UML Diagram in OOP

Model your project before development

Md Saeed Siddik
Assistant Professor
IIT, University of Dhaka

UML: Unified Modeling Language

- UML is an industry standard graphical notation for describing and analyzing software designs.
- UML has been accepted as a standard by the Object Management Group
- Easy to graphically visualize a software before start development.

UML Diagrams

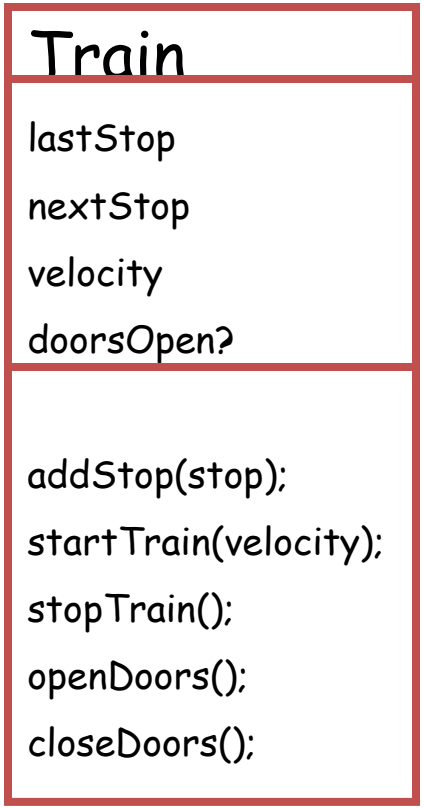
- The UML defines nine types of diagrams:
  - activity diagram
  - class diagram
    - Describes the data and some behavioral (operations) of a system
  - collaboration diagram
  - component diagram
  - deployment diagram
  - object diagram
  - sequence diagram
  - statechart diagram
  - use case diagram

UML Relationships

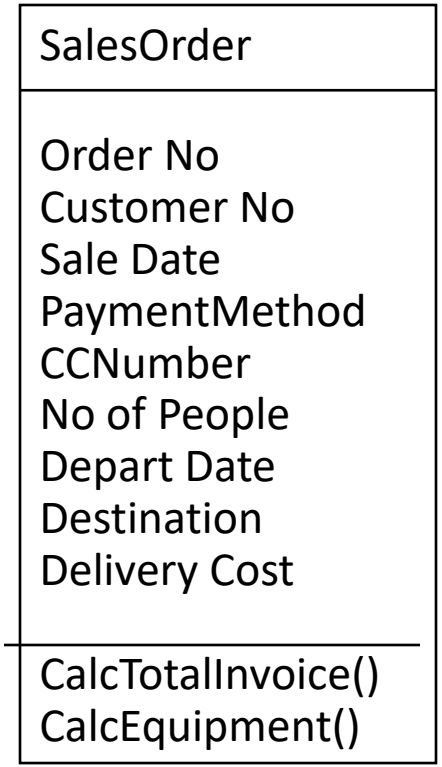
- An relationship is a connection between or among model elements.
- The UML defines four basic kinds of relationships:
  - Association
  - Dependency
  - Generalization

Class Diagrams

- Describe classes
  - In the OO sense
- Class diagrams are static -- they display what interacts but not what happens when they do interact
- Each box is a class
  - List fields
  - List methods



UML Class Diagram



Class Name

List of Attributes

List of operations

Object Diagrams

307: SalesOrder
Order No = 307
Customer No = 1480
Sale Date = 19/5/2025
PaymentMethod = Visa
CCNumber = 12345 678 90
CCExpDate = 29/5/2025
No of People = 2
Depart Date = 21/5/25
Destination = Dhaka
Delivery Cost = 100

Class Diagrams: Relationships

- Many different kinds of edges to show different relationships between classes

- Mention just a couple

Associations

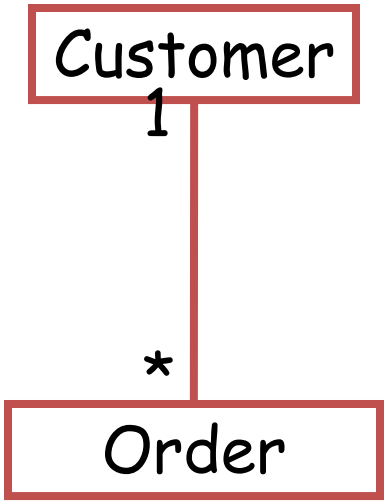
- An association is a relationship that describes a set of links between or among objects.
- An association can have a name that describes the nature of this relationship. You can put a triangle next to this name to indicate the direction in which the name should be read.

