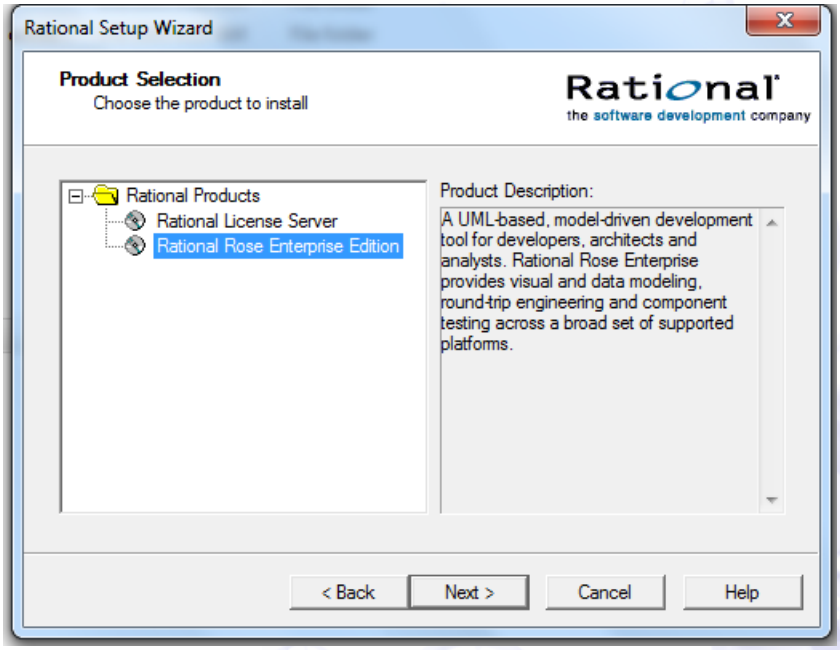
مهندسی نرم افزار

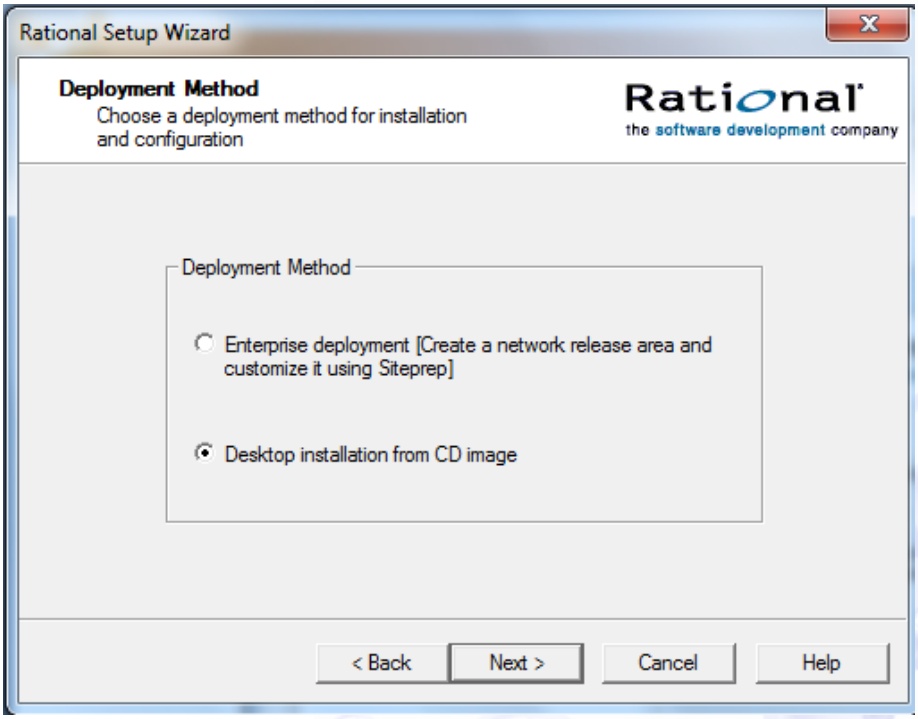
فصل پنجم: آشنایی با Rational Rose

مدرس: اسماعیل نورانی

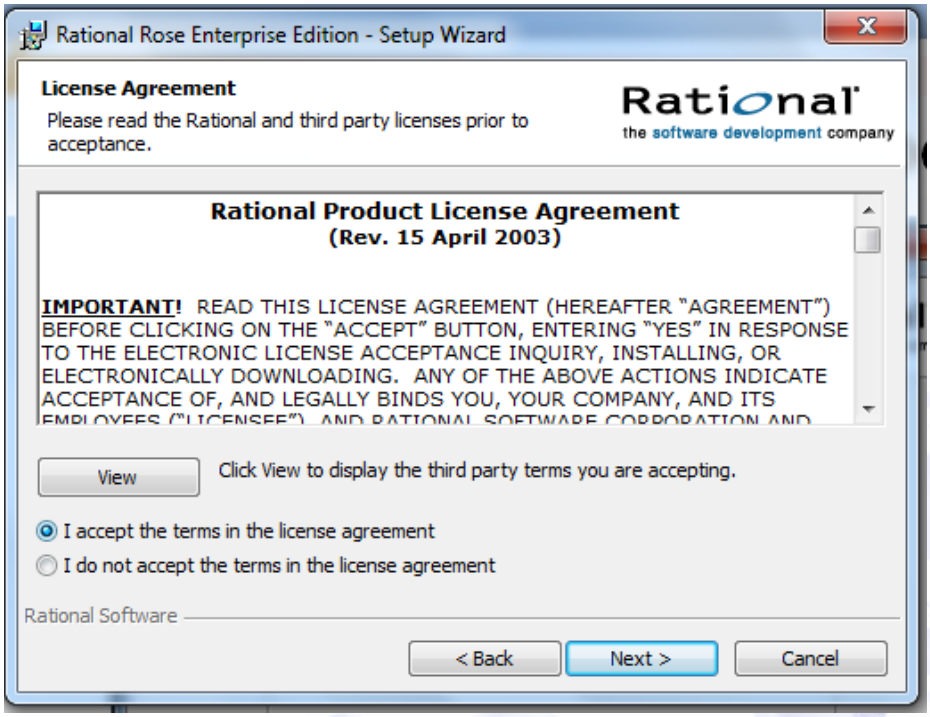
نصب Rational Rose 2003



Rational Rose 2003 نصب

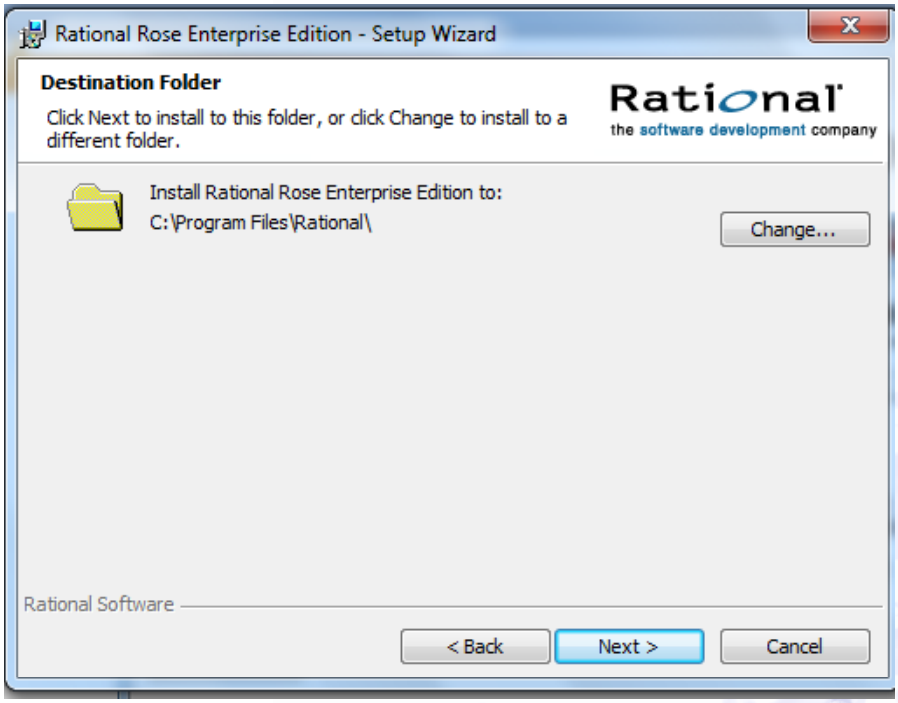


