Unit-5
Class Modeling:

Objects

- The purpose of class modeling is to describe objects.
- An object is a concept, abstraction or thing that has meaning for a domain/application.
- Some objects have real world counterparts while others are conceptual entities.
- The choice of objects depends on judgment and the nature of problem.
- All objects have identity and are distinguishable.

Classes

- An **object** is an **instance** – occurrence – of **a class**
- **A class describes a group of objects with the same properties (attributes), behavior (operations), kinds of relationships, and semantics.**
- **The objects in a class share a common semantic purpose, above and beyond the requirement of common attributes and behavior.**
- **By grouping objects onto classes we abstract a problem.**

UML representation of classes/objects:

- **UML: Unified Modeling Language (OMG Standard): O.O Visual Modeling language**

- **Class/object representation**

Figure 3.1 A class and objects. Objects and classes are the focus of class modeling.

Values and attributes

- **Value : piece of data.**
- **Attribute: a named property of a class that describes a value held by each object of the class.**
  - **Attributes may be discovered by looking for adjectives or by abstracting typical values.**
  - **Don’t confuse values with objects:**
    - **An attribute should describe values, not objects.**
    - **Unlike objects, values lac identity**

UML representation

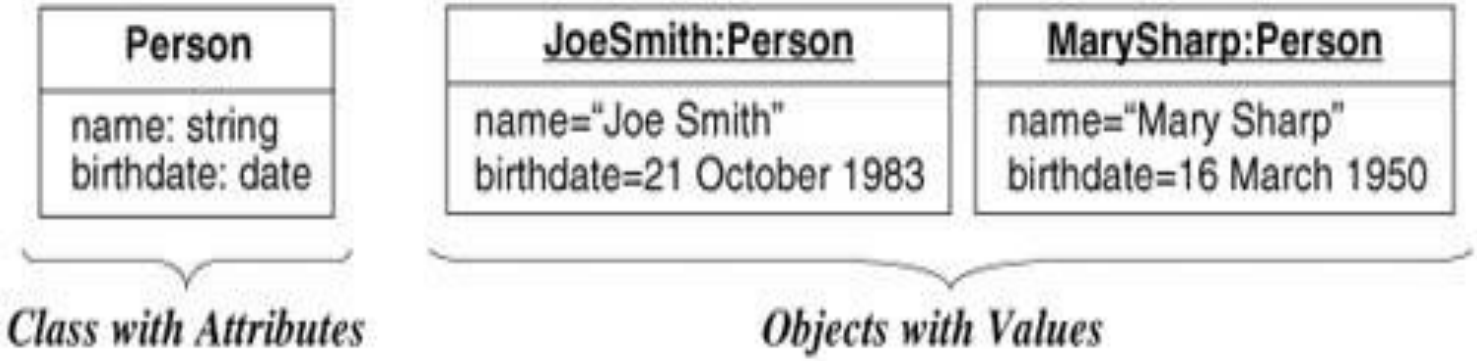

Figure 3.2 Attributes and values. Attributes elaborate classes.

Object identifiers

- Objects identifiers are implicit.
- Objects belonging to the same and having the same attributes values may be different individual objects.

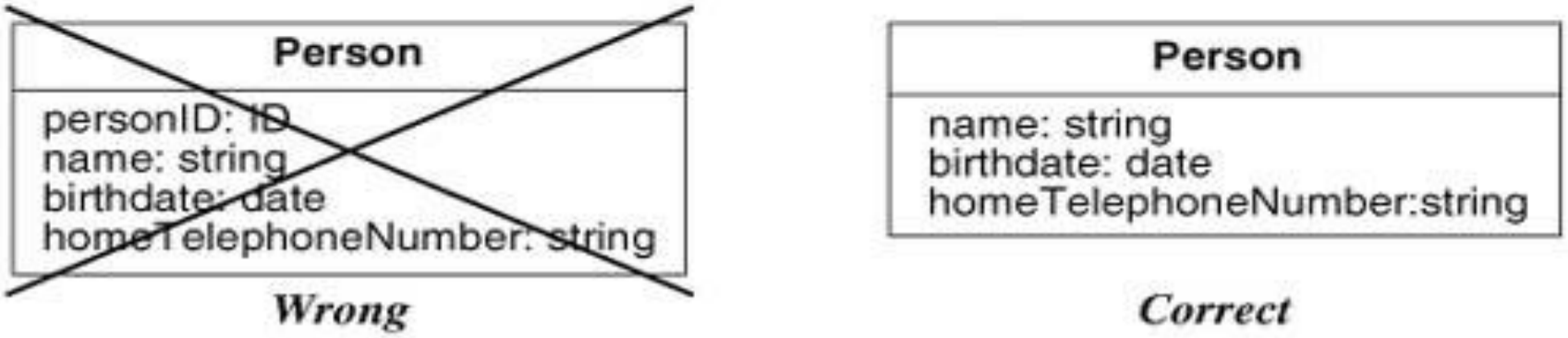

Figure 3.3 Object identifiers. Do not list object identifiers; they are implicit in models.

