Chapter 3: Part-B
Classes, Objects, and Basic Structural Modeling in UML

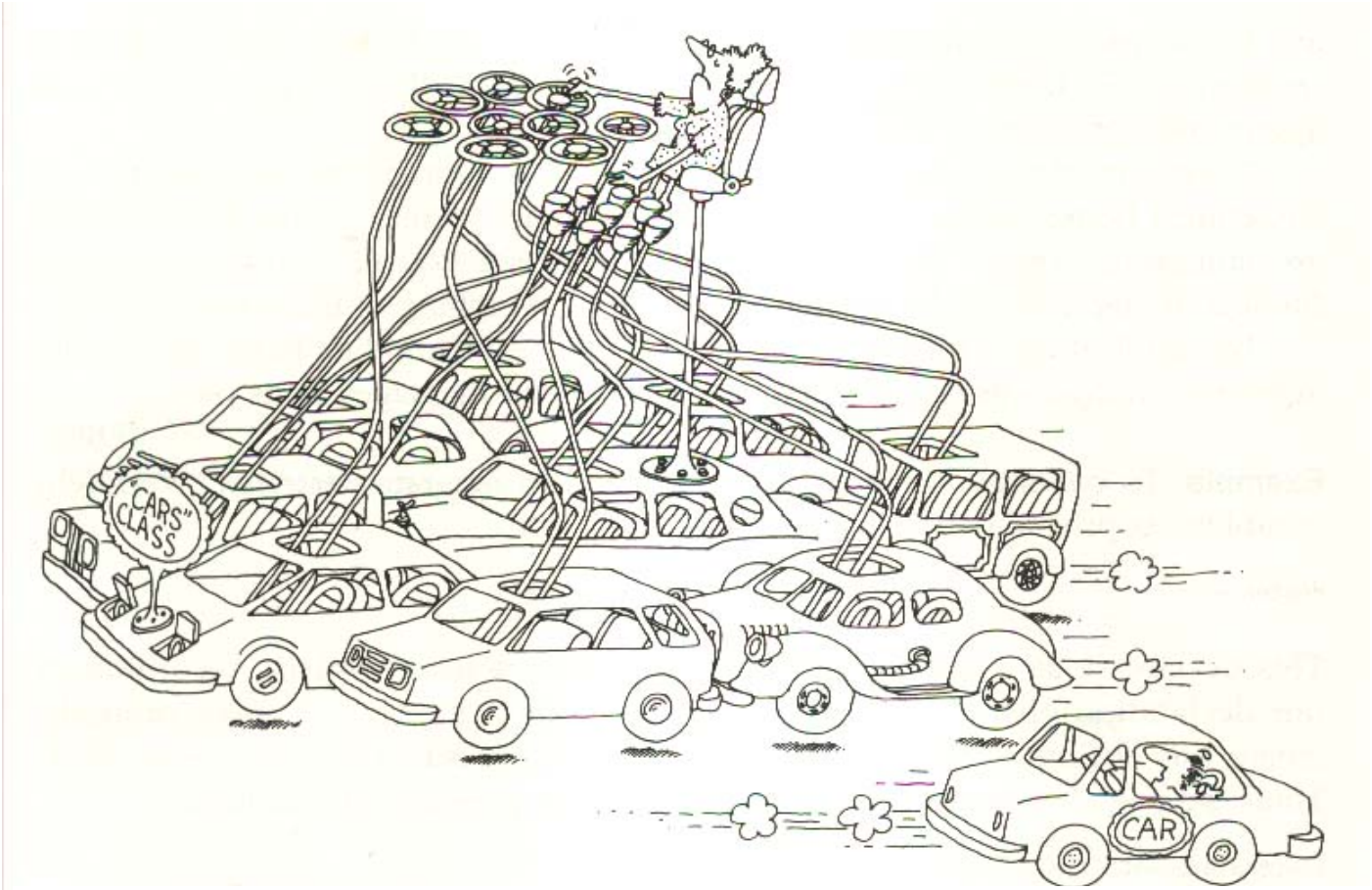
H.S. Sarjoughian

CSE 460: Software Analysis and Design

School of Computing, Informatics and Decision Systems Engineering
Fulton Schools of Engineering

Arizona State University, Tempe, AZ, USA

Class Caricature



a class represents a set of objects that share a common structure and a common behavior

What is a Class (cont.)

- *Objects* which share a **common structure** and a **common behavior** maybe grouped into a *class*

- A class serves as a contract between an abstraction and all of its clients
  - Interface
  - Implementation

- Classes represent abstractions of objects which exist in time and space – classes and objects are closely linked to one another

