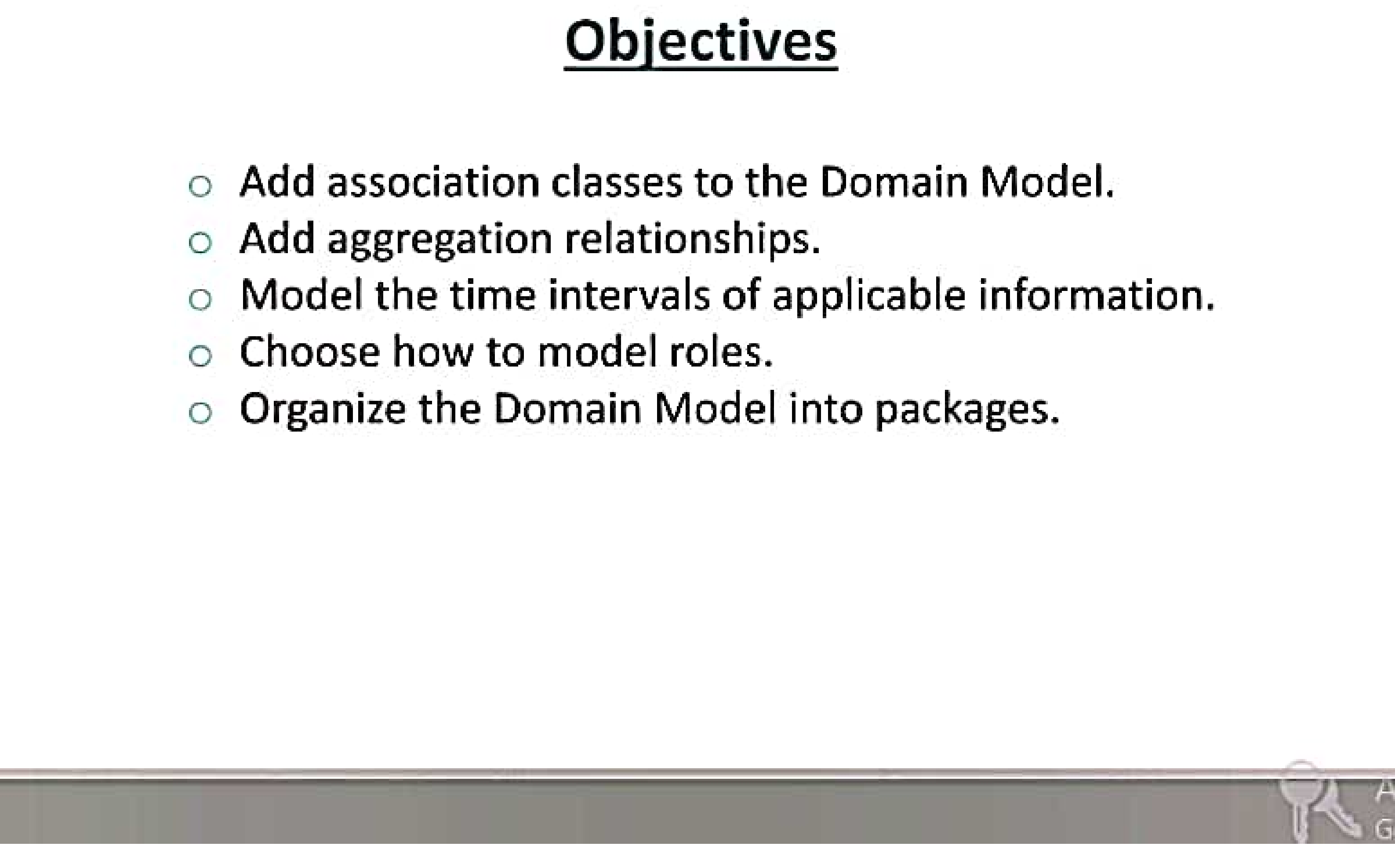
DOMAIN MODEL REFINEMENT



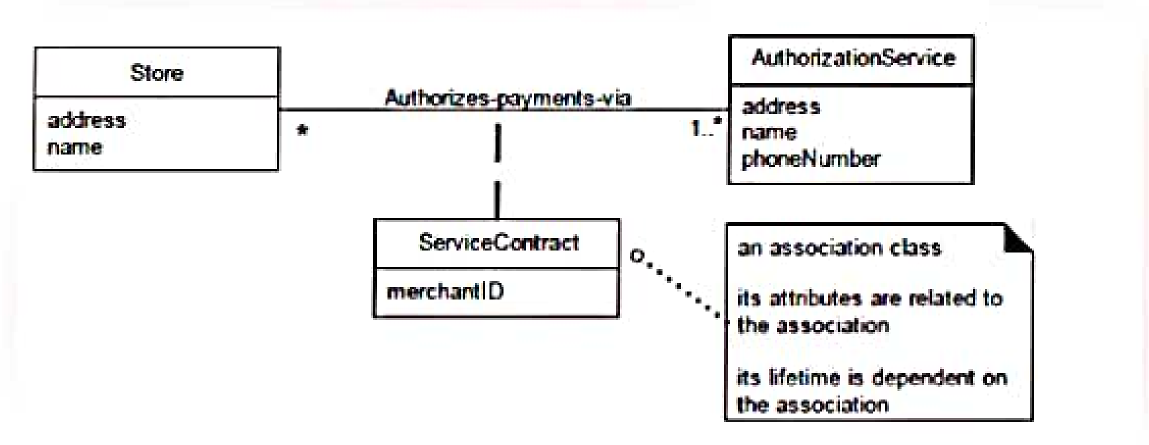
Objectives

- Add association classes to the Domain Model.
- Add aggregation relationships.
- Model the time intervals of applicable information.
- Choose how to model roles.
- Organize the Domain Model into packages.

