Entity-Relationship Diagram

- A popular graphical notational paradigm for representing conceptual models
- Has three core constructs
  - An *entity*: depicted as a rectangle, represents a collection of real-world objects that have common properties and behaviors
  - A *relationship*: depicted as an edge between two entities, with diamond in the middle of the edge specifying the type of relationship
  - An *attribute*: an annotation on an entity that describes data or properties associated with the entity

Entity-Relationship Diagrams

ERD is a data modeling technique used in software engineering to produce a conceptual data model of an information system.
So, ERDs illustrate the logical structure of databases.

major activity of this phase is identifying entities, attributes, and their relationships to construct model using the Entity Relationship Diagram.

Entity → table

Attribute → column

Relationship → line

Entity-Relationship Diagrams

The data modeling revolves around discovering and analyzing organizational and users data requirements.

Requirements based on policies, meetings, procedures, system specifications, etc.

- Identify what data is important
- Identify what data should be maintained

Entity-Relationship Diagrams

Entity:

"…anything (people, places, objects, events, etc.) about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc.)."

Tangible: customer, product

Intangible: order, accounting receivable

Attribute:

Attributes are data objects that either identify or describe entities (property of an entity).

Relationship:

Relationships are associations between entities.

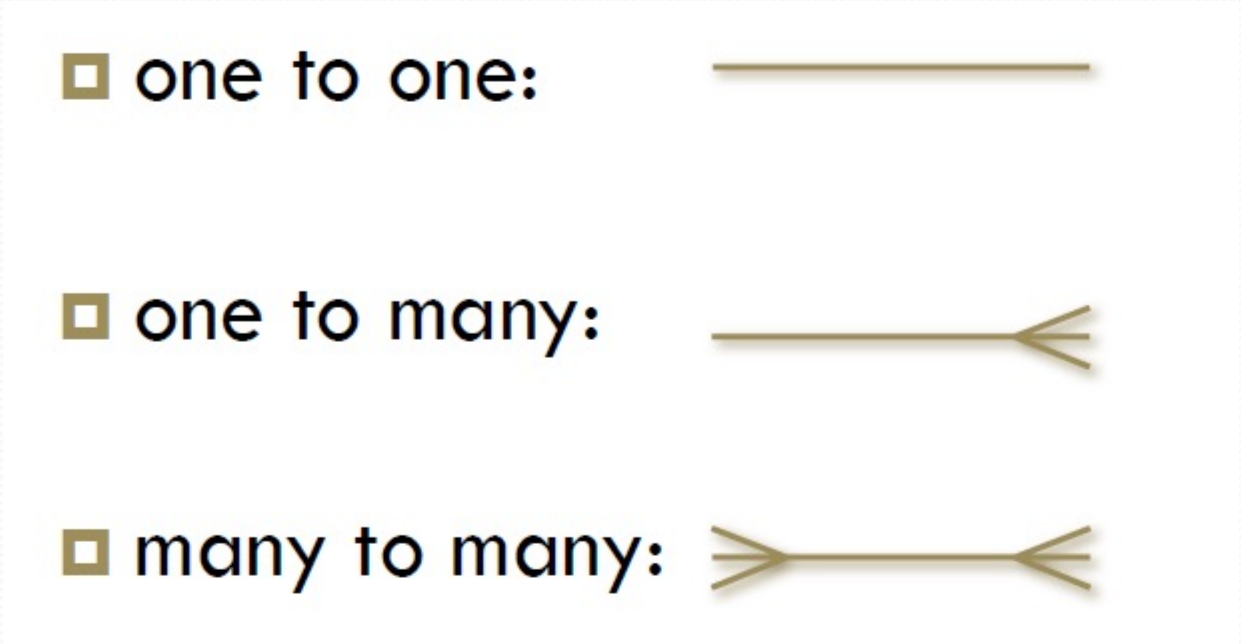
Typically, a relationship is indicated by a verb connecting two or more entities.

Employees are assigned to projects

Relationships should be classified in terms of cardinality.

