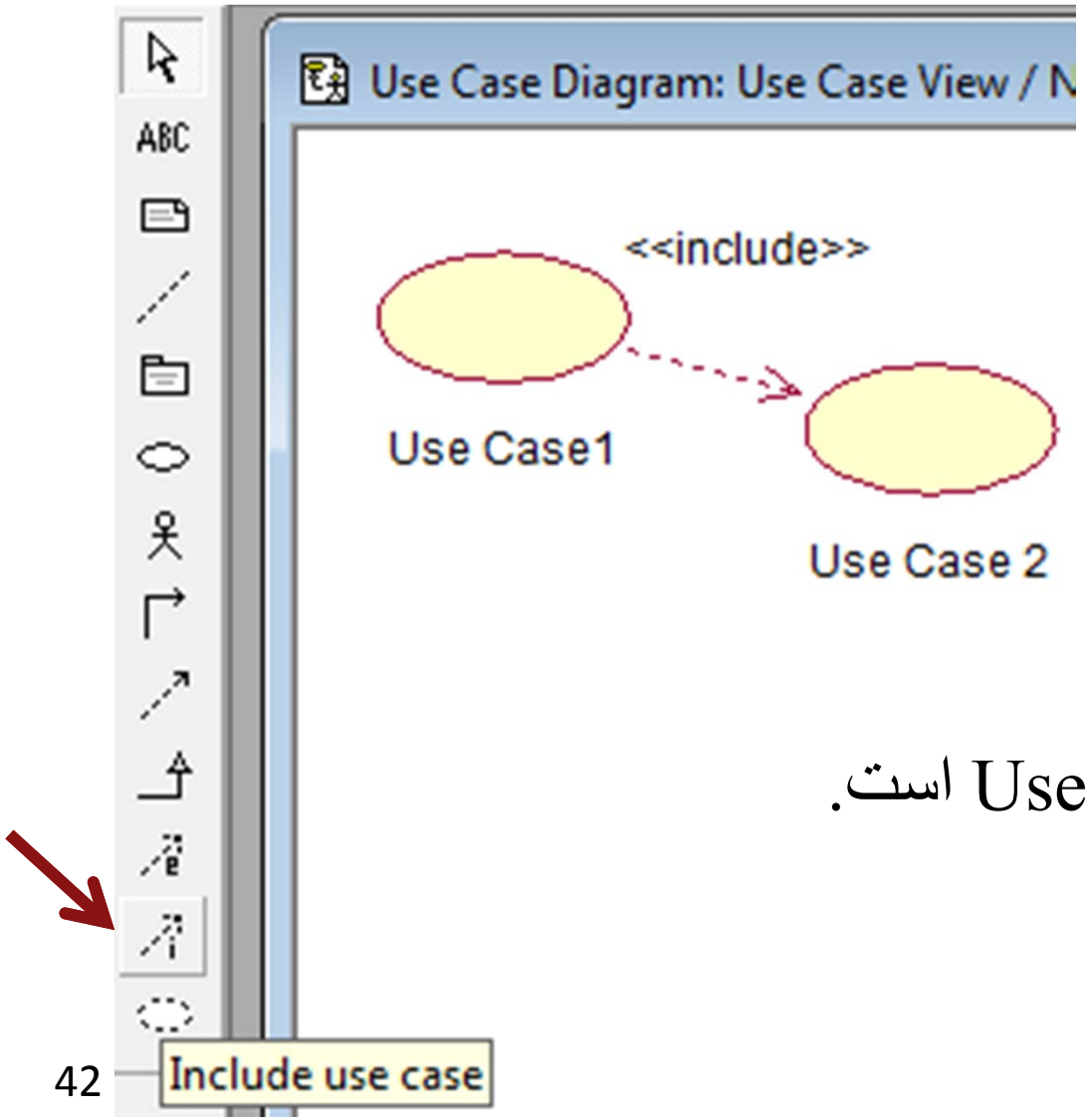
روابط میان Use Case ها

- include
- extend
- generalization

include

- زمانی که یک Use case عملیات Use Case دیگر را مورد استفاده قرار میدهد این رابطه مطرح می شود. اصطلاحا گفته می شود عملیات تعریف شده در یک Use Case دربردارنده عملیات تعریف شده در Use Case دیگر است.
- به عبارت دیگر می توان عملیات مشترک را در یک Use Case تعریف کرد .سایر Use Case ها این عملیات مشترک را عینا مورد استفاده قرار می دهند.
- بعنوان مثال در سیستم کتابخانه، Use Case های *امانت کتاب* و *تمدید امانت* ، شامل Use Case *کنترل عضویت* می باشد.

نماد رابطه include در Rational Rose



Use Case1 شامل Use Case2 است.