Rational Rose 2003 نصب

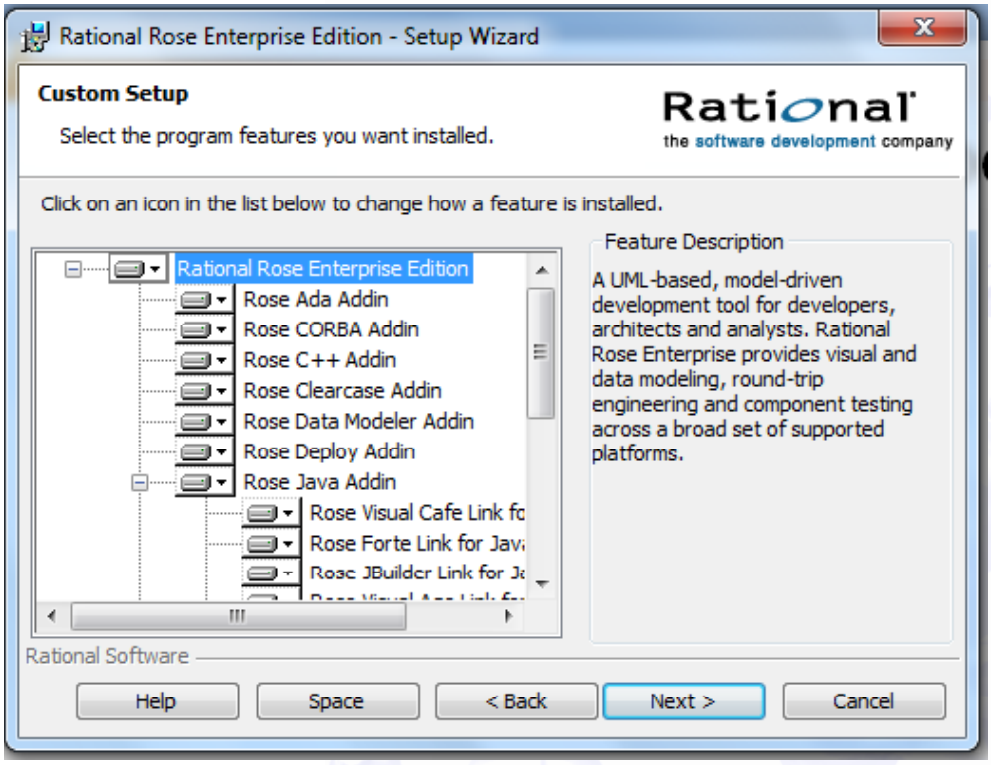


Rational Rose 2003 نصب

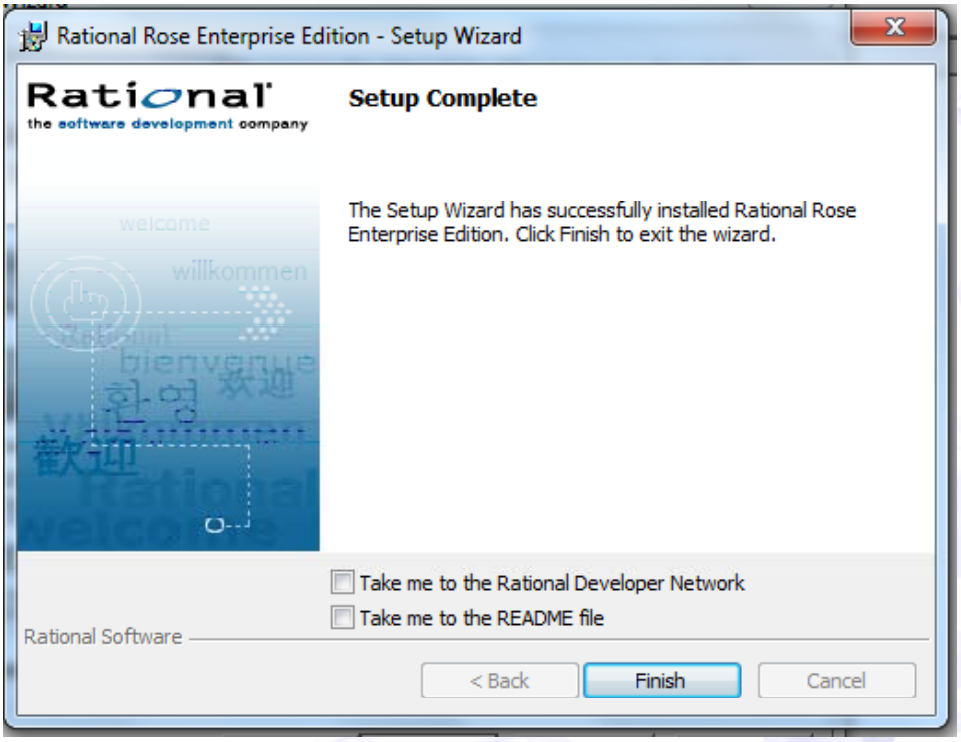
Rational Rose 2003 نصب



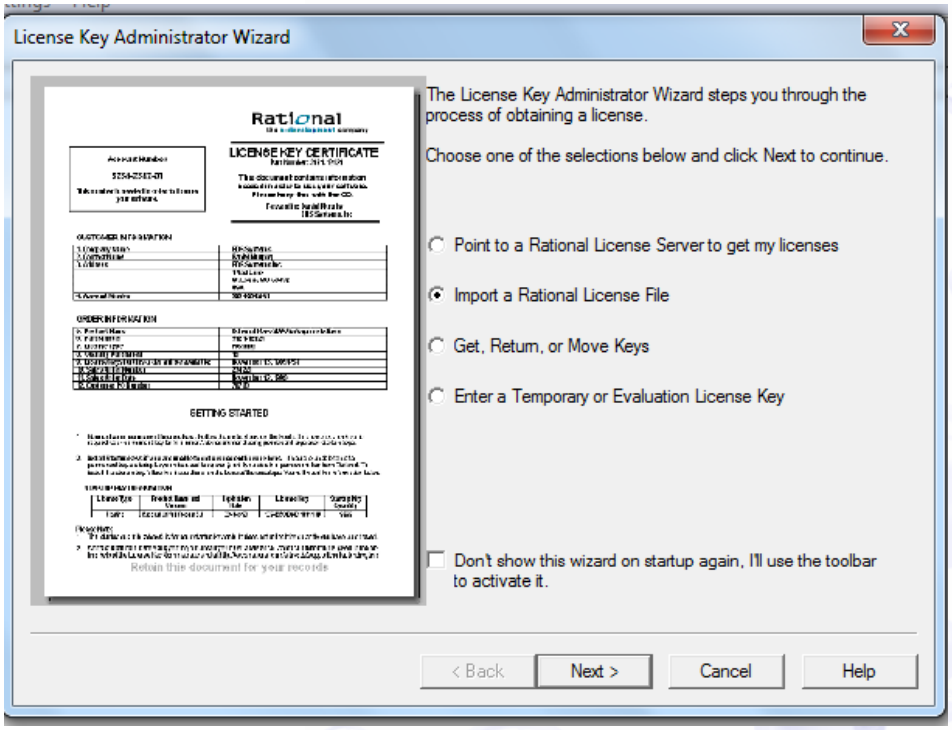
Rational Rose 2003 نصب



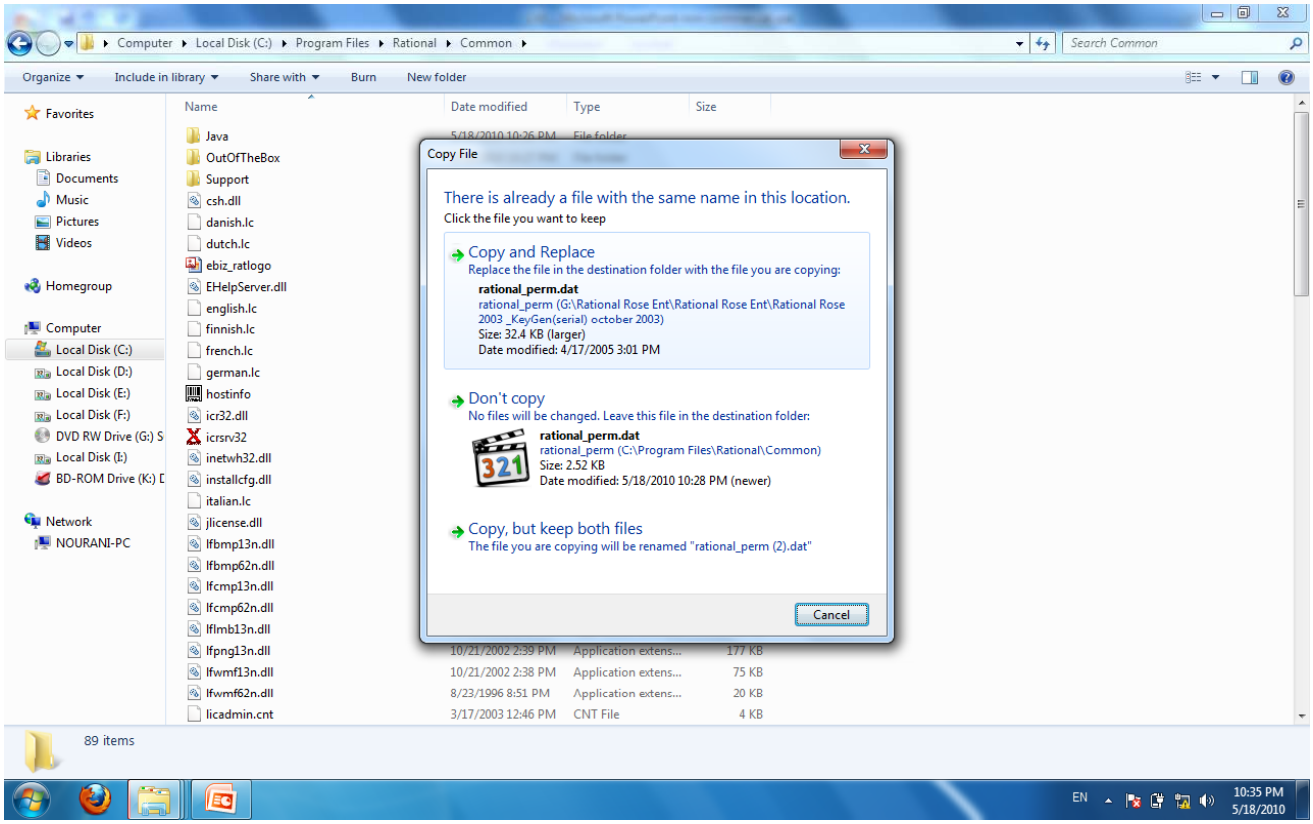
Rational Rose 2003 نصب