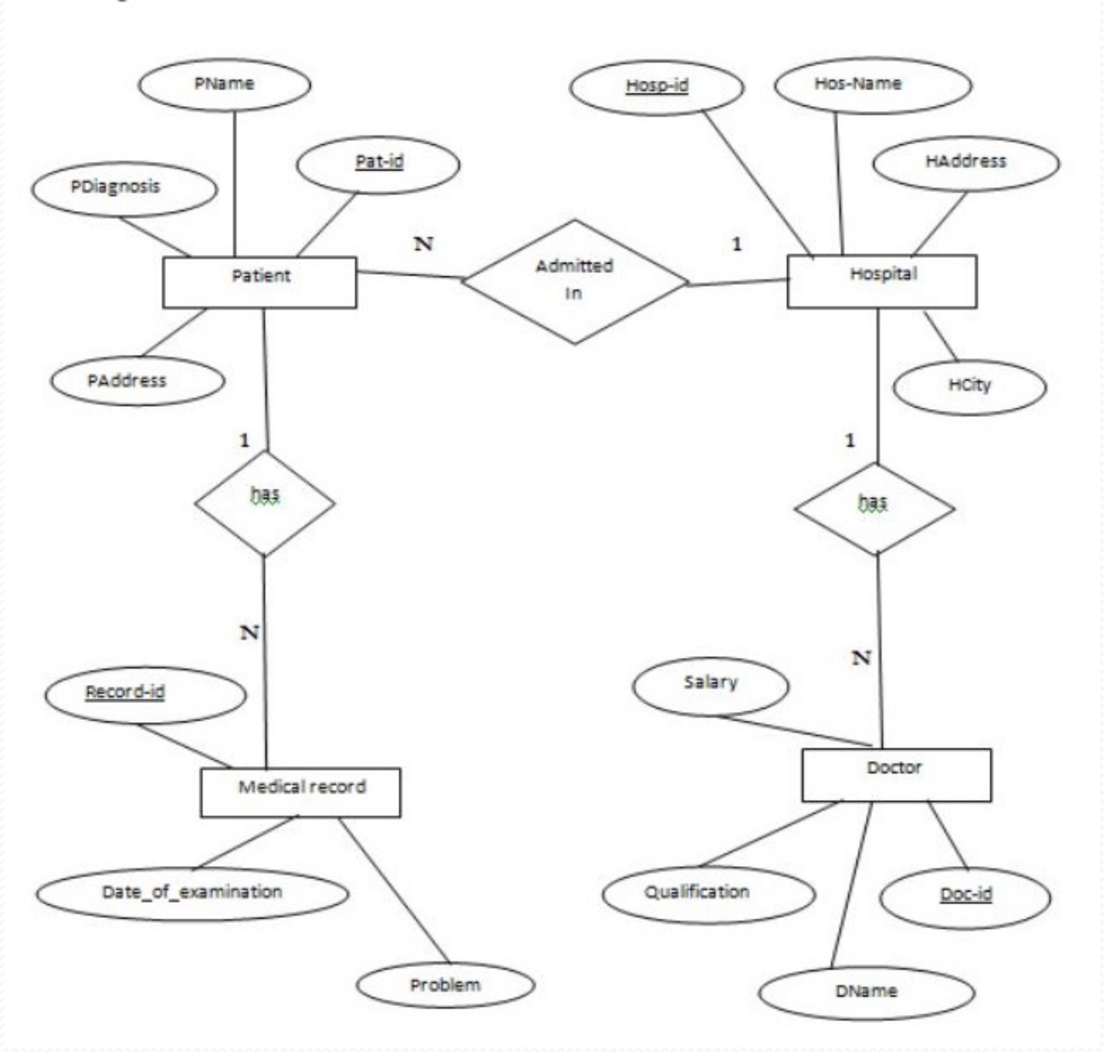
One-to-one, one-to-many, many to many, many to one