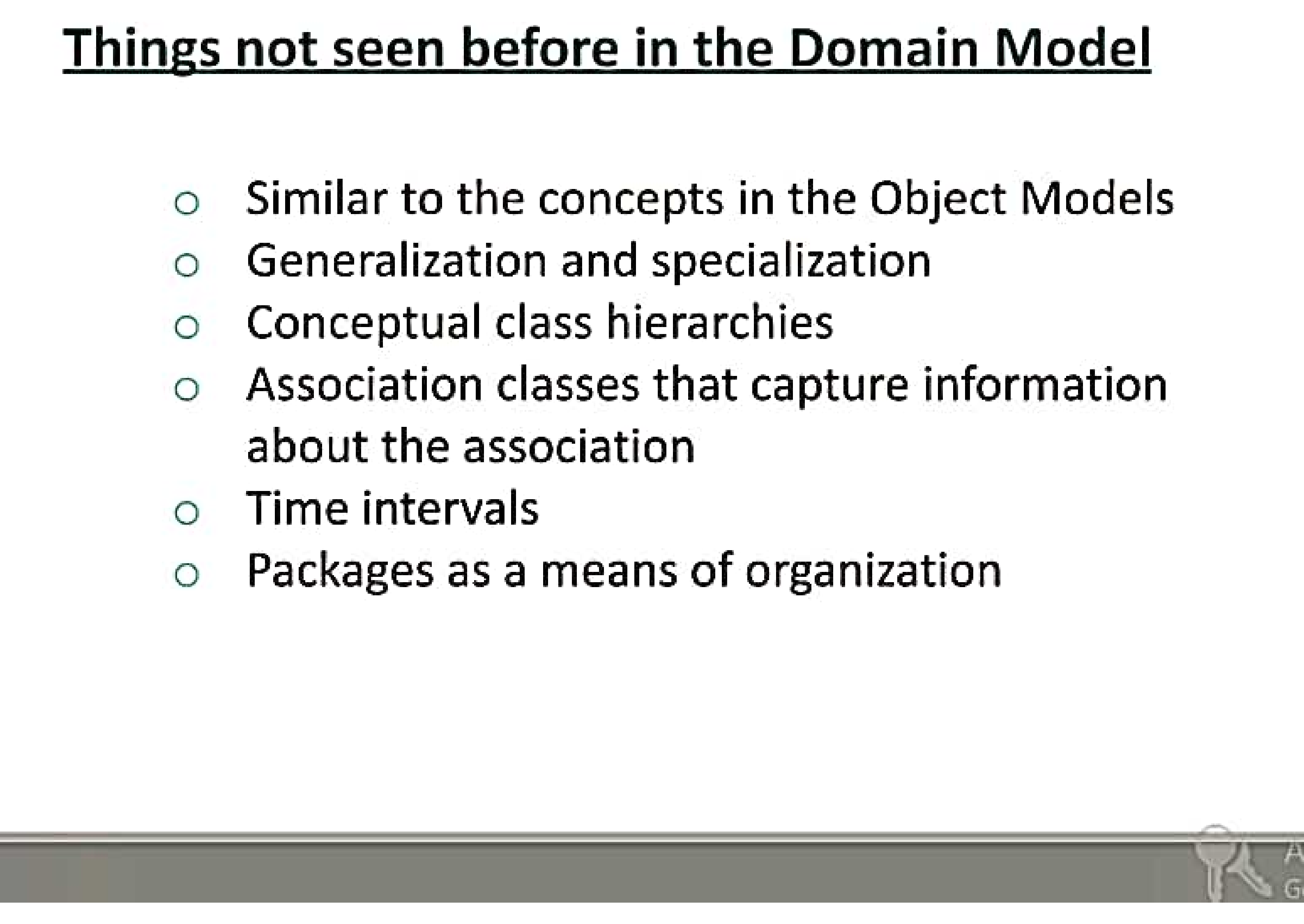
This leads to The notion of an association class ,in which we can add features to the association itself.

Service Contract may be modeled as an association class related to the association between Store and Authorization Service.
In the UML, this is illustrated with a dashed line from the association to the association class.

It visually communicates the idea that a ***Service-Contract*** and its attributes are related to the association between a *Store* and ***Authorization