OBJECT-ORIENTED MODELING AND DESIGN

Class Modeling

CONTENTS:

Class Modeling:

Captures the static structure of a system by characterizing the objects in the system, the relationships between the objects, and the attributes and operations for each class

1. Object and Class Concepts:

Object: is a concept, abstraction, or a thing with identity that has meaning for an application Eg. two apples) each have identity and are distinguishable.
Class: Describes a group of objects with the same properties(attributes), behavior (operations), kinds of relationship, and Semantics. (*Eg: Person, company, Process and Window*)

Class Diagram: Provide a Graphical notation for modeling classes and their relationships, thereby describing possible objects as shown in figure 1.

Object Diagram: Shows individual Objects and their relationships.

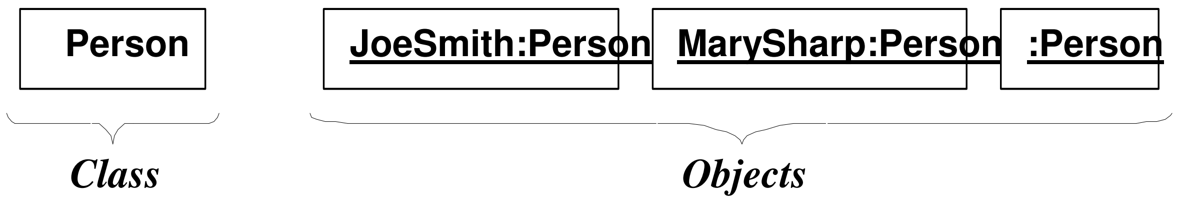


Figure 1: A Class and Objects: Objects and Classes are the focus of Class Modeling

Values: A Value is a piece of data. **Attributes:** named property of class. These elaborate classes as shown in figure 2.

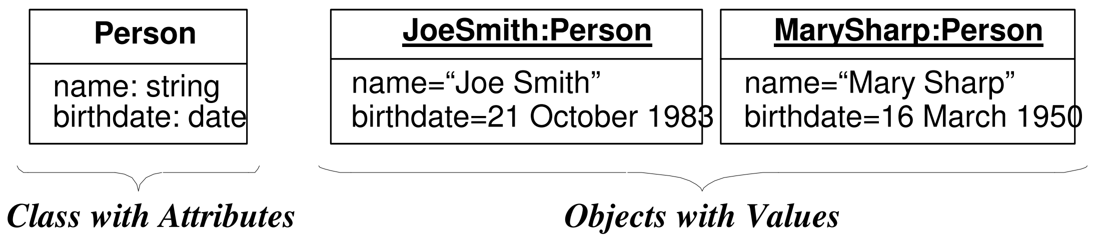


Figure 2: Attributes and Values.

Operations: is a function or procedure that may be applied to or by objects in a class as shown in figure 3.

Methods: is a implementation of an operation for a class.

Polymorphism: same operation applied to many different class.

Signature: all methods must have same signature ie *print* should not have *filename* as argument for one method and *filepointer* as argument for other.

Feature: generic word for either an attribute or operation.

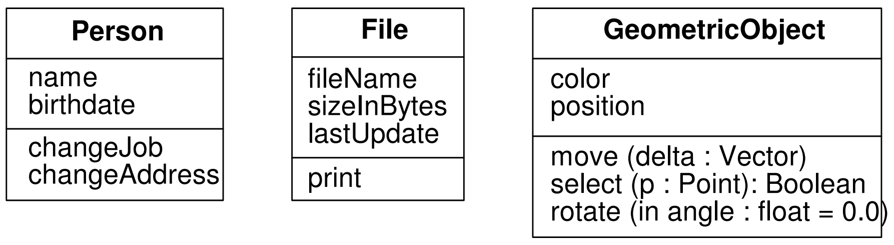


Figure 3 : Operations : function that may be applied to or by objects in a class.

Summary of notations for classes: Figure 4 summarizes the notations for classes.

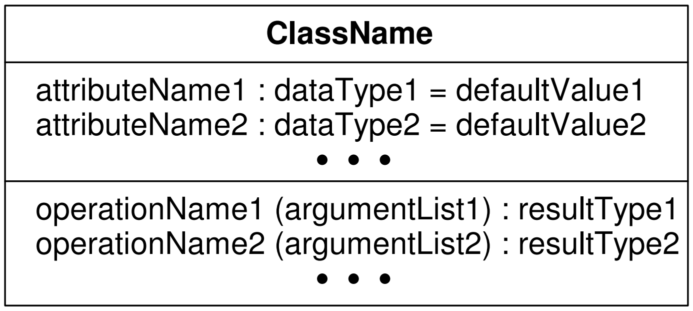

ClassName
attributeName1 : dataType1 = defaultValue1
attributeName2 : dataType2 = defaultValue2
• • •
operationName1 (argumentList1) : resultType1
operationName2 (argumentList2) : resultType2
• • •

Figure 4: Summary of modeling notations for classes: A Box represents a class and may have as many as three components.

The compartments in the box contain, from top to bottom: class name, list of attributes, and list of operations. Optional details are type, default values, argument list and result type may follow each operation name.