Entity-Relationship Diagrams



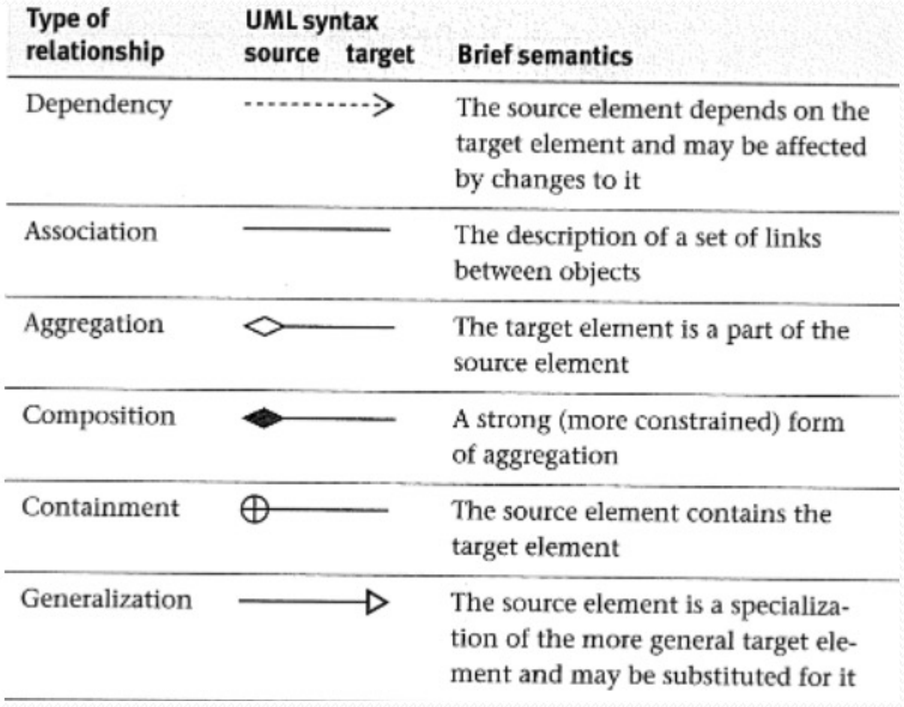
- one to one:
- one to many:
- many to many:

Entity-Relationship Diagrams



Class Diagram

- A Class Diagram represents a system in terms of
  - *objects*: akin to entities, organized in classes that have an inheritance hierarchy
  - *attributes*: object's variables or characteristics
  - *behaviors*: actions on the object's variables



Type of relationship	UML syntax source target	Brief semantics
Dependency	- - - - - - - - ->	The source element depends on the target element and may be affected by changes to it
Association	————	The description of a set of links between objects
Aggregation	◇————	The target element is a part of the source element
Composition	◆————	A strong (more constrained) form of aggregation
Containment	⊕————	The source element contains the target element
Generalization	————▷	The source element is a specialization of the more general target element and may be substituted for it

Class Diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram −

•The name of the class diagram should be meaningful to describe the aspect of the system.

•Each element and their relationships should be identified in advance.

•Responsibility (attributes and methods) of each class should be clearly identified

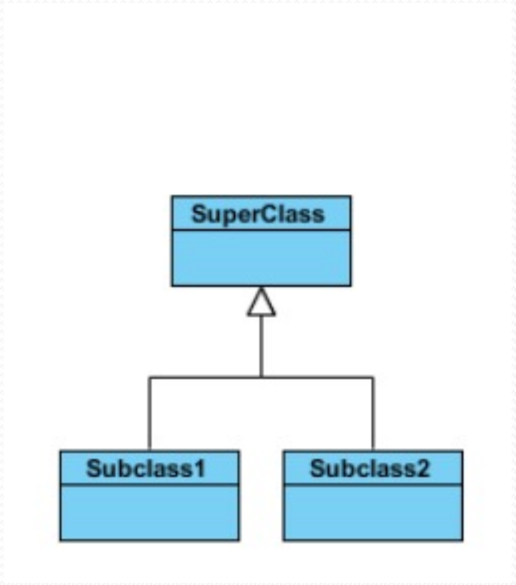
•For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.

