UNIT III
Introduction to UML
Advanced Classes & Relationships

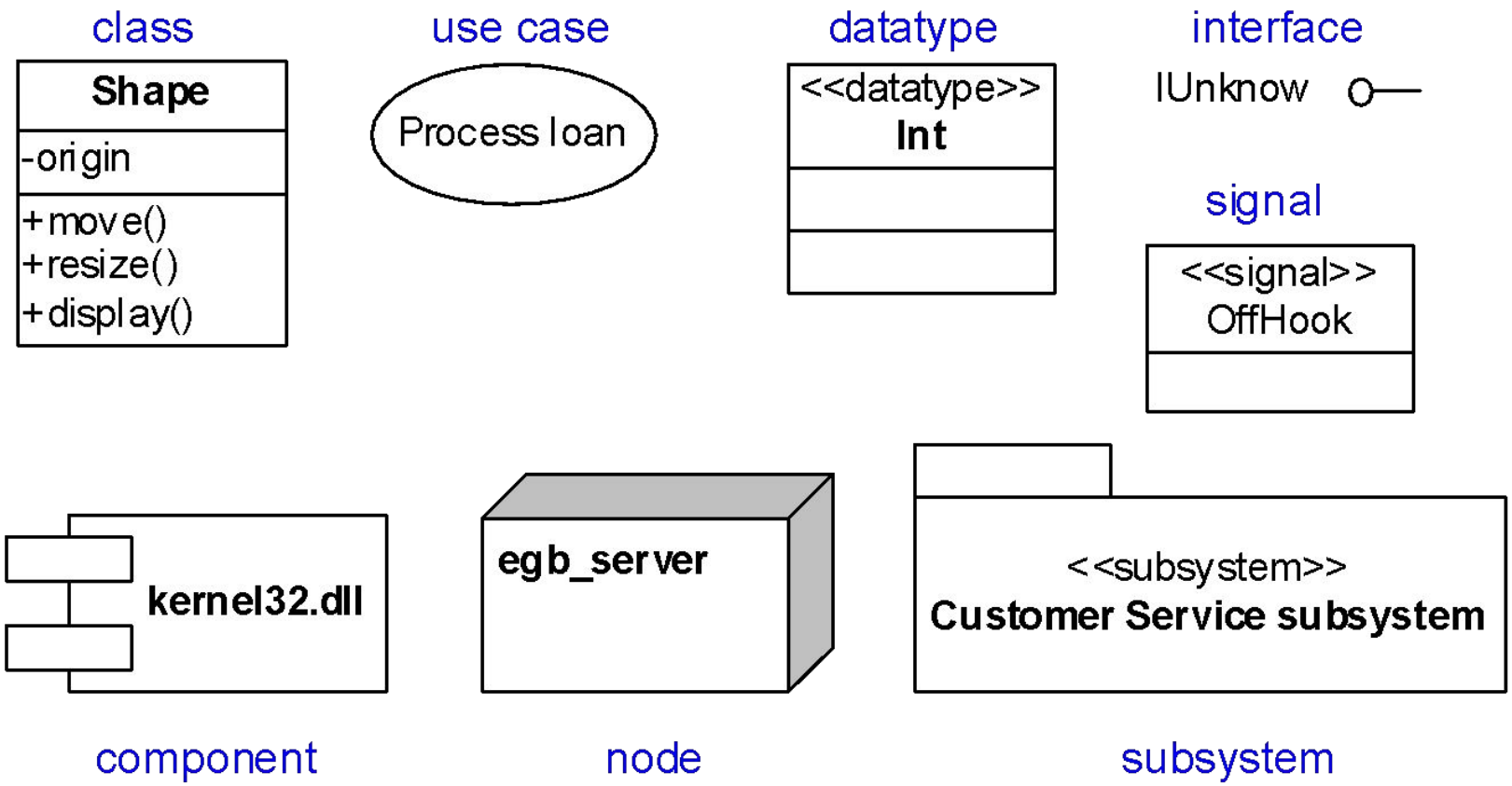
Outline

- What is “Classifier”
- Special properties of attributes and operations and different kinds of classes
- What is “Relationship”
- Important relationships in UML