Associations

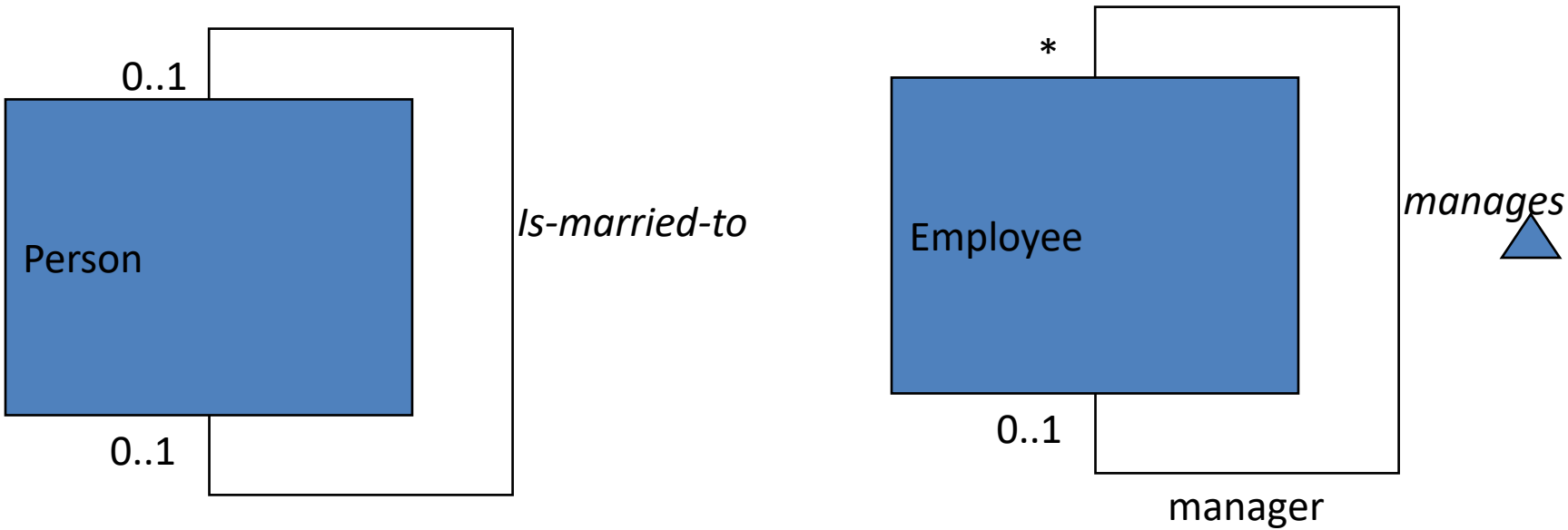
- An association contains an ordered list of association ends.
  - An association with exactly two association ends is called a binary association
  - An association with more than two ends is called an n-ary association.

Association

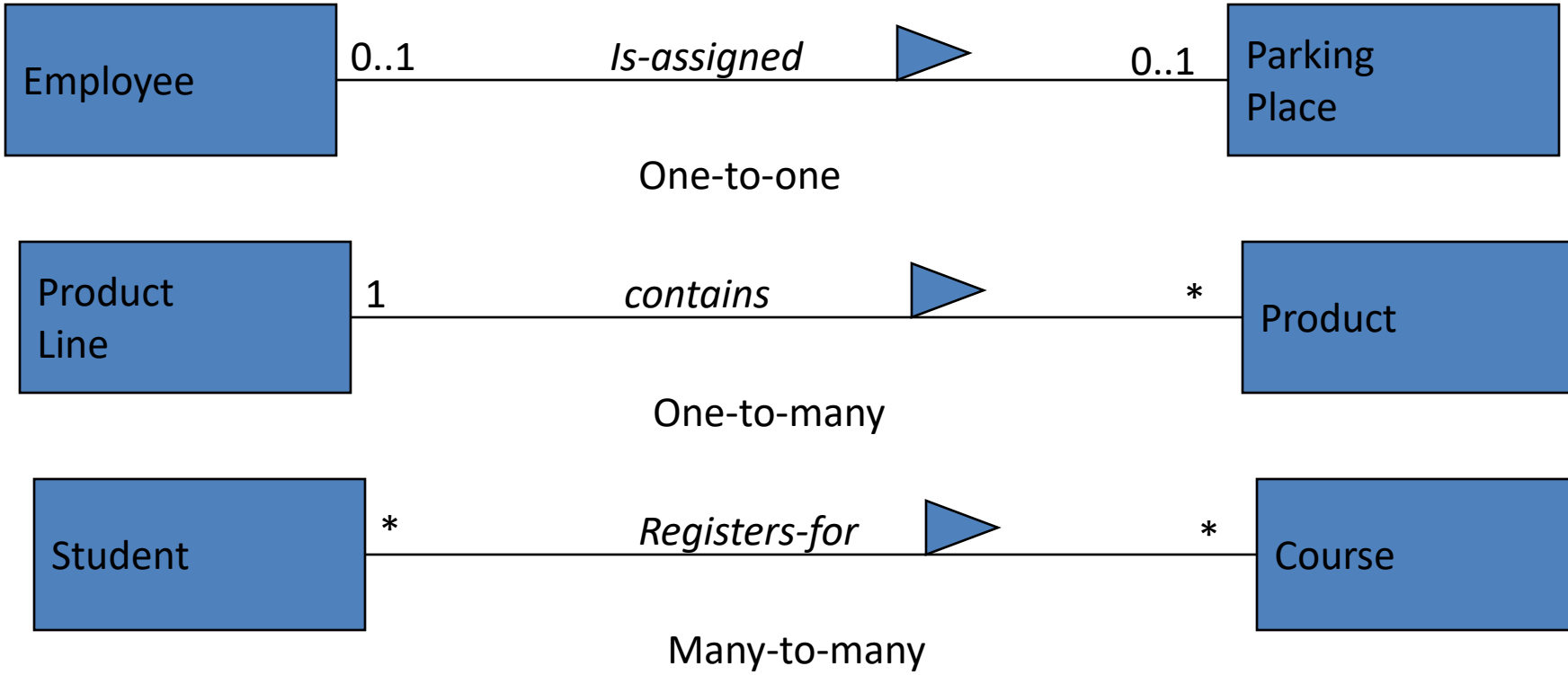
- Association between two classes
  - if an instance of one class must know about the other in order to perform its work.
- Label endpoints of edge with cardinalities
  - Use * for arbitrary
- Can be directional (use arrows in that case)