Class Diagram

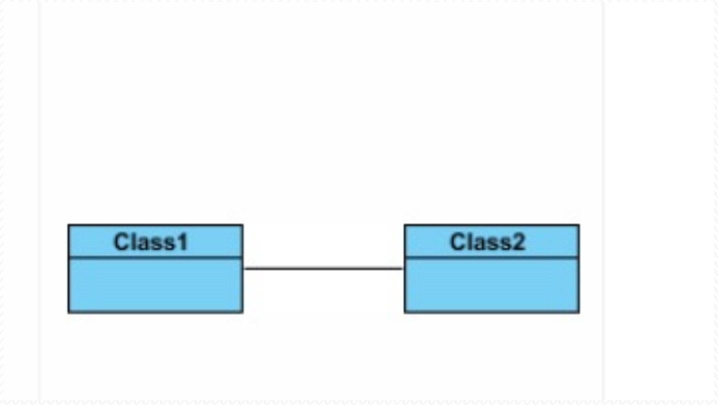
Class Relationships:

A class may be involved in one or more relationships with other classes.

Inheritance (or Generalization)

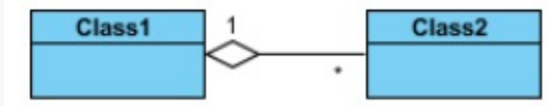


Simple Association:

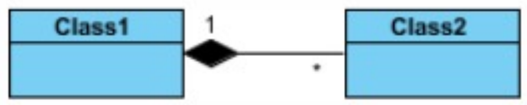


Class Diagram

Aggregation:
A special type of association. It represents a "part of" relationship.



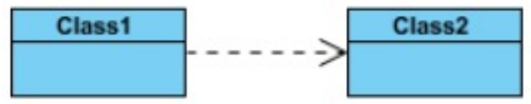
Composition:
A special type of aggregation where parts are destroyed when the whole is destroyed.
•Objects of Class2 live and die with Class1.
•Class2 cannot stand by itself.



Class Diagram

Dependency

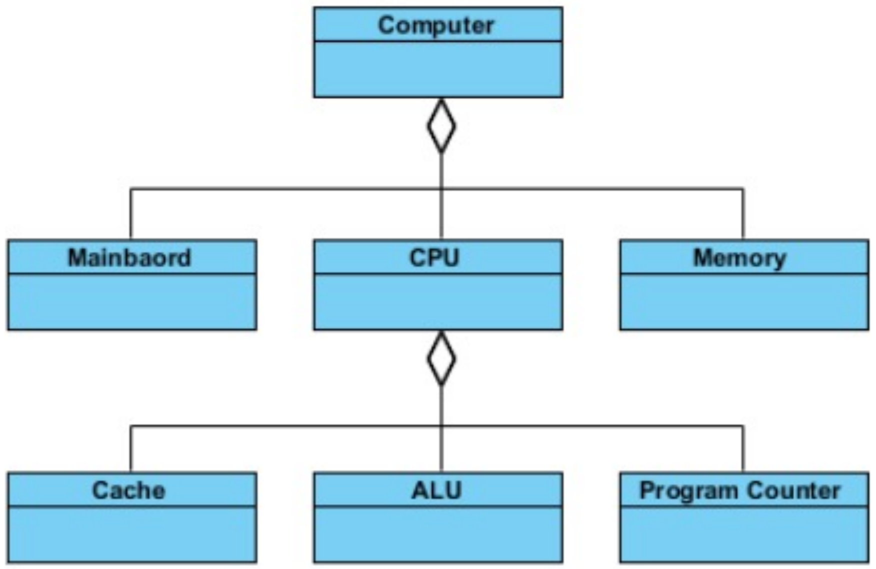
•Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).