2. Link and Association Concepts:

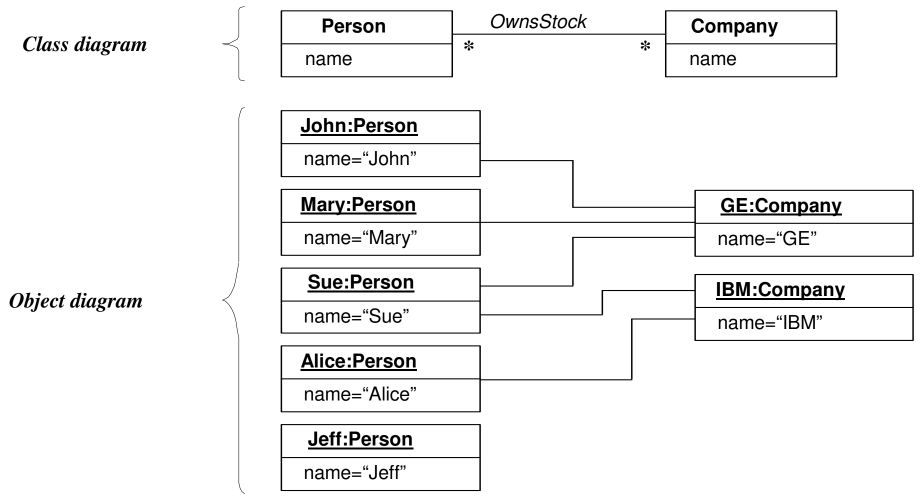


Figure 5: Many to Many Associations

Link: is a physical / conceptual connection among objects most links relate two objects, but some links relate 3 or more object. It is an instance of association as shown in figure 5 above.

Association: is a description of a group of links with common structure and common semantics as in the class diagram shown in figure 5.

Multiplicity: Specifies the number of instances of one class that may relate to single instance of other class. Similar to Cardinality ratio in DBMS as shown in the following figures 6,7,8.

UML specifies: "1": exactly one,
"1 ..*" (one or more)
"3..5" (three to five inclusive)

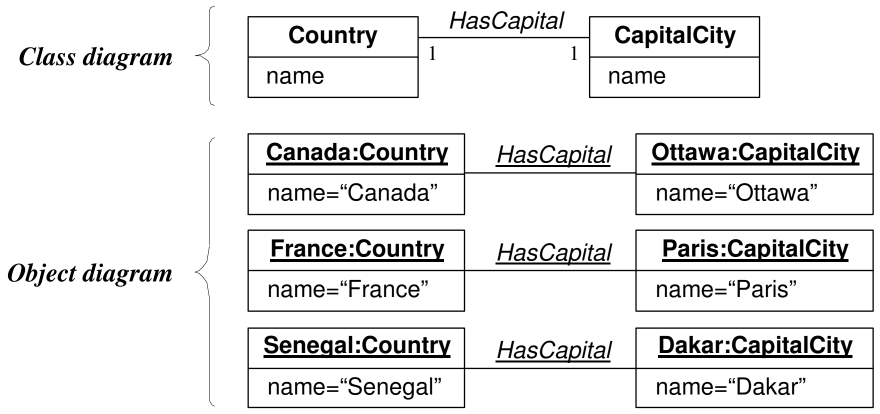


Figure 6: One-to-one Association

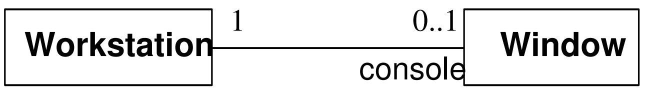


Figure 7:Zero-to-one multiplicity

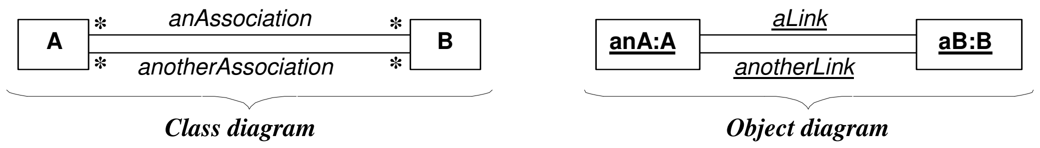


Figure 8: Association Vs Link

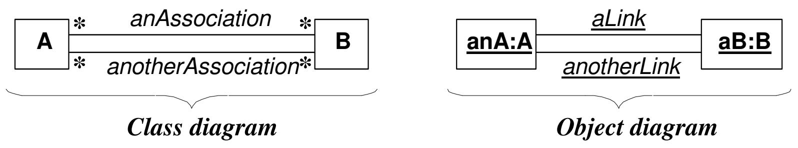


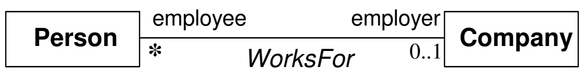
Figure: Multiple association to model multiple links

Figure 9 Association Vs Links. Multiple associations to model multiple links between the same objects

Association End names:

- Association End names is an important concept in UML.
- Association ends can not only be assigned multiplicity they can also be named. In a problem description they are generally identified by nouns

Association end: can give names to the association link.
Use of associations end name: to name links b/w object of same class.



employee	employer
Joe Doe	Simplex
Mary Brown	Simplex
Jean Smith	United Widgets

Figure 10: Association End names. Each end of an association can have a name.

Importance of association end names as in figure 11.