Operations and Methods

- **An operation is a function or procedure that may be applied to or by objects in a class.**
- **Each operation has a target object as an implicit parameter.**
- **All objects in a class share the same operations.**
- **The behavior of the operation depends on the class of its target.**
- **The same operation may apply to many different classes. Such an operation is POLYMORPHIC.**

Operations and Methods

- **A method is the implementation of an operation for a class.**
- **A different piece of code may implement each method.**
- **An operation may have arguments in addition to its target object. These arguments may be placeholders for values and/or for objects.**
- **When an operation has methods on several classes these methods must have the same SIGNATURE: number and types of arguments, type of result value.**

UML representation

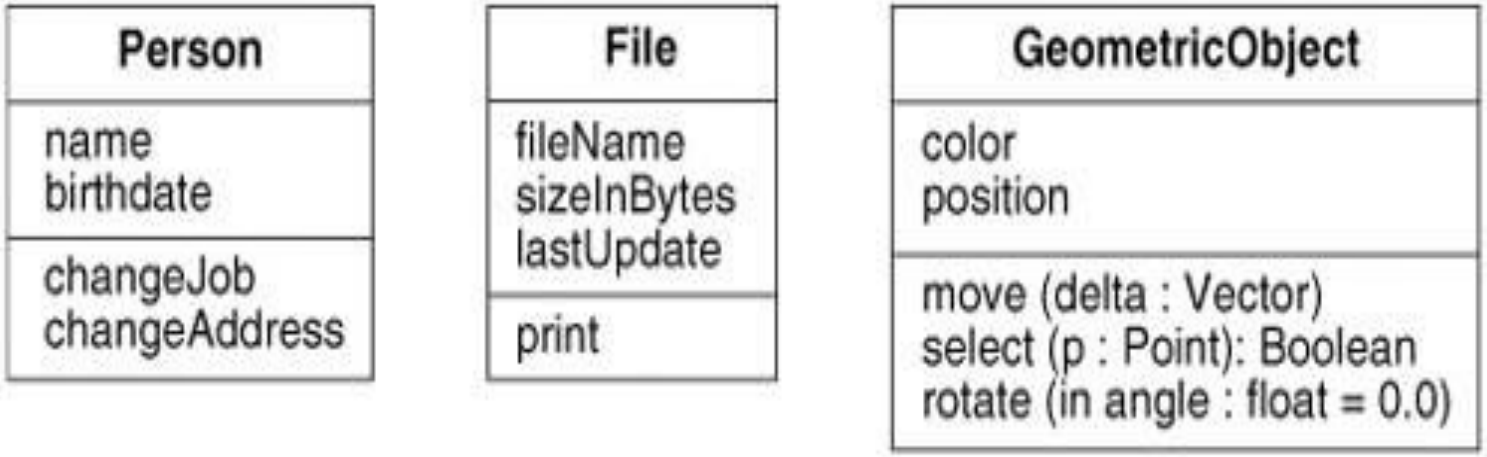

Figure 3.4 Operations. An operation is a function or procedure that may be applied to or by objects in a class.

Links and Association concepts

- A link is a physical or conceptual connection among objects.
- Most links relate two objects, but some links relate three or more objects.
- A link is defined as a tuple, that is a list of objects.
- A link is an instance of an association.
- An association is a description of a group of links with common structure and semantics.
- Association is denoted by a line. Its name is optional if the model is unambigious.