Service,*** and that the lifetime of the ***Service Contract*** is dependent on the relationship.



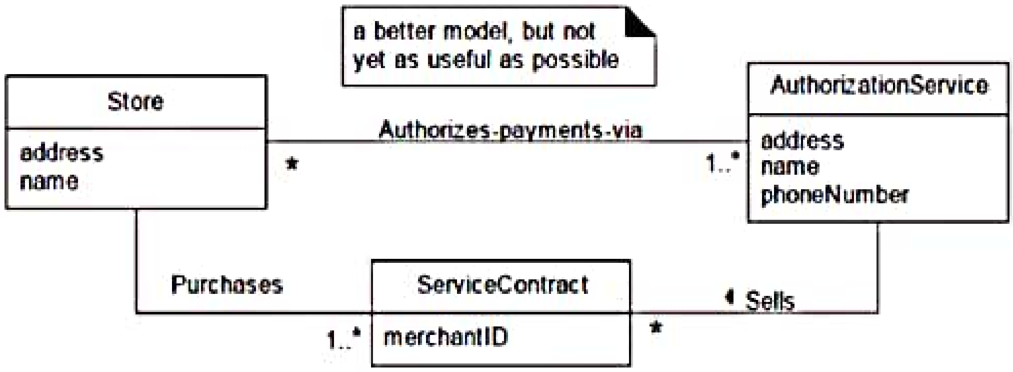
Association class



Things not seen before in the Domain Model

- Similar to the concepts in the Object Models
- Generalization and specialization
- Conceptual class hierarchies
- Association classes that capture information about the association
- Time intervals
- Packages as a means of organization

Association Classes



First attempt at modeling the merchantID problem.