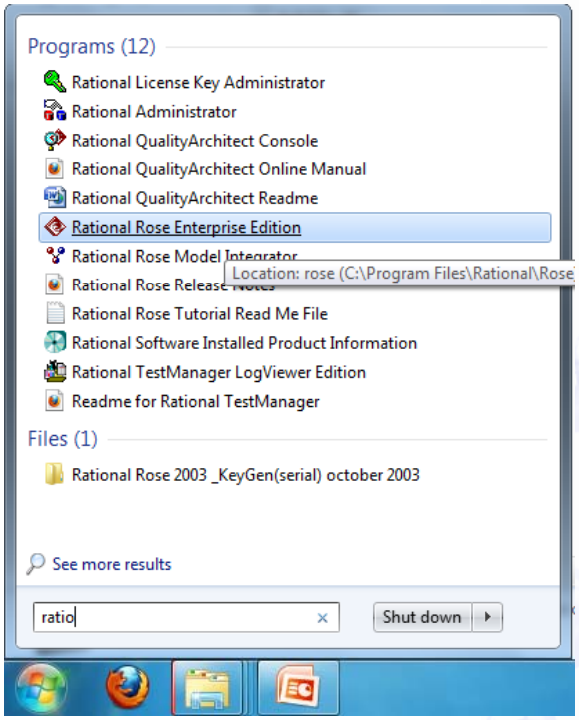
Rational Rose 2003 نصب

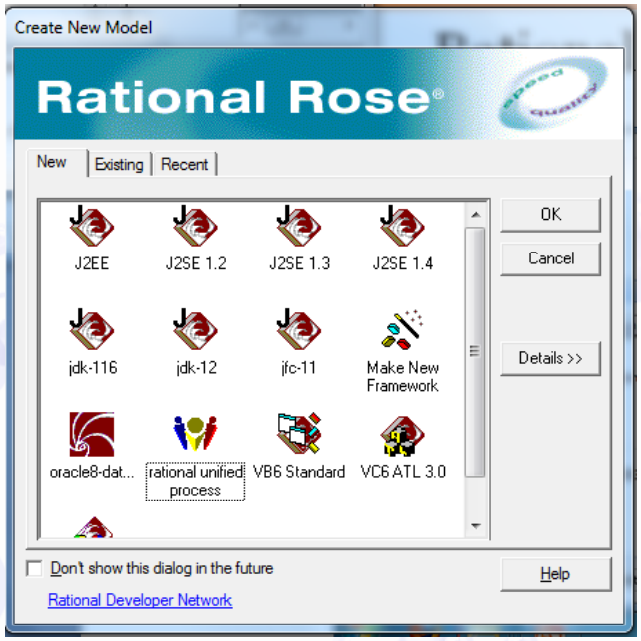


Rational Rose 2003 نصب



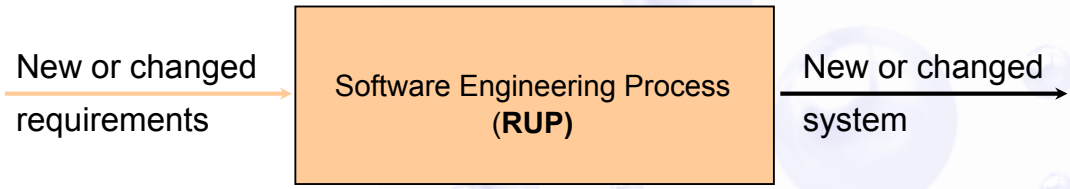
Rational Rose 2003 نصب





UML *vs.* RUP

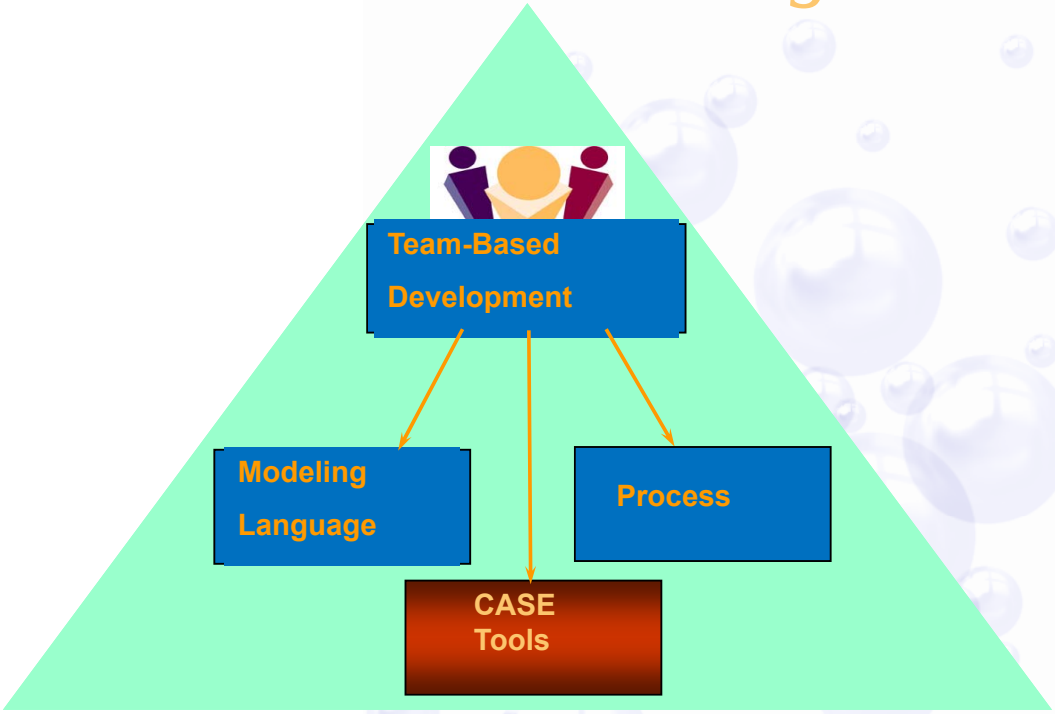
- RUP defines *Who* does *What*, *How*, and *When*