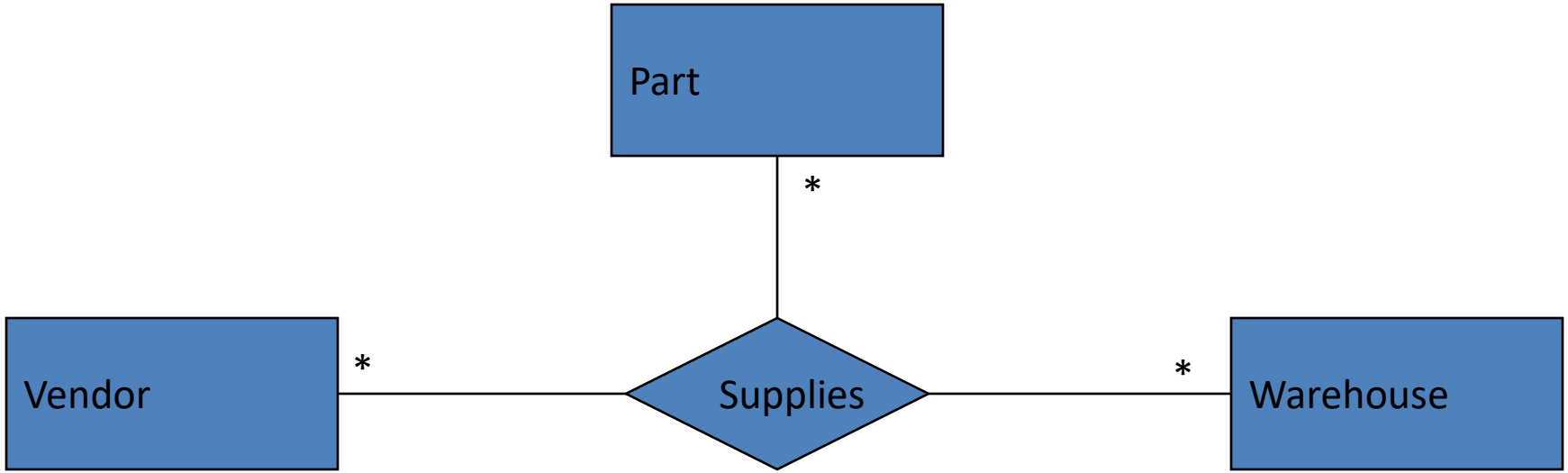
Associations: Unary relationships



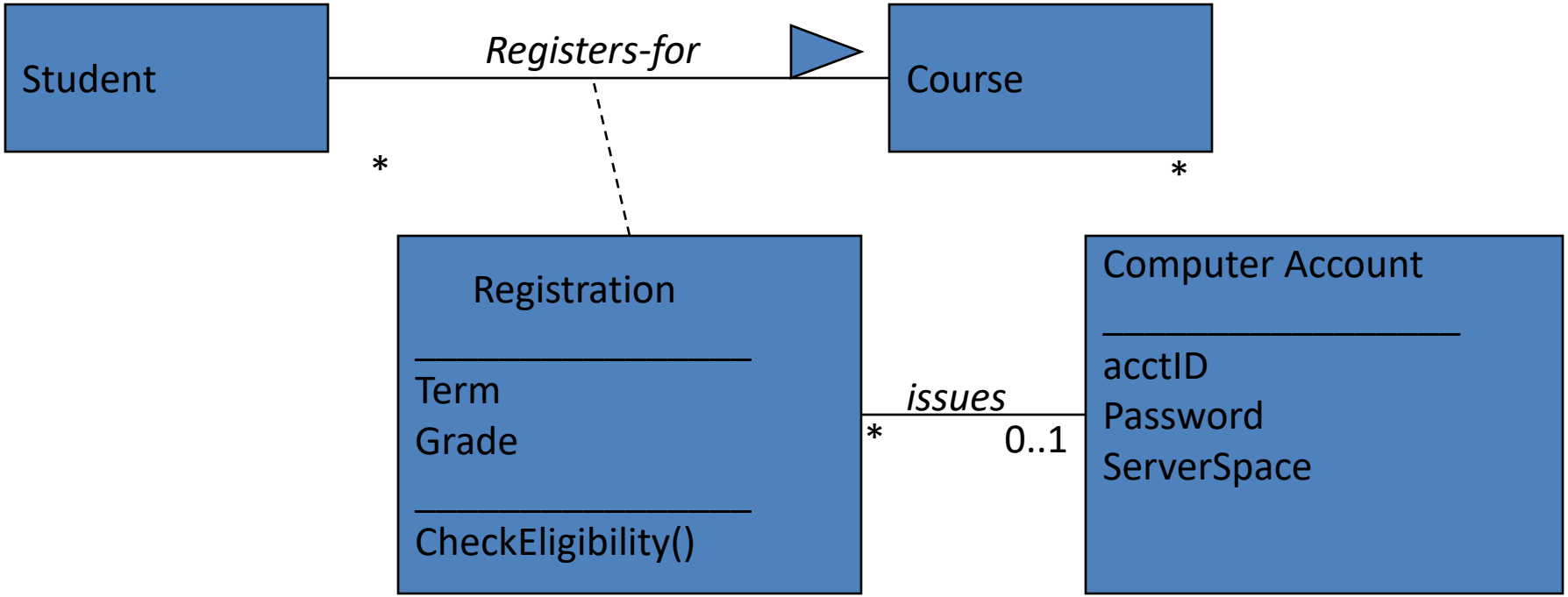
Associations: Binary Relationship



Associations: Ternary Relationships



Association Classes

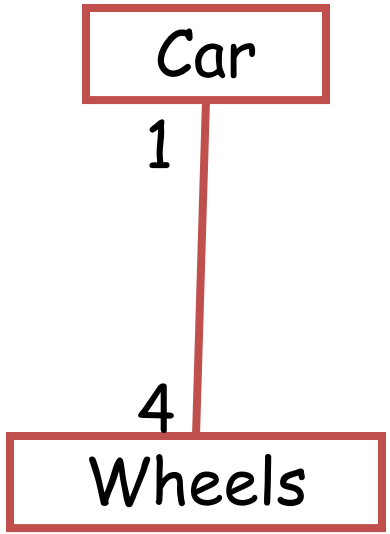


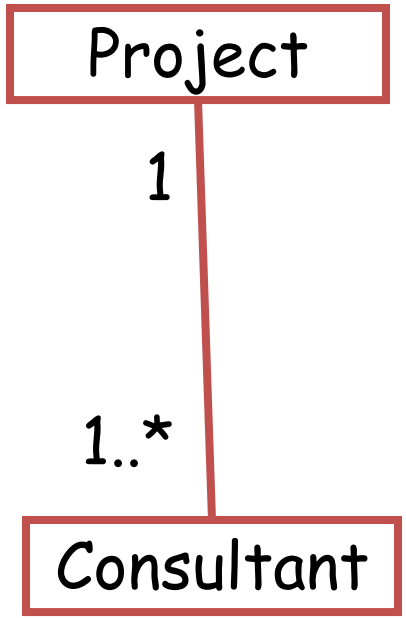
Composition

- An association in which one class belongs to a collection
  - No Sharing: An object cannot exist in more than one collections
  - Strong “has a” relationship
  - Ownership
- Denoted by filled diamond on the “contains” side

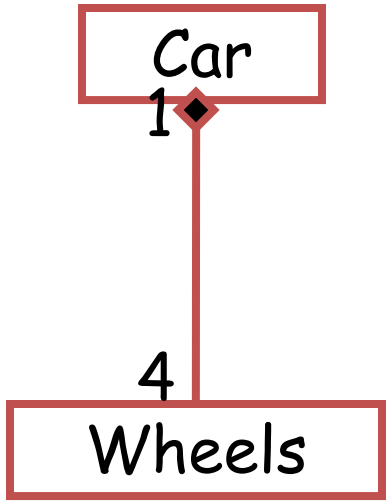
Aggregation

- An association in which one class belongs to a collection