The fact that both *Store* and ***AuthorizationService*** are related to ***ServiceContract*** is a clue that it is dependent on the relationship between the two. The *merchantID* may be thought of as an attribute related to the association between *Store and AuthorizationService.*

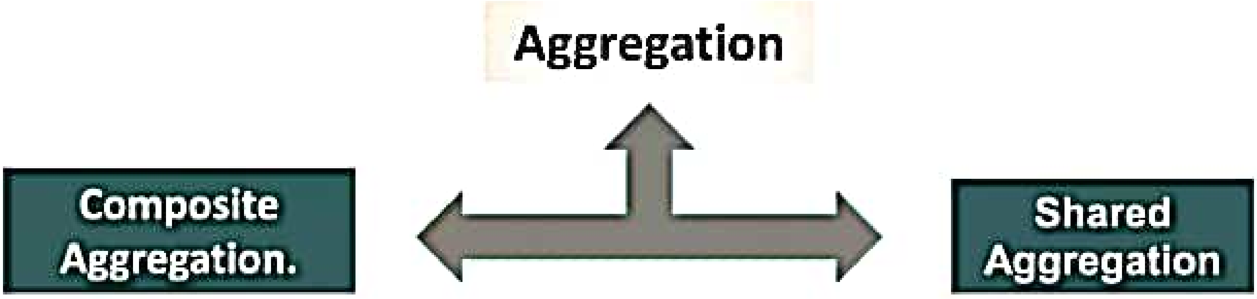
Aggregation and Composition

Aggregation is a kind of association used to model whole-part relationships between things.

The whole is called the **composite.**

Aggregation is shown in the UML with a hollow or filled diamond symbol at the composite end of a whole-part association.

Aggregation is a property of an association role.



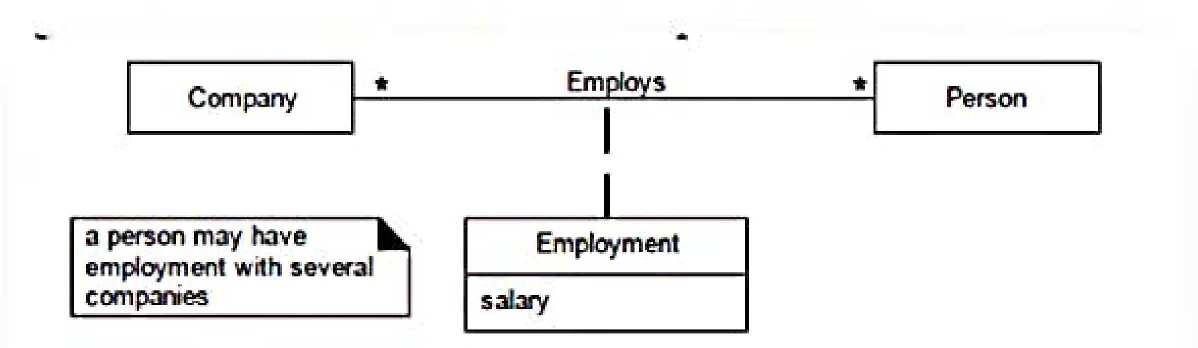
Aggregation

Composite Aggregation.

Shared Aggregation

Guidelines For Adding Association Classes

- ❑ An attribute is related to an association.
- ❑ Instances of the association class have a life-time dependency on the association.
- ❑ There is a many-to-many association between two concepts, and information associated with the association itself.

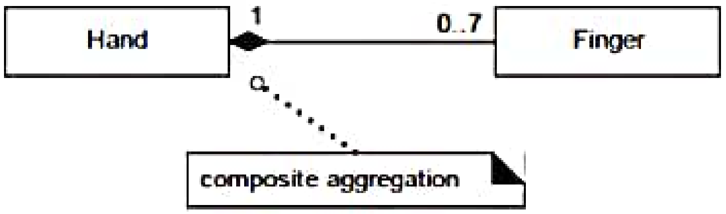


examples of association classe.