What is Classifier

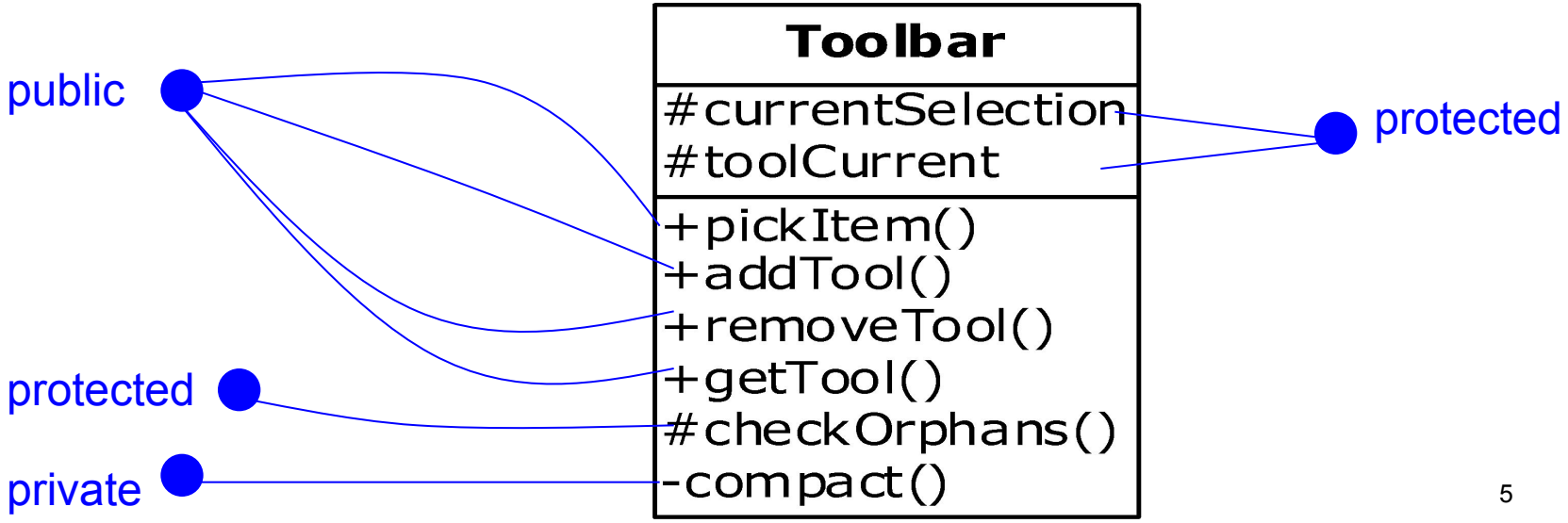
- A *classifier* is a mechanism that describes structural and behavioral features.
- In general, the modeling elements that can have instances are called classifiers.
- Class, Instance, Datatype, Signal, Component, Node, Use case, Subsystem are classifiers. (packages are not.)

What is Classifier cont.



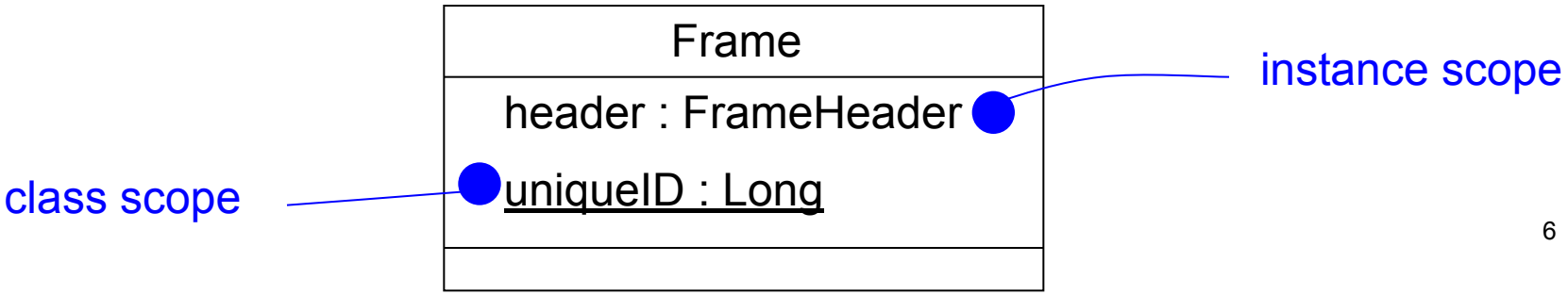
Special properties of attributes and operations

- **Visibility**
  - Public[+]: any outside classifier with visibility to the given classifier can use this feature.
  - Protected[#]: any descendant of the classifier can use the feature.
  - Private[-]: only the classifier itself can use the feature.