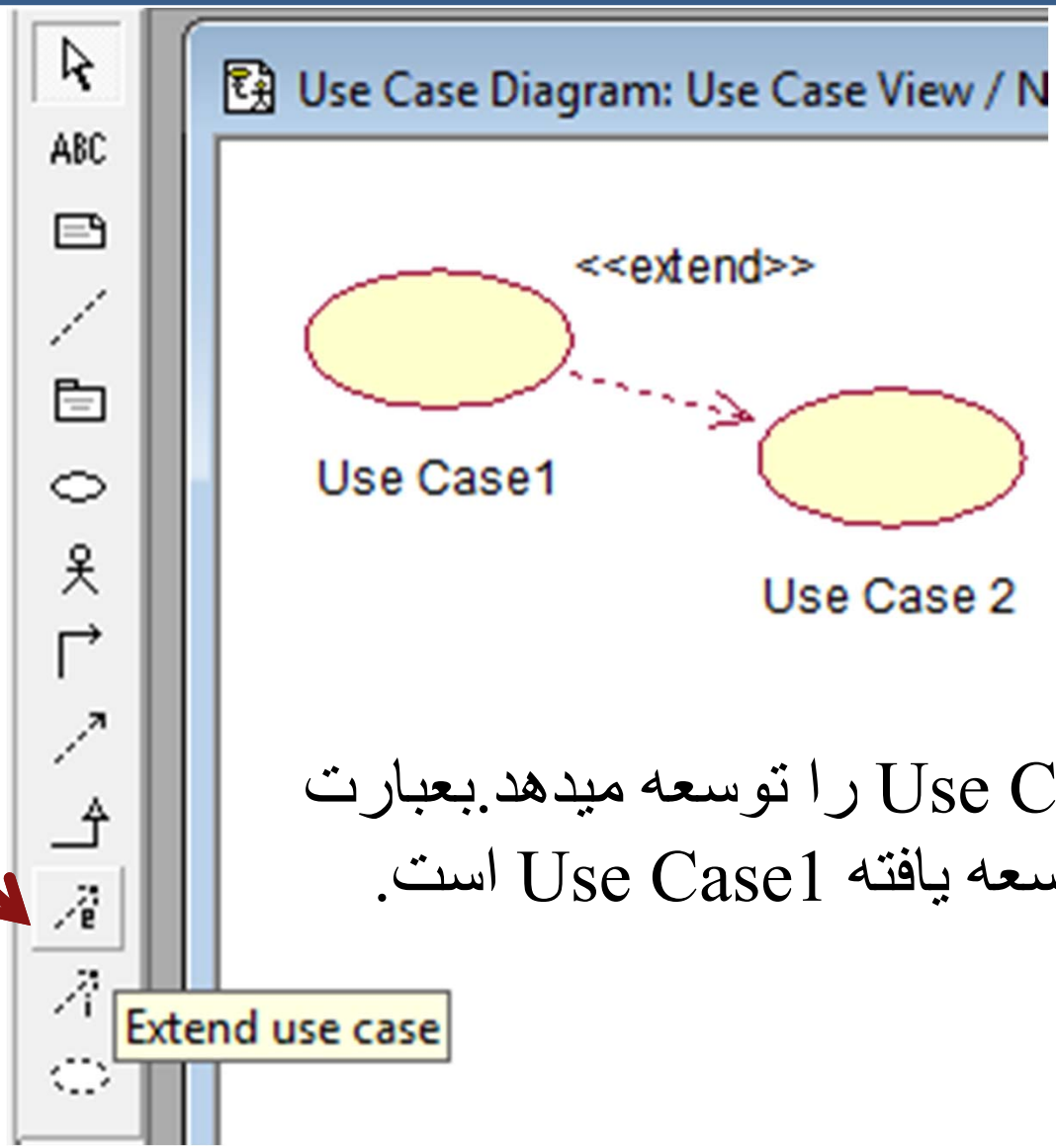
extend

- هرگاه یک Use Case ، Use Case دیگر را توسعه دهد، رابطه extend میان آنها برقرار می شود.

- Use Case های توسعه دهنده در واقع نوع خاصی از Use Case اصلی است. Use Case توسعه یافته بعنوان Use Case اجباری و هر یک از Use Case های توسعه دهنده بعنوان Use Case اختیاری محسوب می شوند که بر حسب نیاز کاربر اجرا می گردند.

- بعنوان مثال در سیستم بانک، *استعلام موجودی از حساب* توسط *چاپ موجودی* و *نمایش موجودی* توسعه داده شده است.

نماد رابطه extend در Rational Rose



Use Case1 ، Use Case2 را توسعه میدهد.بعبارت دیگر Use Case2 توسعه یافته Use Case1 است.

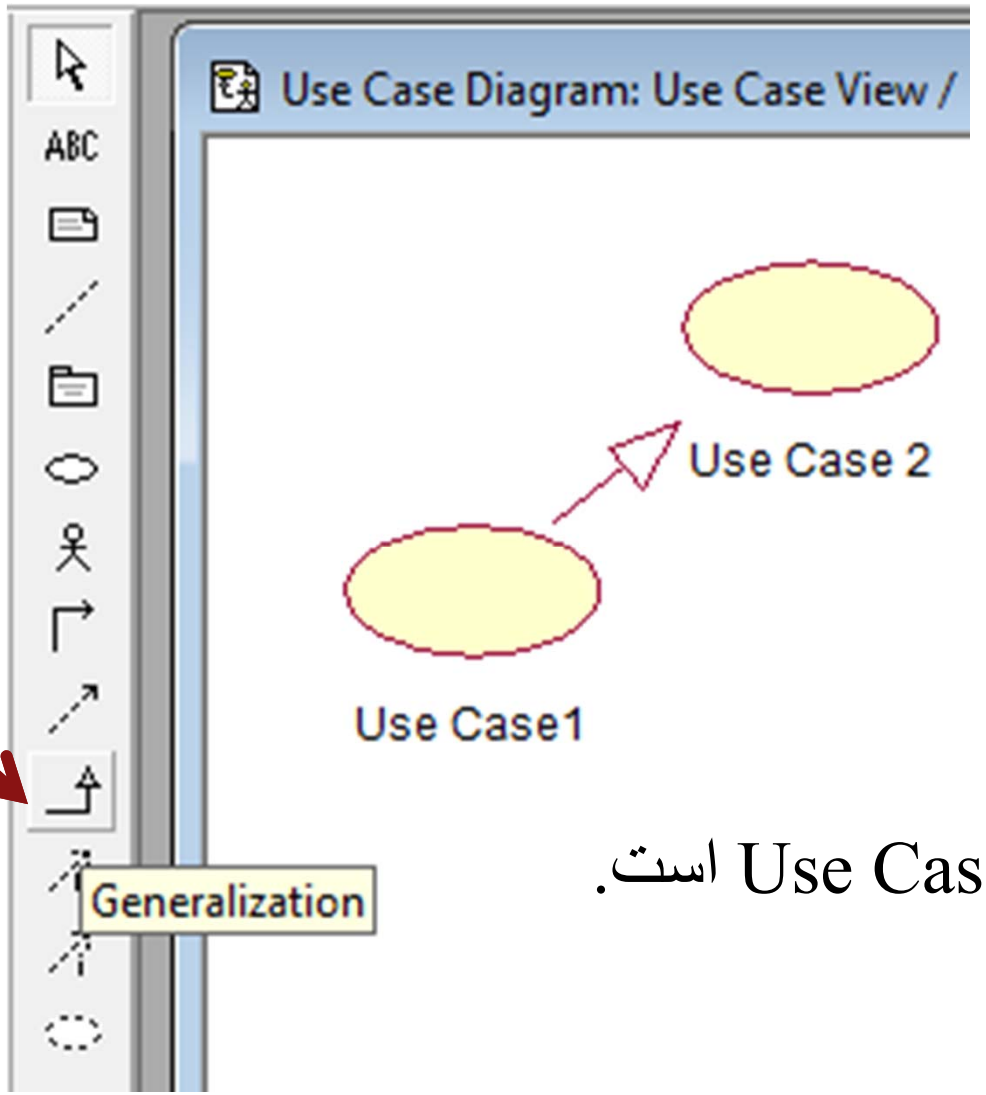
generalization

- زمانی که یک Use Case حالت کلی یک یا چند Use Case دیگر باشد ، رابطه ی وراثت میان آنها تعریف می گردد.

- بعنوان مثال در سیستم بانک *افتتاح حساب* یک Use Case عمومی است که خود در بر دارنده انواع افتتاح حساب مانند *افتتاح حساب جاری* ، *افتتاح حساب قرض الحسنه* می باشد.

