15-413

Object Modeling

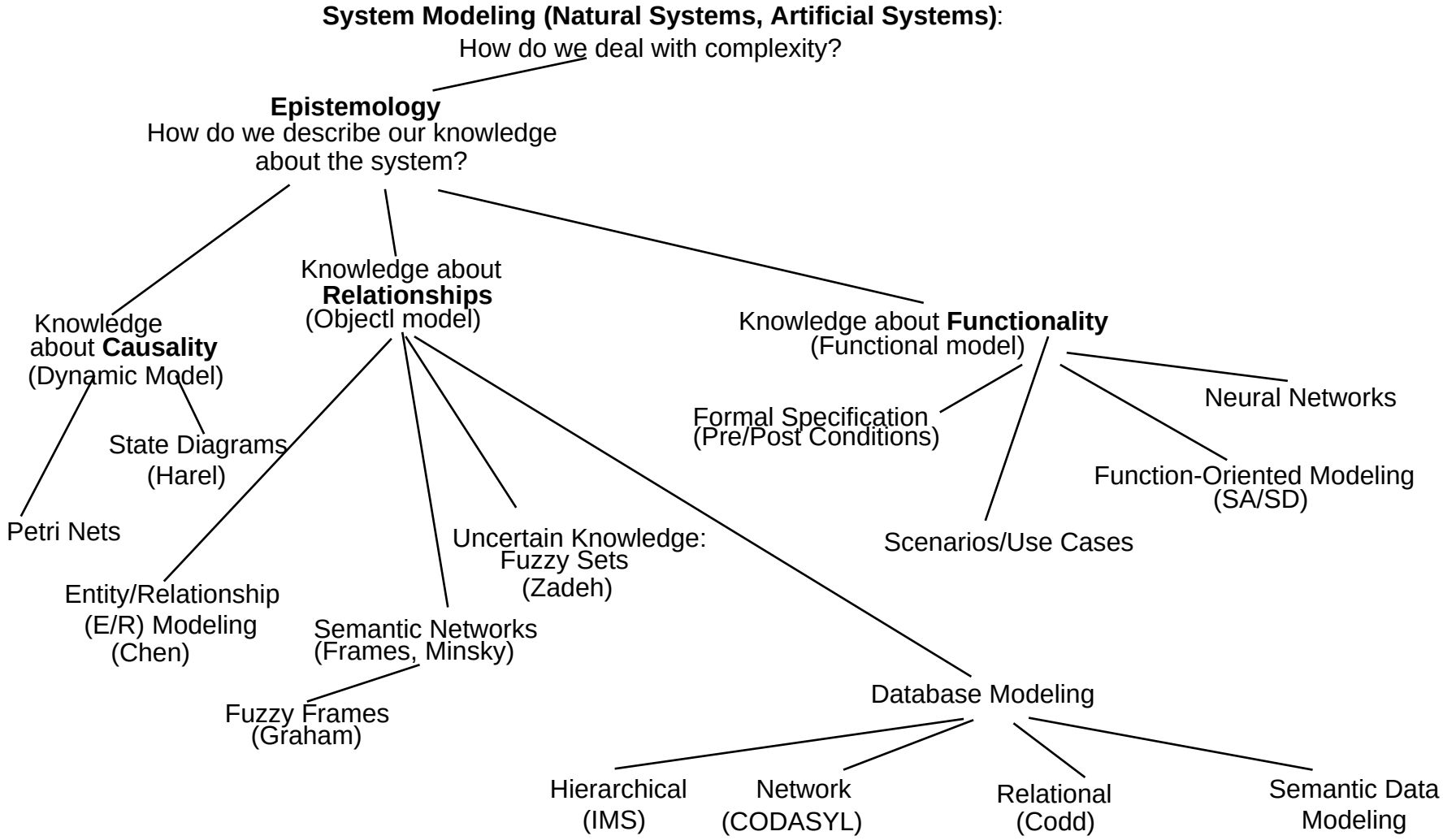
Bernd Bruegge
Carnegie Mellon University
School of Computer Science

10 September 1998

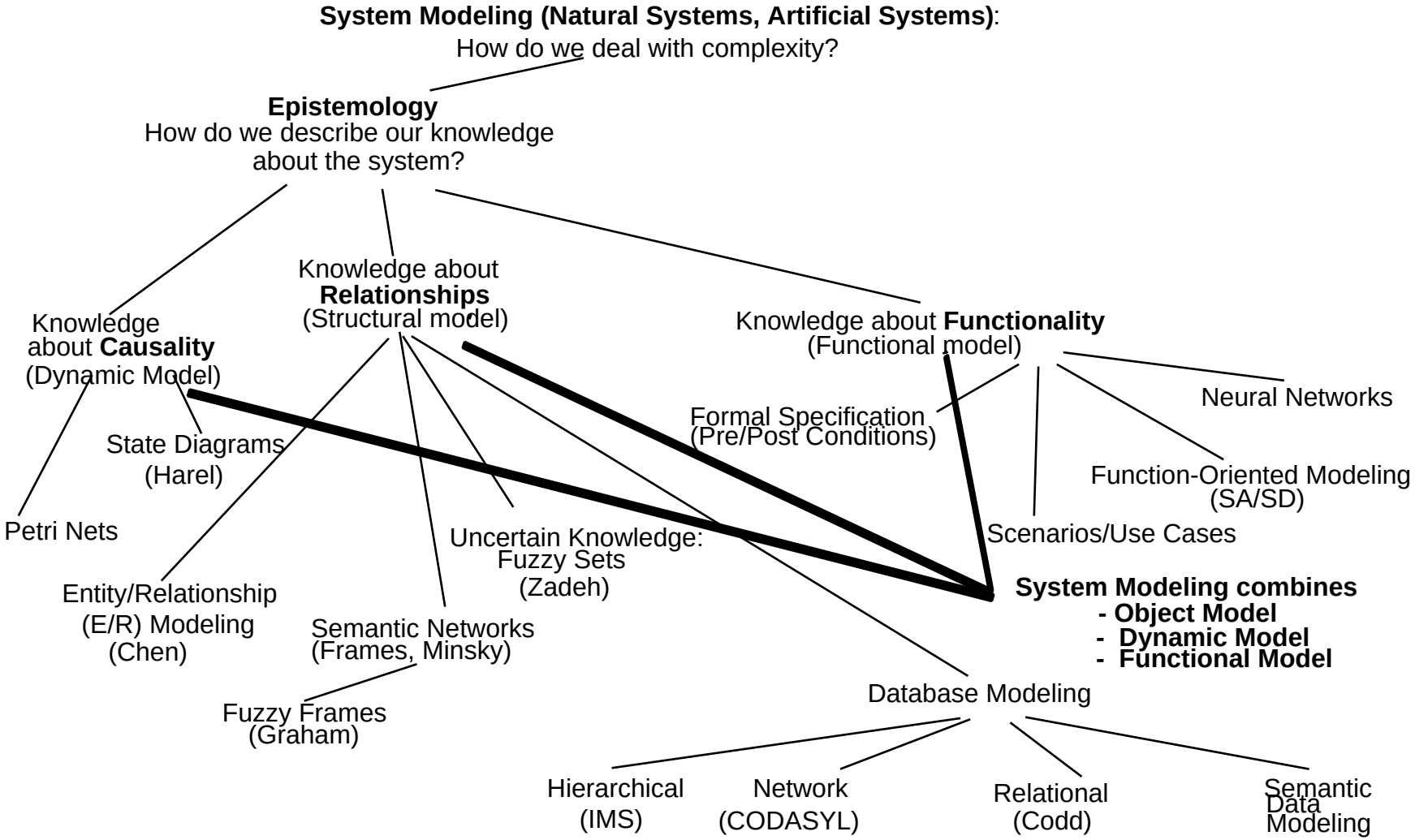
Outline

- From use cases to objects
- Object modeling
- Class vs instance diagrams
- Attributes
- Operations and methods
- Links and associations
- Examples of associations
- Two special associations
  - **Aggregation**
  - **Inheritance**

Software Modeling: Bird's View



Software Modeling



Definition: Object Modeling

- Main goal: Find the important abstractions
- What happens if we find the wrong abstractions?
  - **Iterate and correct the model**
- Steps during object modeling
  - **1. Class identification**
    - **Based on the fundamental assumption that we can find abstractions**
  - **2. Find the attributes**
  - **3. Find the methods**
  - **4. Find the associations between classes**
- Order of steps
  - **Goal: get the desired abstractions**
  - **Order of steps secondary, only a heuristic**
  - **Iteration is important**