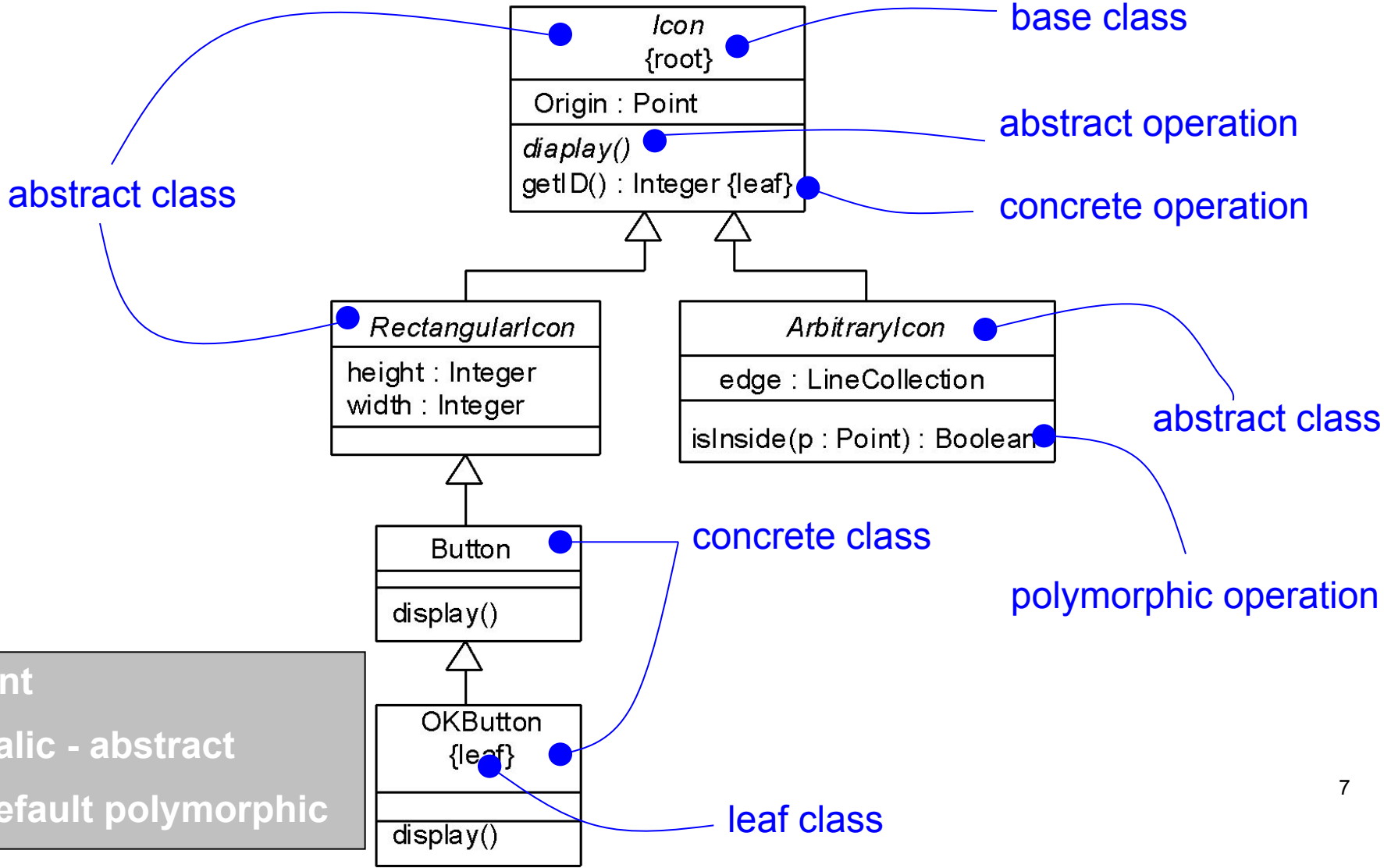


Special properties of attributes and operations

- Scope
  - The owner scope of a feature specifies whether the feature appears in each instance of the classifier or whether there is just a single instance of feature for all instances of the classifier.
    - instance : each instance holds its own value.
    - classifier : just one value for all instances. [*static*]