- Use Case های فرزند مفهوم تعریف شده در Use case پدر را مورد استفاده قرار می دهند.

نماد رابطه Generalization در ational Rose



Use Case1 ، نوعی از Use Case2 است.

Activity Diagram

❑ اجزای نمودار فعالیت:

- Activity
- انتقال یا گذر (Transition)
- نقاط تصمیم
- نقاط آغازین و پایانی
- همگام سازی
- Swim lane

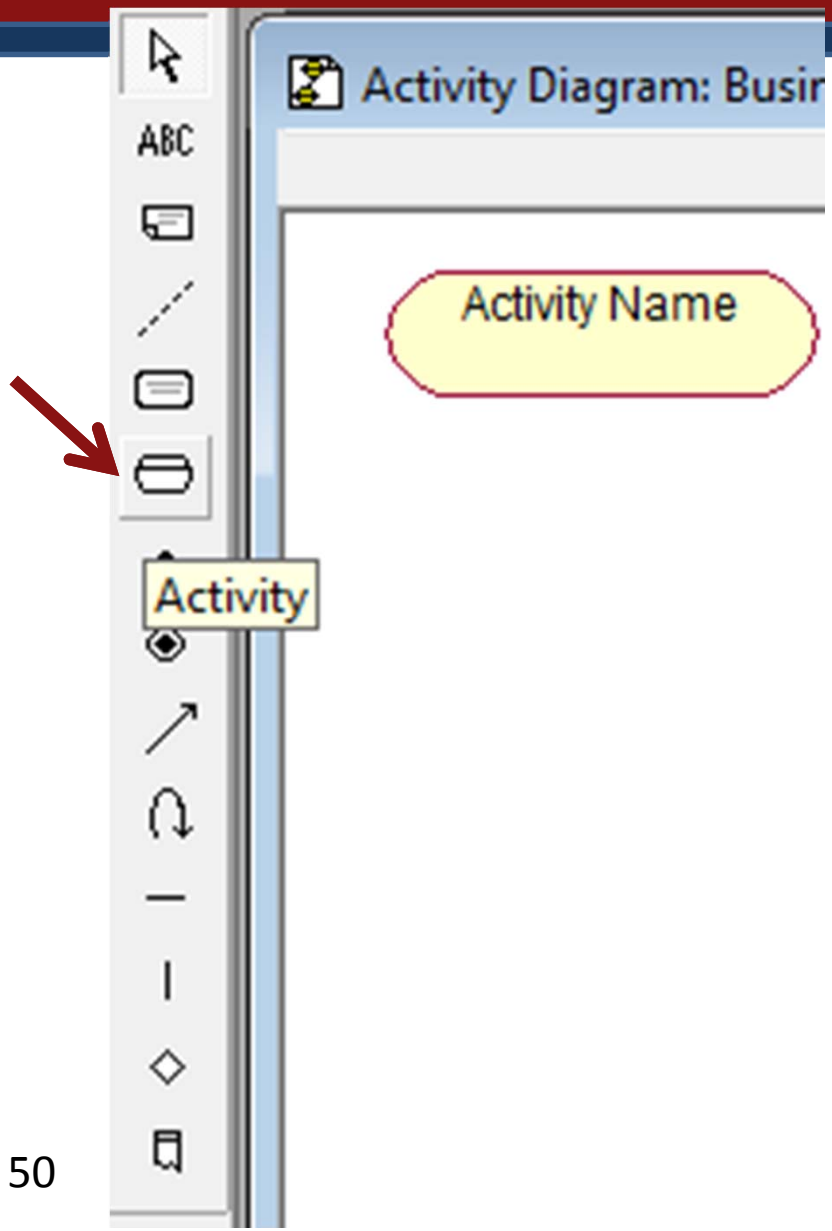
Activity

- نموار فعالیت مشابه فلوچارت است که برای نشان دادن جریان کار سیستم بکار می رود. (نمایش کنترل از یک فعالیت به فعالیت دیگر)

- هر فعالیت نشان دهنده ی یک عملکرد خاص است که در سیستم انجام می شود.

- در Rational Rose فعالیت با بیضی نشان داده می شود که نام فعالیت درون آن نوشته می شود.