- UML provides a standardized graphical notation to represent Software Engineering Process Artifacts

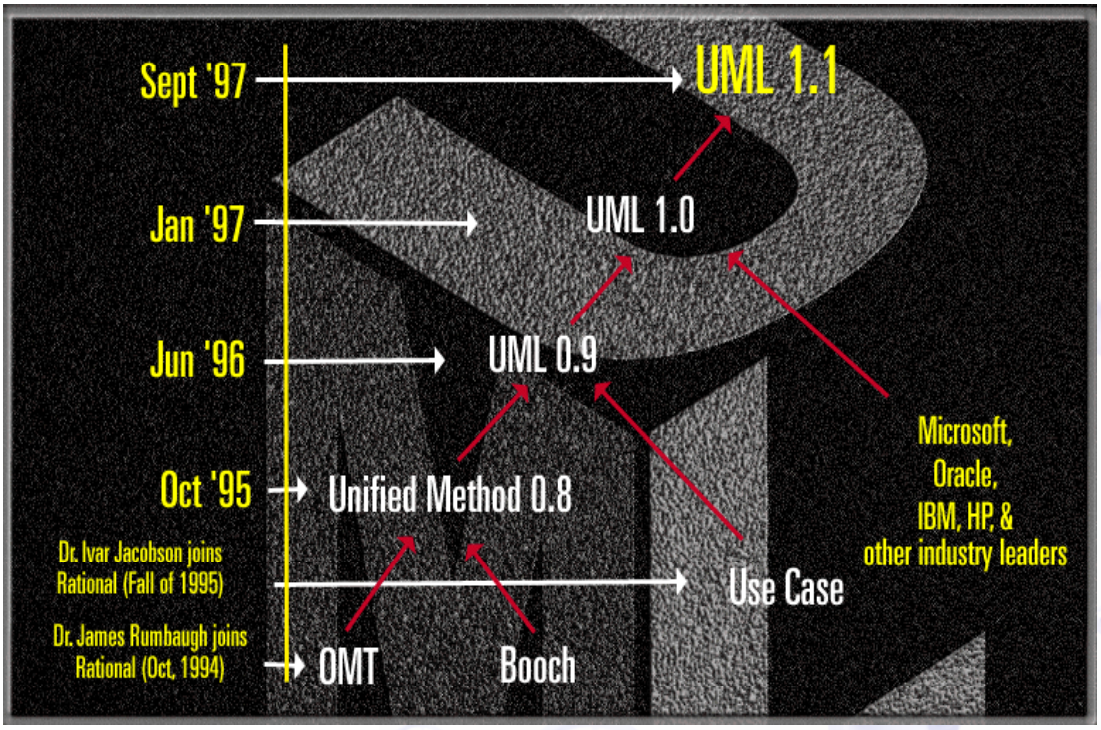
For Building a System -

UML Is Not Enough!

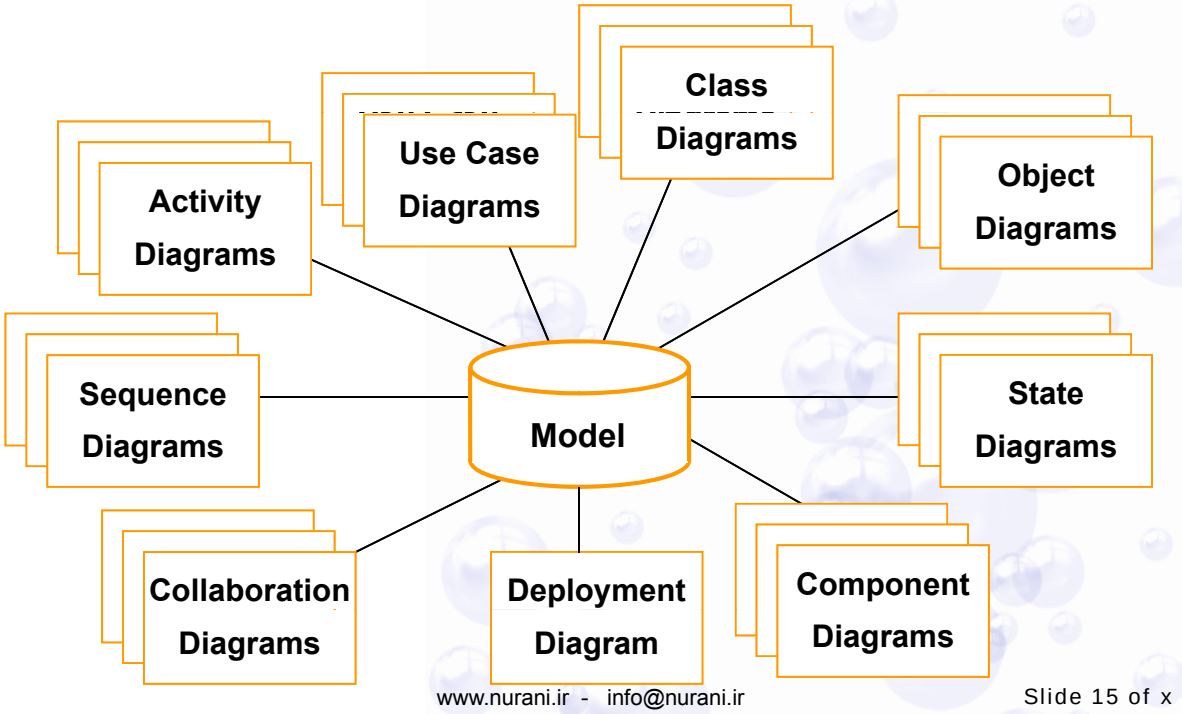


CASE Tools Example: Rational Suite

UML History



The UML Provides Standardized Diagrams



دیدگاه های مختلف سیستم در UML

1- Use Case View:

- Requirements and Features - Analyst

2- Design View:

- Problem and Solution - Designer and Programmer

3- Process View:

- Multithreading - Programmer

4- Implementation View:

- Technologies - Programmer

5- Deployment View:

- Hardware and Topology - Designer and Technologist

نکته: بر حسب نوع برنامه، ممکن است بعضی view ها مهمتر باشند.

دیدگاه های مختلف سیستم در Rational Rose

1- Use Case View:

→ **Use Case View in UML Standard**

2- Logic View:

→ **Design View and Process View in UML Standard**

3- Component View:

→ **Implementation View in UML Standard**

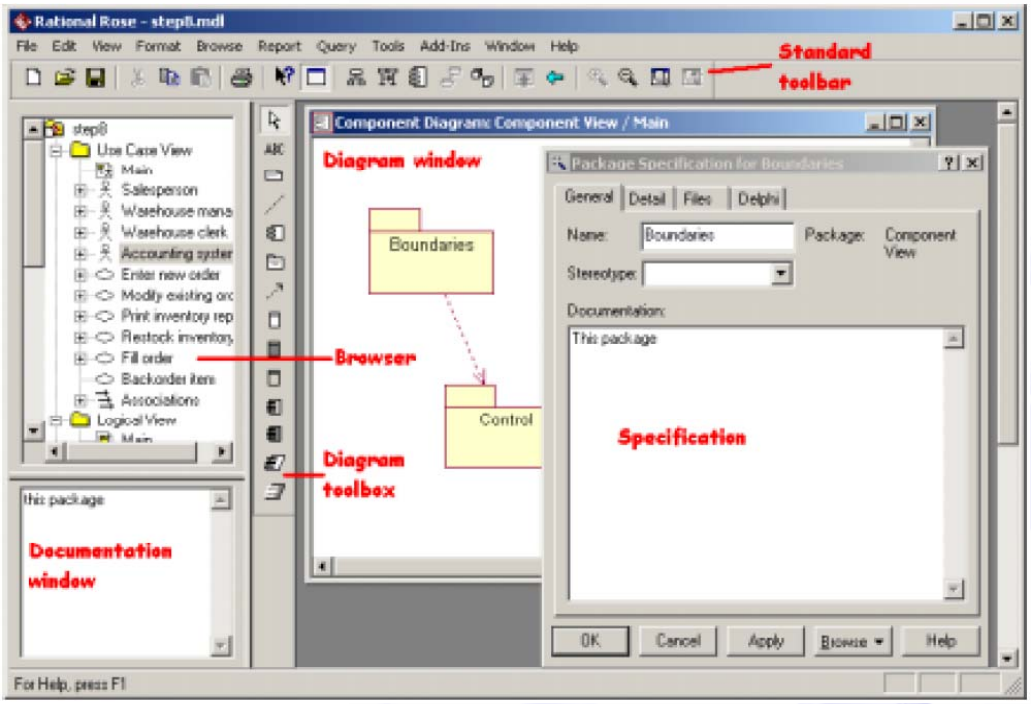
4- Deployment View:

→ **Deployment View in UML Standard**

View های مختلف Rational Rose

Usecase view -
Logical view -
Component view -
Deployment view -

محیط RationalRose



عناصر تشکیل دهنده محیط RationalRose

۱- **Standard toolbar** ، که برای تمام دیاگرام‌ها مشترک است و در قسمت بالای پنجره واقع است.