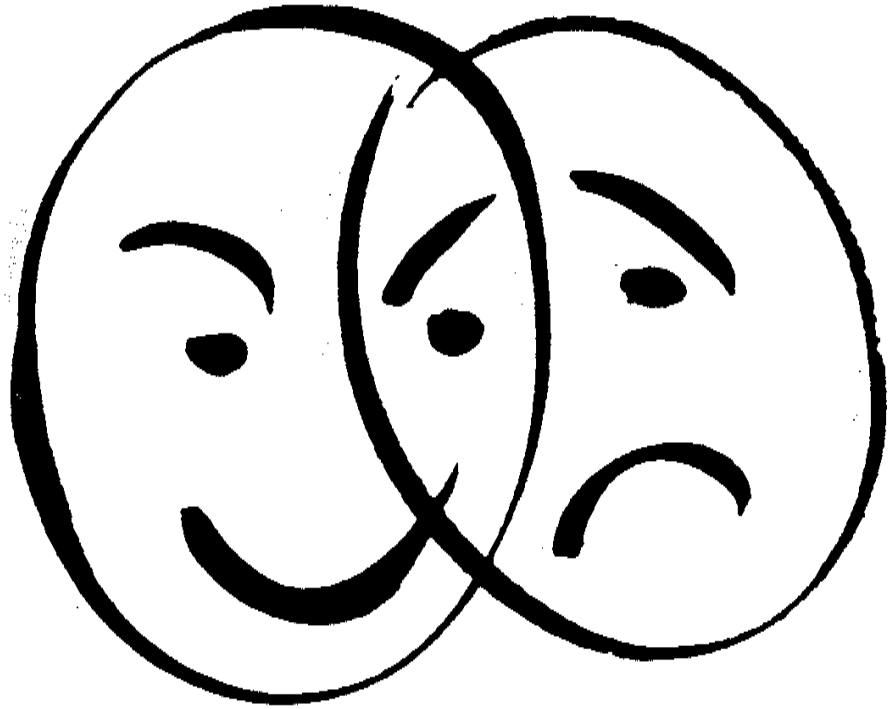
Class Identification

- Identify the boundaries of the system
- Identify the important entities in the system
- Class identification is crucial to object-oriented modeling
- Basic assumption:
  - **1. We can find the classes for a new software system (Forward Engineering)**
  - **2. We can identify the classes in an existing system (Reverse Engineering)**
- Why can we do this?
  - **Philosophy, science, experimental evidence**

Class identification is an ancient problem

- Objects are not just found by taking a picture of a scene or domain
- The application domain has to be analyzed.
- Depending on the purpose of the system different objects might be found
  - **How can we identify the purpose of a system?**
  - **Scenarios and use cases**
- Another important problem: Define system boundary.
  - **What object is inside, what object is outside?**

What is This?



Modeling in Action

- Face
- Mask
- Sad
- Happy
- Is it one Face or two?
- Who is using it?
  - **Person at Carneval?**
  - **Bankrobber?**
  - **Painting collector**
- How is it used?

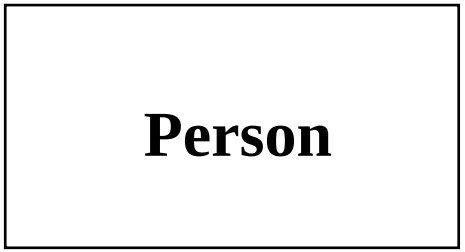
Pieces of an Object Model

- Classes
- Associations (Relations)
  - **Part of- Hierarchy (Aggregation)**
  - **Kind of-Hierarchy (Generalization)**
- Attributes
  - **Detection of attributes**
  - **Application specific**
  - **Attributes in one system can be classes in another system**
  - **Turning attributes to classes**
- Methods
  - **Detection of methods**
  - **Generic methods: General world knowledge, design patterns**
  - **Domain Methods: Dynamic model, Functional model**

Object vs Class

- Object (instance): Exactly one thing
  - **The lecture on Sep 10 on Software Engineering from 12:15 PM - 1:00 PM**
- A class describes a group of objects with similar properties
  - **Car, Fuelpump Serial Number, Dealer Behavior, Maintenance History**
- Object diagram: A graphic notation for modeling objects, classes and their relationships ("associations"):
  - **Class diagram: Template for describing many instances of data. Useful for taxonomies, patters, schemata...**
  - **Instance diagram: A particular set of objects relating to each other. Useful for discussing scenarios, test cases and examples**
- Together-J: CASE Tool for building object diagrams, in particular class diagrams

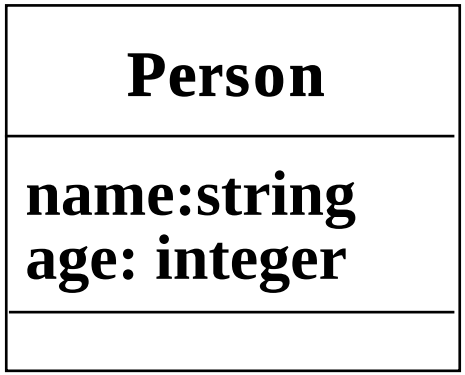
UML: Class and Instance Diagrams

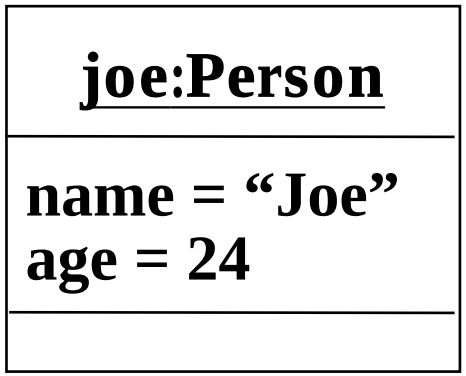


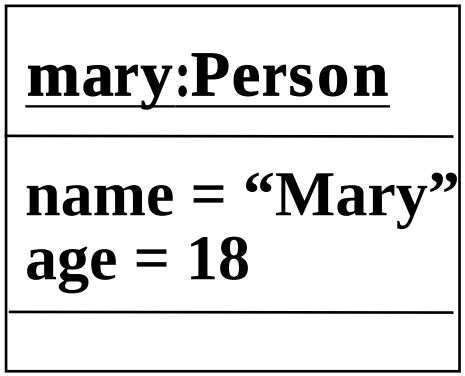
Class Diagram

Instance Diagram

Attributes and Values



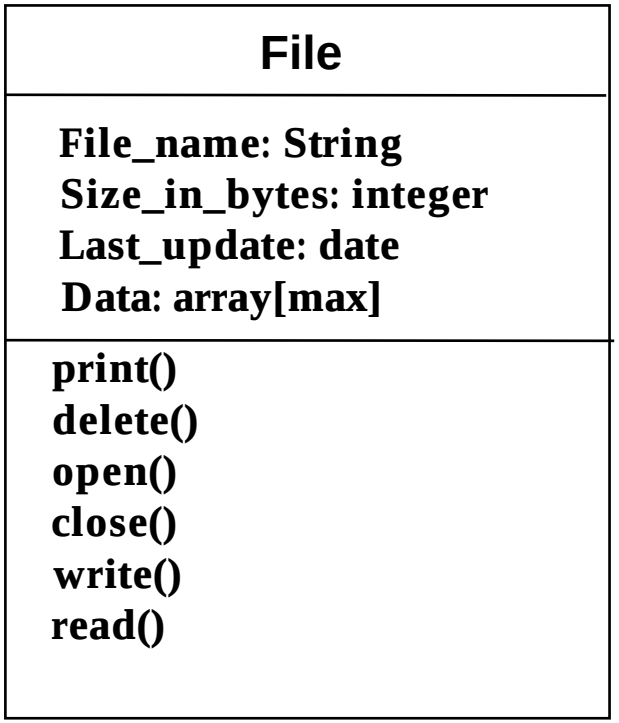




Operation, Signature or Method? What when?

- ***Operation***: A function or transformation applied to objects in a class. All objects in a class share the same operations *(Analysis Phase)*
- ***Signature:*** Number & types of arguments, type of result value. All methods of a class have the same signature *(Object Design Phase)*
- ***Method***: Implementation of an operation for a class *(Implementation Phase)*

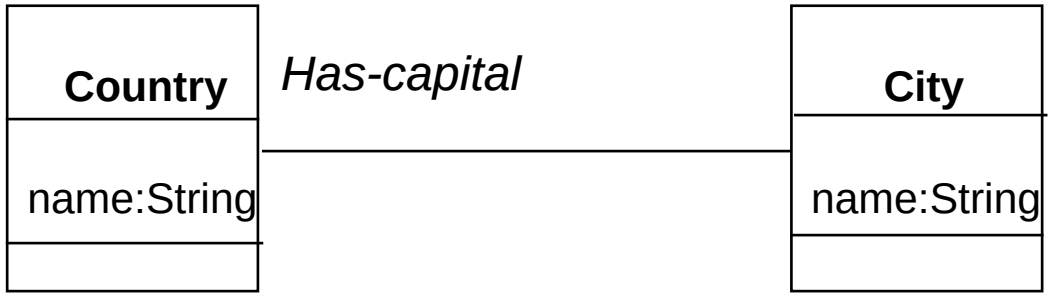
  Polymorphic operation: The same operation applies to many different classes.