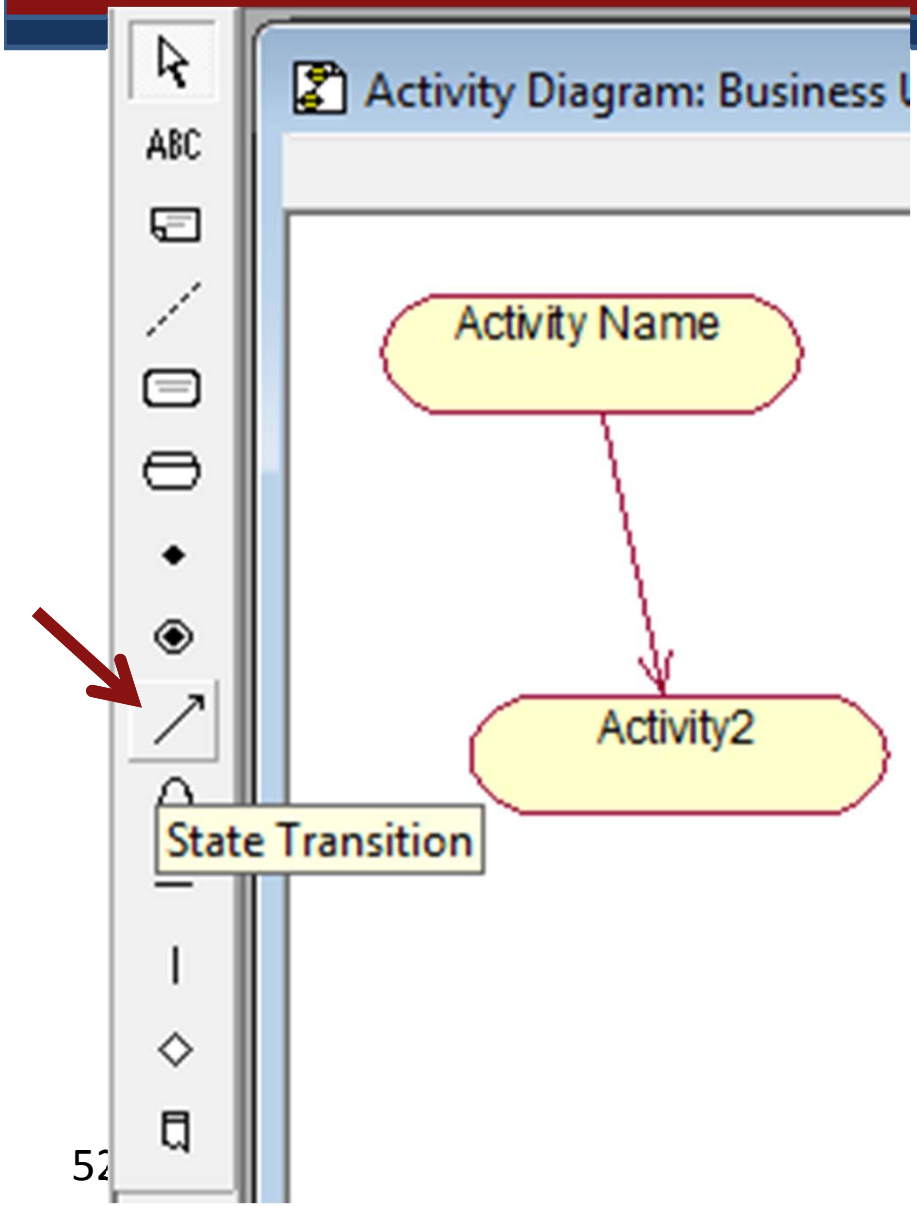
نماد Activity در Rational Rose



انتقال یا گذر (Transition)

- برای نمایش جریان کنترل از یک فعالیت به فعالیت دیگر بکار می رود
- در Rational Rose گذر با یک فلش نمایش داده می شود.(ابزار State Transition)

نماد Transition در Rational Rose

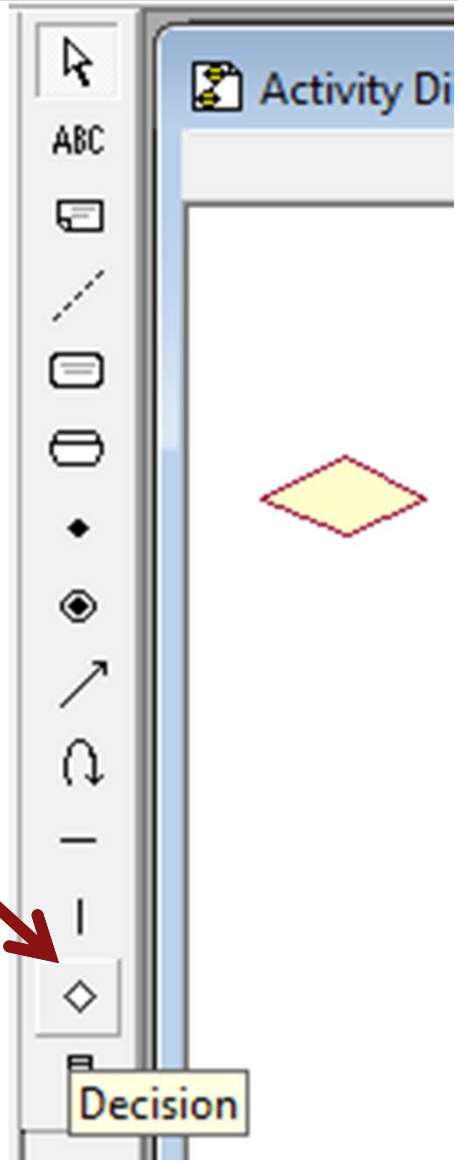


نقاط تصمیم

- به منظور شاخه شدن جریان کنترل در شرایط مختلف از ساختار تصمیم استفاده می شود. در واقع برای بررسی شرط های سیستم از این عنصر استفاده می شود.

- نقاط تصمیم در Rational Rose با لوزی نمایش داده می شود.

نماد نقاط تصمیم در Rational Rose

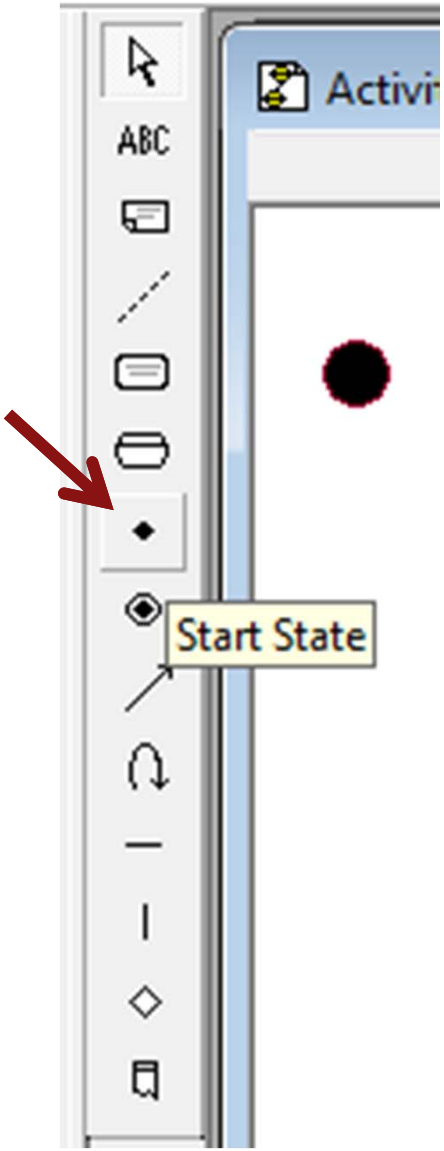


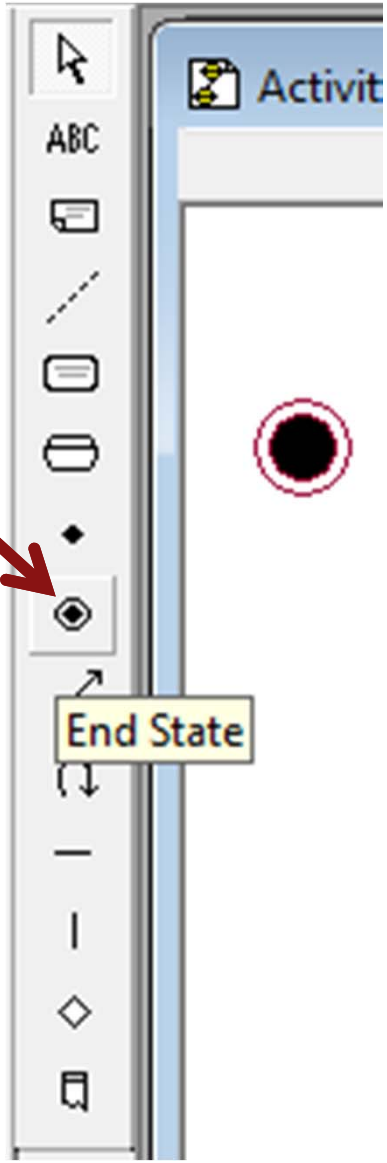
نقاط آغازین و پایانی

- برای نمایش نقاط شروع و پایان مجموعه فعالیتها بکار می رود.

