Software Engineering

Lesson #04 - Practice

Lesson #04 - Practice

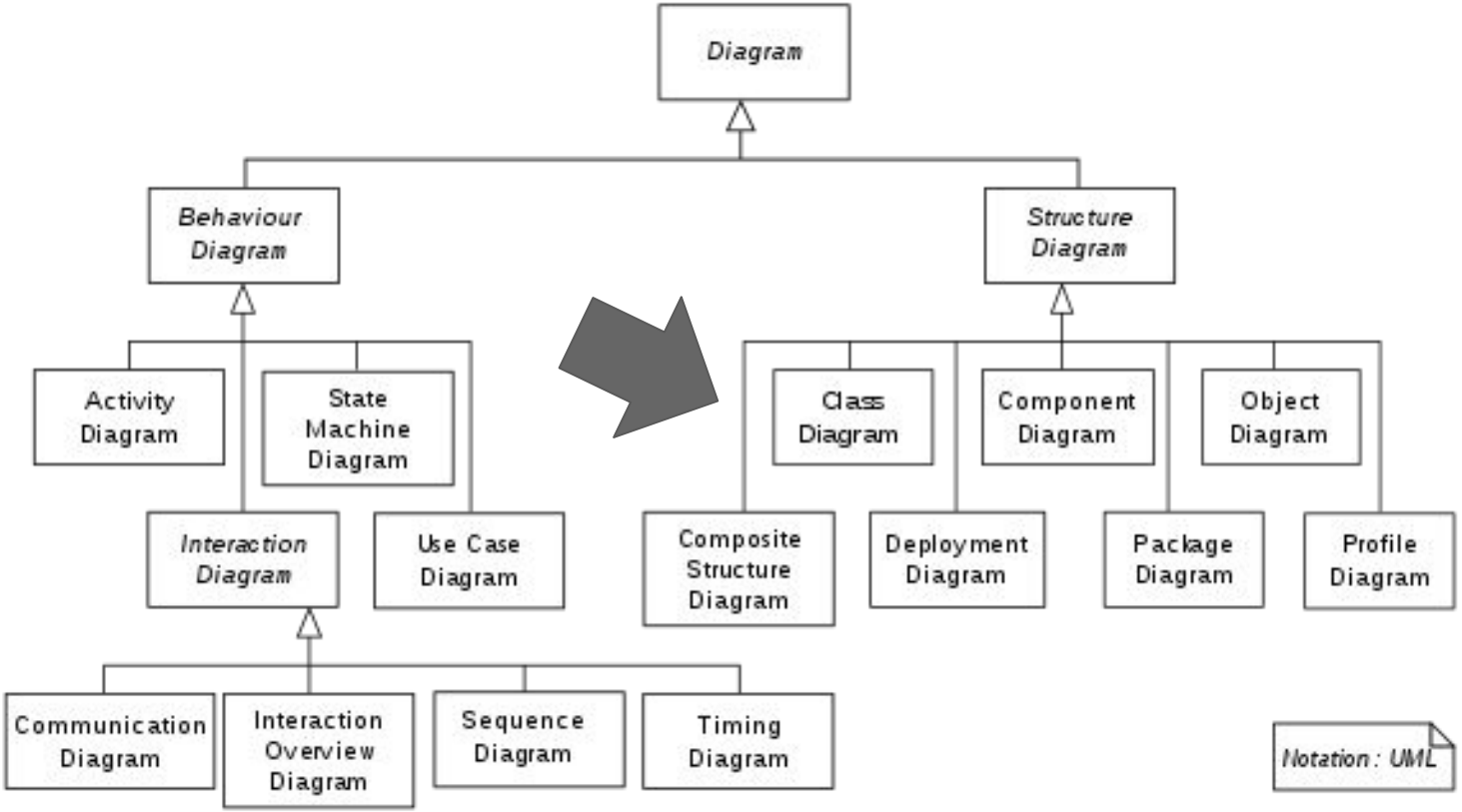
Agenda: Lesson #04 - Software Engineering - Practice