Links and Associations

- Links and associations establish relationships among objects and classes.
- **Link:**
  - **A connection between two object instances. A link is like a tuple.**
  - **A link is an instance of an association**
- **Association:**
  - **Basically a bidirectional mapping.**
  - **One-to-one, many-to-one, one-to-many,**
  - **An association describes a set of links like a class describes a set of objects.**

1-to-1 and 1-to-many Associations

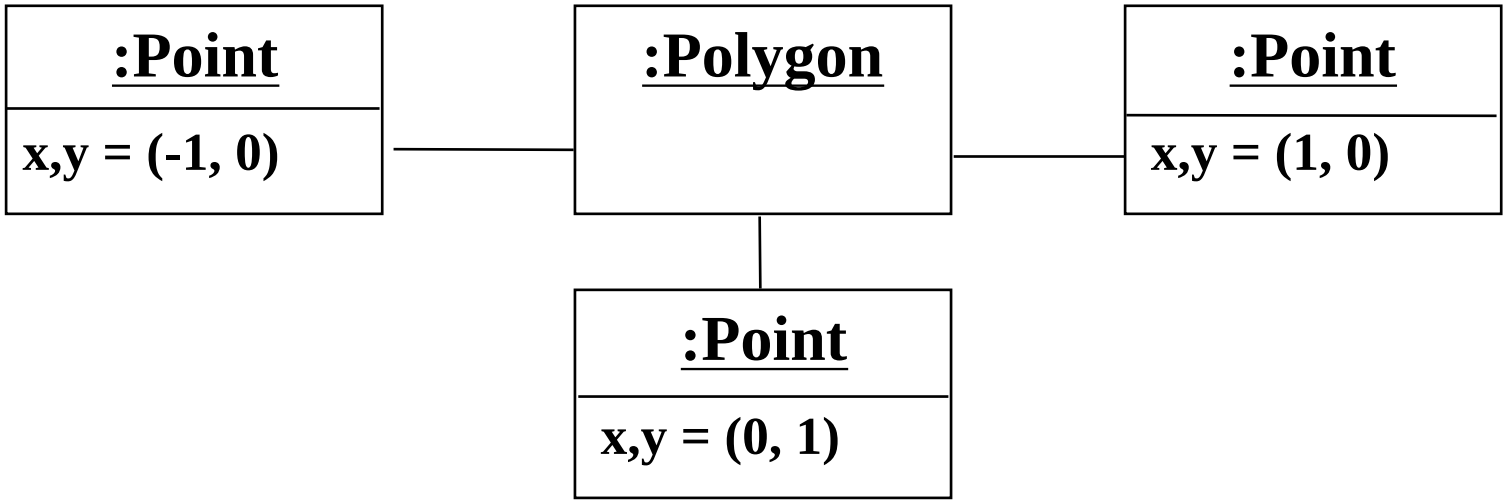


One-to-one association

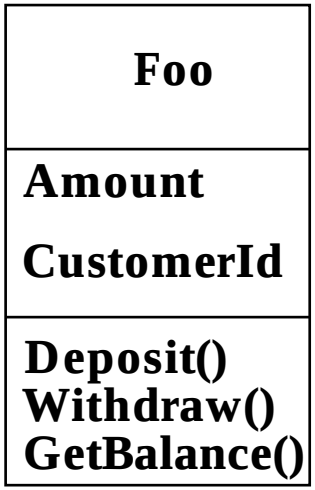
One-to-many association

Object Instance Diagram

Example for 1-to-many

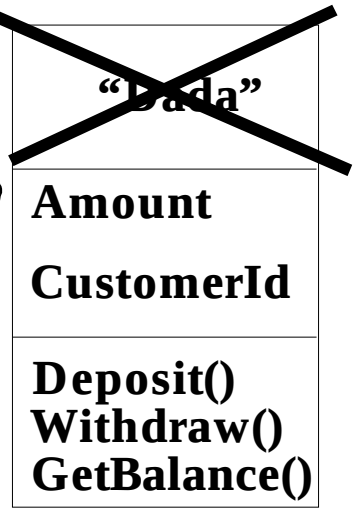


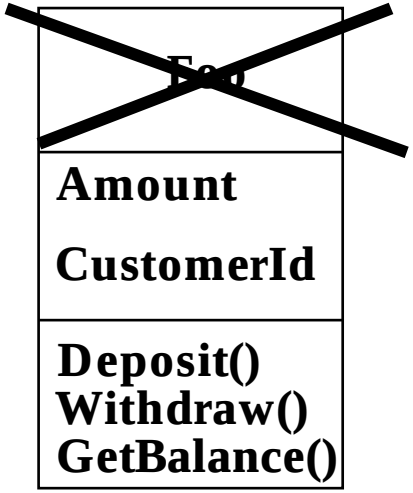
Object Modeling in Practice: Class Identification

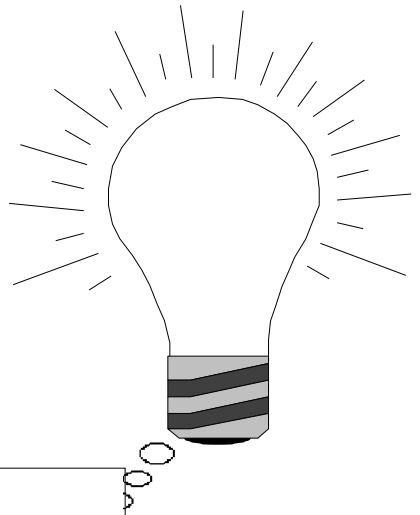


Class Identification: Name of Class, Attributes and Methods

Object Modeling in Practice: Encourage Brainstorming







Account

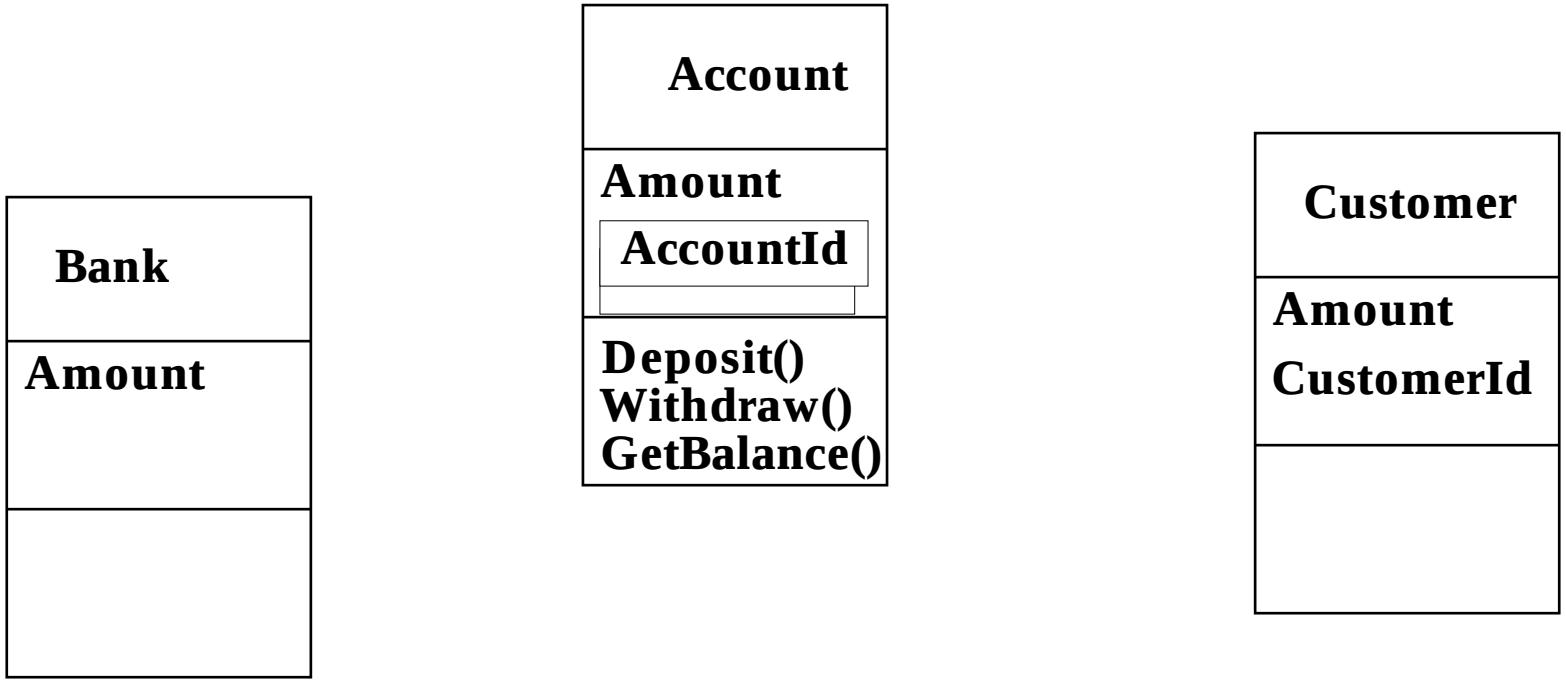
Amount

CustomerId

Deposit()
Withdraw()
GetBalance()

Naming is important!
Is Foo the right name?