  - Shared: An object can exist in more than one collections
  - No ownership implied
- Denoted by hollow diamond on the “contains” side

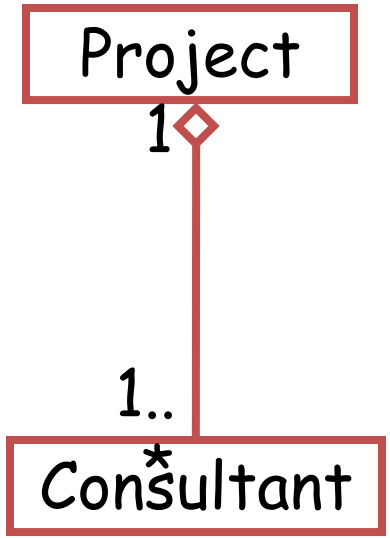

Car
1
4
Wheels


Project
1
1..*
Consultant

Composition



Aggregation

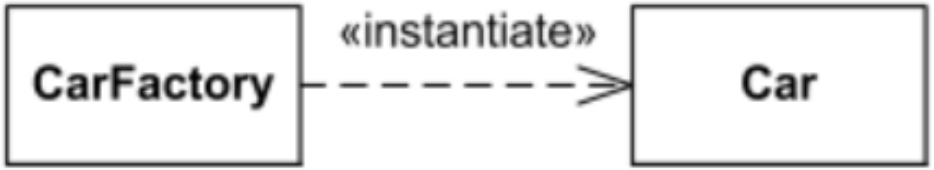


UML Dependency Diagram

- Dependency is a directed relationship which is used to show that some UML element or a set of elements requires, needs or depends on other model.
- Dependency is called a **supplier** - **client** relationship, where supplier provides something to the client, and thus the client is in some sense incomplete while semantically or structurally dependent on the supplier element