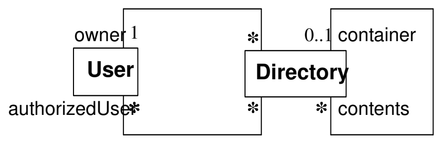


Figure 11: association end names are necessary for association between two objects of same class

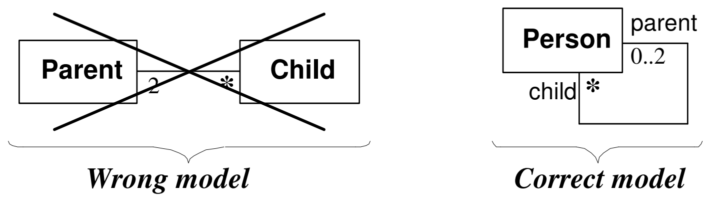


Figure 12: Association end names: Use association end names to model multiple references to the same class

Ordering: association which follows some priority eg. Windows opened on screen follow on explicit order only top most windows is visible on any part of screen
Ordered collection of elements with duplicates not allowed .ref figure 13

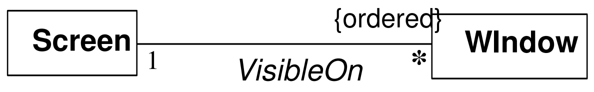


Figure 13 Ordering an object for an association end. Ordering sometimes occurs for " many" multiplicity

Bags and sequences:

- Bag: collection of elements with duplicates allowed.
- Sequence: ordered collection of elements with duplicate allowed eg. An itinerary is a sequence of airports and the same airport can be visited more than once.
- Sequence is ordered bag both allow duplicates, {ordered} and {sequence} is same only difference is sequence allows duplicates as shown in figure 14.

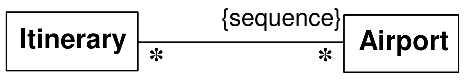


Figure 14: An example of a sequence. A itinerary may visit multiple airports so you should use{ sequence} and not { ordered}

Association Classes:

- It is an association that is also a class.
- Like the links of an association, the instances of an association class derive identity from instances of the constituent classes.
- You can find association classes by looking for adverbs in a problem statement or by abstracting values. ref figure 15,16

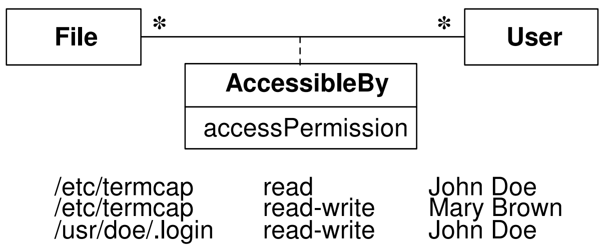


Figure 15: The links of an association can have attributes.

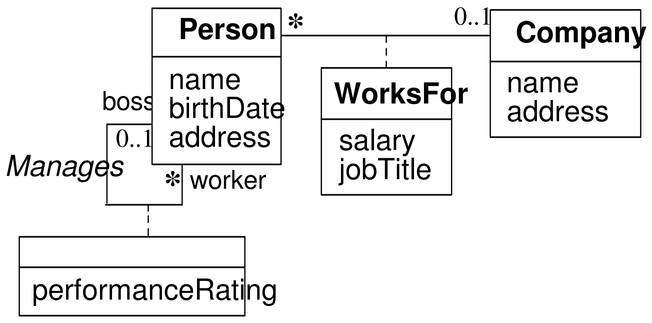


Figure 16: Association classes. Attributes may also occur for one-to-many and one-to-one associations

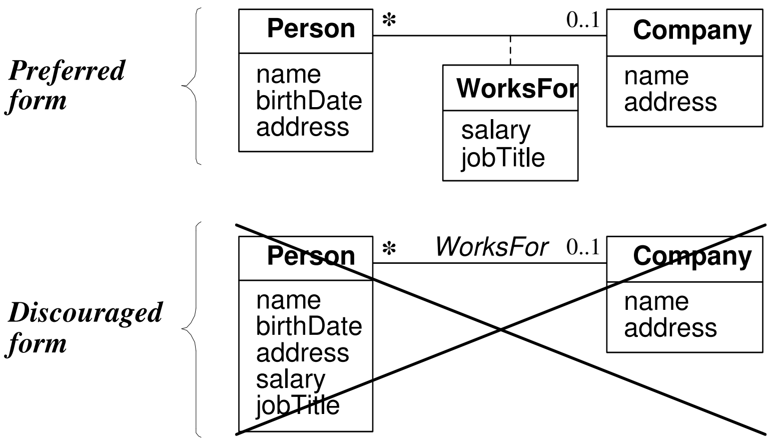


Figure 17: Proper use of association class: Do not fold attributes of an association into a class

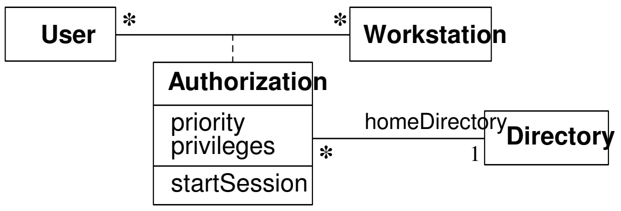


Figure 18: An association class participating in an association.