- هر جریان کاری(مجموعه فعالیتها)، دارای یک نقطه ی شروع می باشد. همچنین هر جریان کاری می تواند در بخشهای مختلف و در شرایط گوناگون خاتمه یابد.

نماد نقاط شروع و پایان در Rational Rose



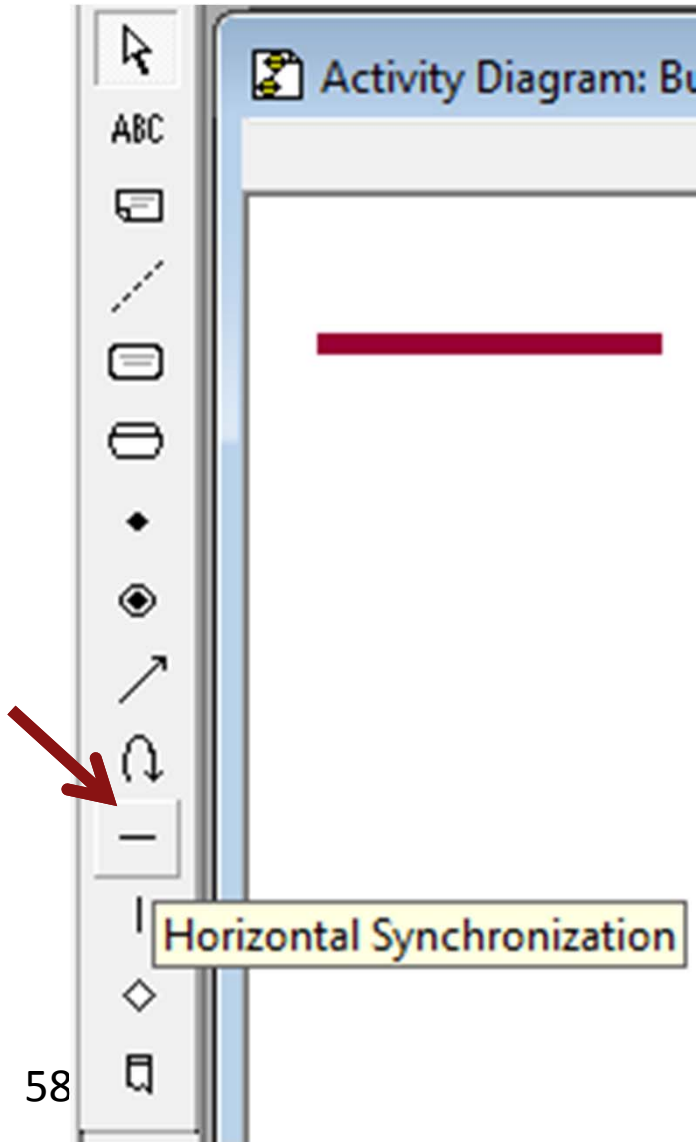


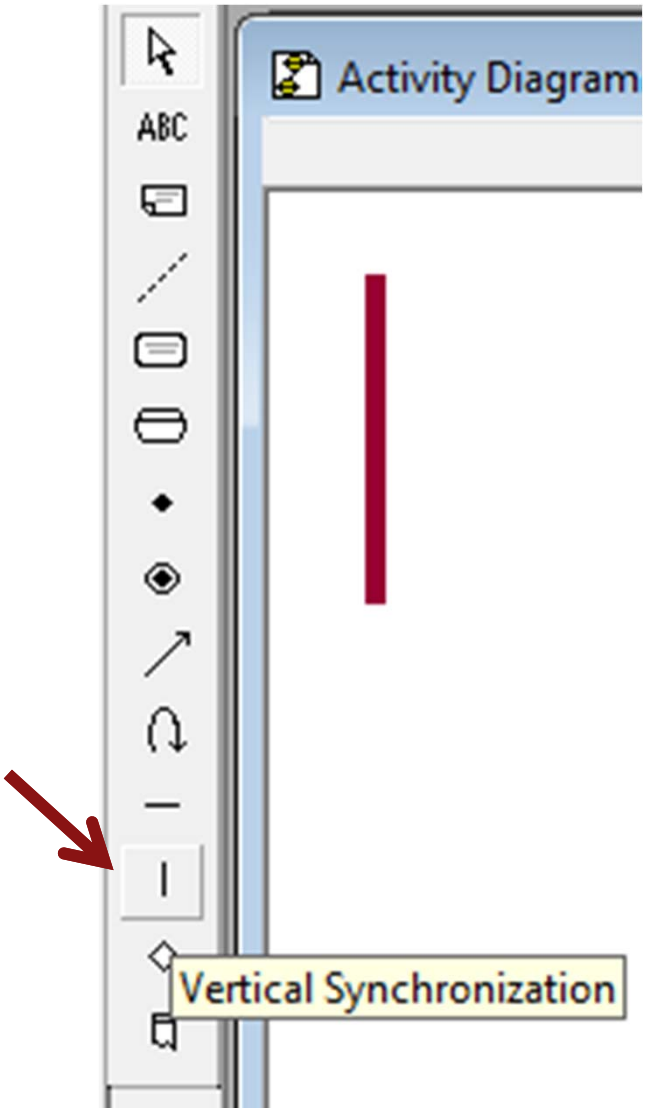
همگام سازی (Synchronization)

- برای نمایش فعالیتهایی که بطور موازی انجام می شود بکار می رود.

- در Rational Rose از میله های همگام سازی افقی یا عمودی (Horizontal synchronization یا Vertical synchronization) استفاده می شود.

نماد همگامسازی در Rational Rose

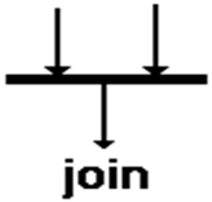




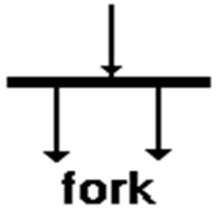
انواع همگام سازی (Synchronization)

- دو نوع همگامسازی مطرح می شود:

  - Join: با اتمام چند فعالیت، یک فعالیت شروع می شود.