1. Class Diagrams: Advanced Concepts
2. Class Work
3. Q & A

Agenda: Lesson #04 - Software Engineering - Practice

1. **Class Diagrams: Advanced Concepts**
2. Class Work
3. Q & A

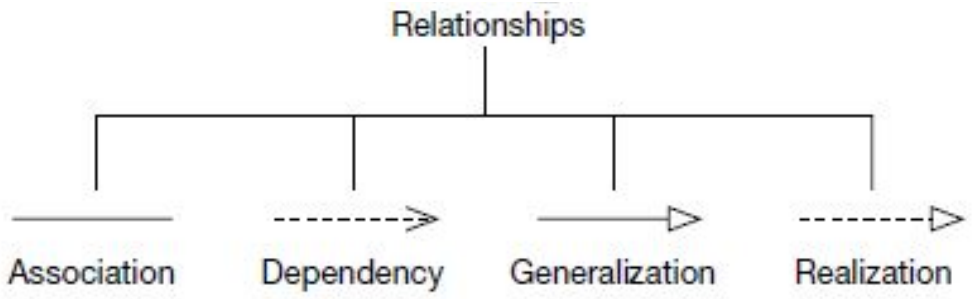
Class Diagrams



Classification and Generalization

The UML uses the generalization symbol to show generalization

If we need to show classification, use a dependency with the «instantiate» keyword



Multiple and Dynamic Classification

Classification refers to the relationship between an object and its type

Mainstream programming languages assume that an object belongs to a single class

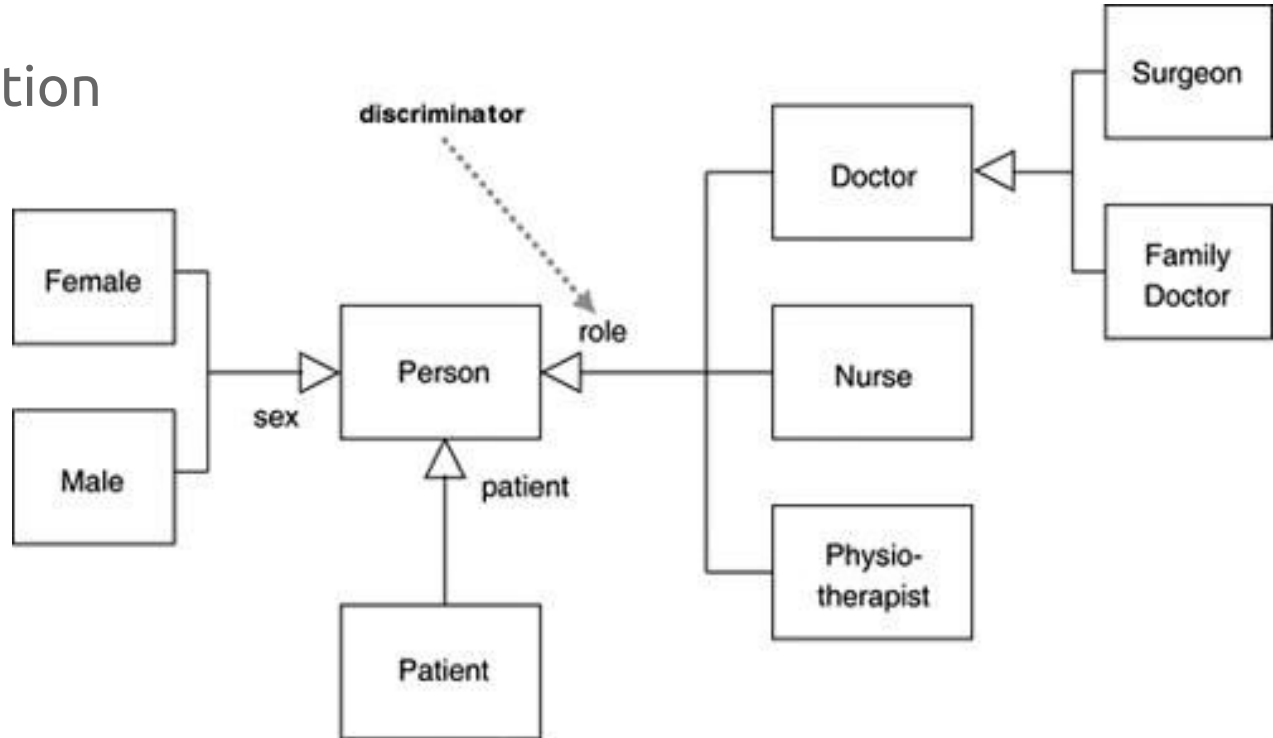
Multiple and Dynamic Classification

In single classification, an object belongs to a single type, which may inherit from supertypes