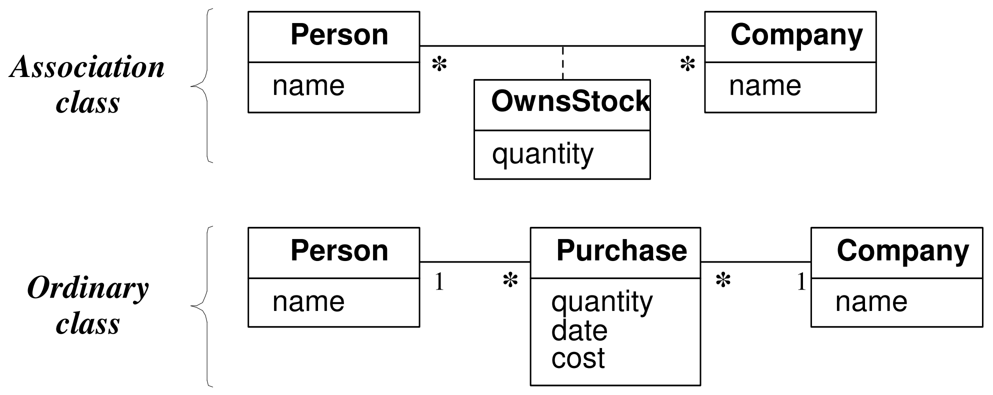


Figure 19: Association class Vs. Ordinary class An association class is much different from an ordinary class

Qualified Associations:

- An association in which an attribute called the qualifier disambiguates the objects for a "many" association end.
- Job of qualifier is to reduce the many association to one. Can only work for m-m and 1-m
- Also facilitates traversal of class models
- Stock exchange gives special unique symbol to each company
- Using qualified association we can make traversal easy.
- Ref Figure 20,21.

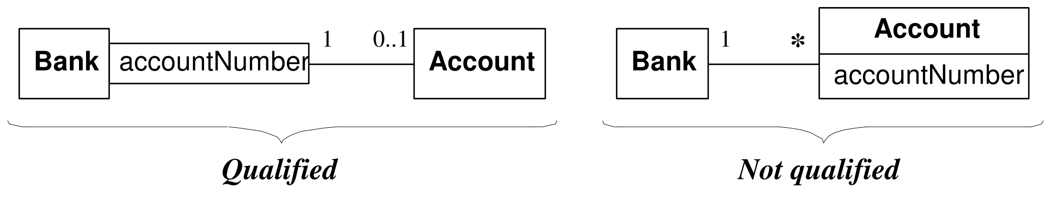
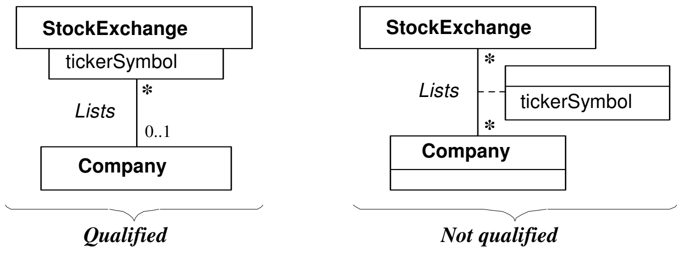


Figure 20: Qualified association. Qualification increases the precision of a model.



3.
Figure 21 : Qualified association. Qualification also facilitates traversal of class models.

3. Generalization and inheritance

- Generalization is the relationship between a class (the super class) and one or more variations of the class (sub class)
- Super class holds the common attributes, operations and associations. Subclass adds specific attributes
- Each subclass inherits features of super class Ancestor and descendents
- Refer figure 22.
- Figure 23 shows the inheritance definition for the case study of describing graphical figures.