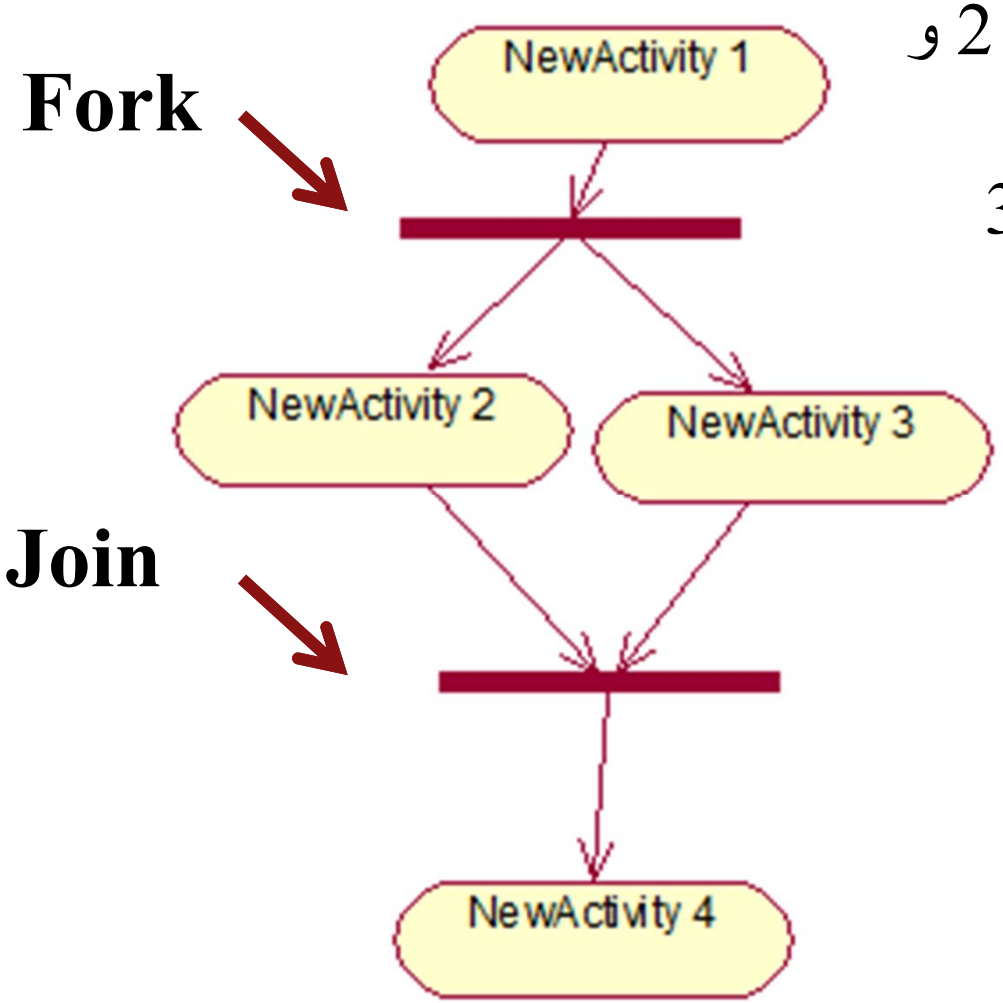


  - Fork : با اتمام یک فعالیت، چند فعالیت شروع می شوند.



مثال

پس از انجام فعالیت 1، فعالیتهای 2 و 3 بطور همزمان انجام می شود.
همچنین پس اتمام فعالیتهای 2 و 3 فعالیت 4 انجام می شود.

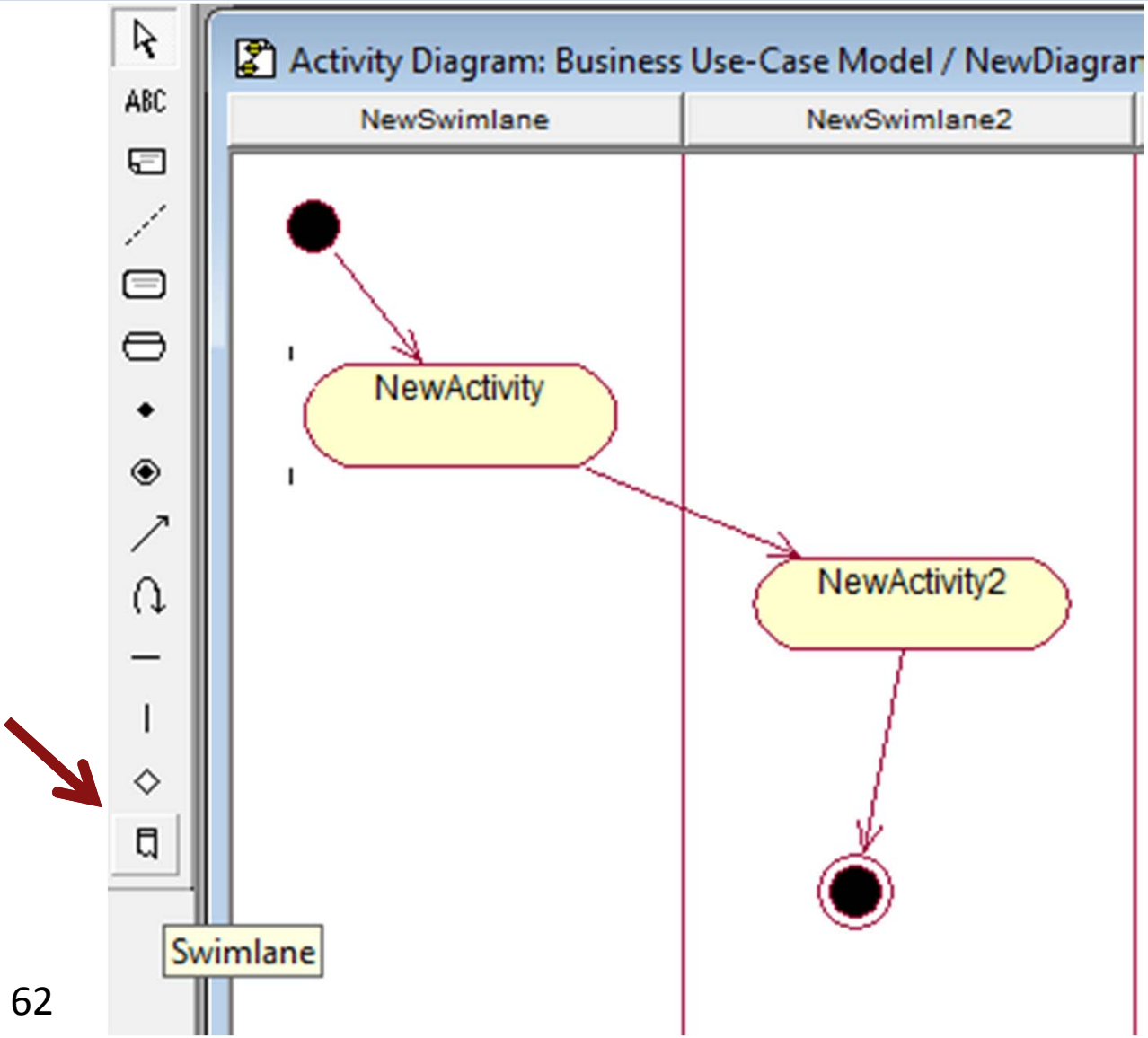


Swim lane

- برای قطعه بندی نمودار فعالیت مورد استفاده قرار می گیرد. منظور از قطعه بندی نمودار آن است که مشخص می کند چه عنصری مسئول یک فعالیت است.