Class Diagram

Aggregation Example - Computer and parts

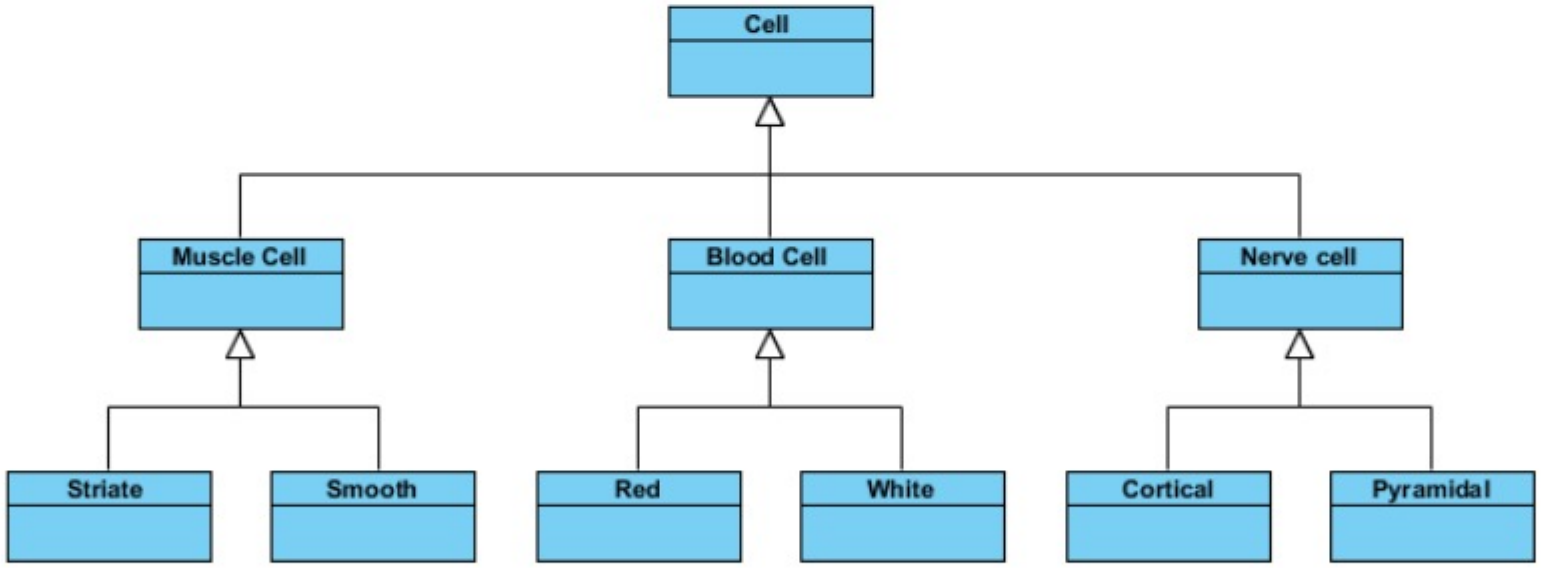
- An aggregation is a special case of association denoting a "consists-of" hie
- The aggregate is the parent class, the components are the children classe