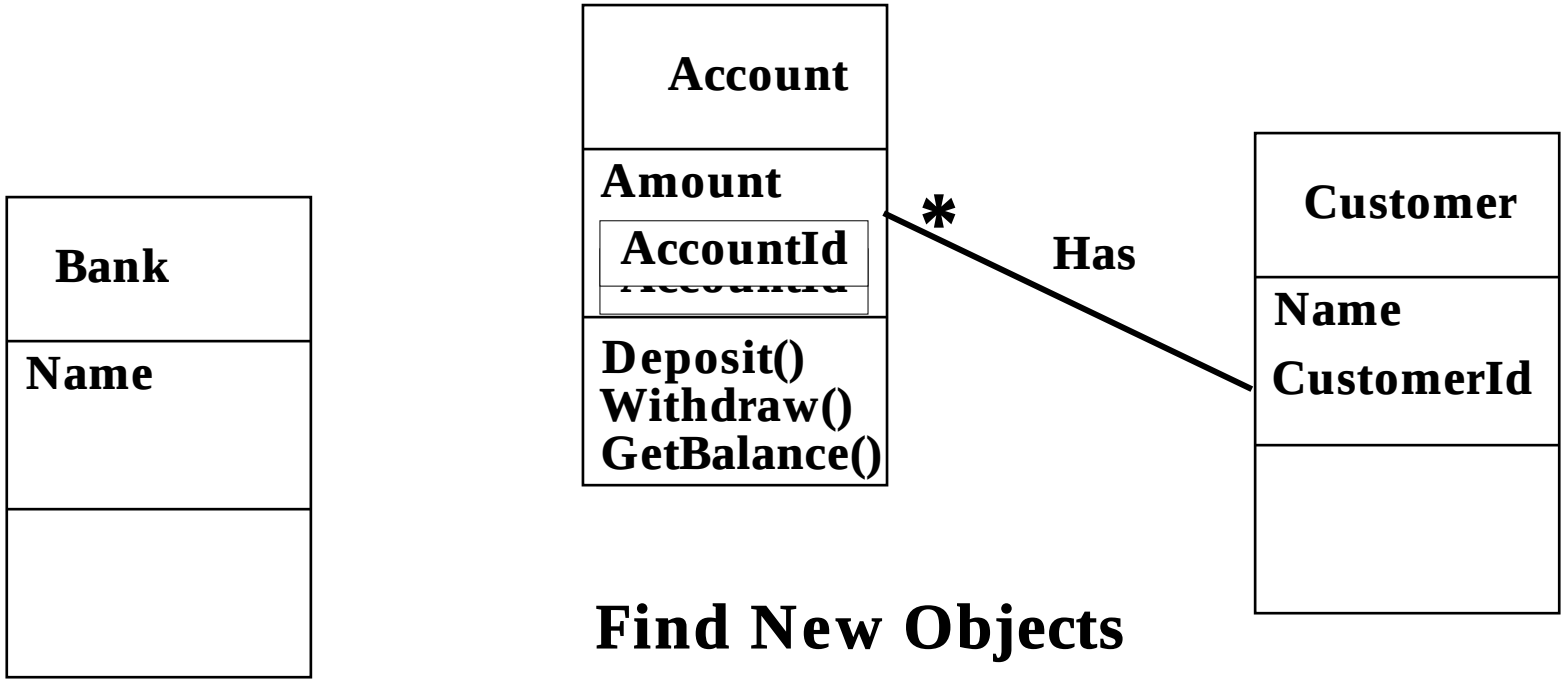
Object Modeling in Practice ctd



Find New Objects

Iterate on Names, Attributes and Methods

Object Modeling in Practice: A Banking System



Find New Objects

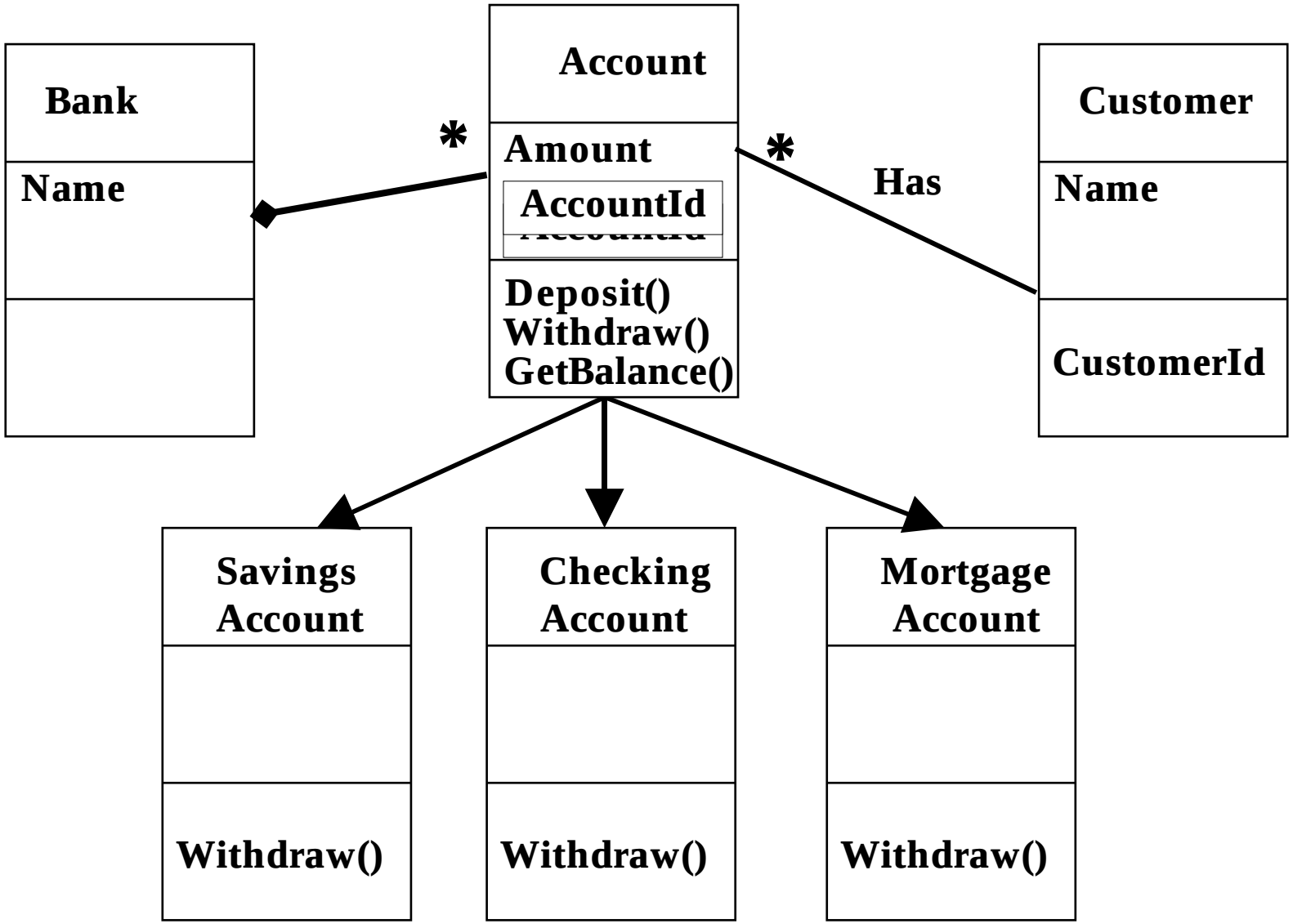
Iterate on Names, Attributes and Methods