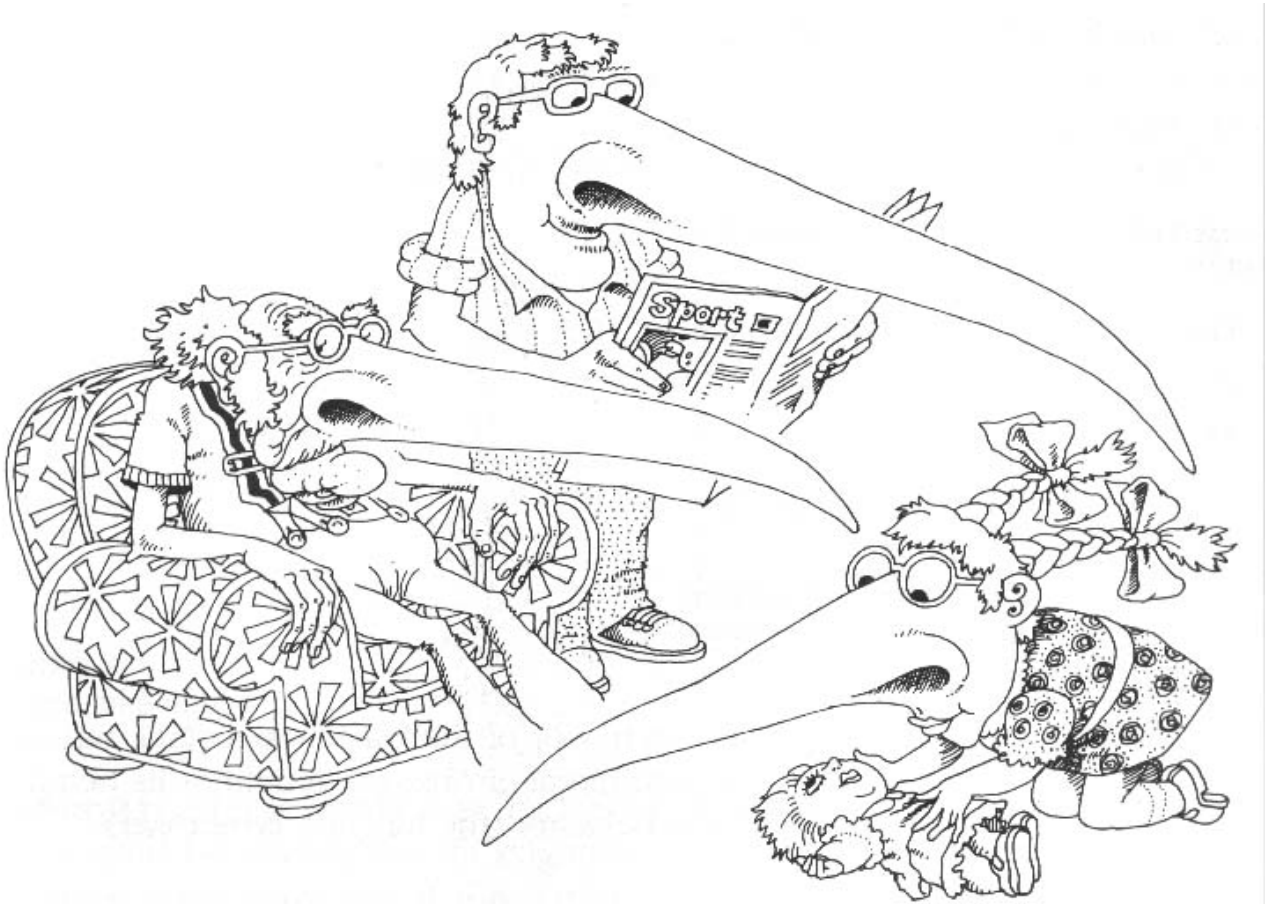
Classes and Objects Relationships

- A relationship between two objects states the assumptions each makes about the other. Most important are the operations and their expected results.

- Some common ways to combine or relate abstractions
  - Association
  - Dependency
  - Generalization/specialization
  - Aggregation

Inheritance Caricature

G. Booch, OOAD



a subclass may inherit the structure and behavior of its superclass

Inheritance

- Inheritance: a relationship among classes where one class shares the structure and behavior of one or more classes. Types of inheritance relationships are
  - Single inheritance
  - Multiple inheritance

multiple inheritance
should be used with care

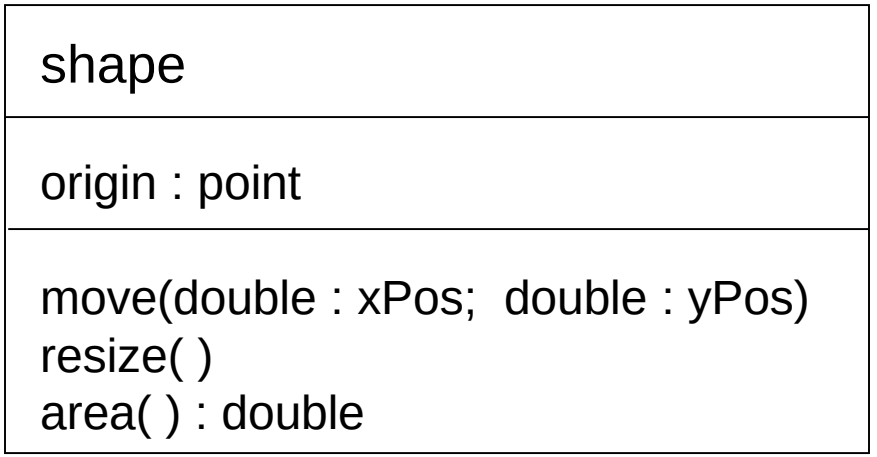
- Inheritance defines an "is-a" hierarchy among classes in which a subclass inherits from generalized superclasses.
  - Multi-level inheritance (superclass, class, subclass)

- A subclass generally specializes its class and superclasses structure and behavior – the specialization can be achieved through augmentation and/or partial redefining of class and superclasses.