۲- **Diagram toolbar** ، که وابسته به پنجره‌ی دیاگرام فعال است و در سمت چپ پنجره‌ی دیاگرام واقع است.

۳- **Browser** ، به شما اجازه می‌دهد تا بصورت یک ساختار درختی دیاگرام‌های موجود و عناصر مدلهایتان را مشاهده کنید.

۴- **Diagram window** ، ساخت و ویرایش دیاگرام‌ها در این قسمت صورت می‌پذیرد.

۵- **Documentation window** ، به شما اجازه می‌دهد تا به مدلهایتان مستندات لازم را نیز اضافه نمایید. می‌توانید مستنداتتان را در این قسمت یا در قسمت specification ویرایش نمایید.

۶- **Specification** ، محیط ویرایشی برای اضافه کردن مستندات به مدل.

The **Tools** Menu

- Under the **Tools** menu item, can:
  - Generate Code in
    - Ada
    - Java
    - Oracle8
    - C++
    - XML_DTD
  - Reverse Engineer Models from Code
  - ...

Use Case View

- این دید که تشریح رفتار سیستم از دید کاربر است و فعل و انفعالات متقابل Actor ها و مورد های استفاده را نمایش می دهد شامل ۴ نمودار زیر است:

دیاگرامهای موردهای استفاده (usecase diagrams)

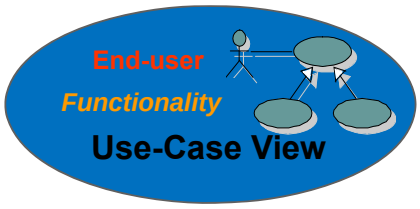
دیاگرامهای توالی (sequence diagrams)

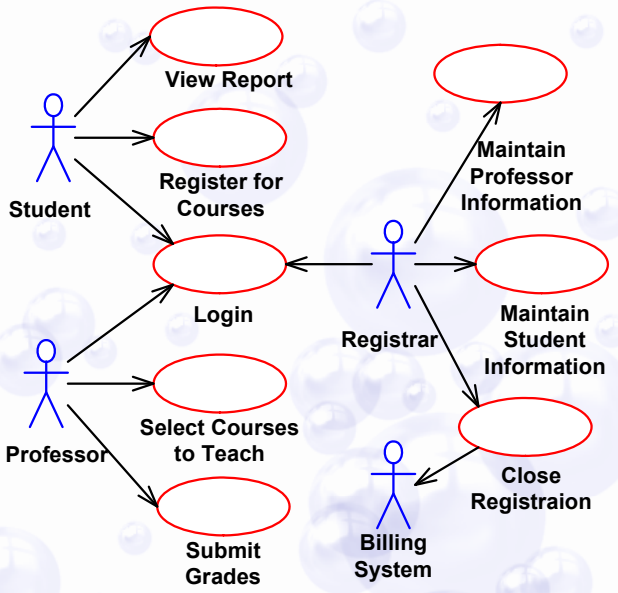
دیاگرامهای همکاری (collaboration diagrams) (اختیاری)

دیاگرامهای فعالیت (activity diagrams) (اختیاری)

Use-Case View

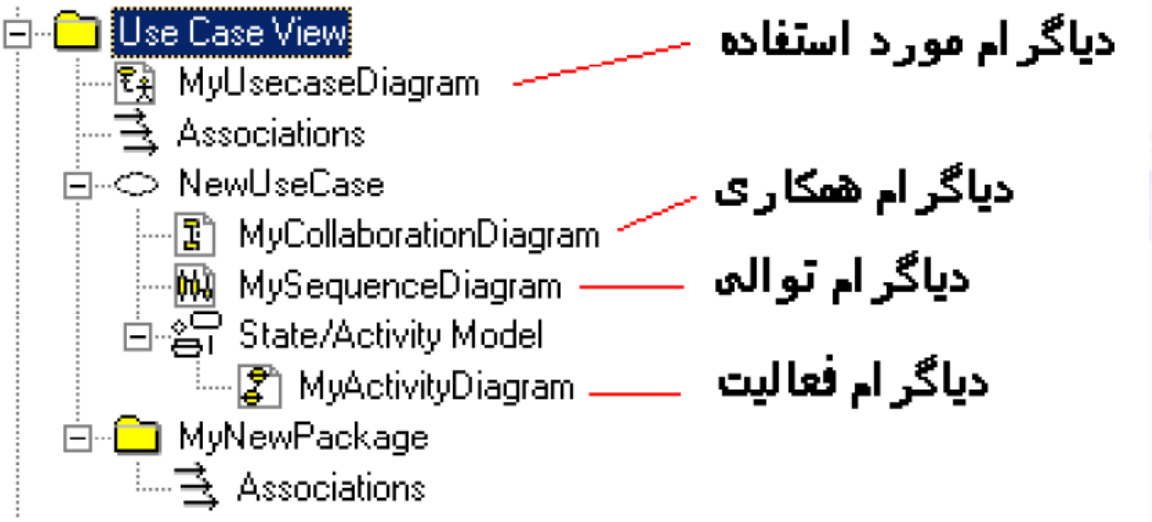
Describes What the system must do





A Use Case Diagram

Use Case View

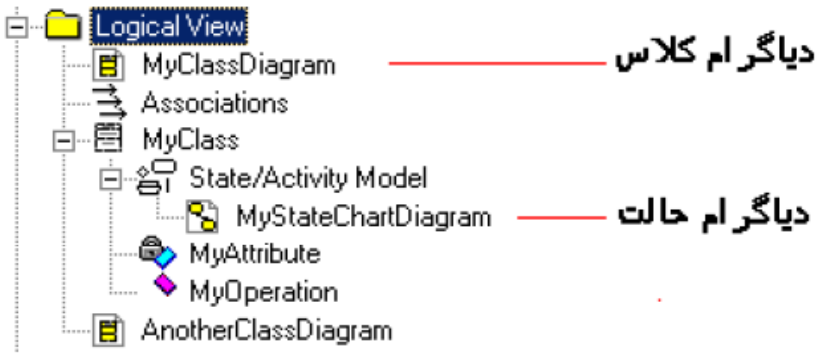


Logical View

- این دید شامل نیازمندیهای عملیاتی سیستم از دید طراح و تحلیلگر می باشد. شامل دو نمودار زیر است.