Composite Aggregation

- Represented with filled diamond.
- Means that the part is a member of only one composite object, and that there is an existence and disposition dependency of the part on the composite.
- Composition is signified with a filled diamond. It implies that the composite solely owns the part, and that they are in a tree structure part hierarchy.
- it is the most common form of aggregation shown in models.
- **Example Composition Aggregation**

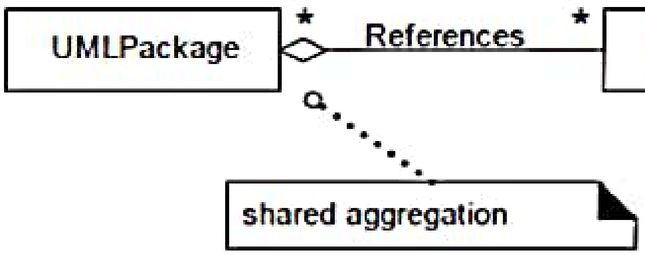
  A finger is a part of at most one hand ,thus the aggregation diamond is filled to indicate composite aggregation



For Example, in a human body domain model ,one thinks of the hand as including the fingers, so if one says, A hand has come into existence, we understand this to also mean that fingers have come into existence as well.

Shared Aggregation

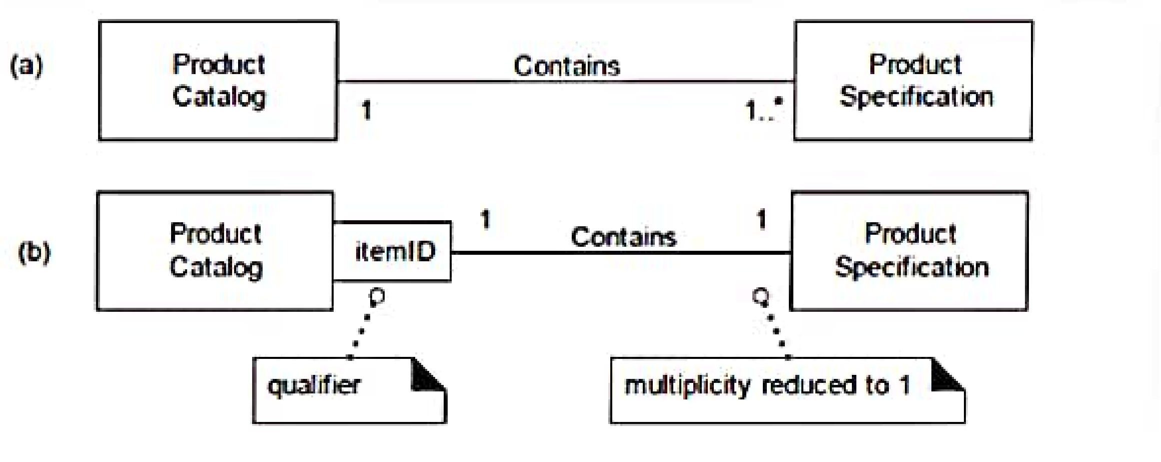
- **Represented with Hollow diamond.**
- Means that the multiplicity at the composite end may be more than one.
- Implies that the part may be simultaneously in many composite instances.
- For instance, a UML package may be considered to aggregate its elements. But
- an element may be referenced in more than one package.