Abstract, Root, Leaf and Polymorphic Elements

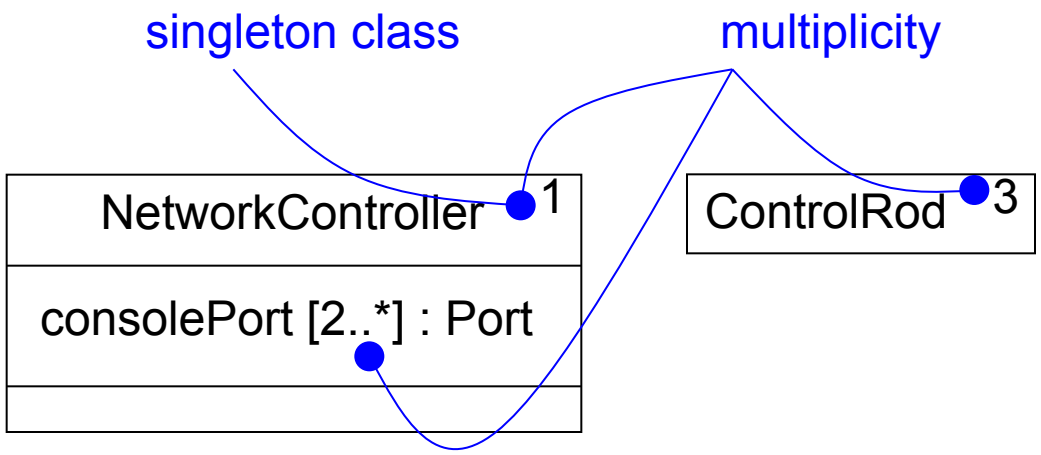


Hint

- **italic - abstract**
- **default polymorphic**

Multiplicity

- It's reasonable to assume that there may be any number of instances of classes.
- The number of instances a class may have is called multiplicity.



Attributes

- The syntax of an attribute in the UML is

[visibility] name [multiplicity] [: type] [= initial-value] [{ property-string }]

- There are three defined properties
  1. changeable : no restrictions on modifying the attribute's value
  2. addOnly : additional value may be added for attributes with a multiplicity > 1, but once created, a value may not be removed or altered.
  3. frozen : the attribute's value may not be changed after object is initialized. [*const*]

Operations

- The syntax of an operation in UML is

[visibility] name [(parameter-list)] [: return-type] [{ property-string }]

[direction] name : type [= default-value]