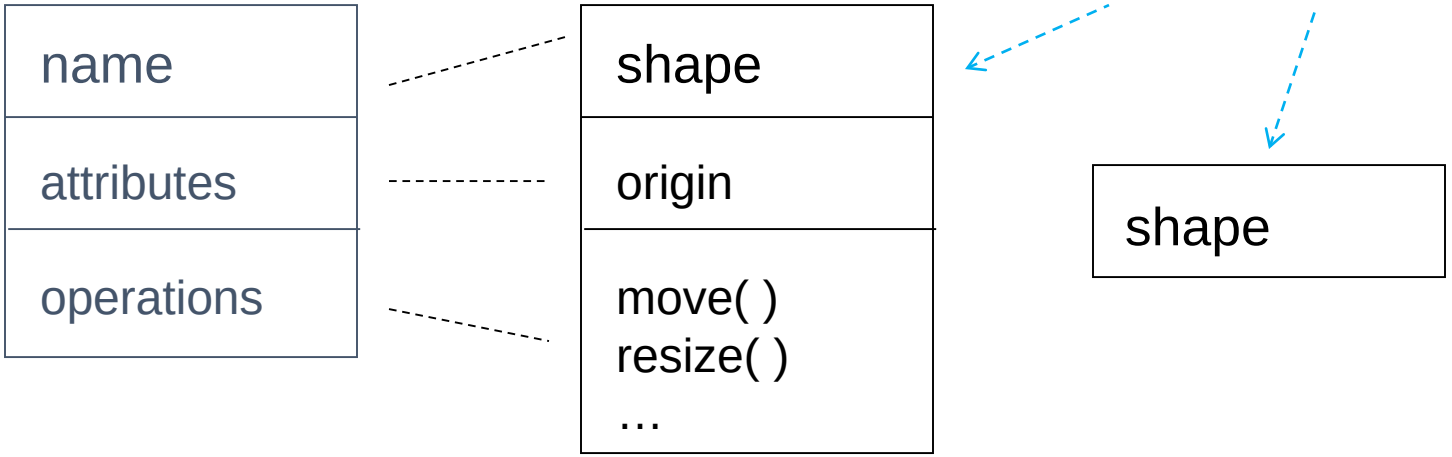
Classes

semantics:

- name
- attributes
- operations



notation:



Classes (cont.)