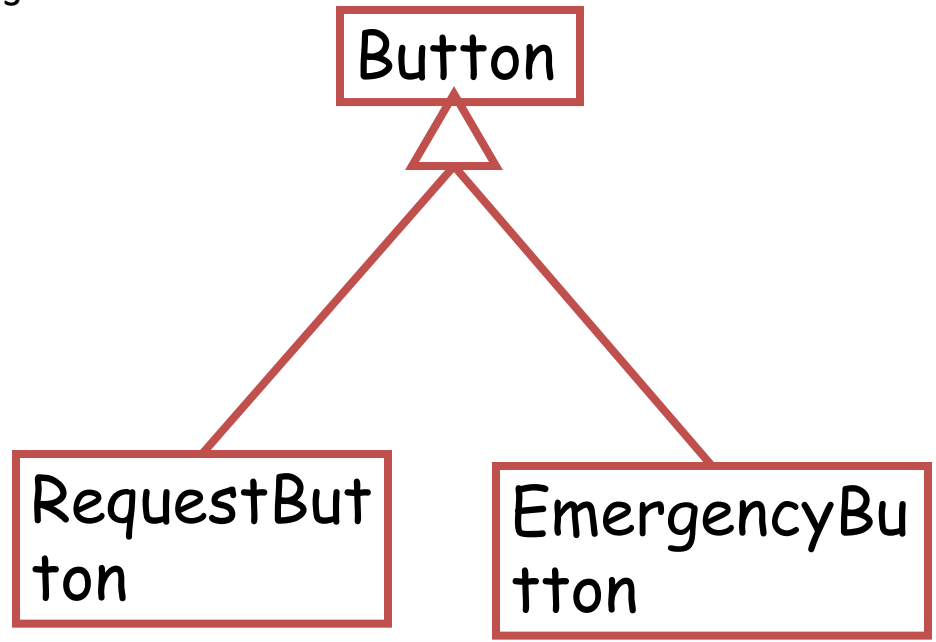
UML Dependency

- *"In the example below, the Car class has a dependency on the CarFactory class. In this case, the dependency is an instantiate dependency, where the Car class is an instance of the CarFactory class."*