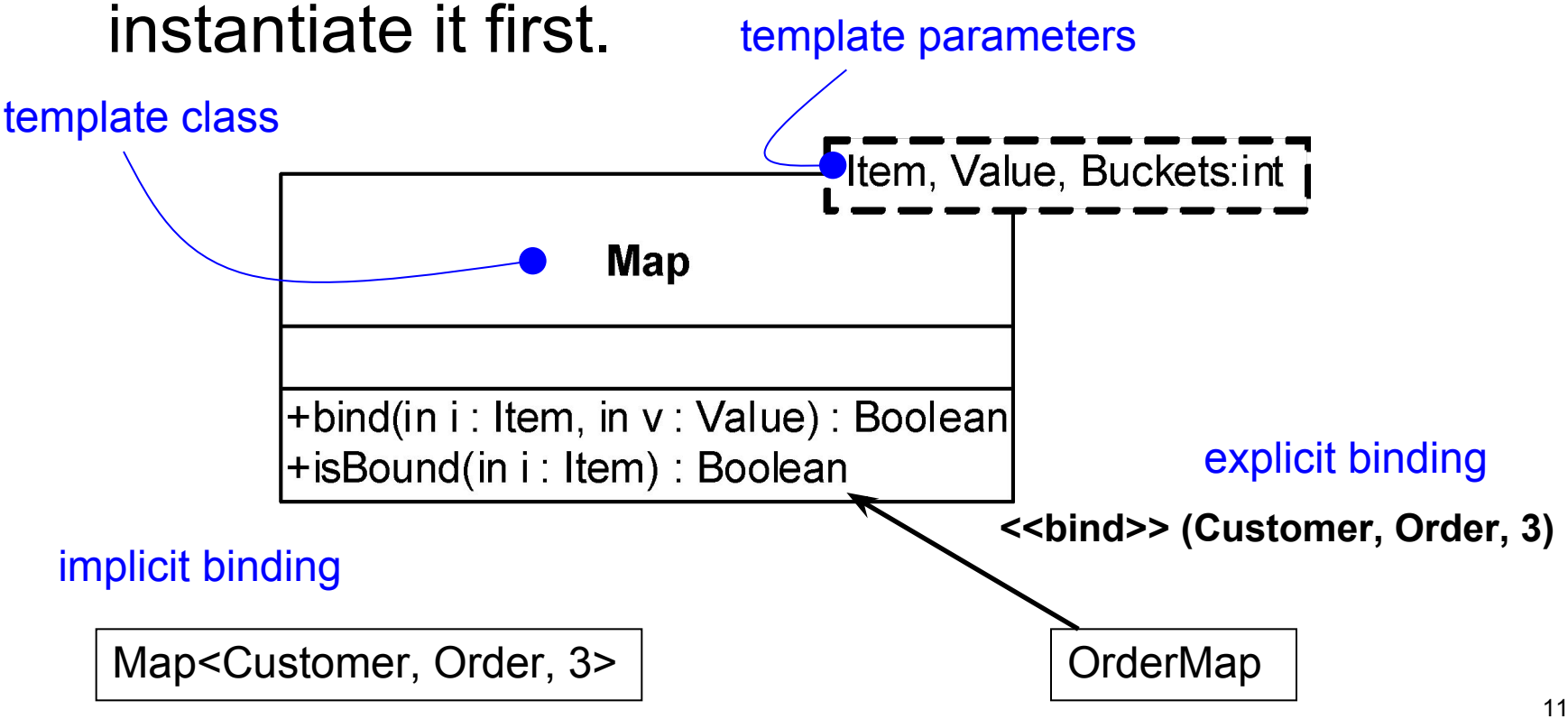
in, out, inout *:* means parameter may be modified or not.

- There are five defined properties
  1. leaf : may not be overridden <page 7>
  2. isQuery : leave the state of subsystem unchanged.
  3. sequential : only one flow is in the object at a time.
  4. guarded : sequentializing all calls.
  5. concurrent : treating the operation as atomic.
  - 3. 4. 5. are for concurrence.

In Rational Rose is **Synchronize**

Template Classes

- Like template classes in C++ and Ada.
- Cannot use a template directly; you have to instantiate it first.



What is Relationship

- A relationship is a connection among things.
- There are four most important relationships in object-oriented modeling:
  - Dependencies
  - Generalizations
  - Associations
  - Realizations

Dependency

- Specifying a change in the specification of one thing may affect another thing, but not necessarily the reverse.
- Rendering as a dashed line []
- UML defines a number of stereotypes.
- There are eight stereotypes that apply to dependency relationships among *classes* and *objects* in <u>class diagrams</u>.

Dependency

bind	the source instatiates the target template
derive	the source may be computed from target
friend	the source is given special visibility into target
instanceOf	source object is an instance of the target classifier
instantiate	source object creates instance of the target
powertype	target is a powertype of the source
refine	source is at a finer degree of abstraction than target
use	the semantics of the source element depends on the semamtics of the public part of the target