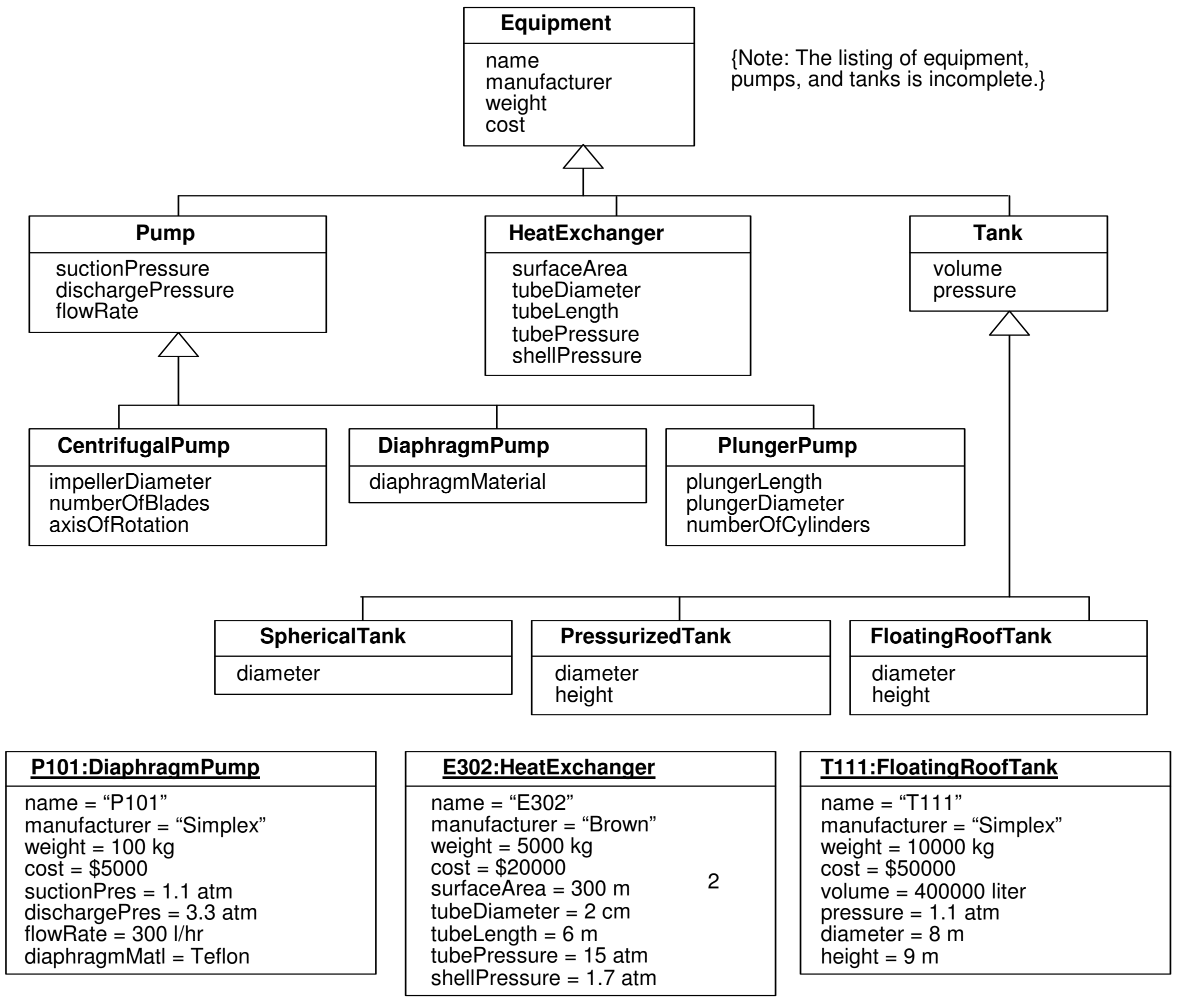


Figure 22: A multilevel Inheritance hierarchy with instances: Generalization organizes classes by their similarities and differences, structuring the description of objects.

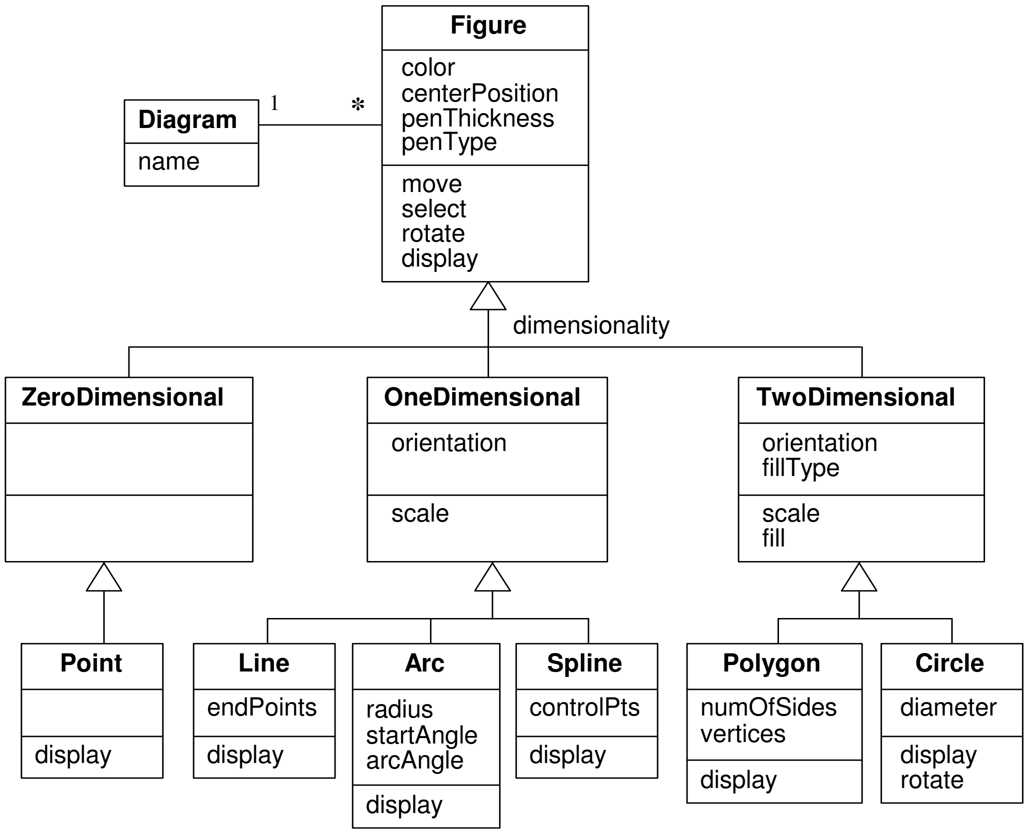


Figure 23: Inheritance for graphic figures. Each subclass inherits the attributes, operations, and associations of its super classes.

Use of Generalization:

- Serves three purposes:
- Support for polymorphism. (call at super class level automatically resolved)
- Second purpose is to structure the description of objects.(a taxonomy is formed)
- Third purpose is to enable reuse of code.

The terms generalization, specialization and inheritance all refer to aspects of the same idea.

- Generalization: derives from the fact that the superclass generalizes the subclasses.
- Specialization: refers to the fact that the subclasses refine or specialize the Superclass.
- Inheritance: is the mechanism for sharing attributes, operations, and associations via generalization / specialization relationship.

Overriding Features:

- A subclass may override a super class feature by defining a feature with the same name.
- We can override methods and default values of attributes.
- Never override the signature, or form, of a feature.