Examples

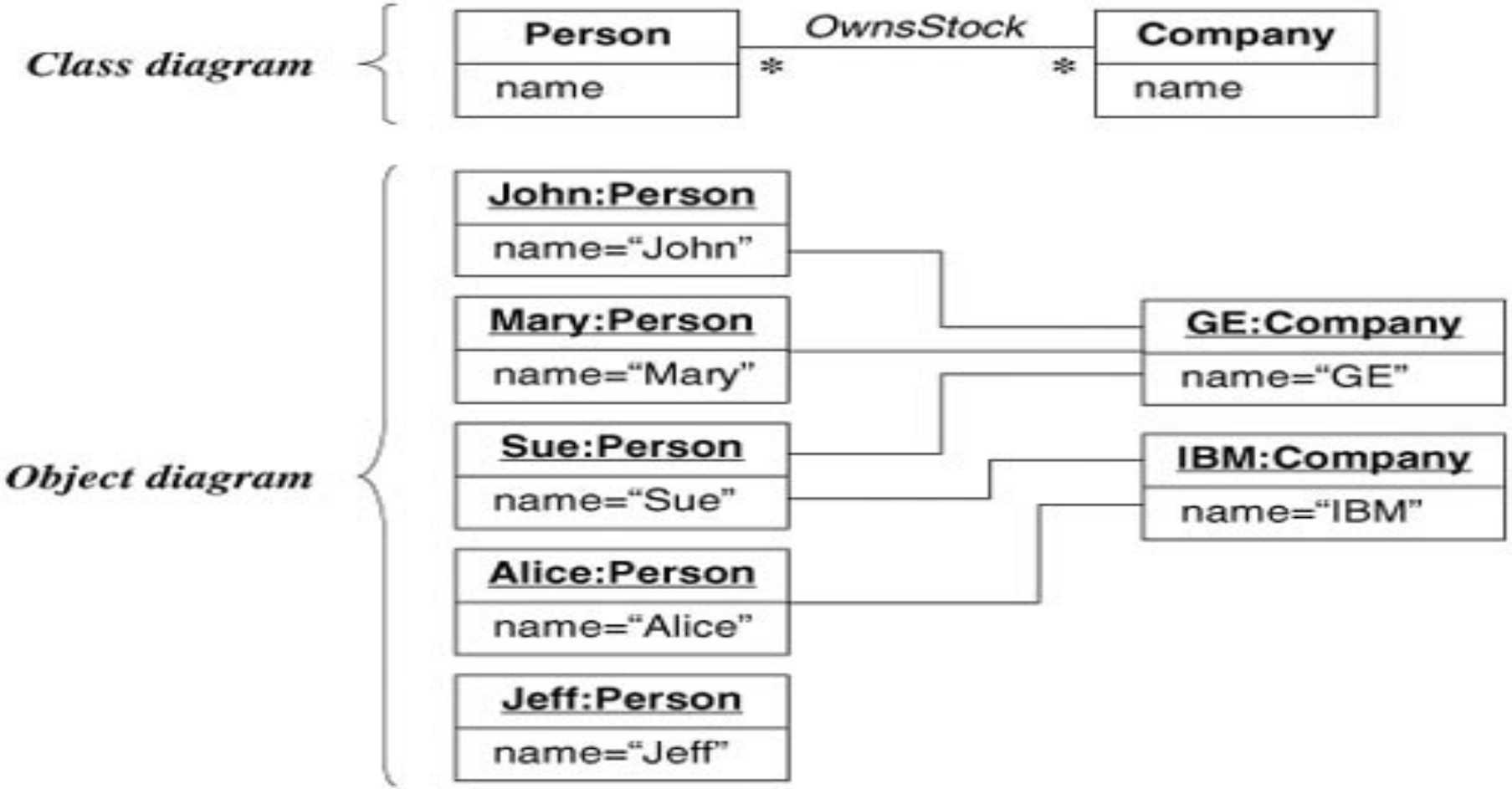

Figure 3.7 Many-to-many association. An association describes a set of potential links in the same way that a class describes a set of potential objects.

- Associations are inherently bi-directional.
- The association name is usually read in a particular direction but the binary association may be traversed in either direction.

A reference is an attribute in one object that refers to another object.

Multiplicity

- It specifies the number of instances of one class that may relate to a single instance of the associated class.

- UML diagrams explicitly list multiplicity at the end of association lines.
- Intervals are used to express multiplicity:
  - 1 (exactly one)
  - 0..1 (zero or one)
  - 1..* (one or more)
  - 0..* (zero or more)
  - 3..5 (three to five inclusive)

Association: ordering, bag, sequence

- On a ‘many” association end, sometimes, it is required that objects have an explicit order.
- In this case the ordering is an inherent part of the association
- Example:

Figure 3.15 Ordering the objects for an association end. Ordering sometimes occurs for “many” multiplicity.