Find Associations between Objects

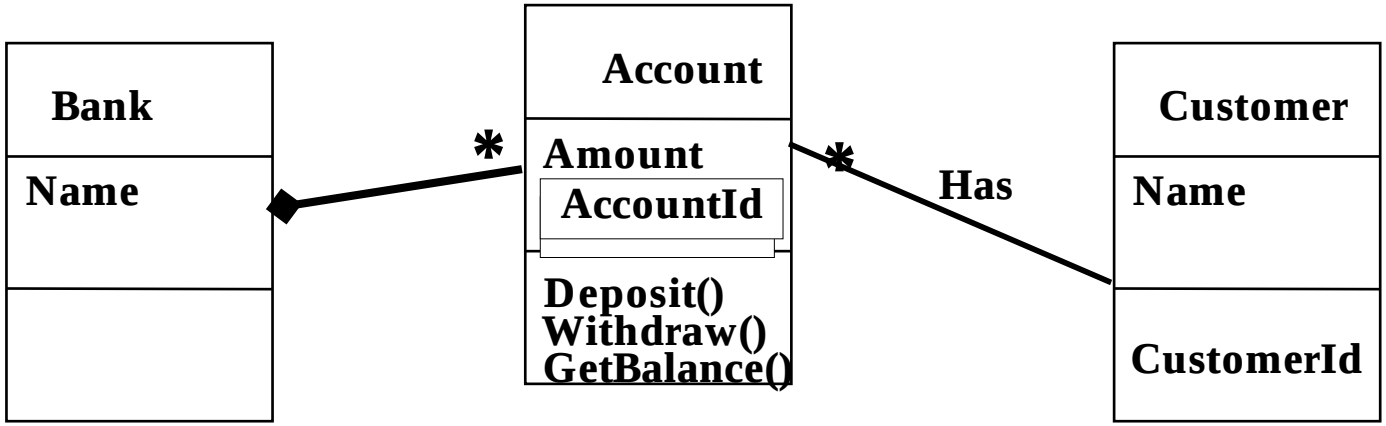
Label the assocations

Determine the multiplicity of the assocations

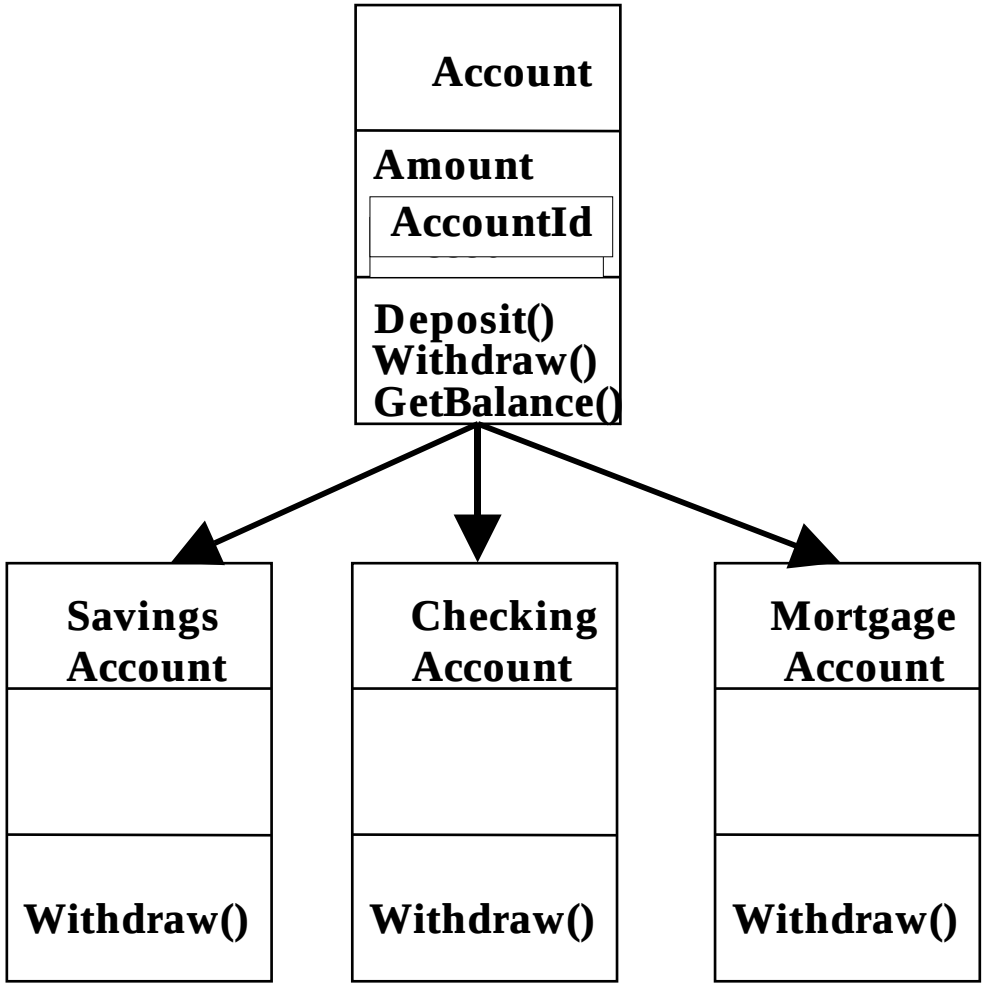
Object Modeling in Practice: Iterate, Categorize!



How to Avoid Ravioli Models:
Don't put too many classes on the same page



Avoid Ravioli Models:
Put Taxonomies on separate Page



Object Modeling in Practice: Heuristics

- Allow time for brainstorming
- Iterate, iterate
- Find associations and their multiplicity
  - **Unusual multiplicities usually lead to new objects or categories**
- Find Inheritance: Look for a Taxonomy, Categorize
- Find Aggregation
- Avoid Ravioli Models
  - **If an object model has more than 7 objects on a page, start partitioning the object model over more than one page**
- Iterate, iterate

Many-to-Many Associations

From Problem Statement To Object Model

Problem Statement: A stock exchange lists many companies. Each company is uniquely identified by a ticker symbol

Class Diagram:

From Problem Statement to Code

Problem Statement : A stock exchange lists many companies. Each company is identified by a ticker Symbol

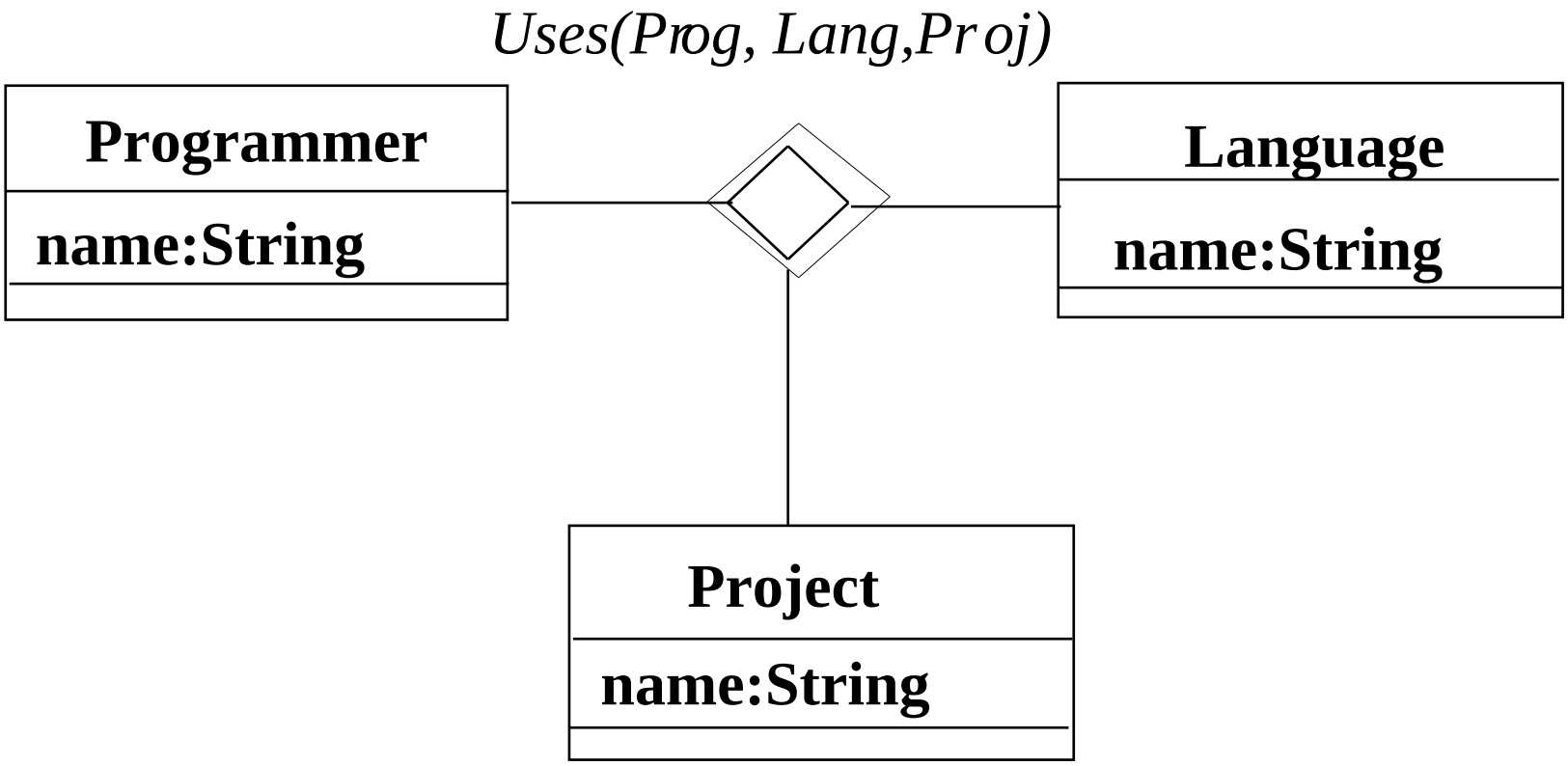
Class Diagram:

Java Code

```
public class StockExchange
{
 public Vector m_Company = new Vector();
};

public class Company
{
 public int m_tickerSymbol
 public Vector m_StockExchange = new Vector();
};
```