In multiple classification, an object may be described by several types that are not necessarily connected by inheritance

Multiple and Dynamic Classification

In multiple classification



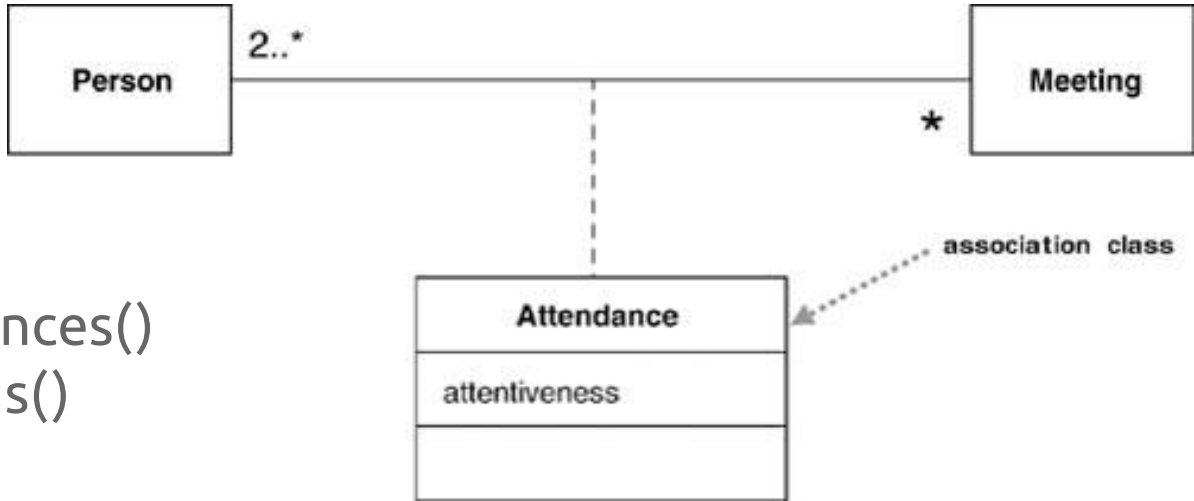
Multiple and Dynamic Classification

Dynamic classification allows objects to change class within the subtyping structure; static classification does not

With static classification, a separation is made between types and states; dynamic classification combines these notions

Association Class

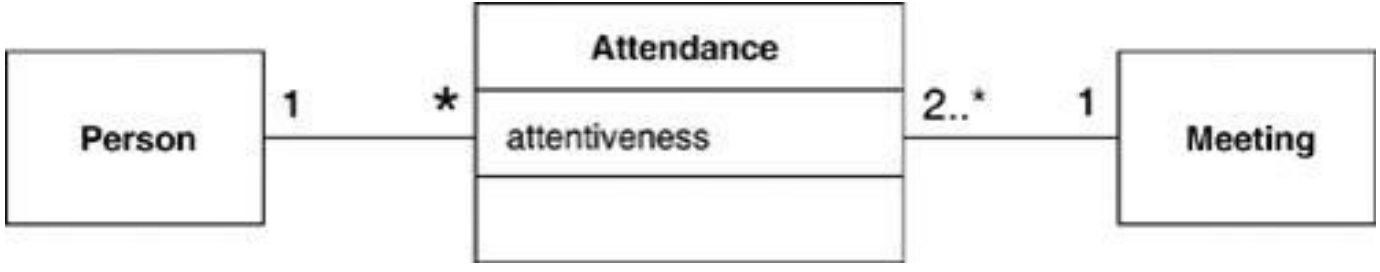
Association classes allow you to add attributes, operations, and other features to associations



class Person
    List getAttendances()
    List getMeetings()