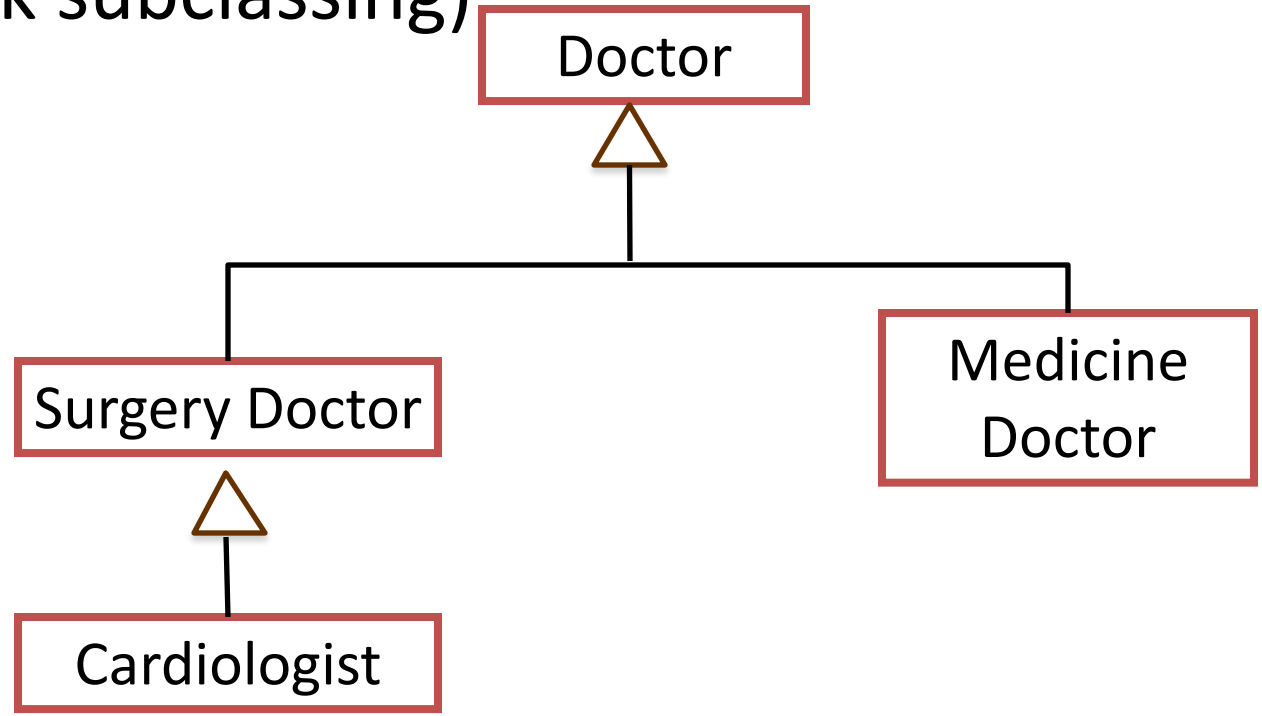
Generalization

- Inheritance between classes
- Denoted by open triangle

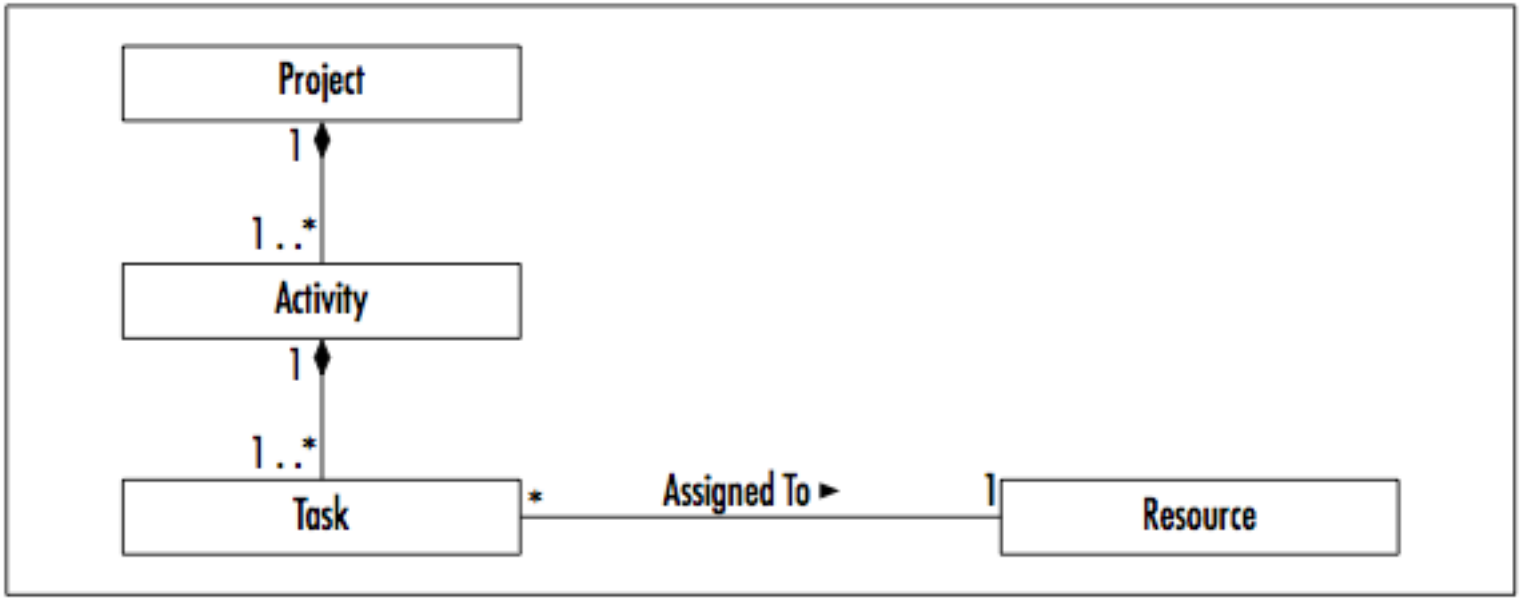


Generalization

- (Think subclassing)