4. A Sample Class Model: figure 24

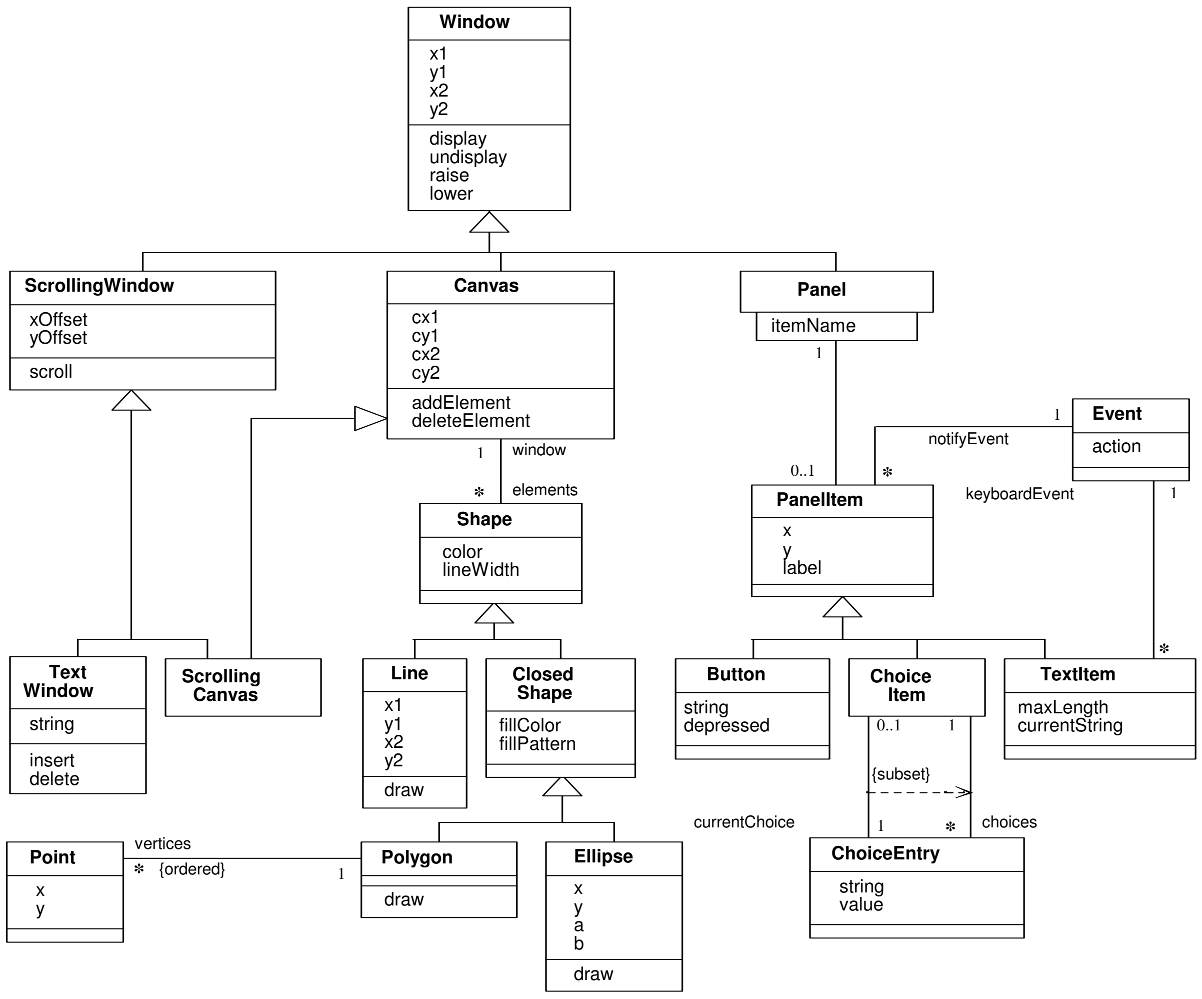


Figure 24. Class model of a windowing system.

The figure 24 shows a class model of a workstation window management system. This model is greatly simplified – a real model would require a number of pages- but it illustrates many class modeling constructs and shows how they fit together.

This simple mode gives a flavor of the use of class modeling.

5. Navigation of class model

Until now we have seen how class model can express the structure of an application. Now we will in this section see how they can also express the behavior of navigating among classes. Navigation is important because it lets you express the behavior of navigating among classes. Navigation is important because it lets you exercise a model and uncover hidden flaws and omissions so that you can repair them. You can perform navigation manually (an informal technique) or write navigation expressions.

Consider the simple model for credit card accounts: we can pose a variety of questions against the model.

- How many credit card account customer has?
- What transaction occurred in a time interval for a customer?
- The UML incorporates a language that can express these kinds of questions- the Object Constraint Language (OCL)