Ternary Associations

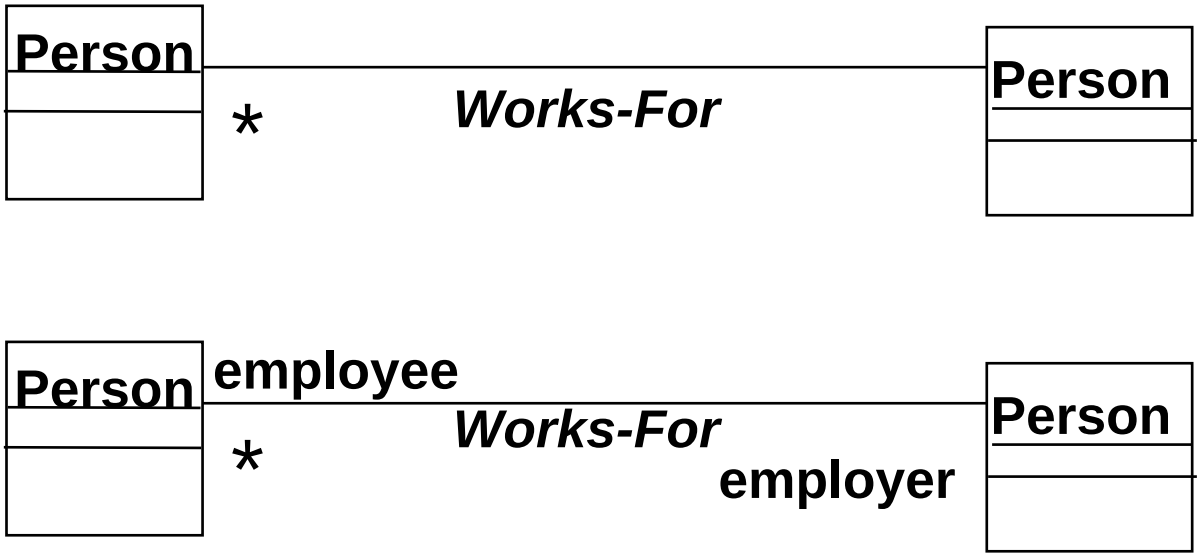


Roles

- A role name is the name that uniquely identifies one end of an association.
- A role name is written next to the association line near the class that plays the role.
- When do you use role names?
  - **Necessary for associations between two objects of the same class**
  - **Also useful to distinguish between two associations between the same pair of classes**
- When do you not use role names?
  - **If there is only a single association between a pair of distinct classes, the names of the classes serve as good role names**

Example of Role

Problem Statement : A person assumes the role of employee with respect to another person, who assumes the role of employer with respect to the first person.



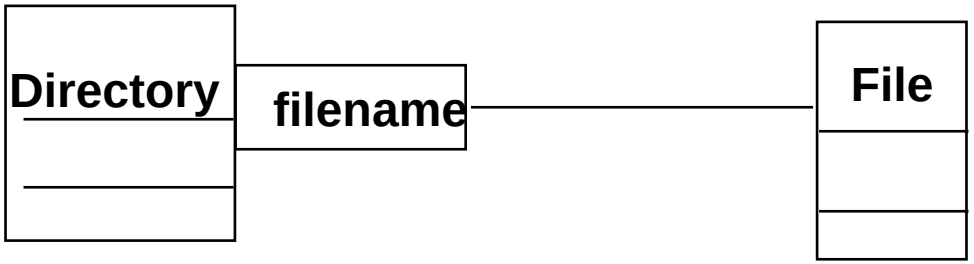
Roles in Associations

- **Client Role:**
  - **An object that can operate upon other objects but that is never operated upon by other objects.**
- **Server Role:**
  - **An object that never operates upon other objects. It is only operated upon by other objects.**
- **Agent Role:**
  - **An object that can both operate upon other objects and be operated upon by other objects. An agent is usually created to do some work on behalf of an actor or another agent.**

Qualification

- The qualifier improves the information about the multiplicity of the association between the classes.
- It is good for reducing 1 to many multiplicity to 1-1 multiplicity

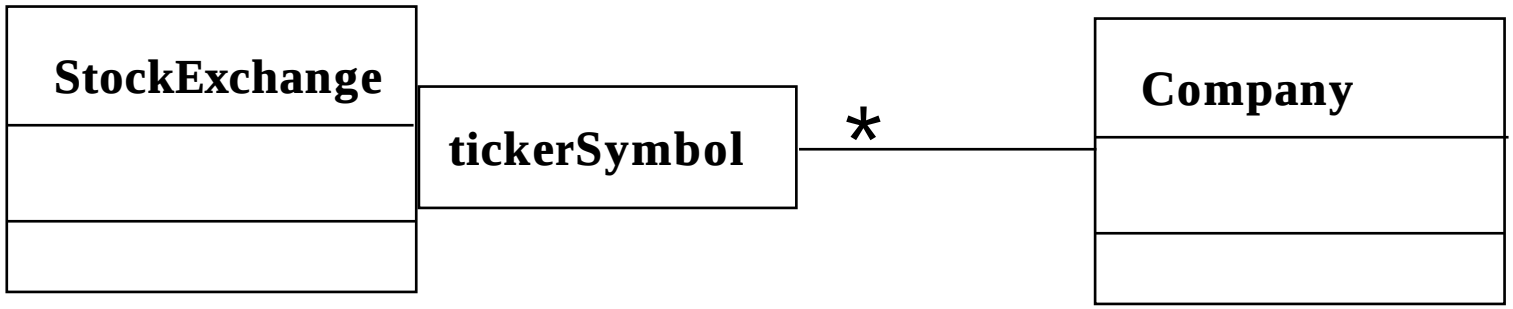
A directory has many files. A file belongs only to one directory



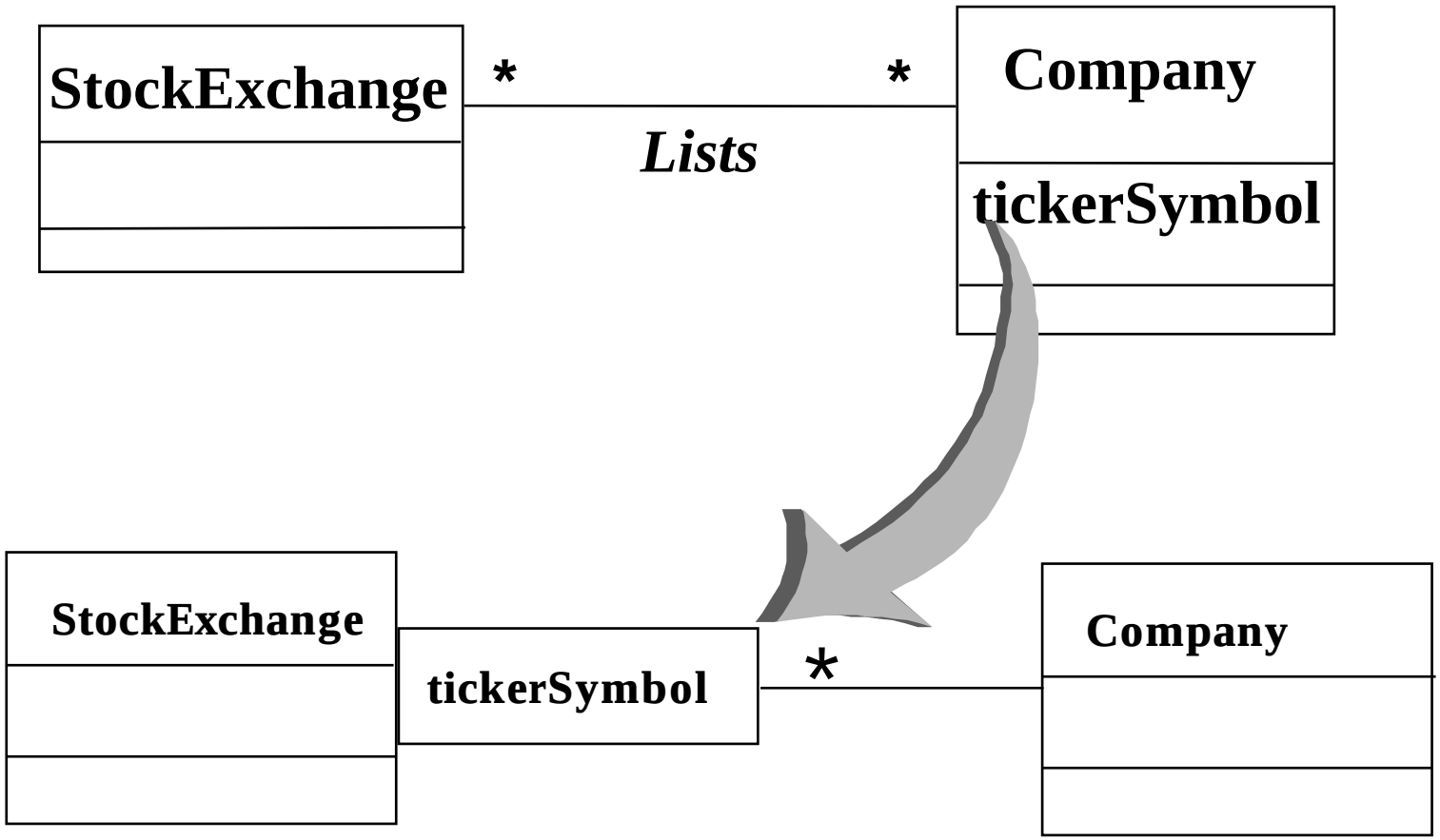
A directory has many files, each with a unique name

Use of Qualification

Problem Statement : A stock exchange lists many companies. However, a stock exchange lists only one company with a given ticker symbol. A company may be listed on many stock exchanges, possibly with different ticker symbols.