Association class

- UML offers the ability to describe links of association with attributes like any class.
- An association class is an association that is also a class.

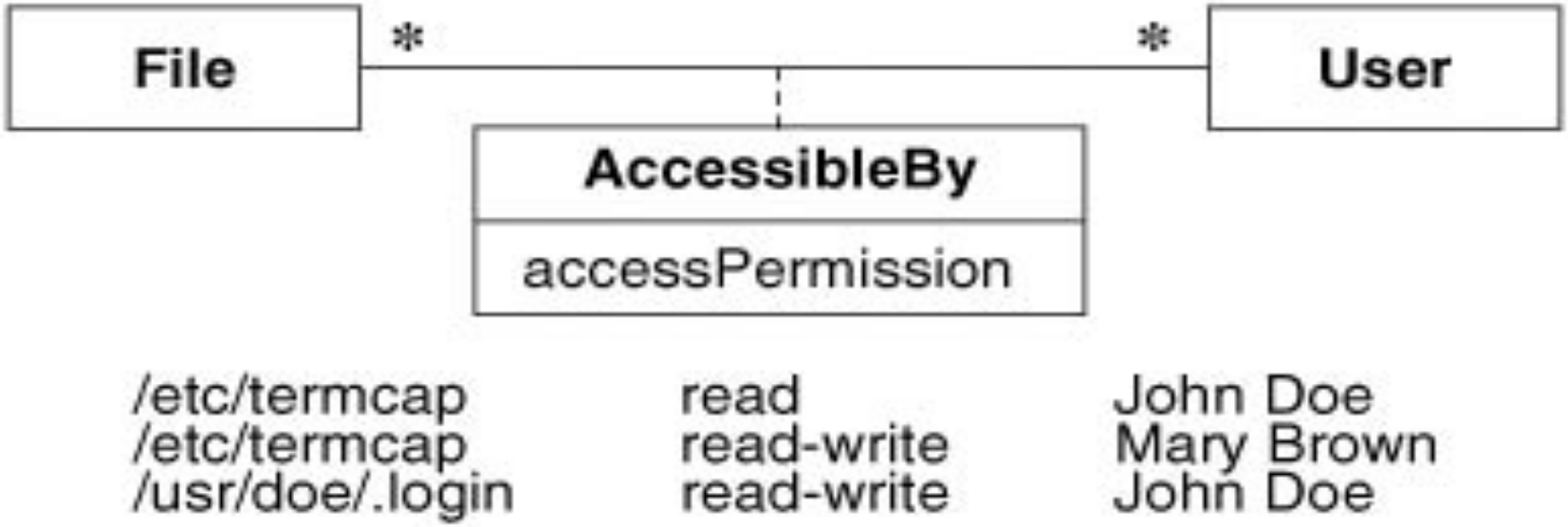

Figure 3.17 An association class. The links of an association can have attributes.

Association class

- Examples:

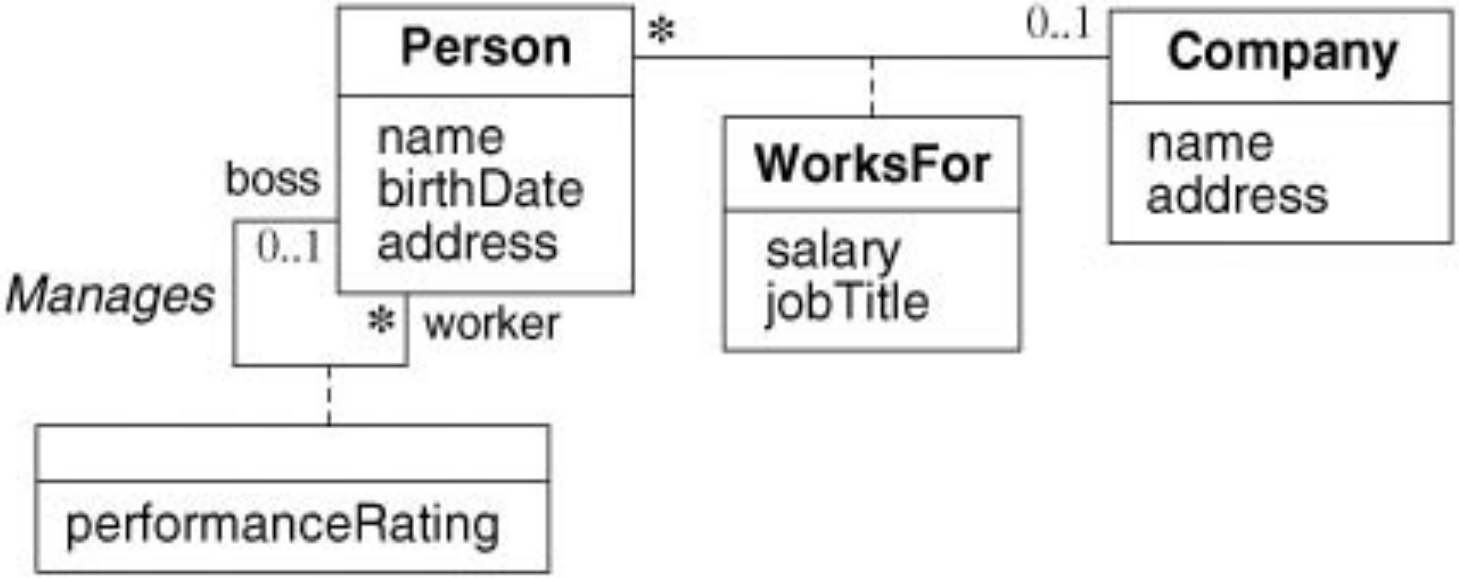

Figure 3.18 Association classes. Attributes may also occur for one-to-many and one-to-one associations.

Generalization/Inheritance

- Generalization is the relationship between a class (**superclass**) and one or more **variations** of the class (**subclasses**).
- Generalization organizes classes by their **similarities** and their **differences, structuring** the descriptions of objects.
- A superclass holds **common** attributes, attributes and associations.
- The subclasses **adds specific** attributes, operations, and associations. They i**nherit** the features of their superclass.
- Often **Generalization** is called a "**IS A"** relationship
- **Simple generalization** organizes classes into a **hierarchy**.
- A subclass may **override** a superclass **feature (**attribute default values, operation) by **redefining a feature with the same name**.
- Never override the signature of methods.