Association Class

Promoting an association class to a full class

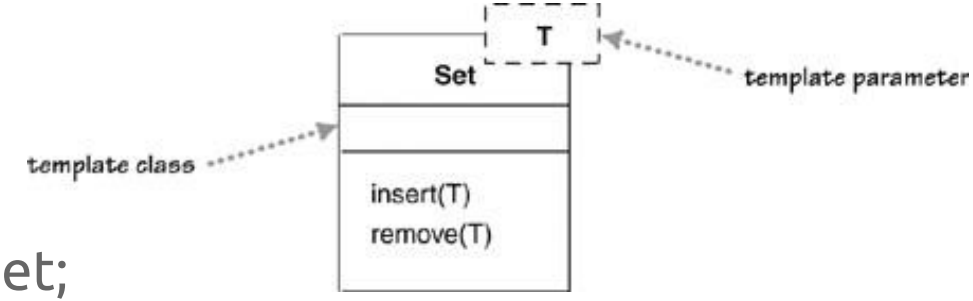


Template (Parameterized) Class

Several languages, most noticeably C++, have the notion of a parameterized class, or template

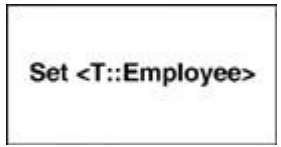
Templates are on the list to be included in Java and C# in the near future

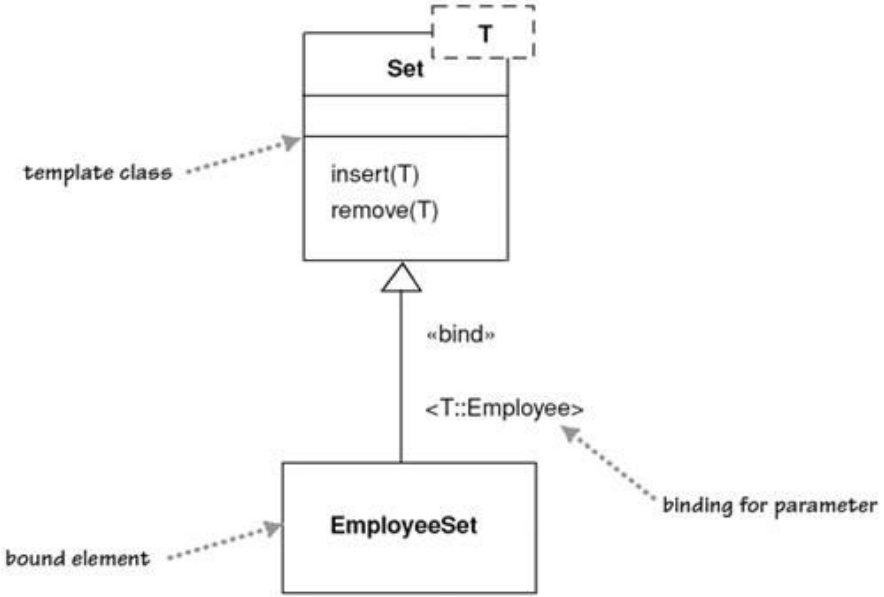
```
class Set <T> {
void insert (T newElement);
void remove (T anElement);

Set <Employee> employeeSet;
```



Template (Parameterized) Class

A use of a parameterized class, such as Set<Employee>, is called a derivation

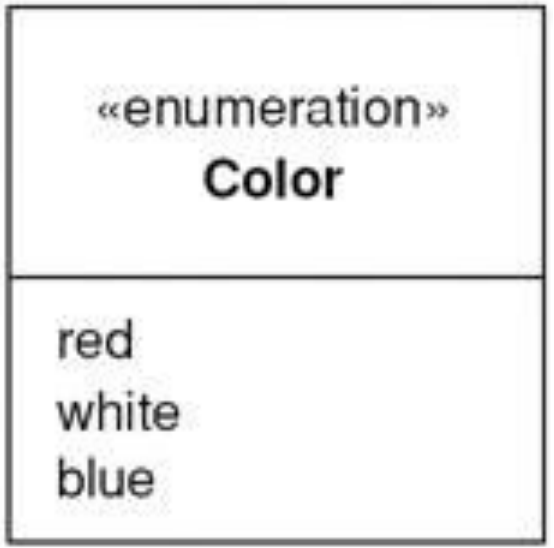




Enumerations

Enumerations are used to show a fixed set of values that don't have any properties other than their symbolic value