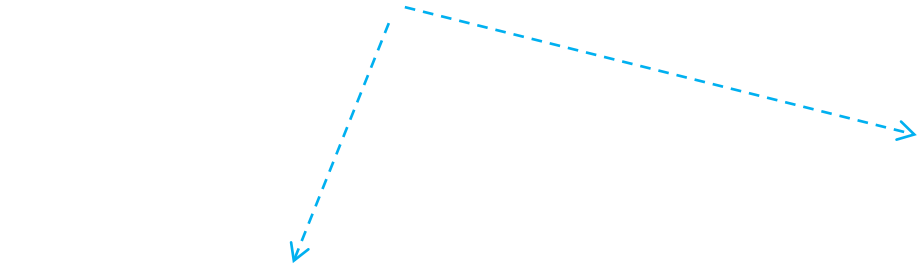
name

simple names

java::awt::Rectangle

path names

- non-software things



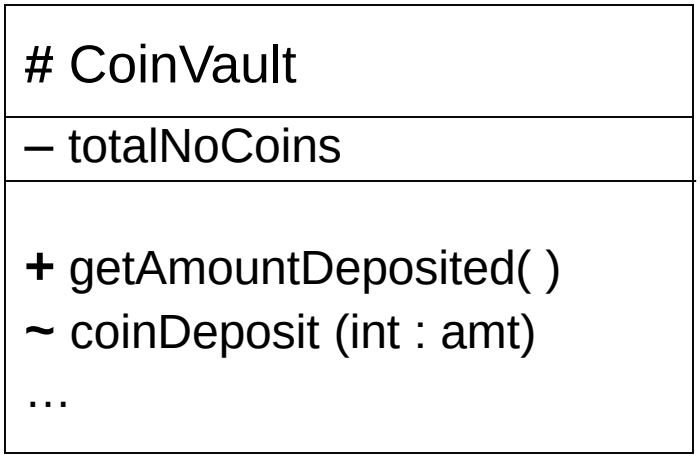
Accounts Receivable Person

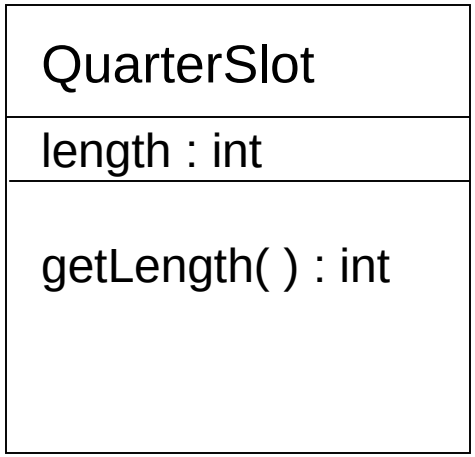
Robot

processOrder( )
status( )
…

Class Example

- Two classes for a virtual vending machine





+ public
# protected
~ package
– private

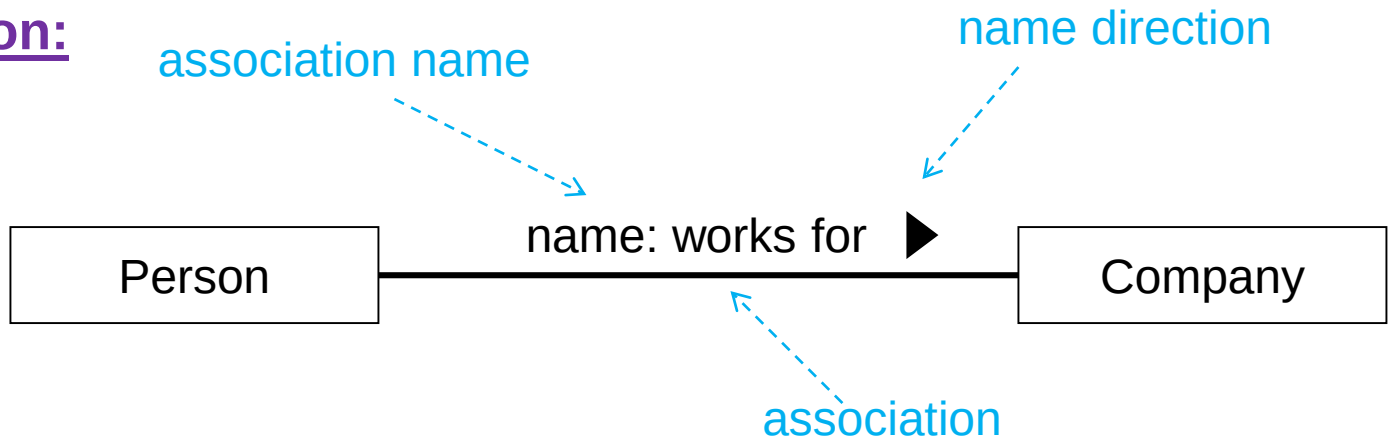
UML notation for visibility

Association Relationship

semantics: objects of one thing are connected are connected to objects of another. Four adornments are possible

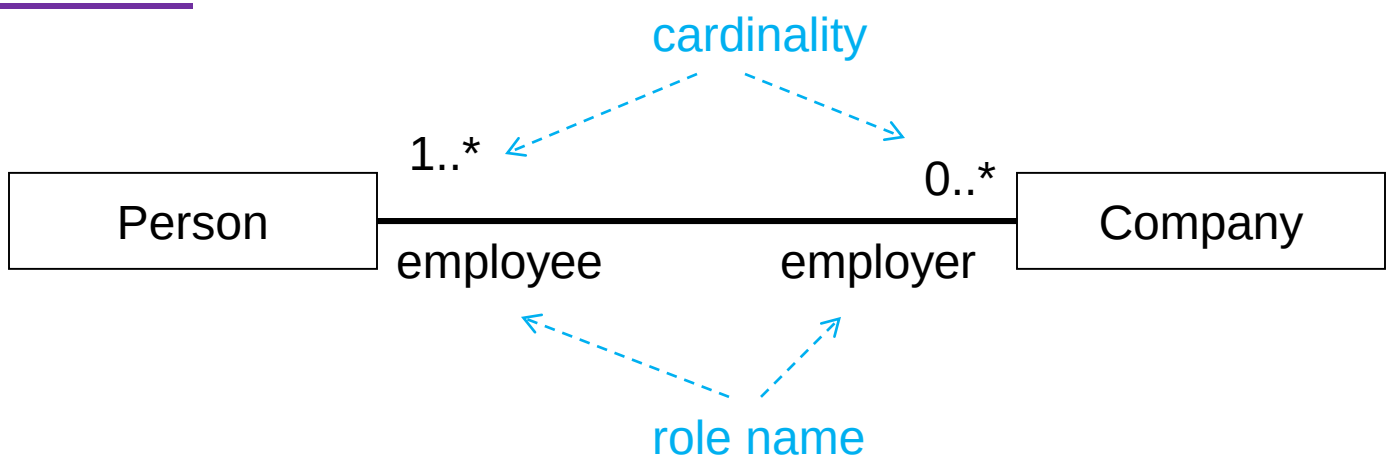
- name
- role
- multiplicity: multiplicity at one of an association means that for each object of the class at the opposite end, there must exist that many objects at the near end. Choices are 1, 0..1; 0..* (*); 1..*; 1..9; 1, 4, 9; 1, 3, 5..*, an exact number.
- direction

notation:



Association Relationship (cont.)

notation:



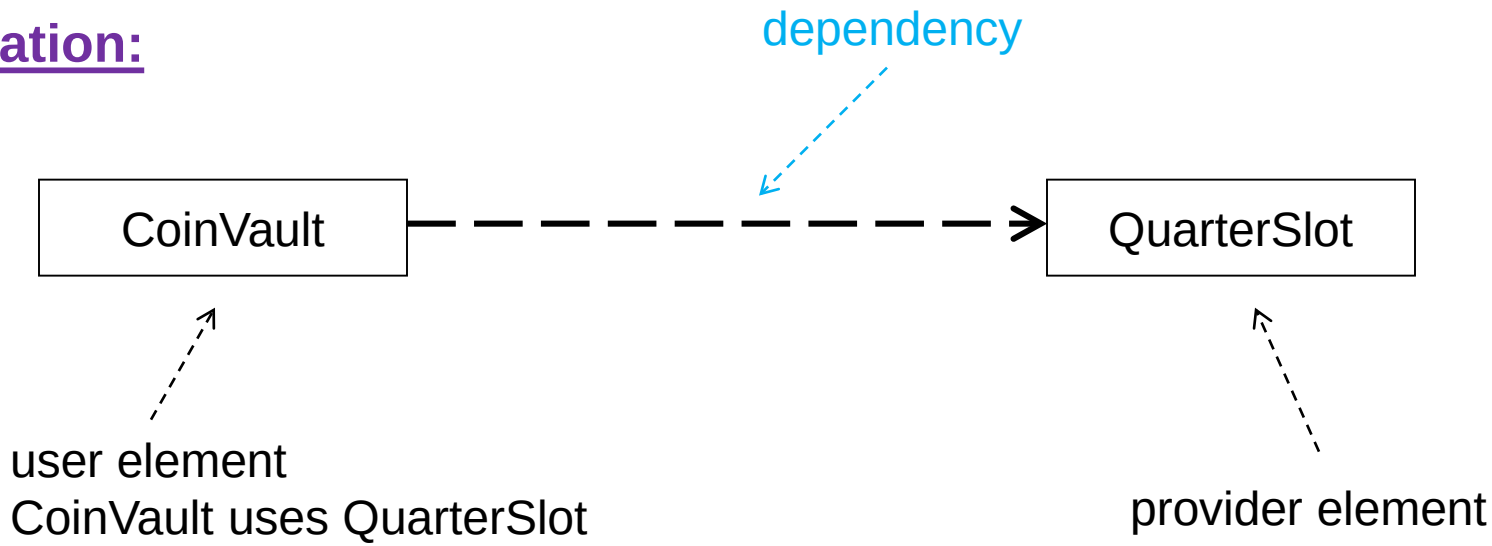
semantics: a class participating in an association has a role to play

- A company (playing the role of an employer) can have one or more persons (playing the role of employee)
- A person P (employee) can belong to zero or more companies C1, … Cn (employers)

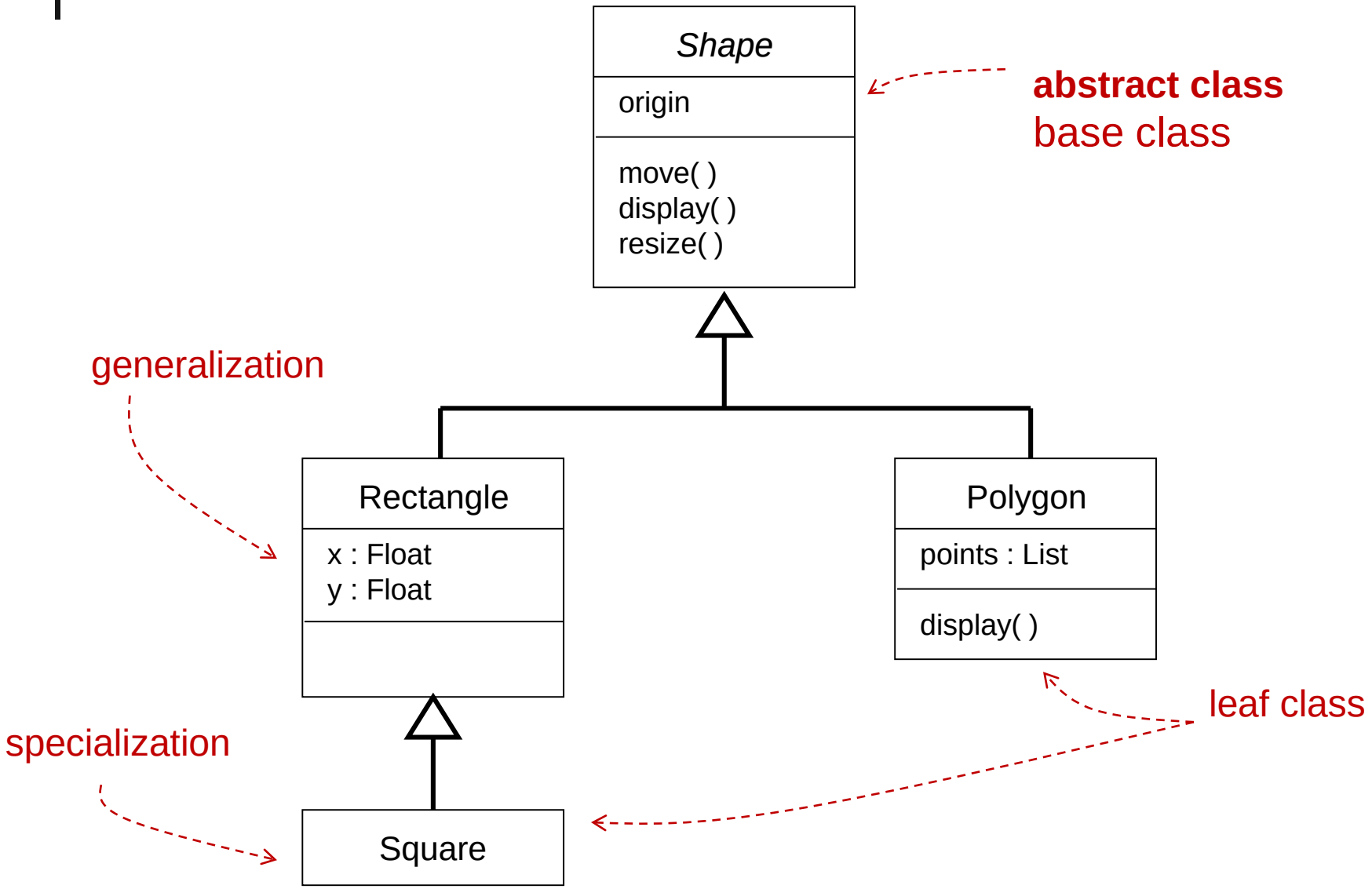
Dependency Relationship

semantics: indicates that a change to the provider element may require a change to the user element. Usually, dependency relationship is used to show one class (user) uses another class (provider) as an argument in the signature of one of its operations. There can be one adornment (name) but generally not used.