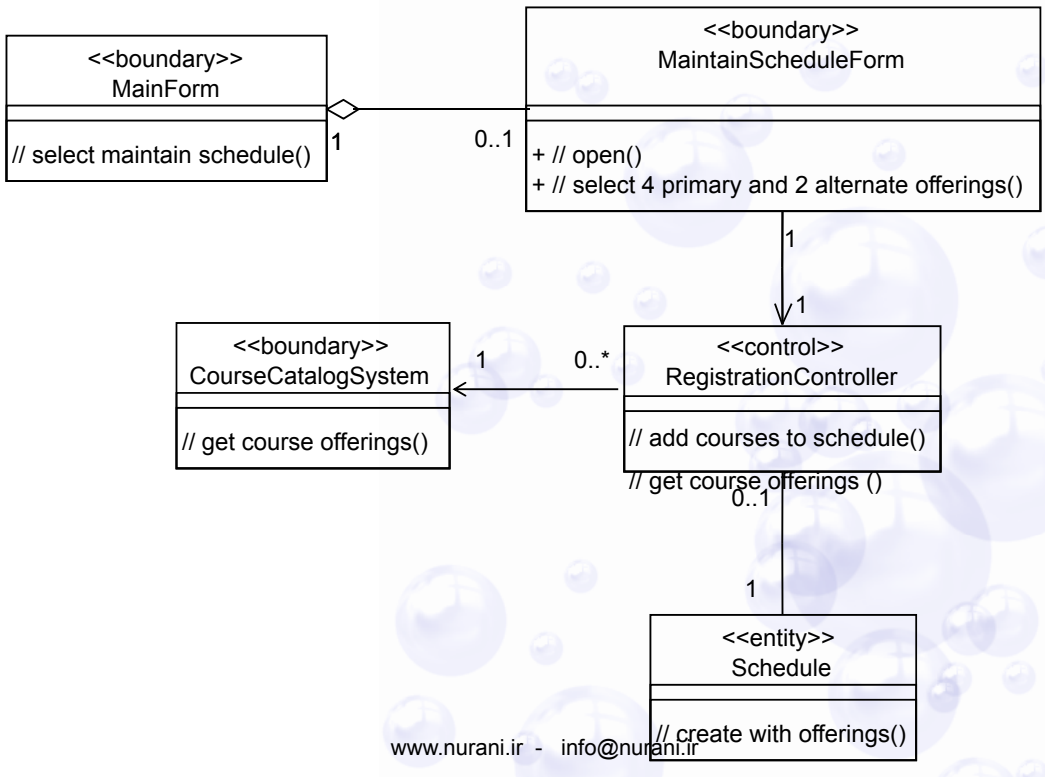
دیاگرامهای کلاسها (class diagrams)

دیاگرامهای حالت (statechart diagrams)



Logical View

Describes HOW the system will do...



Component View

- سازماندهی مولفه های تشکیل دهنده سیستم نرم افزاری را در قالب نموداری به نام Component Diagram نمایش می دهد.
- مولفه هایی همچون:
  - source code files, data files, components, executable, etc.

Deployment View

- وظیفه System Engineer می باشد
- فقط در سیستمهای توزیع شده کاربرد دارد.
- چگونگی نگاشت مولفه های نرم افزار روی اجزای سخت افزاری سیستم در قالب Deployment Diagramنمایش می دهد.

Rational Rose

- در حقیقت Rational Rose از طریق نمودارها و عناصر زیر کمک می کند تا به مدلسازی رفتار سیستم بپردازیم:
  - Use Case Diagram
  - Class Diagram
  - Collaboration Diagram
  - Sequence Diagram
  - State-chart Diagram
  - Activity Diagram
  - Component Diagram
  - Packages
  - Deployment Diagram

Diagrams : Class Diagram

تعریف: نمودار کلاس، نموداری است که مجموعه کلاس ها، نحوه تعامل آنها و ارتباط بین آنها را تشریح می کند.

- در قسمت Logical view تهیه میشود.
- جزو Structural Modeling میباشد
- جنبه Static سیستم را مدل میکند

Class Diagram اجزای اصلی تشکیل دهنده

- Class
- Attributes
- Operations
- Relationships
  - Associations
  - Generalization
  - Dependency
  - Realization
- Constraint Rules and Notes

Finding Classes

- Do we have that should be stored or analyzed ?
- Do we have external system ? external system is modeled as class
- Do we have any patterns, class libraries, components, and so on ?

Finding Classes

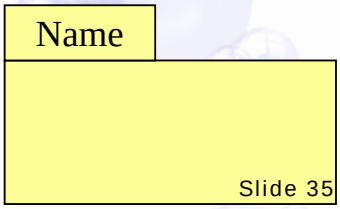
- Classes are usually derived from the following real-world items :
  - Tangible or "real-world" things: book, course
  - Roles: library member, student
  - Events: arrival, leaving, request (help find associations)

Finding Classes

- Special Class – Actors as Classes
  - interact with system, but are not a part of the system itself
  - *black box* to the system : internal implementations are irrelevant
  - don't have to implement it or understand its internal details

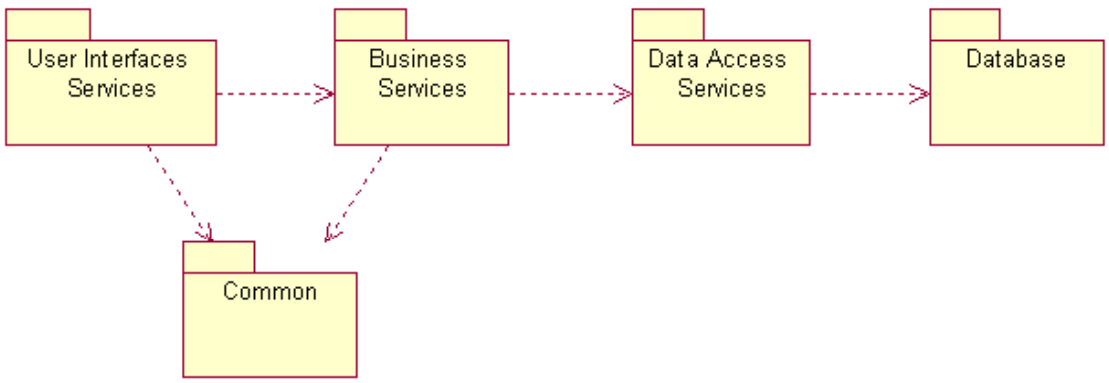
UML Packages

- A package is a general purpose grouping mechanism.
  - Can be used to group any UML element (e.g. use case, actors, classes, components and other packages.
- Commonly used for specifying the logical distribution of classes.
- A package does not necessarily translate into a physical sub-system.



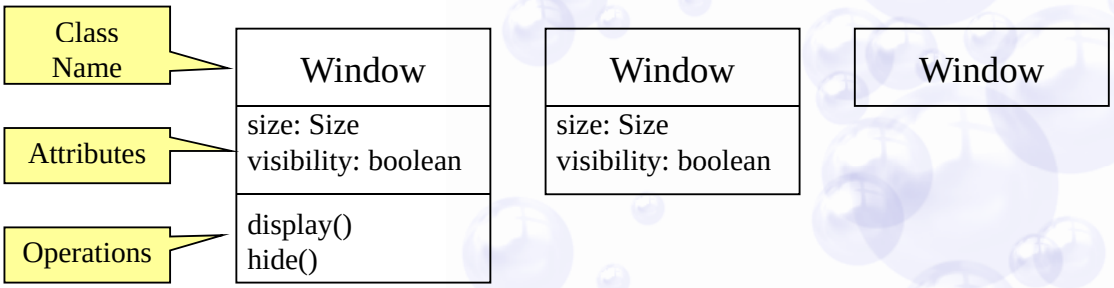
Diagrams : Class Diagram

Package بندي كلاسها:



Classes

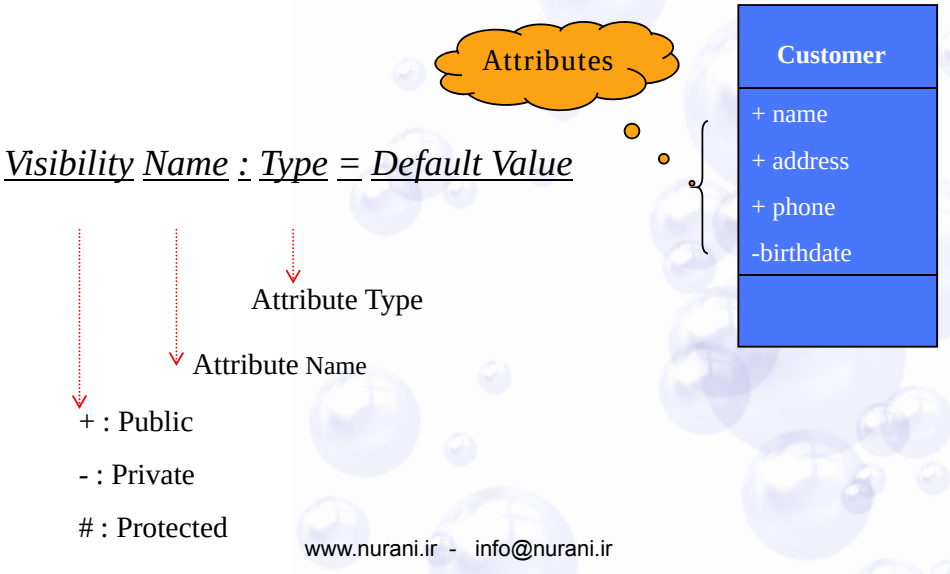
- A class is the description of a set of objects having similar attributes, operations, relationships and behavior.