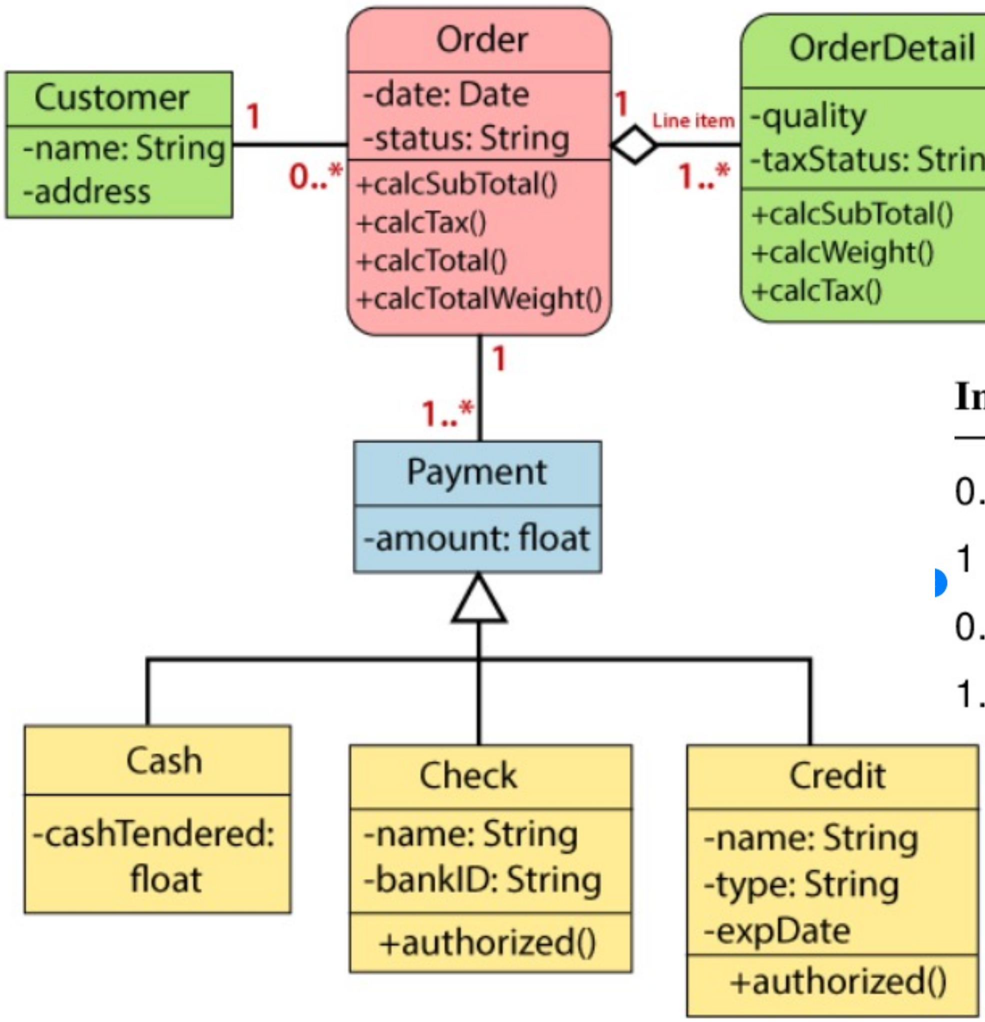
Class Diagram

Inheritance Example - Cell Taxonomy

- Inheritance is another special case of an association denoting a "kind-of" hierarchy
- Inheritance simplifies the analysis model by introducing a taxonomy
- The child classes inherit the attributes and operations of the parent class.