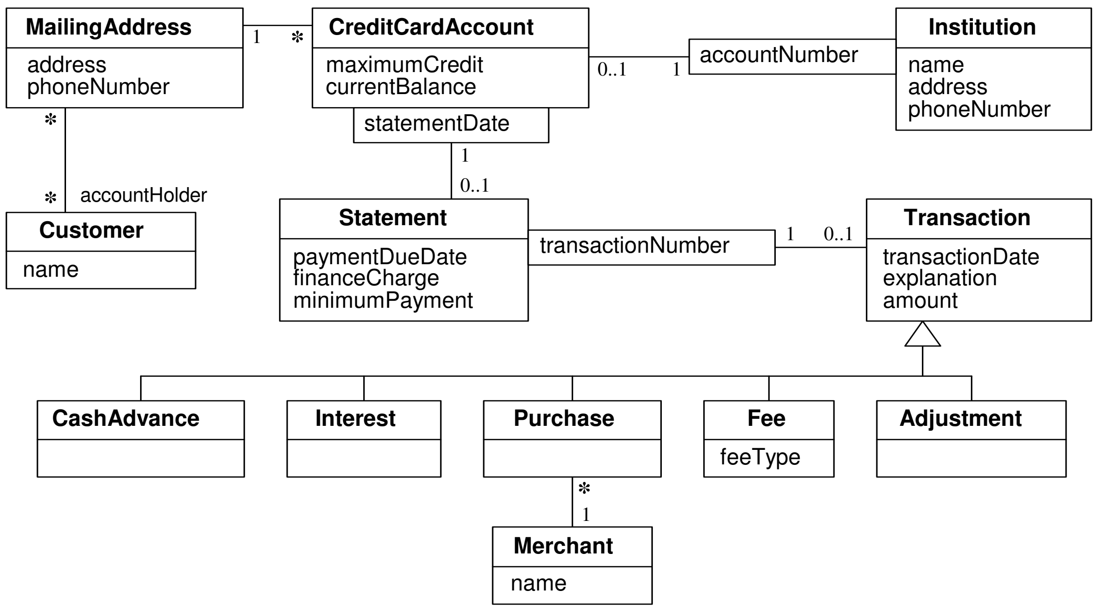


Figure 24: Class model for managing credit card accounts.

Object constraint language :

- Used to answer question pertaining to a class model
- Attribute syntax *aCreditCardAccount.maximumCredit* takes a *CreditcardAccount* object and finds the value of *maximumCredit*
- Operations : The syntax for a collection operation is the source object collection, followed by by -> and then the operation.
- Simple association: A third use of the dot notation is to traverse an association to a target end. The target end may be indicated by an association end name or, where there is no ambiguity, a class name. In the example *aCustomer.MailingaAddress* yields a set of addresses for a customer.
- Qualified association. A qualifier lets you make a more precise traversal.
- Association classes: Given a link of an association class, you can find the constituent objects. Alternatively, given a constituent object, you can find the multiple links of an association class.
- Generalizations: Traversal of a generalization hierarchy is implicit for thr OCl notation.
- Filters: There is often a need to filter the objects in a set. The OCL has several kinds of filters, the most common of which is the *select* operation.

Examples of OCL Expressions:

What transactions occurred for a credit card account within a time interval ?

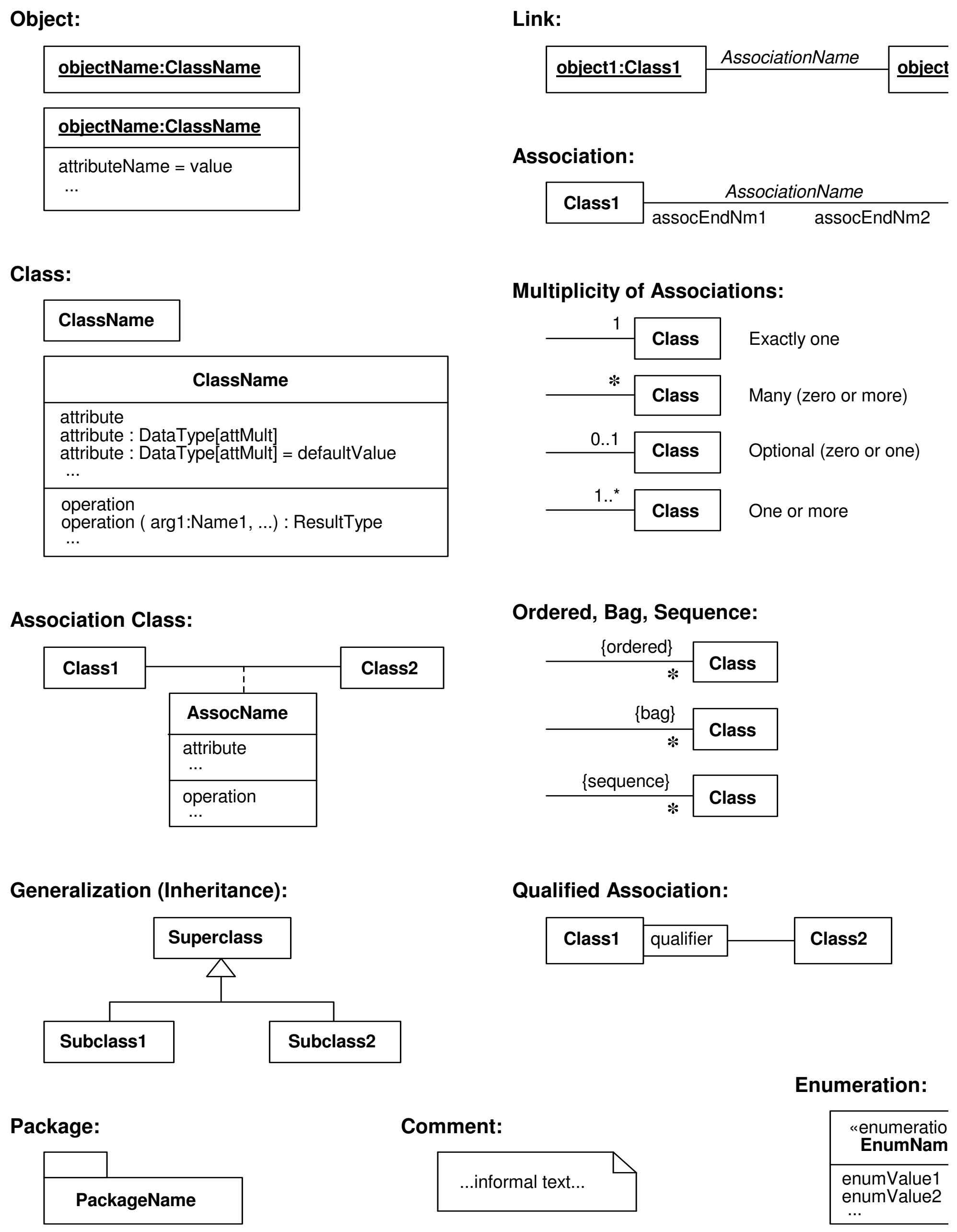
```
aCreditCardAccount.Statement.Transaction-->
select (aStartDate <=transactionDate and
transactiondate <= anEndDate )
```