Dependency

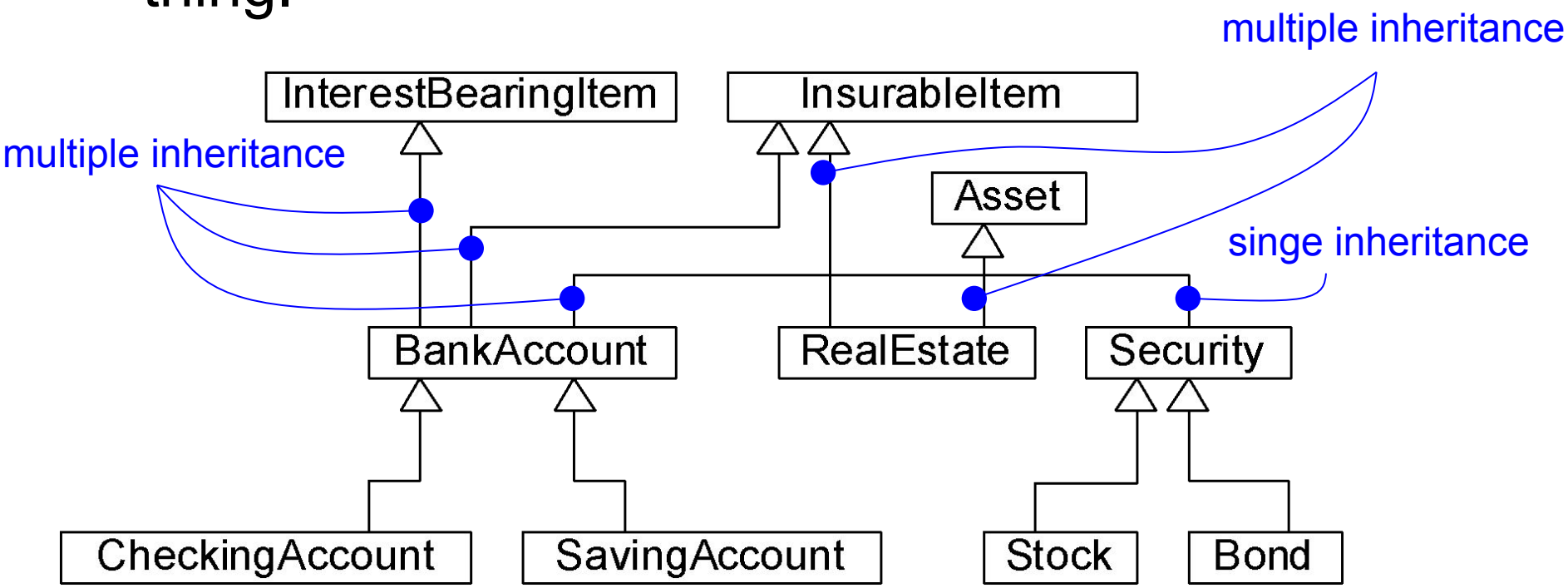
- Two stereotypes that apply to dependency relationships among packages.
  - access – source package is granted the right to reference the elements of the target package.
  - import – a kind of access, but only public content.
- Two stereotypes that apply to dependency relationships among use case.
  - extend – target use case extends the behavior of source.
  - include – source use case explicitly incorporates the behavior of another use case at a location specified by the source

Dependency

- Three stereotypes when modeling interactions among objects.
  - become – target is the same object of source at later time
  - call – source operation invoke the target operation
  - copy – target is an exact, but different, copy of source
- In the context of state machine
  - send – source operation sends the target event
- In the context of organizing the elements of your system into subsystem and model
  - trace – target is an historical ancestor of the source (*model relationship among elements in different models*)

Generalization

- A generalization is a relationship between a general thing and a more specific kind of that thing.