- در Rational Rose از ابزار Swimlane برای این منظور استفاده می شود که با یک خط عمودی نمایش داده می شود. نام مسئول هر فعالیت در بالای خطوط نوشته می شود.

نماد swimlane در Rational Rose



Sequence & Collaboration Diagrams

- نمودارهای توالی (Sequence) و همکاری (Collaboration) تعاملات میان اشیاء سیستم را نمایش می دهند. (همانطور که می دانیم اشیاء از طریق ارسال پیام با یکدیگر در تعامل اند).

- تفاوت این دو نمودار آن است که نمودار توالی، ترتیب زمانی ارتباطات میان اشیاء را بیان می کند، درحالیکه نمودار همکاری، صرفا ارتباطات اشیاء را بدون توجه به ترتیب زمانی ارسال پیامها، نمایش می دهد.

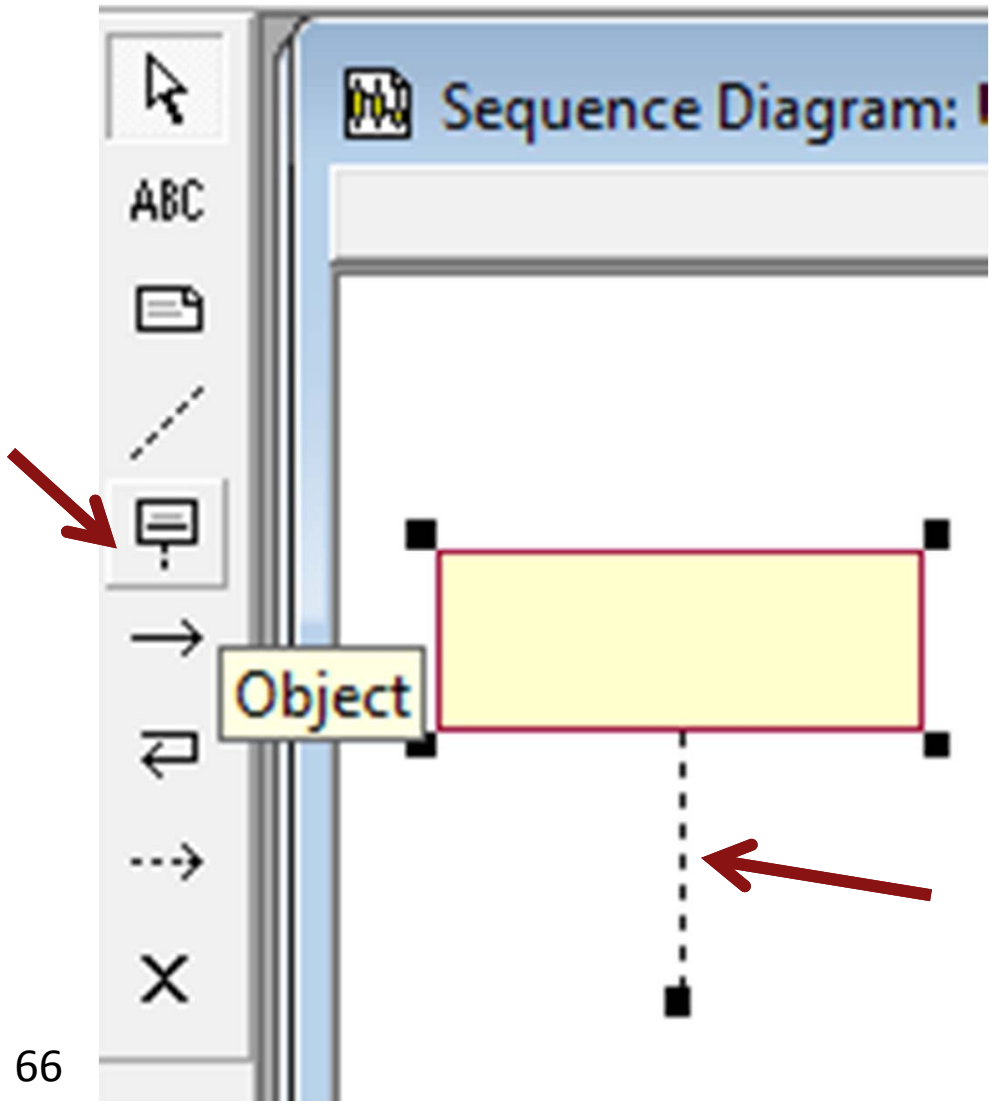
اجزای نمودار های همکاری و توالی

- Object: اشیاء همان نمونه های یک کلای اند و در این نمودارها با یک مستطیل نمایش داده می شود که نام شئ بهمراه نام کلاس آن درون مستطیل ذکر می گردد.

نکته: در نمودار توالی در انتهای مستطیل خط زمان (time line) وجود دارد که زمان حیات شئ را در ارتباط میان سایر اشیاء نشان می دهد.