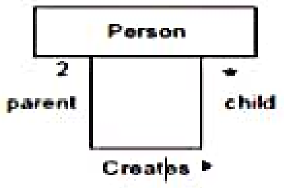
Qualified Associations

A qualifier may be used in an association; it distinguishes the set of objects at the far end of the association based on the qualifier value. An association with a qualifier is a **qualified association.**

For example, *ProductSpecifications* may be distinguished in a *ProductCatalog* by their ***itemID,***



Reflexive Association

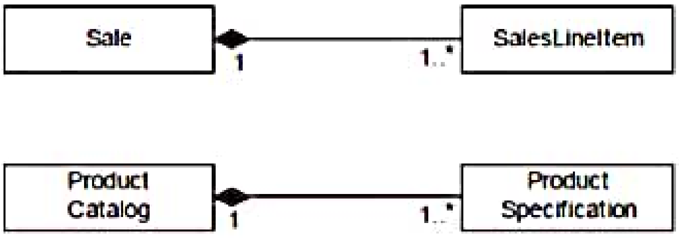


an association to itself; this is known as a **reflexive association,**

Aggregation in the POS Domain Model

In the POS domain, the ***SalesLineItems*** may be considered a part of a composite *Sale;*

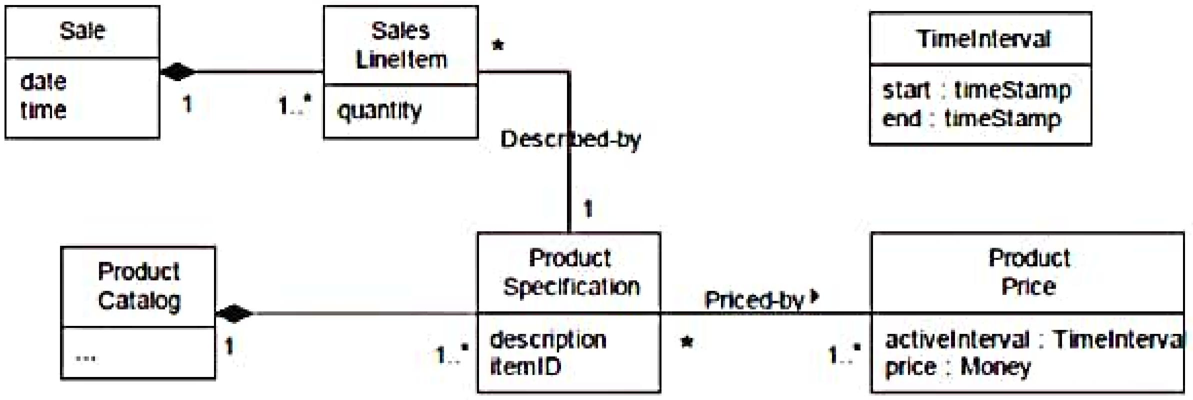
Similarly ***ProductCatalog*** is an aggregate of ***Product-Specifications***.



Time Intervals

If a *SalesLineItem* always retrieved the current price recorded in a *Product-Specification,* then when the price was changed in the object, old sales would refer to new prices, which is incorrect.
So we can use timeinterval class to solve this problem .