Example



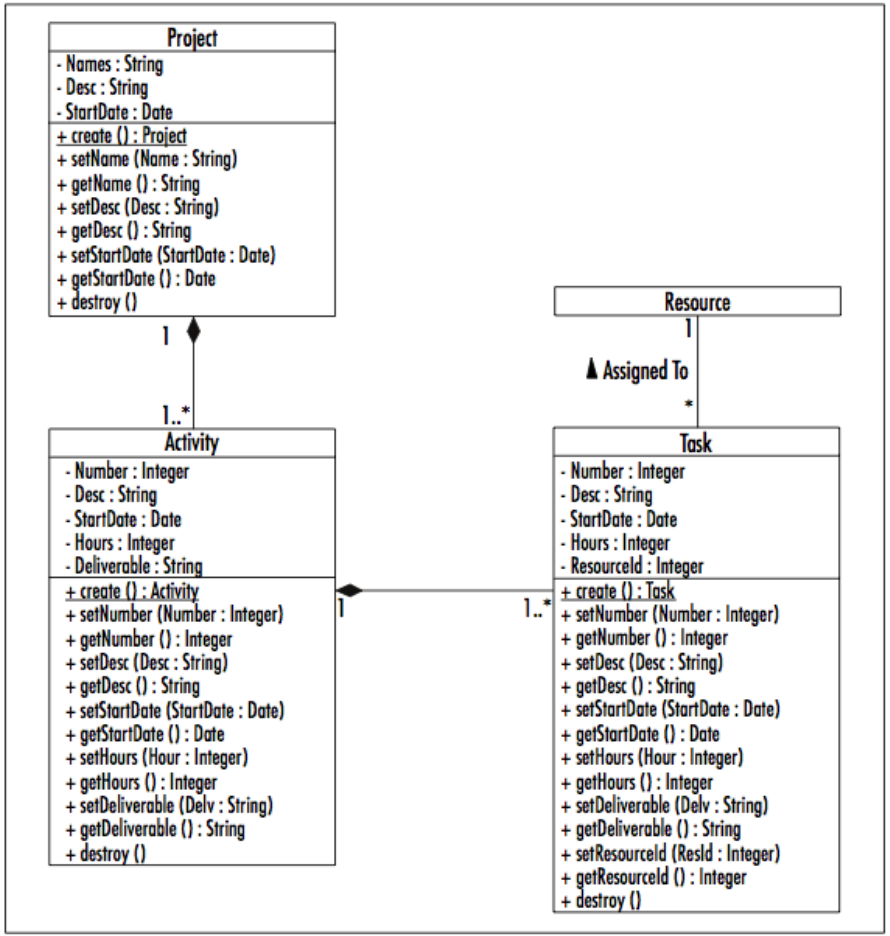


Figure 4-7: Detailed Project Class Diagram

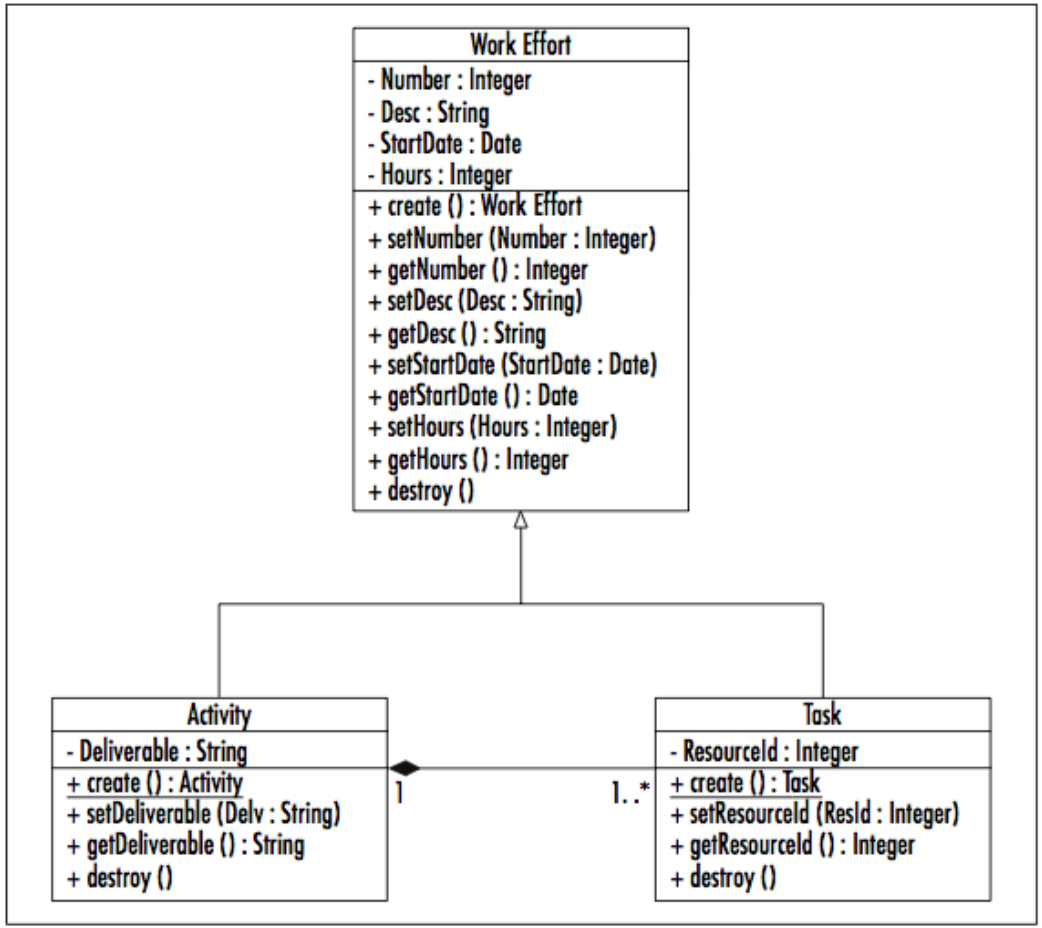
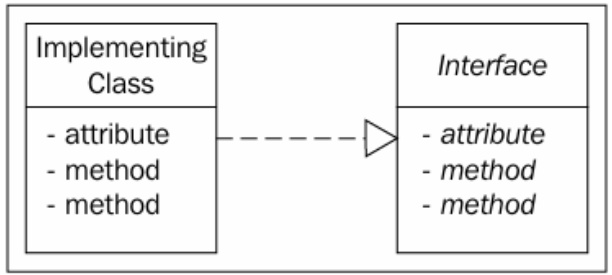


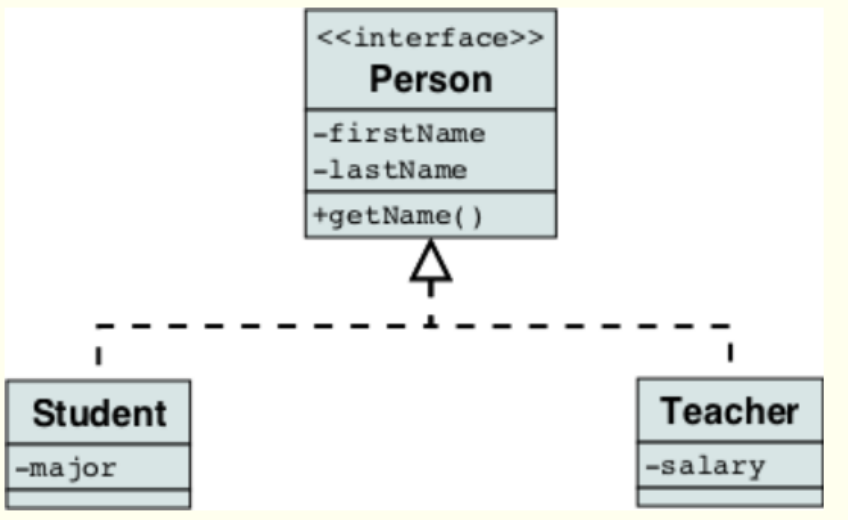
Figure 4-8: Detailed Activities and Tasks Class Diagram

UML Diagram: Realization

- Realization is a semantic relationship between two UML elements, most commonly between an interface and the class that implements that interface.
- Notation: Dotted line with a hollow triangle pointing to the interface.
- Direction: From implementing class to interface
- Example: A class Document realizes the Printable interface.