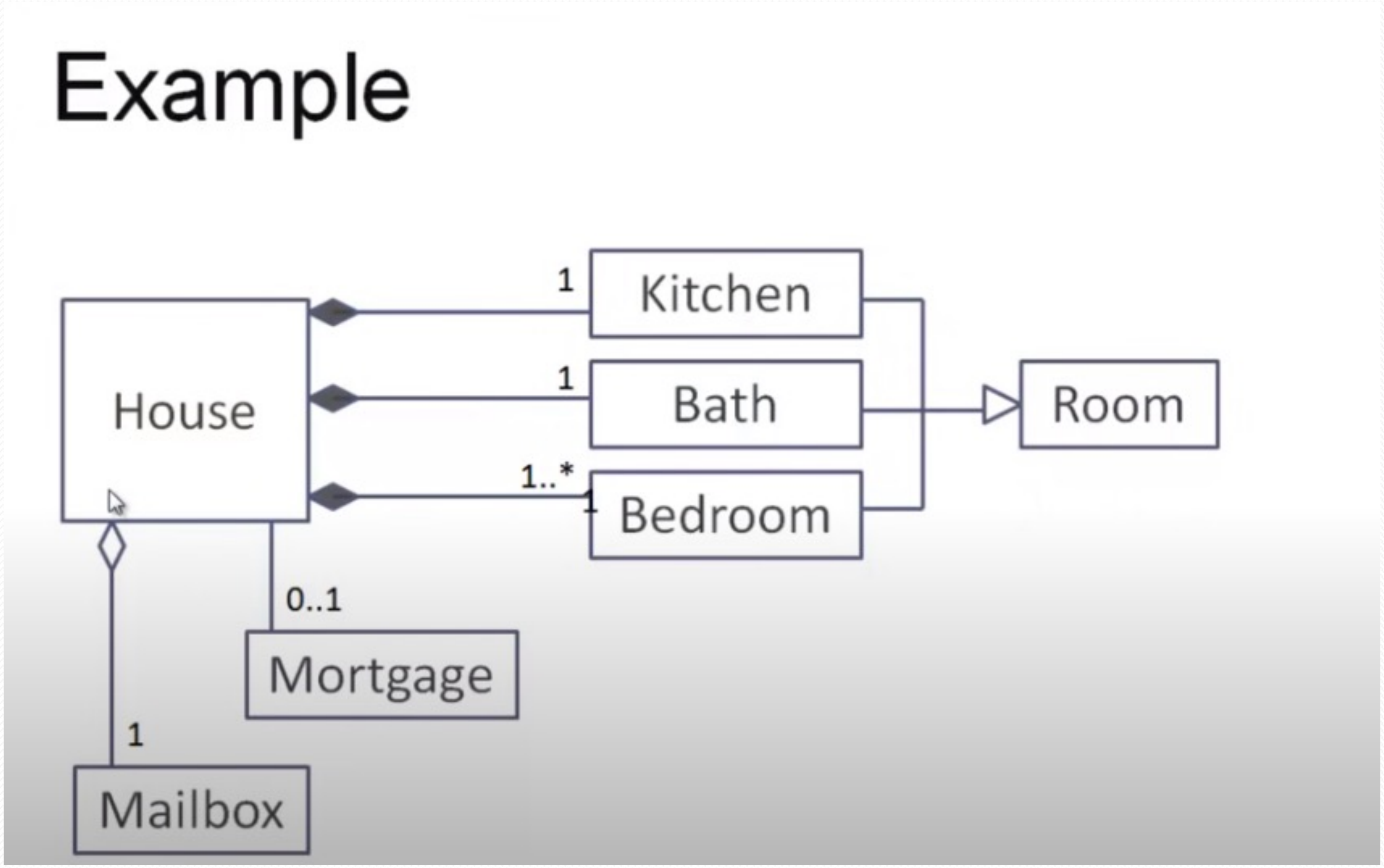


A class diagram describing the sales order system is given below

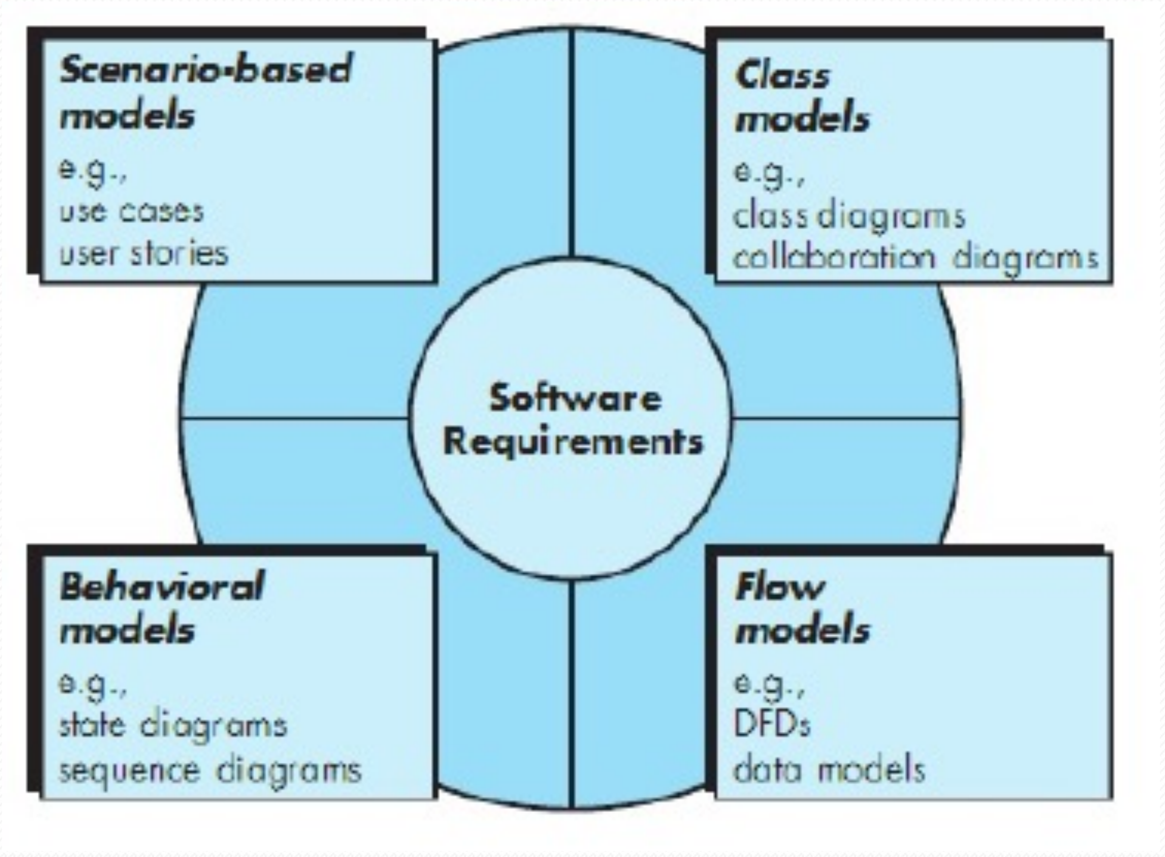


Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more

Class Diagram

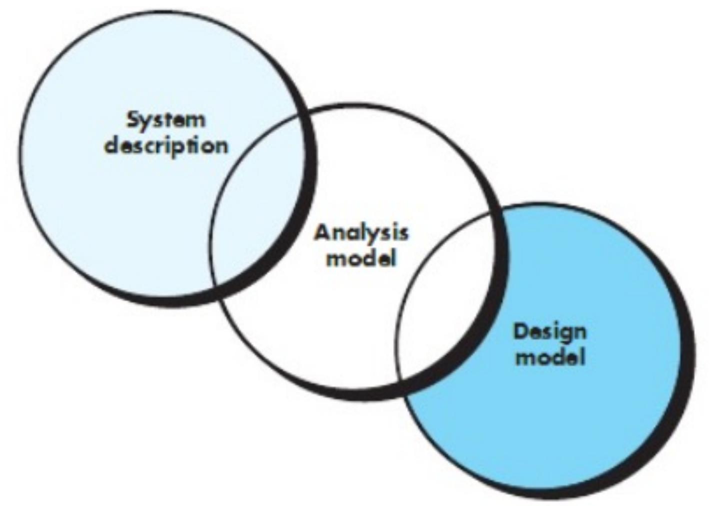


Elements in Analysis Model



Analysis

- Focus on requirements
- Each element should improve understanding of requirements
- Delay consideration of infrastructure till design
- Requirements model provides value to all stakeholders
- Keep the models simple



Prioritizing Requirements

- Different stakeholders have different set of requirements
  - potential conflicting ideas
- Need to prioritize requirements to resolve conflicts
- Prioritization might separate requirements into three categories
  - *essential*: absolutely must be met
  - *desirable*: highly desirable but not necessary
  - *optional*: possible but could be eliminated

References

- Pfleeger Book slides from UCF

Acknowledgement!

- A few slides have been reused from UCF slides for the SE course