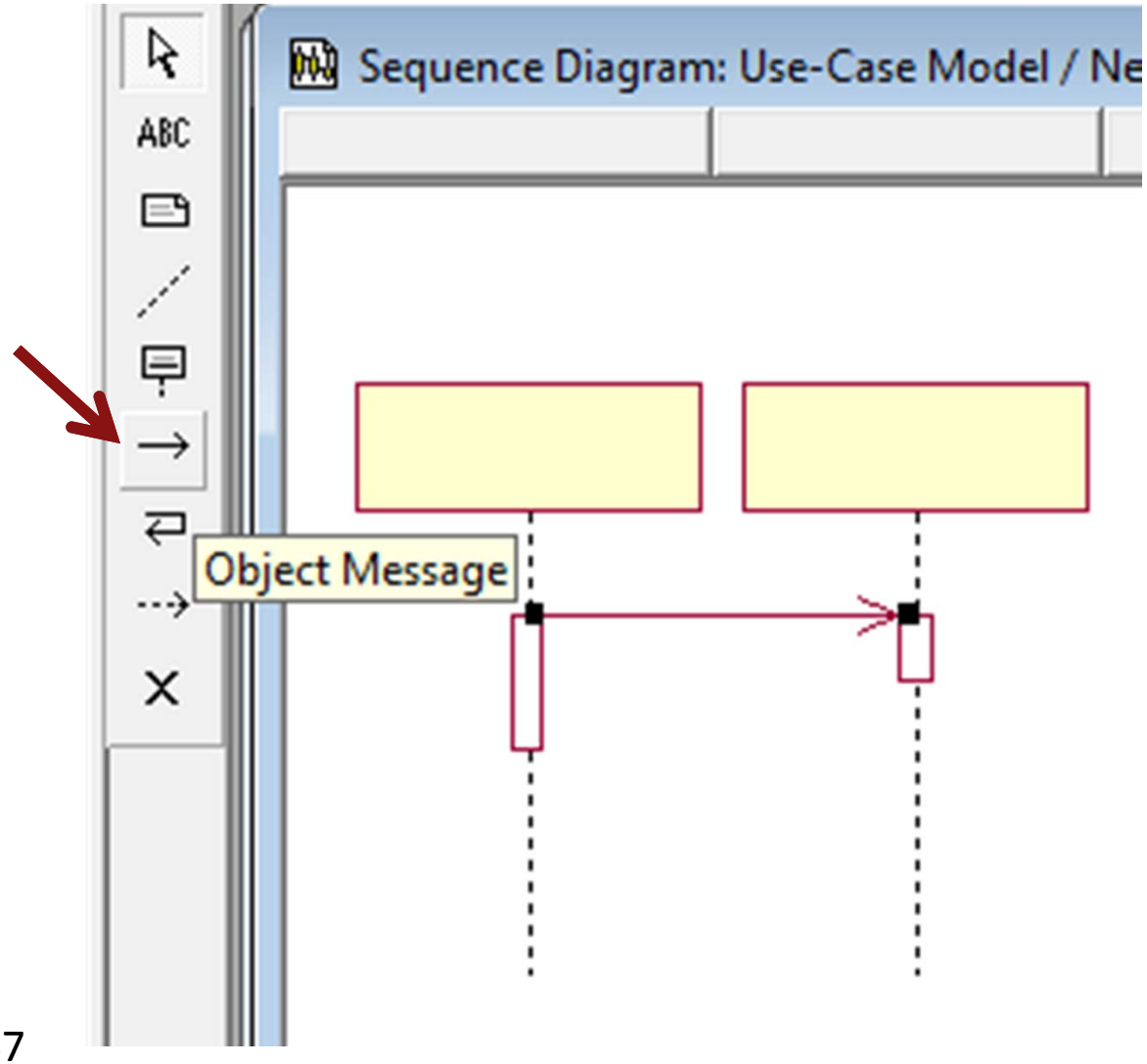
- Message:ارتباط میان اشاء سیستم است که با فلش نشان داده می شود و از سمت فرستنده به گیرنده رسم می شوند.

نماد Object در Rational Rose



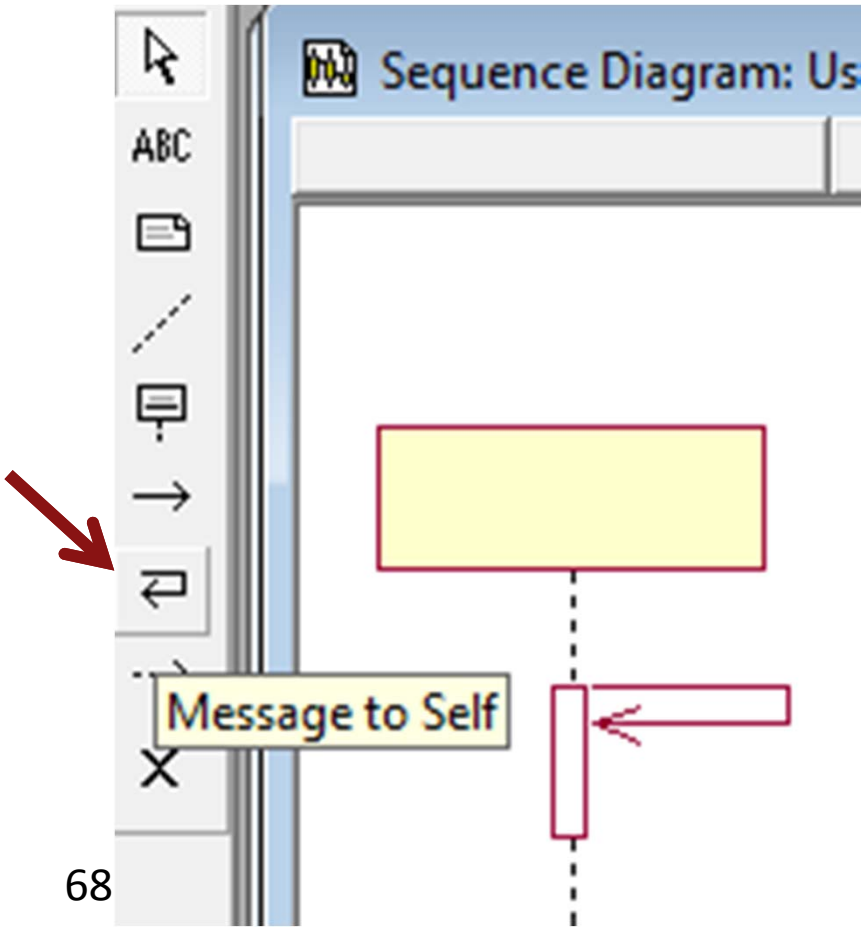
Time Line

نماد Message در Rational Rose



نکات

- یک شئ می تواند به خودش پیام ارسال کند. به عبارت دیگر یک شئ می تواند متدهای خود را فراخوانی کند.



نکات (..)

- در نمودار توالی پیامهایی که در بالا ی نمودار رسم شده اند نسبت به پیامهای پایین تر زودتر ارسال شده اند.

- در Rational Rose هریک از نمودارهای همکاری و توالی با فشردن دگمه ی F5 از روی دیگری ساخته می شود.

Stereotype

- **Stereotype** یا کلیشه ی یک کلاس به منظور گسترش نمودارهای UML مطرح می شود.
- برخی از کلیشه های پر کاربرد کلاس که در نمودار توالی و همکاری مورد استفاده قرار می گیرد عبارتند از:
  - **Boundary**: برقرار کننده ی تعامل میان کاربران و سیستم
  - **Control** : کنترل کننده ی اشیاء و رفتارهای سیستم
  - **Entity**: ذخیره کننده ی ساختمان داده های منطقی سیستم

موفق باشید