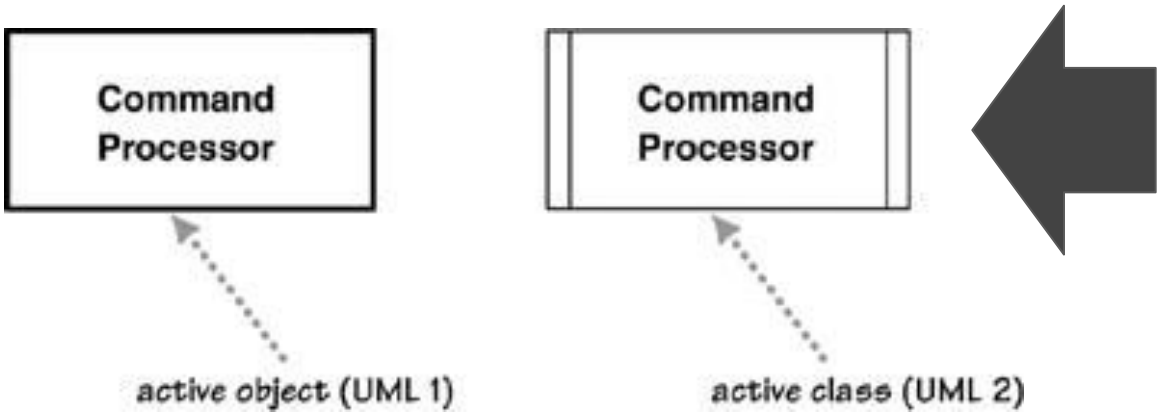
They are shown as the class with the «enumeration» keyword



Active Class

An active class has instances, each of which executes and controls its own thread of control

Method invocations may execute in a client's thread or in the active object's thread



Visibility

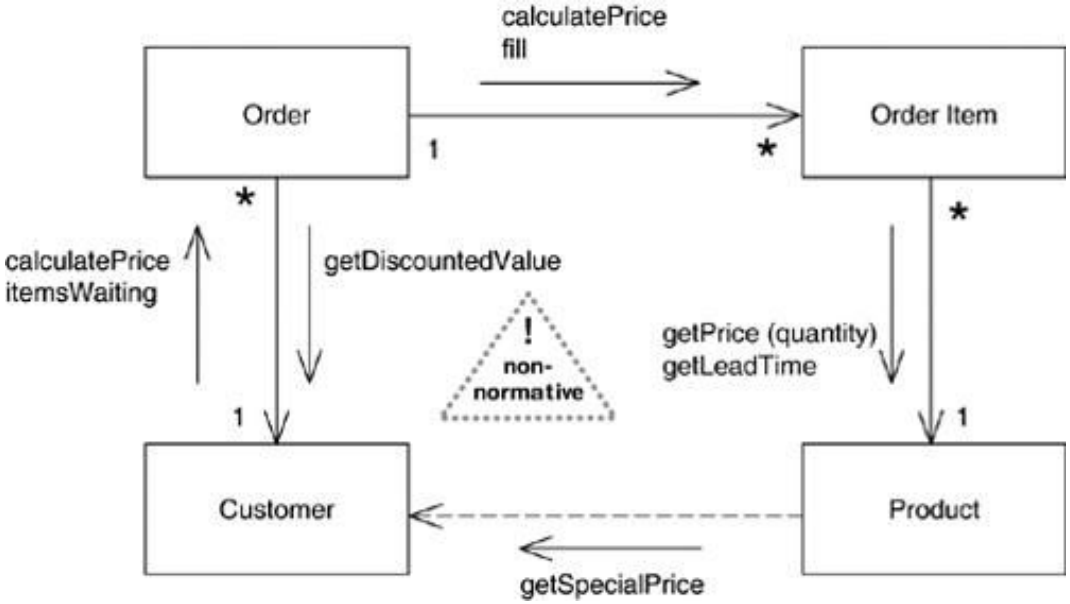
Visibility is a subject that is simple in principle but has complex subtleties