The above expression traverses from a CreditCardAccount object to Statement and then to Transaction, resulting in a set of transactions. (Traversal of the two associations results in a set, rather then a bag, because both associations are one-to-many.) Then we use the OCL select operator (a collection operator) to find the transactions within the time interval bounded by aStartDate and anEndDate.

Summary:
- Object and Class Concepts.
- Link and Association Concepts
- Generalization and Inheritance
- A Sample Class Model
- Navigation of Class Models

The summary of the basic notations of the class model is listed below:

Class Model Notation — Basic Concepts



Exercise:

1. Prepare an Object diagram for an imaginary round tripe you took last weekend to London. Include at least one instance of each class. Fortunately, direct flights on a hyper sonic plane were available. A friend went with you but decided to stay a while and is still there. Captain Johnson was your pilot on both flights. You had a different seat each way, but noticed it was on the same plane because of a distinctive dent in the tail section. Students should indicate unknown values with a "?".

Answer:

Figure A3.10 shows an object diagram that corresponds to the exercise statement. Note that most attribute values are left unspecified by the problem statement, yet the object diagram is still unambiguous. All objects are clearly identified by their attribute values and links to other objects. The exercise states that "you took a round trip between cities last weekend"; we make the assumption that this is also a round trip between airports.The two seats connected by the dotted line may be the same object.

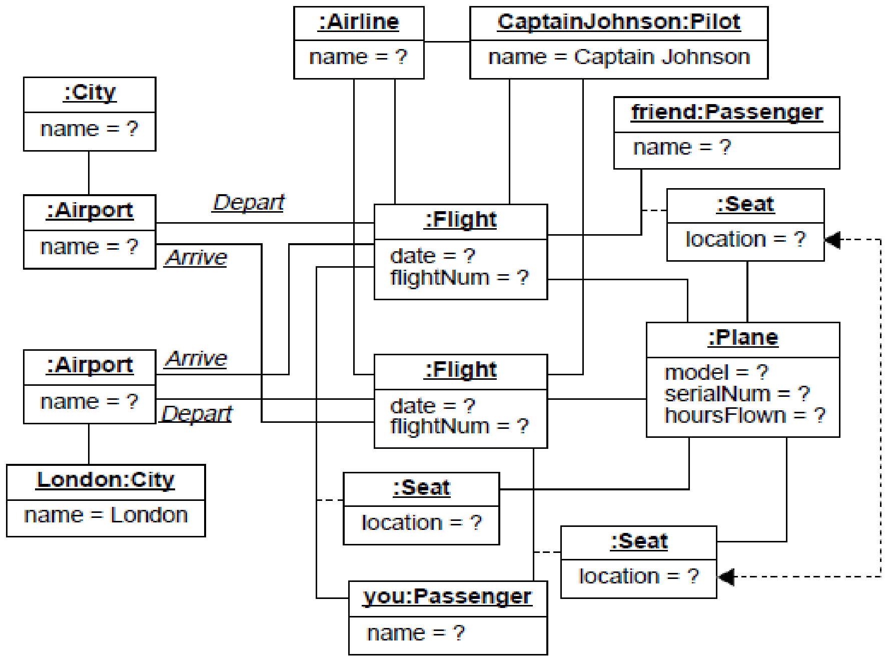


Figure A3.10 Object diagram for an air transportation system

2. Prepare a class diagram for the dinning philosopher problem. There are 5 philosophers and 5 forks around a circular table. Each philosopher has access to 2 forks, one on either side. Each fork is shared by 2 philosophers. Each fork may be either on the table or in use by one philosopher. A philosopher must have 2 forks to eat.

   Answer:

   Figure A3.34 shows a class diagram for the dining philosopher's problem. The one-to one associations describe the relative locations of philosophers and forks. The *InUse* association describes who is using forks. Other representations are possible, depending on your viewpoint. An object diagram may help you better understand this problem

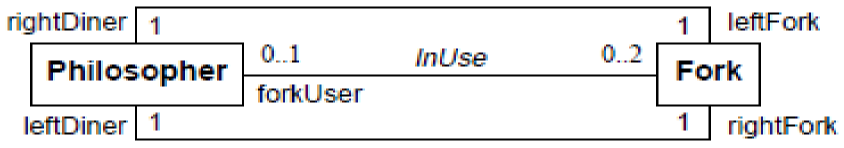


Figure A3.34 Class diagram for the dining philosopher problem