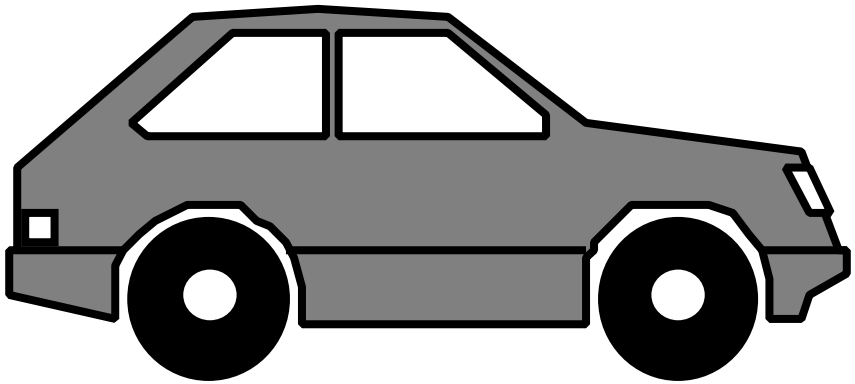


Use of Qualification: Reducing multiplicity

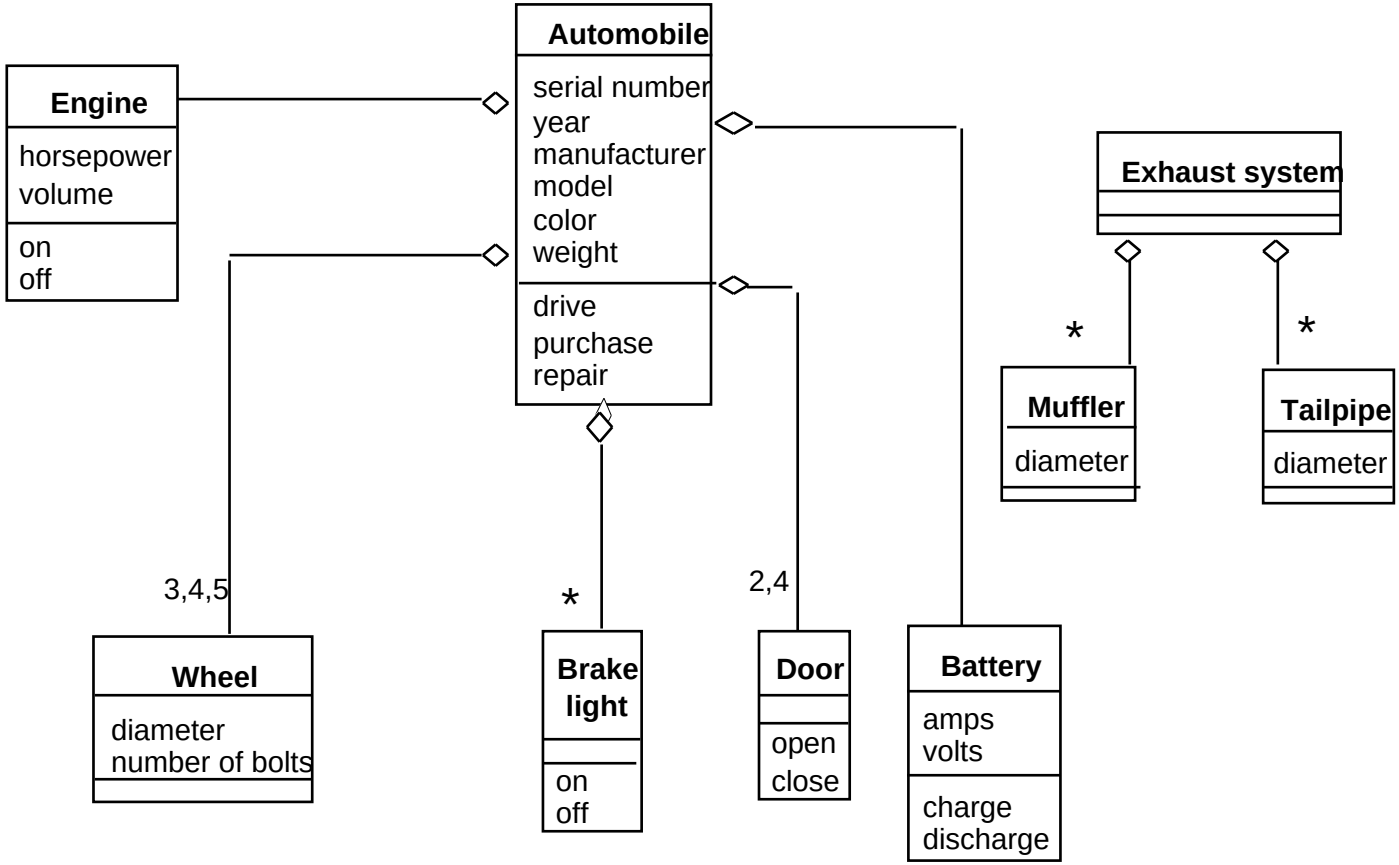


Aggregation

- Models "part of" hierarchy
- Useful for modeling the breakdown of a product into its component parts (sometimes called bills of materials (BOM) by manufacturers)
- UML notation: Like an association but with a small diamond indicating the assembly end of the relationship.



Aggregation

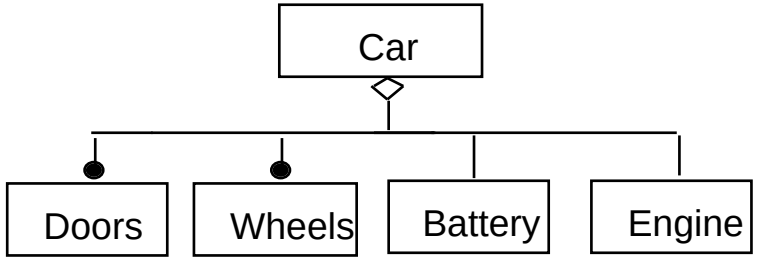


Types of Aggregation

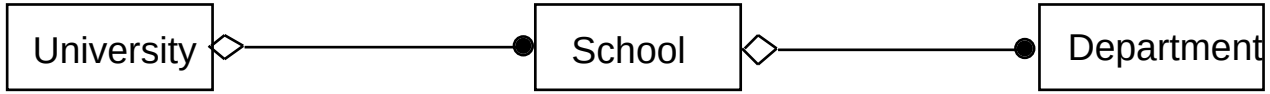
- Aggregation can be fixed, variable or recursive
- Fixed aggregation
  - Fixed number of **parts**
- Variable aggregation
  - *Fixed* number of **levels**, *variable* number of **parts**
- Recursive aggregation
  - *Variable* number of **levels**, *variable* number of **parts**
- A recursive aggregate constains an instance of the same kind of aggregate
  - **Usual form: A superclass and two subclasses, one of which is an assembly of instances of the superclass (intermediate node), the other is a terminal node**

UML patterns for typical aggregations

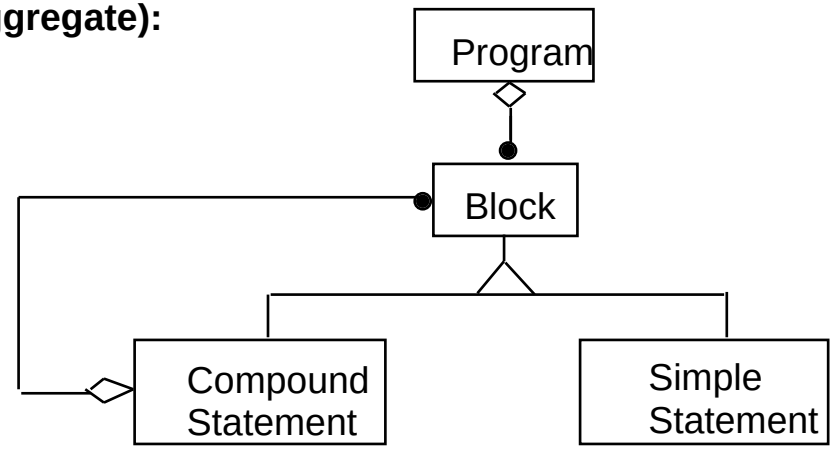
Fixed Structure:



Organization Chart (variable aggregate):