The simple idea is that any class has public and private elements

Public elements can be used by any other class; private elements can be used only by the owning class

Messages

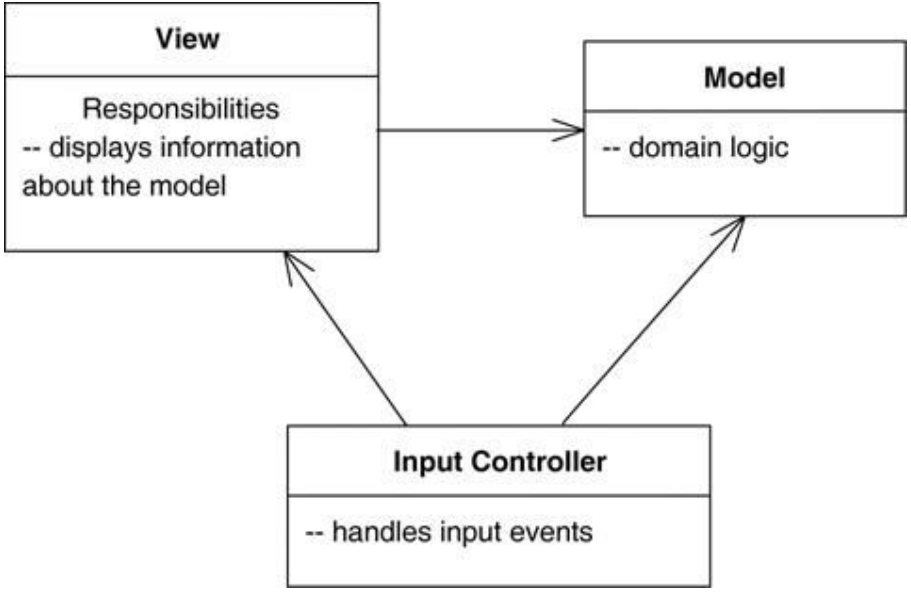
Standard UML does not show any information about message calls on class diagrams



Responsibilities

Often, it's handy to show responsibilities on a class in a class diagram

The best way to show them is as comment strings in their own compartment in the class

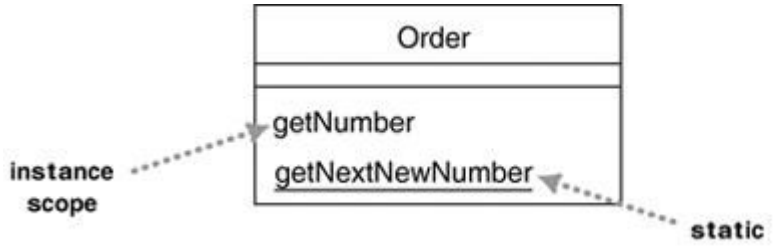


Static Operations and Attributes

The UML refers to an operation or an attribute that applies to a class rather than to an instance as static

This is equivalent to static members in C - based languages

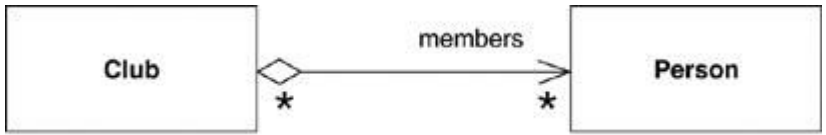
Static features are underlined on a class diagram