notation:

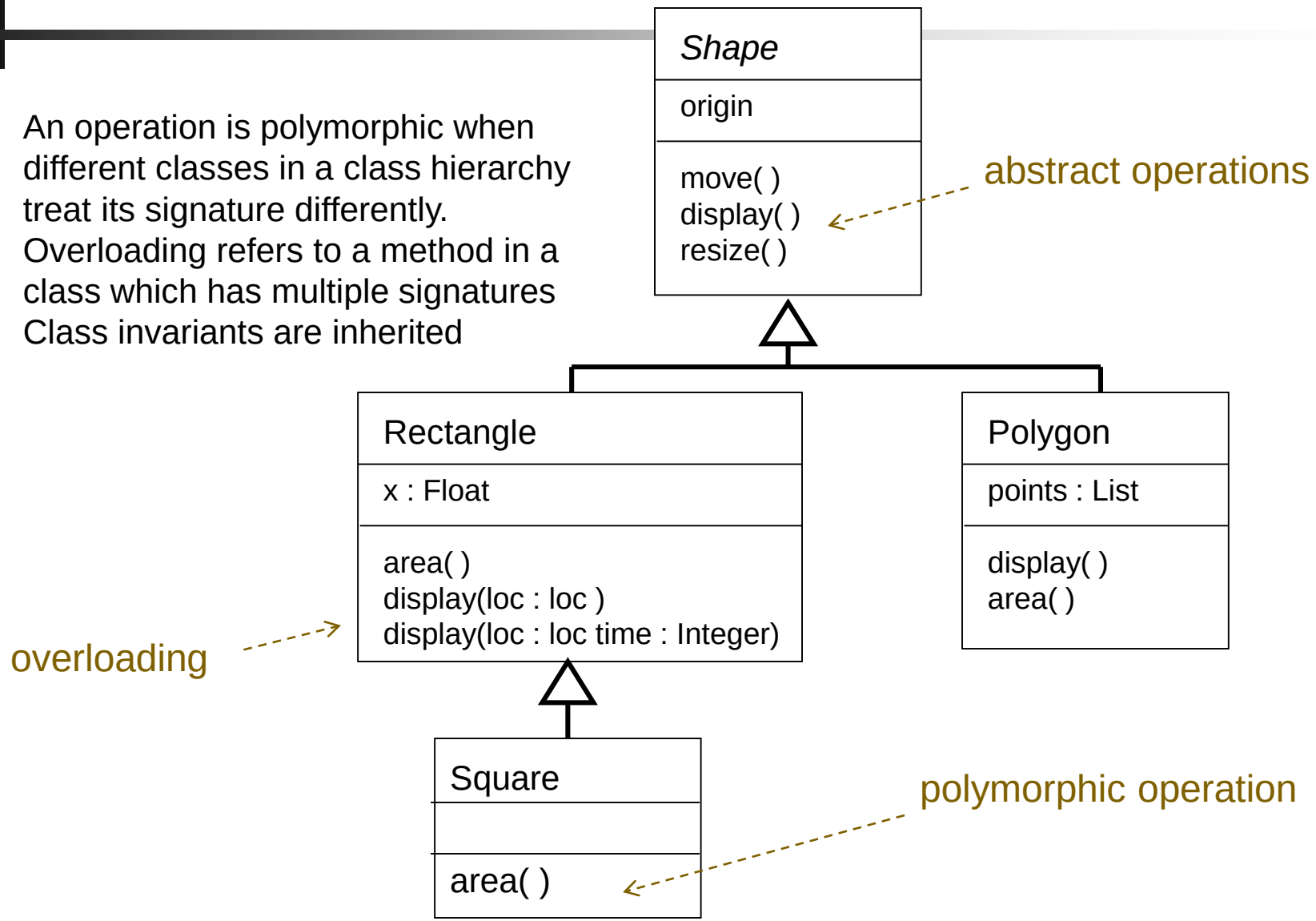


Class (Single Inheritance) Relationship



Polymorphism and Overloading

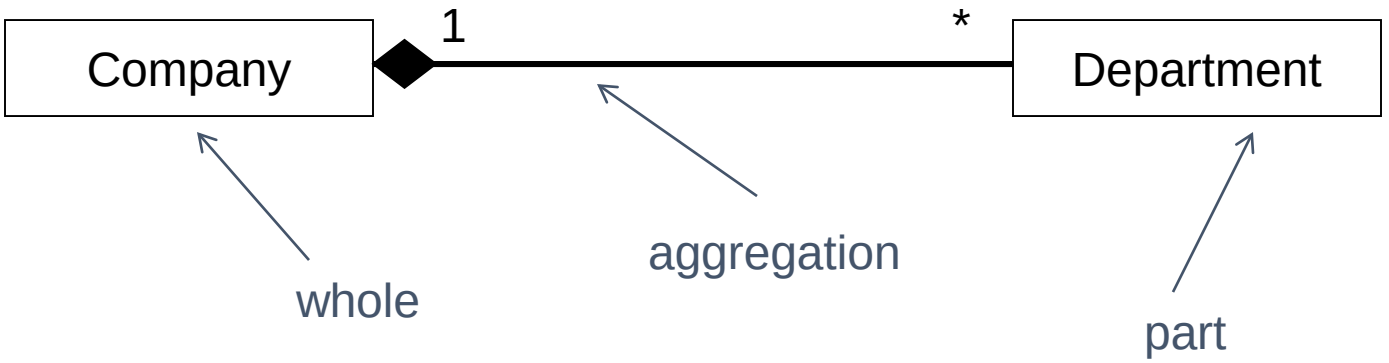
- An operation is polymorphic when different classes in a class hierarchy treat its signature differently.
- Overloading refers to a method in a class which has multiple signatures
- Class invariants are inherited