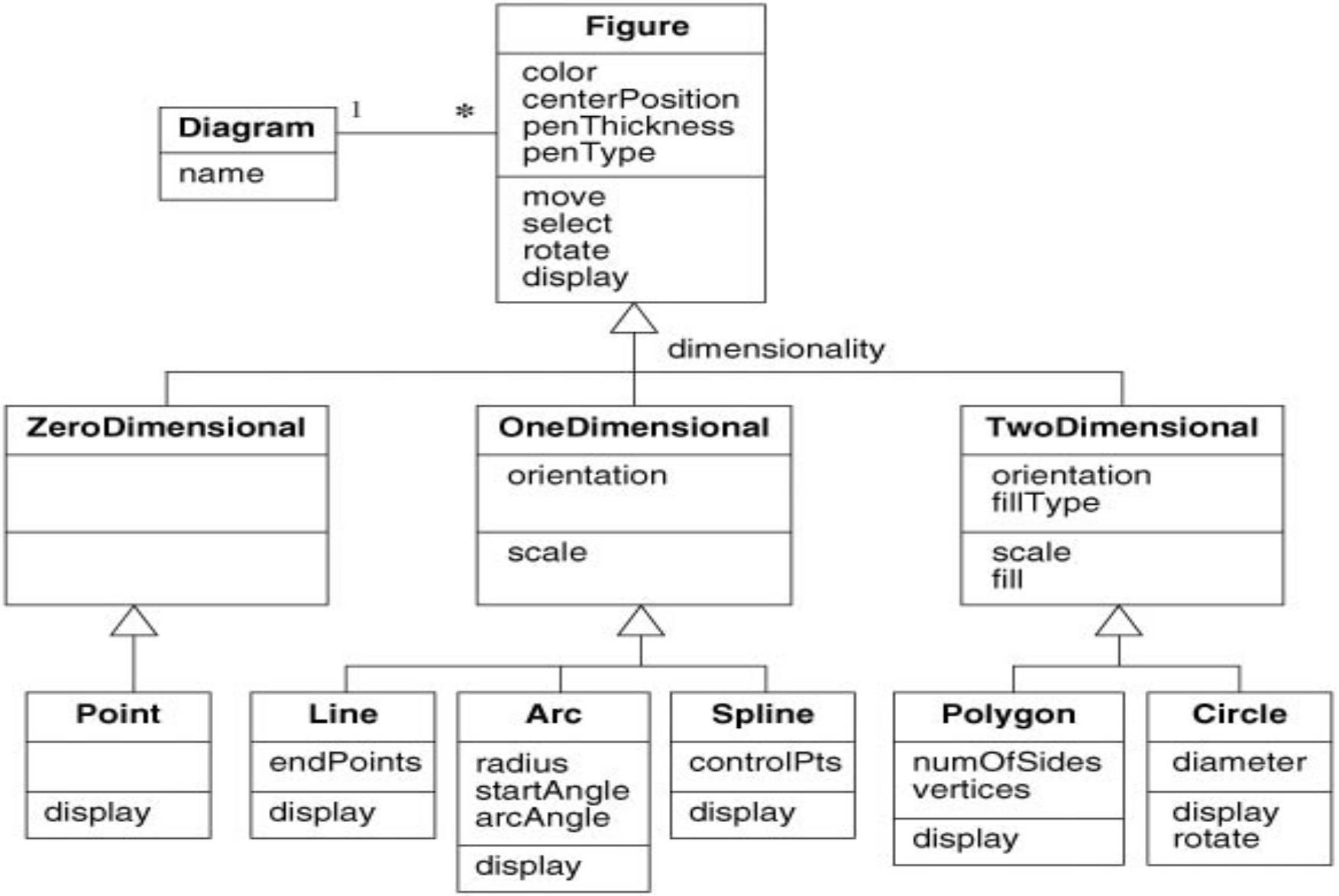

Figure 3.25 Inheritance for graphic figures. Each subclass inherits the attributes, operations, and associations of its superclasses.

Use of generalization

- **Used for three purposes:**
  - Support of polymorphism:
    - polymorphism increases the flexibility of software.
    - Adding a new subclass and automatically inheriting superclass behavior.
  - Structuring the description of objects:
    - Forming a taxonomy (classification), organizing objects according to their similarities. It is much more profound than modeling each class individually and in isolation of other similar classes.
  - Enabling code reuse:
    - Reuse is more productive than repeatedly writing code from scratch.

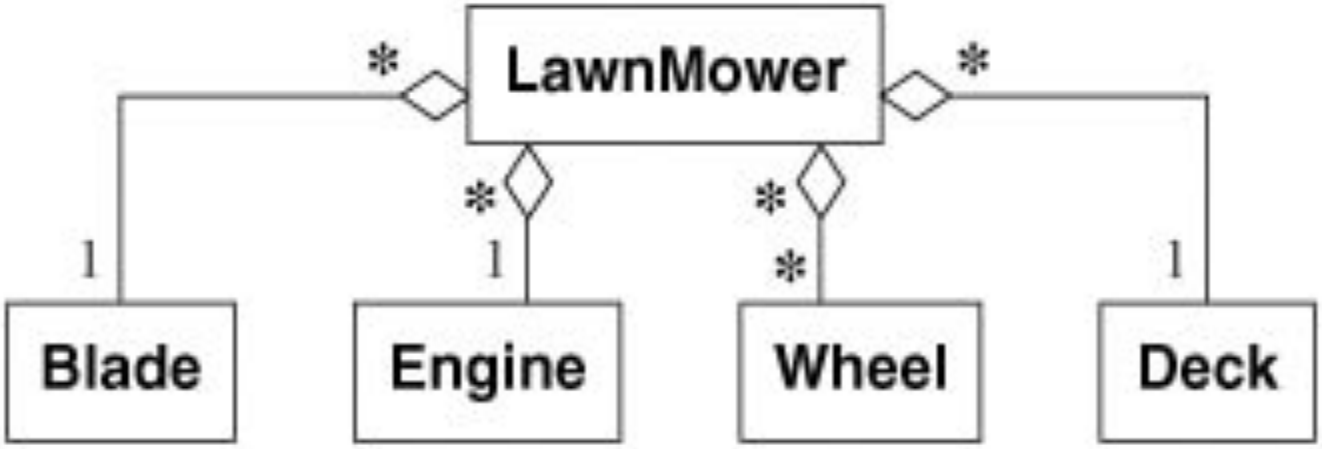

Figure 4.9 Aggregation. Aggregation is a kind of association in which an aggregate object is made of constituent parts.

Object-Oriented Modeling and Design with UML, Second Edition
by Michael Blaha and James Rumbaugh.
ISBN 0-13-1-015920-4.