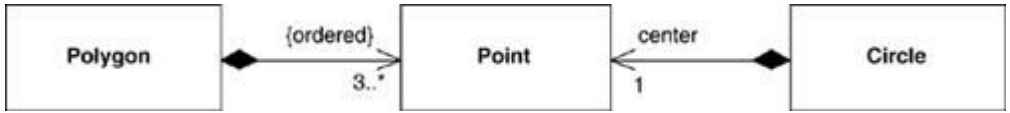


Aggregation and Composition

Aggregation is the part-of relationship

As well as aggregation, the UML has the more defined property of composition

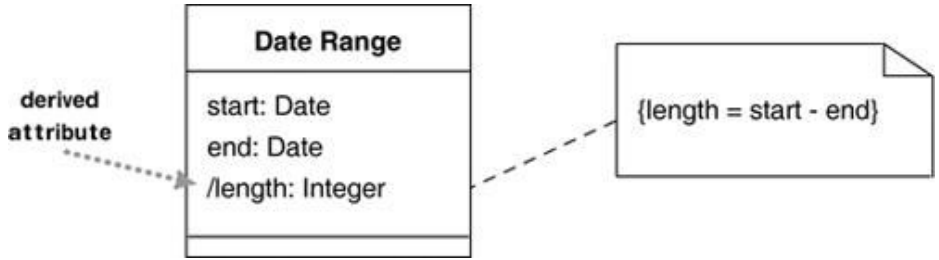




Derived Properties

Derived properties can be calculated based on other values. When we think about a date range, we can think of three properties: the start date, the end date, and the number of days in the period

These values are linked, so we can think of the length as being derived from the other two values

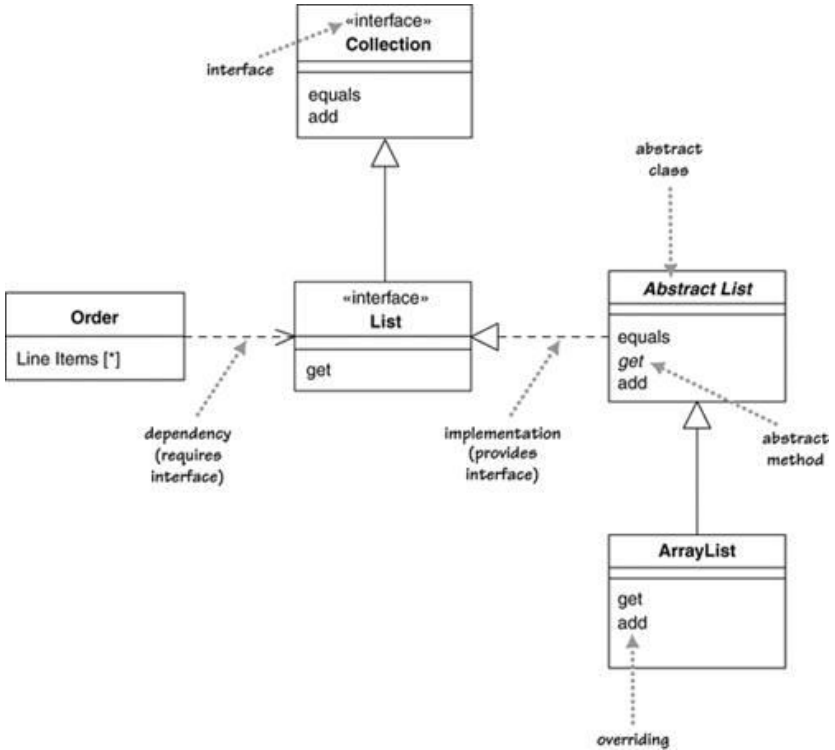


Interfaces and Abstract Classes

An abstract class is a class that cannot be directly instantiated.
Instead, you instantiate an instance of a subclass
Typically, an abstract class has one or more operations that are abstract
Classes have two kinds of relationships with interfaces: providing and requiring
A class provides an interface if it is substitutable for the interface

Interfaces and Abstract Classes

A Java example of interfaces and an abstract class



Reference Objects and Value Objects

Reference objects are such things as Customer. Here, identity is very important because you usually want only one software object to designate a customer in the real world

Value objects are such things as Date. You often have multiple value objects representing the same object in the real world

Qualified Associations

A qualified association is the UML equivalent of a programming concept variously known as associative arrays, maps, hashes, and dictionaries. Diagram shows a way that uses a qualifier to represent the association between the Order and Order Line classes

Agenda: Lesson #04 - Software Engineering - Practice

Q & A