Aggregation Relationship

semantics: it distinguishes whole from part. Aggregation is a "has-a" relationship – one object of the whole has objects of the part.

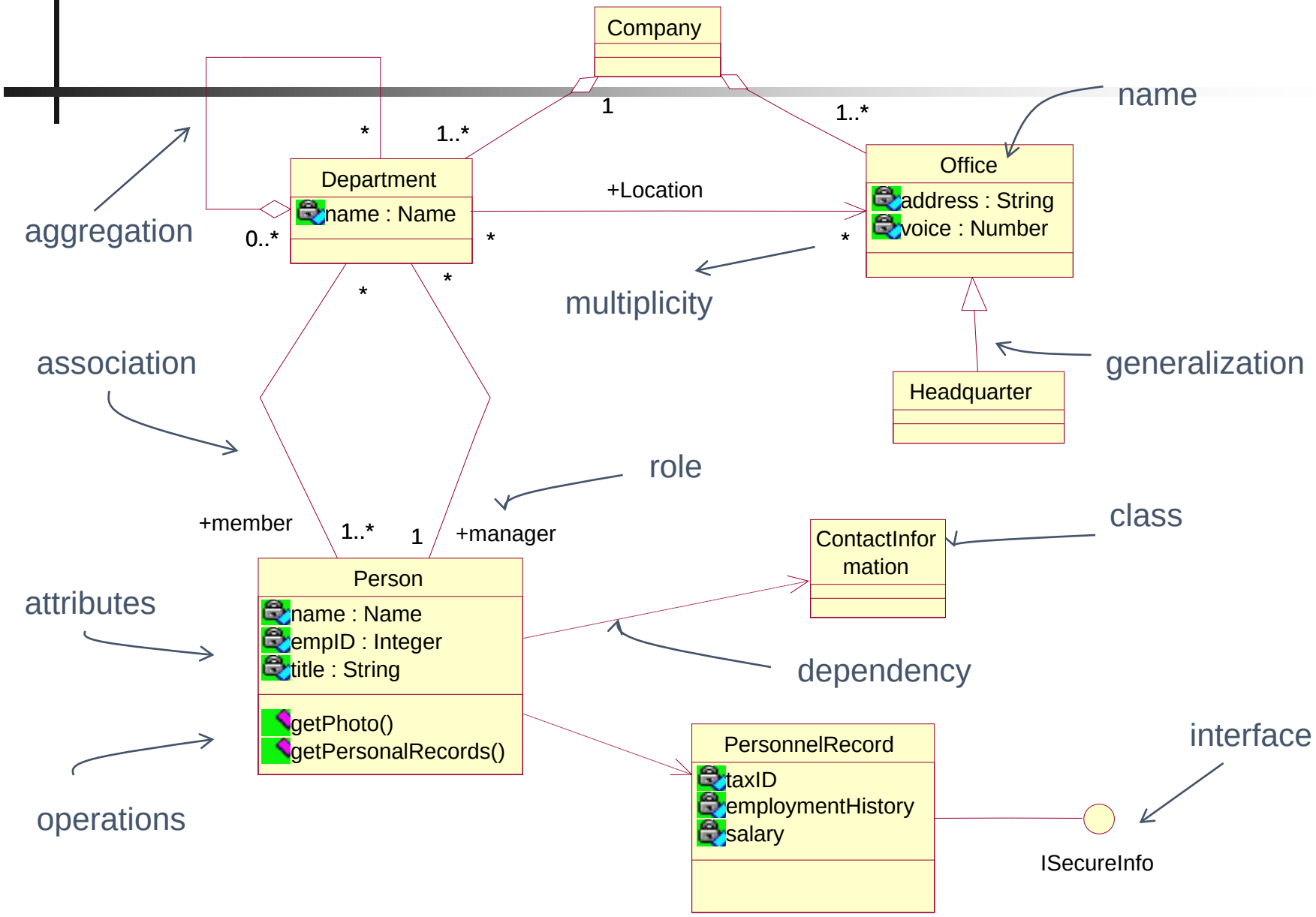
notation:



Class Diagram

- A graph of class elements connected by their various static relationships is called a static class diagram
- A class diagram is generally used to capture static view of a system – modeling vocabulary and collaborations
- Class diagrams are declarative – they are important for visualizing, specifying, documenting, and constructing executable systems through forward engineering
- There may exist more than one class diagram – a class diagram captures a part of the things and relationships that make up an analysis or a design view
- A class diagram may be decomposed into sub-class diagrams
- Class diagrams are required for creating component and deployment diagrams

Class Diagram Example



Class Diagram (cont.)