Class Notation

Attribute

describe the characteristics of the objects



Class Diagram

Visibility: Specify whether attribute or Operation cab be used by another classes.

1- Public:

members of a class are accessible to all clients. This is the default access.

2- Protected:

members of a class are accessible only to subclasses, friends, or to the class itself.

3- Private:

members of a class are accessible only to the class itself or to its friends.

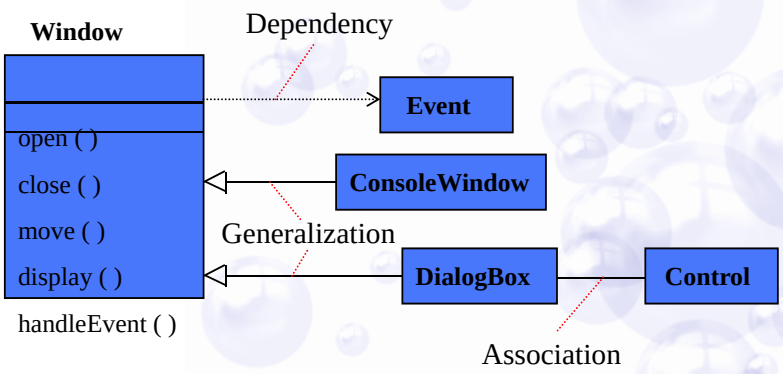
4- Implemented :

the class is accessible only by the implementation of the package containing the class.

- Public (default)
- Protected
- Private
- Implemented

What is a Relationship

- Relationship
  - Relationship Type : Dependency, Generalization, Association

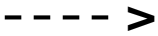


Association

representing either logical relationships or message paths

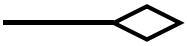
generalization

the abstraction of common features among classes by the creation of a hierarchy of more general classes (superclasses) that encapsulate the common features



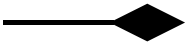
dependency

a relationship between two model elements such that a change in one element may require a change in the dependent element



aggregation

a whole-part association between two or more objects that represent the whole and the other parts of that whole



composition

aggregation where no part can belong to more than one composition at a time and if the composite whole is deleted its parts are deleted with it

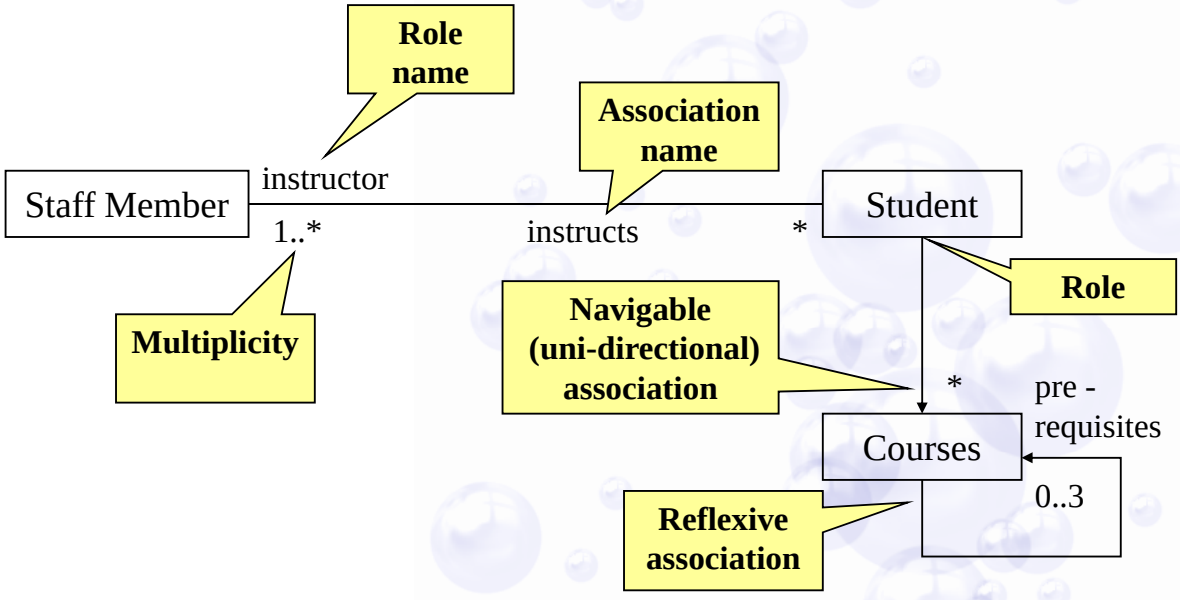
Class Diagram

- An association between two classes indicates that objects at one end of an association "recognize" objects at the other end and may send messages to them.

Association

- Each association has two roles, one for each direction.
  - Role name: verb phrase or noun (responsibility or operation).
- Multiplicity : 0..1; 1 (default); * (0..∞); 1..*; 2,4; 5.
- Navigability: An arrow at the end of the association line indicates that the assoc. can be used in only that direction.

Association



Association

Multiplicity:

- نشان دهنده آن است که در یک لحظه حداکثر چند نمونه از آن کلاس میتوانند وجود داشته باشند.

- Syntax: min .. max
- example:

0 .. 1 بیشتر کلاسهاي ورود اطلاعات از این دسته هستند

1 .. 1 example: main form

0 .. 0 با حالت Abstract (No Instance) فرقدارد

Utility مانند کلاسي که تمام Operation هایش توابع محاسباتي و رشته اي که به آنها Class گویند.

```
Dim m as math
    s = m.sin(90)
```

- حالت هاي N .. 0 ، N .. 1 و N نیز استفاده میشود.

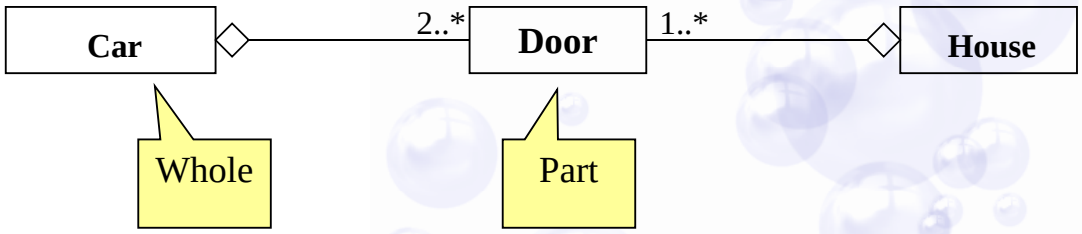
Association

- Multiplicity Indicators

Exactly one	1
Zero or more (unlimited)	* (0..*)
One or more	1..*
Zero or one (optional association)	0..1
Specified range	2..4

Class Diagram

- Aggregation
  - A special form of association that models a whole-part relationship between an aggregate (the whole) and its parts.
  - Models a “is a part-part of” relationship.

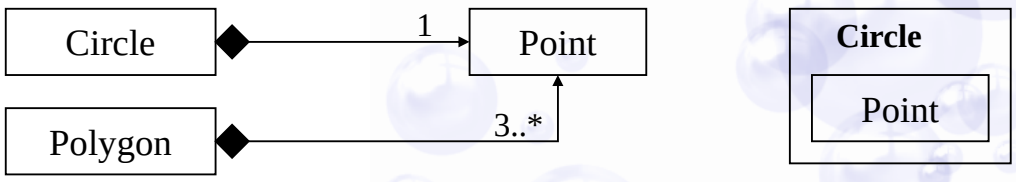


Class Diagram

- Composition

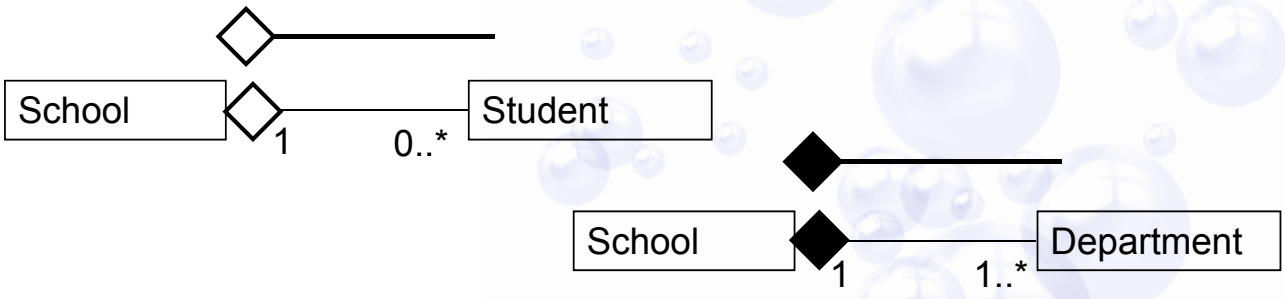
A strong form of aggregation

  - The whole is the sole owner of its part.
  - The life time of the part is dependent upon the whole.