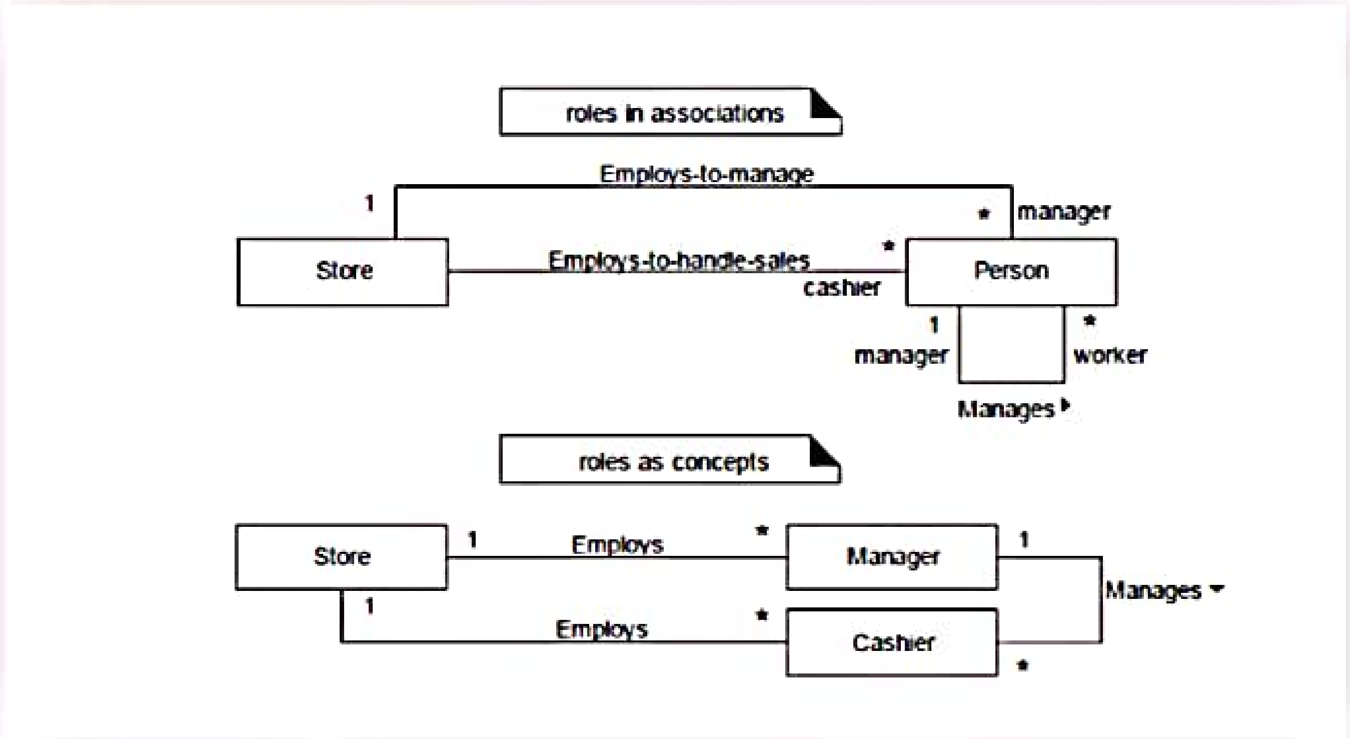
This is to associate a collection of *ProductPrices* with a *ProductSpecification,* each with an associated applicable time interval. so that it can record all past prices and also record future planned prices



Two ways to Model Human Roles

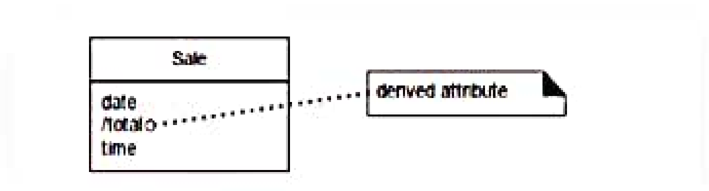
Roles as Concepts

Roles in Associations



Derived Elements

A derived element can be determined from others. Attributes and associations are the most common derived elements.



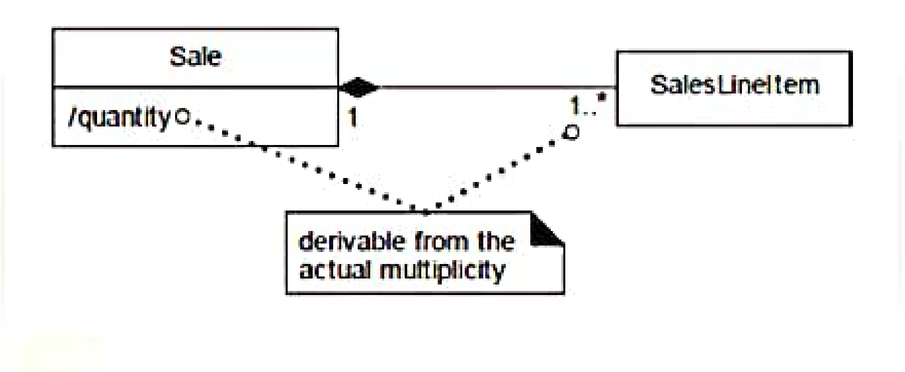
Association Role Names

Each end of an association is a role, which has various properties, such as:

- name
- multiplicity

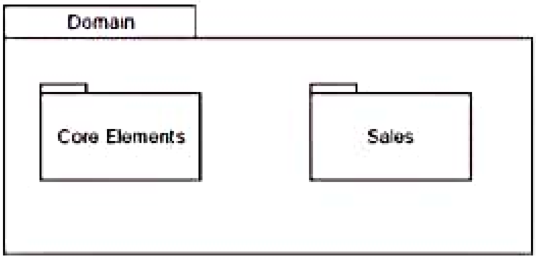
A role name identifies an end of an association and ideally describes the role played by objects in the association.

Role name usually starts with a lowercase letter.



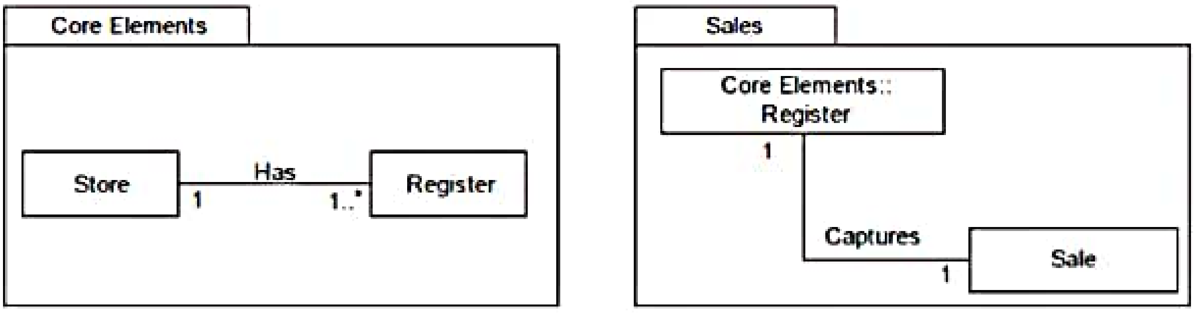
Package Notation

- In UML package is shown as a tabbed folder.
- Subordinate packages may be shown within it.
- The package name is within the tab if the package depicts its elements;
- otherwise, it is centered within the folder itself.



Ownership and References

- An element is owned by the package within it is defined, **but may be referenced in other packages.**
- In that case, the element name is qualified by the package name using the pathname format Package Name Element Name.
- **A class shown in a foreign package** may be modified with new associations, but must otherwise remain unchanged.

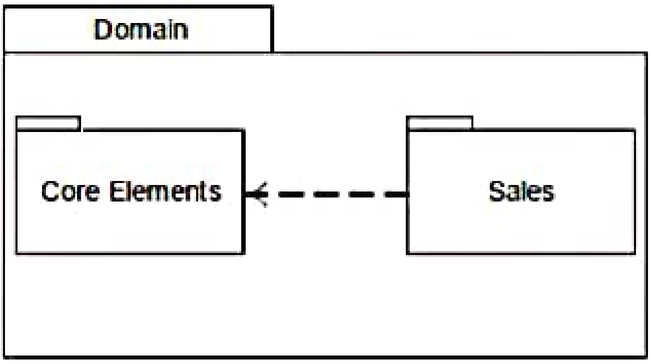


Package Dependencies

- A package dependency indicates that elements of the dependent package in some way know about or are coupled to elements in the target package.

- If a model element is in some way dependent on another, **the dependency shown with a dependency relationship, depicted with an arrowed line.**

For example, if a package references an element owned by another, a dependency Exists.


Domain
Core Elements
Sales