Realization: Example





Similarity between Generalization and Realization

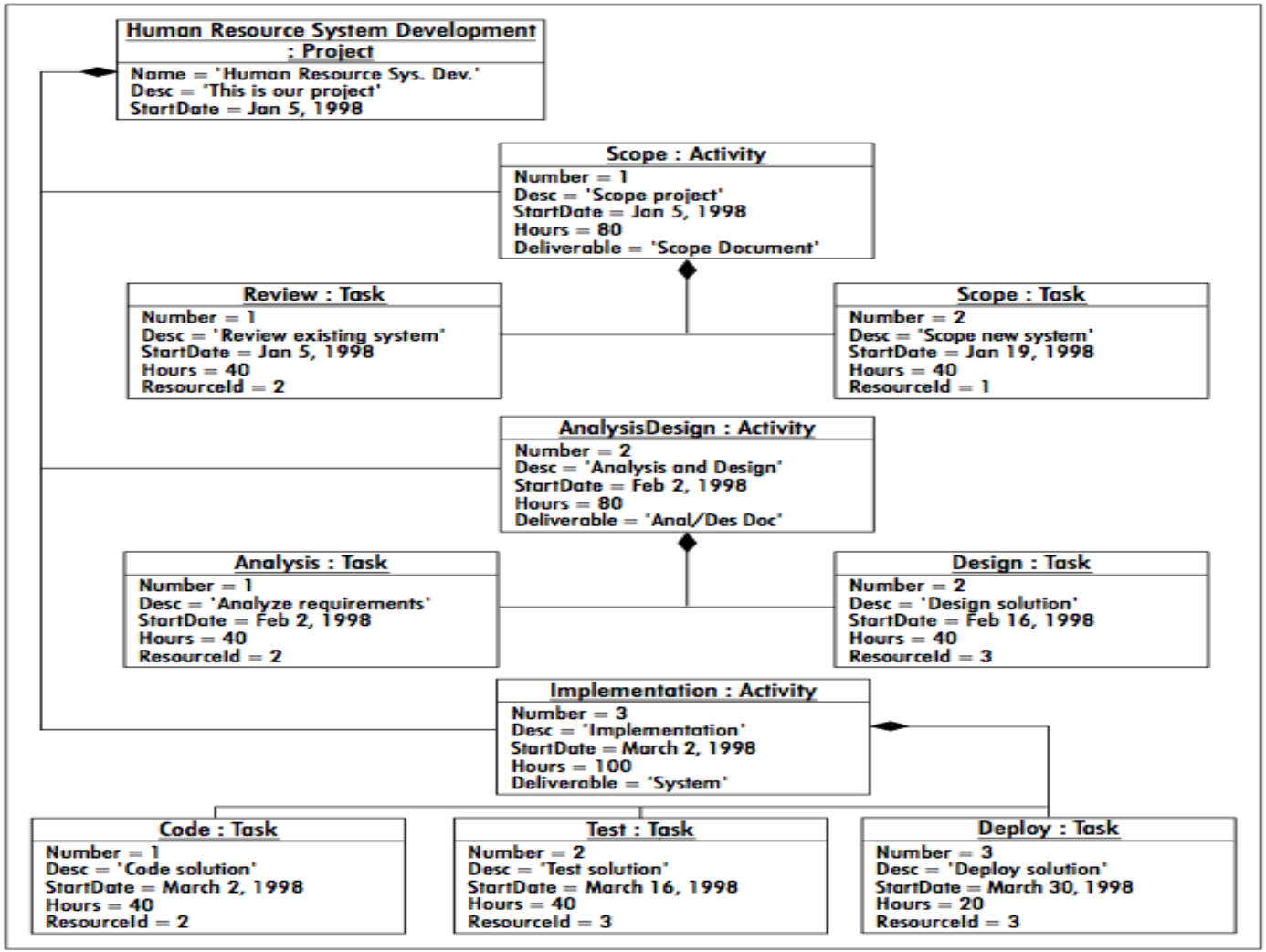
- Generalization always models inheritance relationships between classes. Realization always models interface implementation relationships between classes.

Summary Table

Relationship	Line Type	Symbol	Meaning
Association	Solid	None	Connection between classes
Aggregation	Solid	White diamond	Whole-part (independent)
Composition	Solid	Black diamond	Whole-part (dependent)
Dependency	Dotted arrow	None	"Uses" relationship
Generalization	Solid arrow	Hollow triangle	Inheritance
Realization	Dotted arrow	Hollow triangle	Implements interface

Object Diagram

- Object diagram is an instantiation of a class diagram
- Represents a static structure of a system at a particular time


Human Resource System Development : Project
Name = 'Human Resource Sys. Dev.'
Desc = 'This is our project'
StartDate = Jan 5, 1998
Scope : Activity
Number = 1
Desc = 'Scope project'
StartDate = Jan 5, 1998
Hours = 80
Deliverable = 'Scope Document'
Review : Task
Number = 1
Desc = 'Review existing system'
StartDate = Jan 5, 1998
Hours = 40
ResourceId = 2
Scope : Task
Number = 2
Desc = 'Scope new system'
StartDate = Jan 19, 1998
Hours = 40
ResourceId = 1
AnalysisDesign : Activity
Number = 2
Desc = 'Analysis and Design'
StartDate = Feb 2, 1998
Hours = 80
Deliverable = 'Anal/Des Doc'
Analysis : Task
Number = 1
Desc = 'Analyze requirements'
StartDate = Feb 2, 1998
Hours = 40
ResourceId = 2
Design : Task
Number = 2
Desc = 'Design solution'
StartDate = Feb 16, 1998
Hours = 40
ResourceId = 3
Implementation : Activity
Number = 3
Desc = 'Implementation'
StartDate = March 2, 1998
Hours = 100
Deliverable = 'System'
Code : Task
Number = 1
Desc = 'Code solution'
StartDate = March 2, 1998
Hours = 40
ResourceId = 2
Test : Task
Number = 2
Desc = 'Test solution'
StartDate = March 16, 1998
Hours = 40
ResourceId = 3
Deploy : Task
Number = 3
Desc = 'Deploy solution'
StartDate = March 30, 1998
Hours = 20
ResourceId = 3

Summary

- UML Diagram
- UML Relationship
- Object Diagram

Thank You