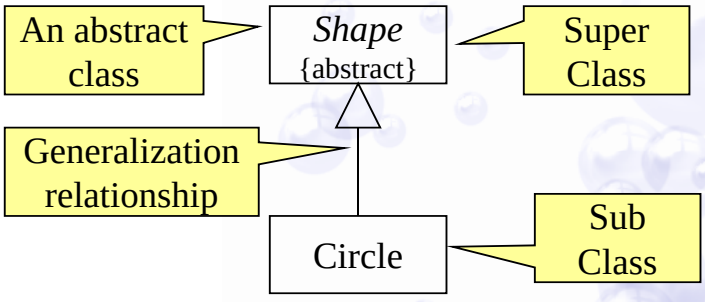
Class Diagram

- Composition and Aggregation:



Class Diagram

- Generalization
  - "is kind of" relationship.



Class Diagram

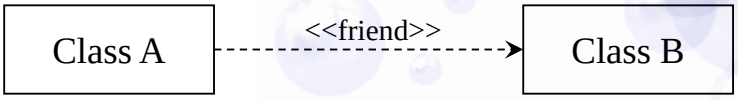
- Generalization
- A sub-class inherits from its super-class
  - Attributes
  - Operations
  - Relationships
- A sub-class may
  - Add attributes and operations
  - Add relationships
  - Refine (override) inherited operations

Class Diagram

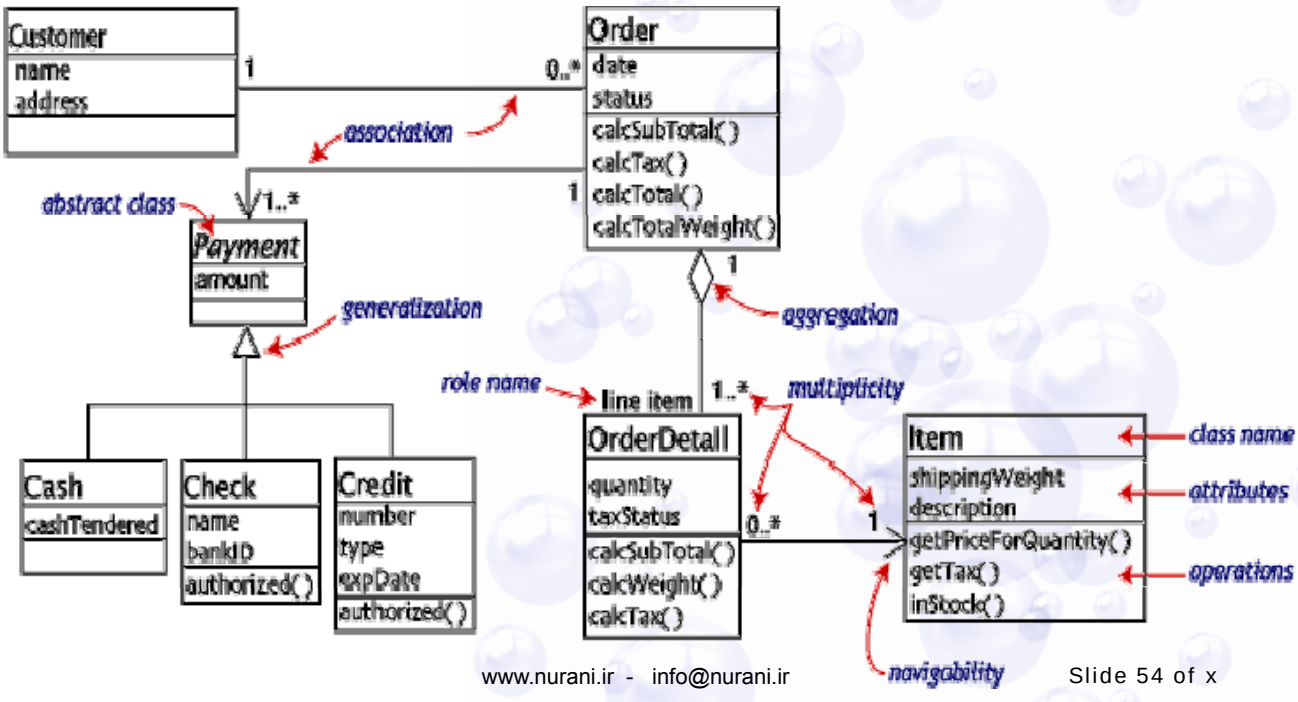
- Dependency
- A dependency is a relation between two classes in which a change in one may force changes in the other

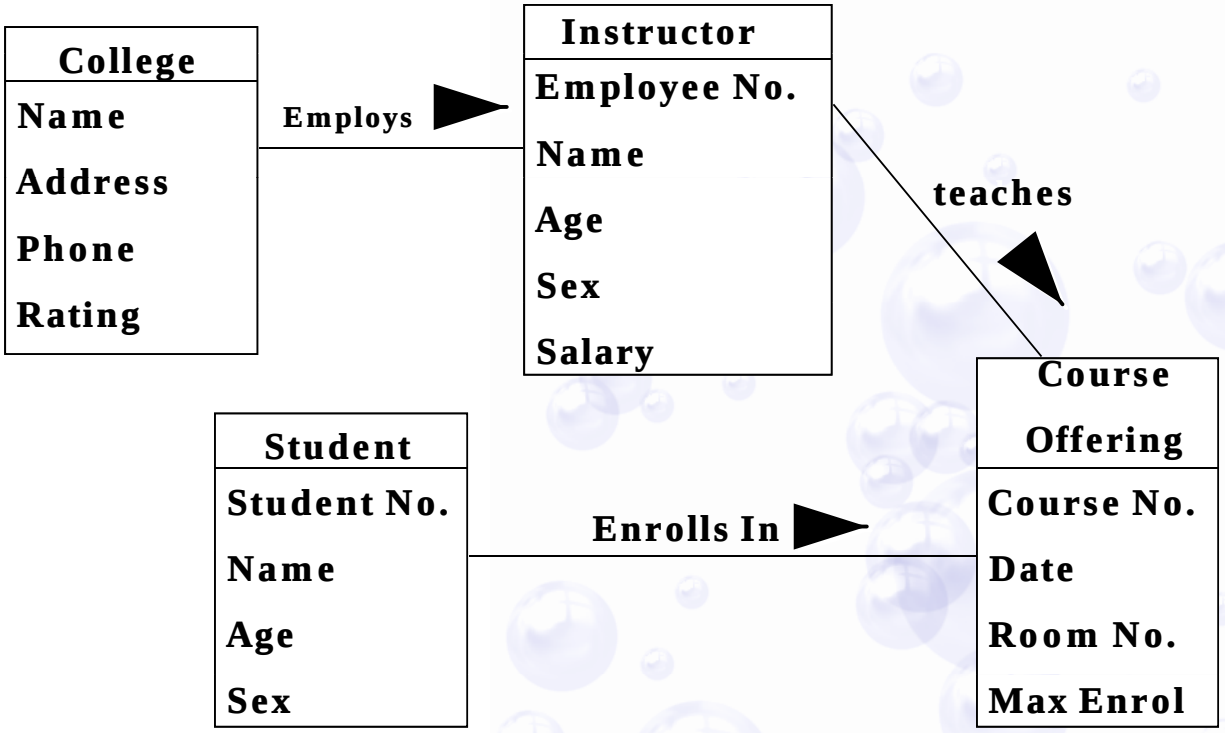
Class Diagram

- Dependency
- Examples:
  - A class is a friend of another class.
  - A class contains an operation that takes an object of another class as a parameter.
  - A class accesses a global object of another class.
- A stereotype may be used to denote the type of the dependency.



Example : Customer order





Example Class Diagram