- Class diagram generally contains:
  - Classes
  - Abstract classes
  - Interfaces
  - Collaborations
  - Packages and subsystems
  - Association, dependency, generalization, and aggregation relationships
  - Notes

An abstract class is like an ordinary class with the exception that it can not be instantiated

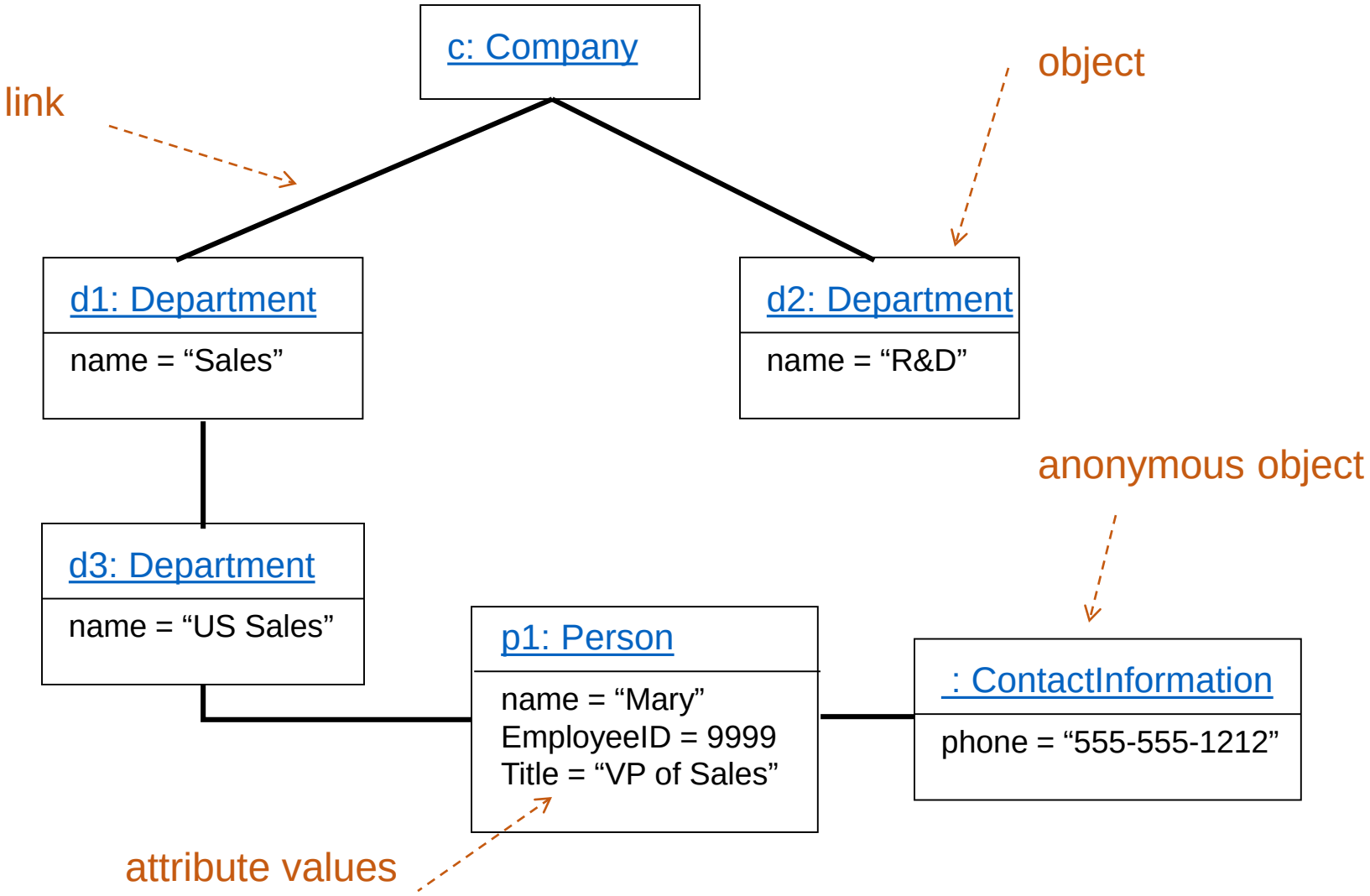
Object Diagram

- An object diagram is a graph – collection of vertices (**objects**) and arcs (**links**)

- Object diagram models a set of instances of classes and their relationships at an instance of time – it is an instance of a class diagram; there can exist multiple object diagrams for a class diagram

- Object diagram represent **static view of a system** – it includes a set of objects, their states, and their relationships

Object Diagram Example



Summary

- Objects and classes are most elementary (important) artifacts of object-oriented software engineering – the choice of these entities is often non-trivial
- Specifying state, behavior, and identity of each object is essential for defining compound objects and classes via relationships
- Class diagrams and object diagrams capture complementary static views of a system's "aggregate" structure and behavior
- Unified Modeling Language provides a standard syntax and semantic suitable for capturing most key aspects of a software intensive system

References

- *Object-Oriented Analysis and Design with Applications, 3nd Edition, G. Booch, et al. Benjamin Cummings, 2007*
- *OMG Unified Modeling Language Specification, UML Standard, http://www.omg.org/technology/documents/formal/uml.htm, Computer Associates International, Inc., 2001*
- *The Unified Modeling Language User Guide,G. Booch, J. Rumbaugh, I. Jacobson, Addison Wesley Object Technology Series, 1999*