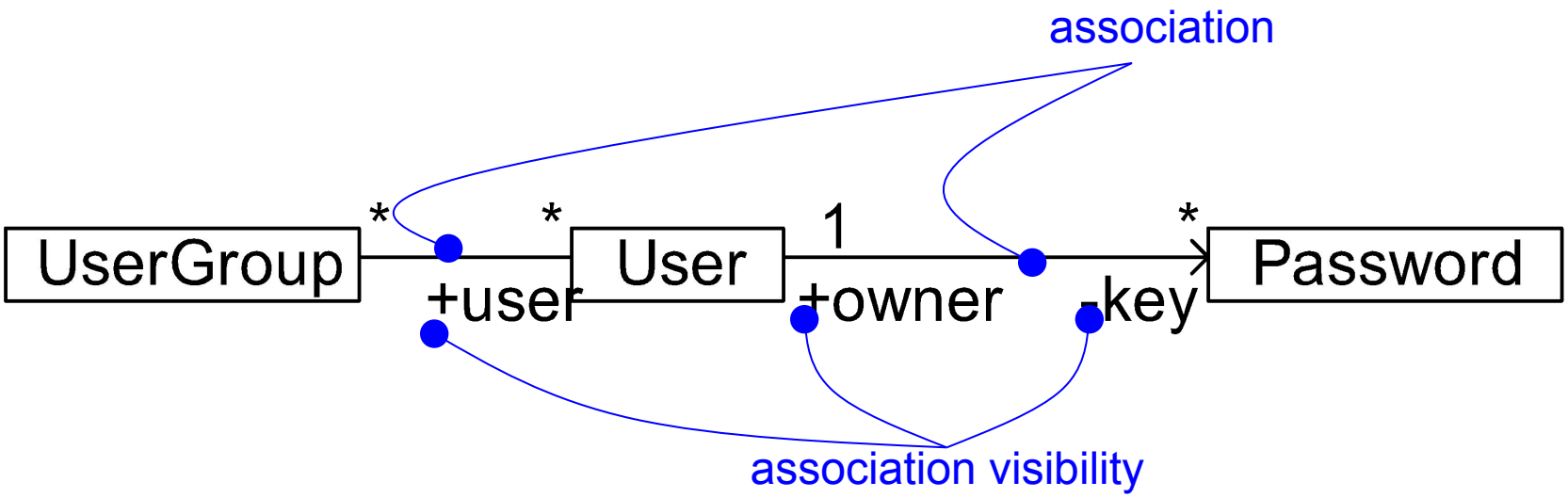


Association

- An association is a structural relationship, specifying that objects of one thing are connected to object of another.
- Basic adornments: name, role, multiplicity, aggregation.
- Advanced adornments: navigation, qualification, various flavors of aggregation.

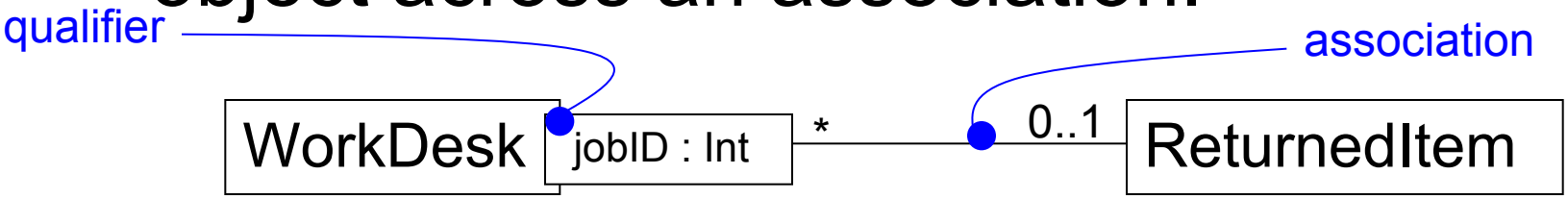
Association

- Navigation : adorning an association with an arrowhead pointing to the direction of traversal.
- Visibility : objects at that end are not accessible to any objects outside the association.

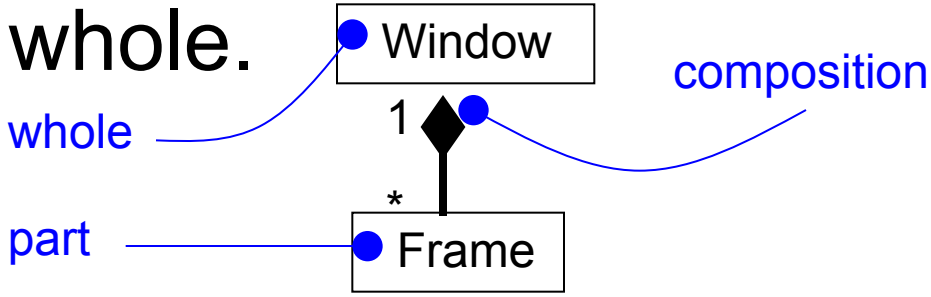


Association

- Qualification : is an attribute whose values partition the set of objects related to an object across an association.



- Composition : A form of aggregation with strong ownership and coincident lifetime of the parts by the whole.



Association

- Constraints
  1. implicit: The relationship is not manifest but, rather, is only conceptual.
  2. ordered: the set of objects at one end of an association are in an explicit order.
  3. changable: links between objects may be changed.
  4. addOnly: new links may be added from an object on the opposite end of association.
  5. frozen: a link added may not be modified or deleted.
  6. xor: over a set of associations, exactly one is manfest for each associated object.

Realization

- A realization is a semantic relationship between classifiers in which one classifier specifies a contract that anther classifier guarantees to carry out.
- Use in two circumstances:
  - In the context of interfaces.
  - In the context of collaborations.
- Rendering as [ ]