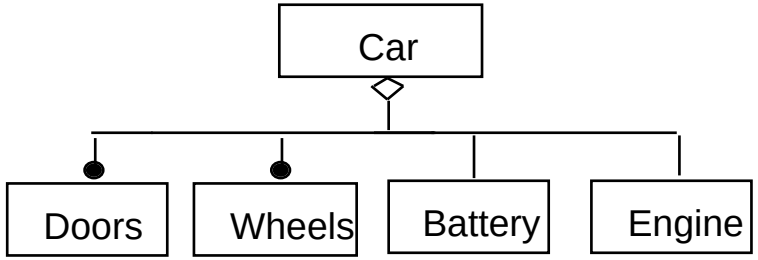


Dynamic tree (recursive aggregate):

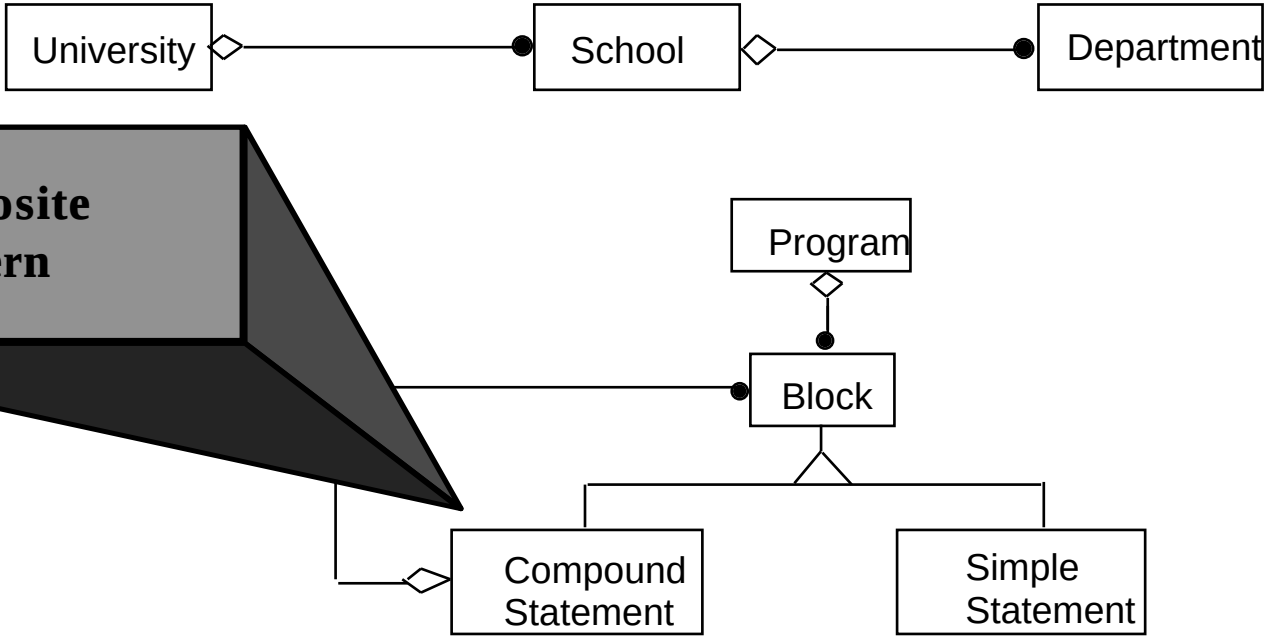


UML patterns for typical aggregations

Fixed Structure:



Organization Chart (variable aggregate):



Patterns and Frameworks

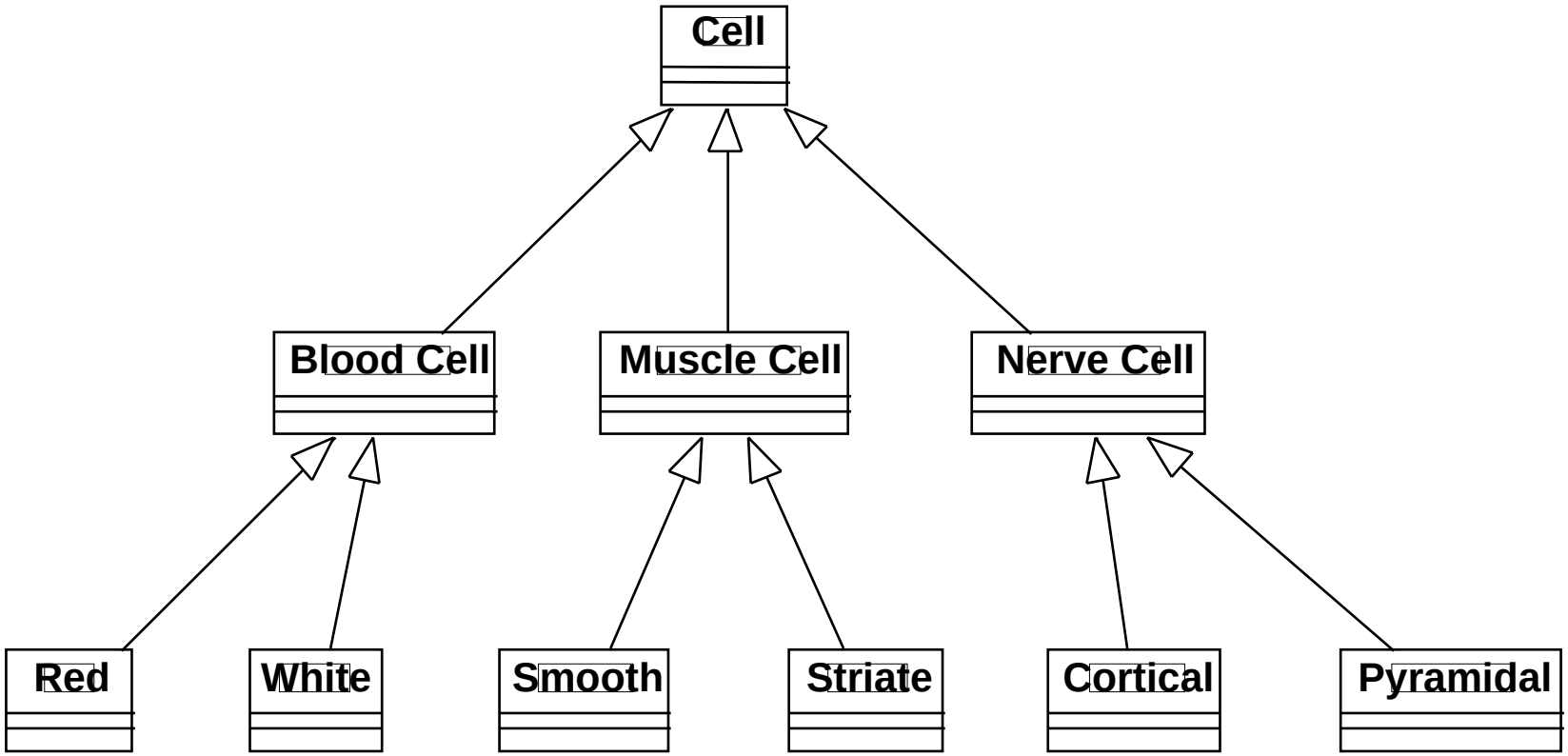
- Literature
  - **Christopher Alexander, “A Timeless Way of Building”, Oxford University Press, 1979.**
  - **Christopher Alexander, “A Pattern Language”, Oxford University Press, 1977.**
  - **E. Gamma et. al , “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison Wesley, 1995**
- Design Pattern:
  - **Definition: Class diagram for problem that occurs again and again.**
  - **Patterns are a way to reuse common knowledge about the interaction of a few classes (“Chess end game”)**
- Framework:
  - **Definition: The design of a set of object that collaborate to carry out a set of responsibilities in a certain application domain**
  - **Frameworks are a way to reuse high-level design (“Chess middle game”)**

Key Benefits of Patterns and Frameworks

- Prefabricated infrastructure with certain properties
  - **Extensibility, scalability, reusability, ...**
- Provide Architectural guidance
- Lead to less monolithic applications
- Lead to reduced maintenance
- Lead to the reuse of designs in the development of other complex systems
- Possible foundation for a software components industry

Inheritance

- Models "kind of" hierarchy
- Powerful notation for sharing similarities among classes while preserving their differences
- UML Notation: An arrow with a triangle



Aggregation vs Inheritance

- Both associations describe trees (hierarchies)
  - **Aggregation tree describes a-part-of relationships (also called and-relationship)**
  - **Inheritance tree describes "kind-of" relationships (also called or-relationship)**
- Aggregation relates instances (involves two or more *different* objects)
- Inheritance relates classes (a way to structure the description of a *single* object)

Other Associations

- Uses:
  - **A subsystem uses another subsystem (System Design)**
- Contains:
  - **Sometimes called “spatial aggregation”**
  - **... contains ...**
  - **Example: A UML package contains another UML package**
- Parent/child relationship:
  - **... is father of ...**
  - **... is mother of ...**
- Seniority:
  - **... is older than ...**
  - **... is more experienced than ...**

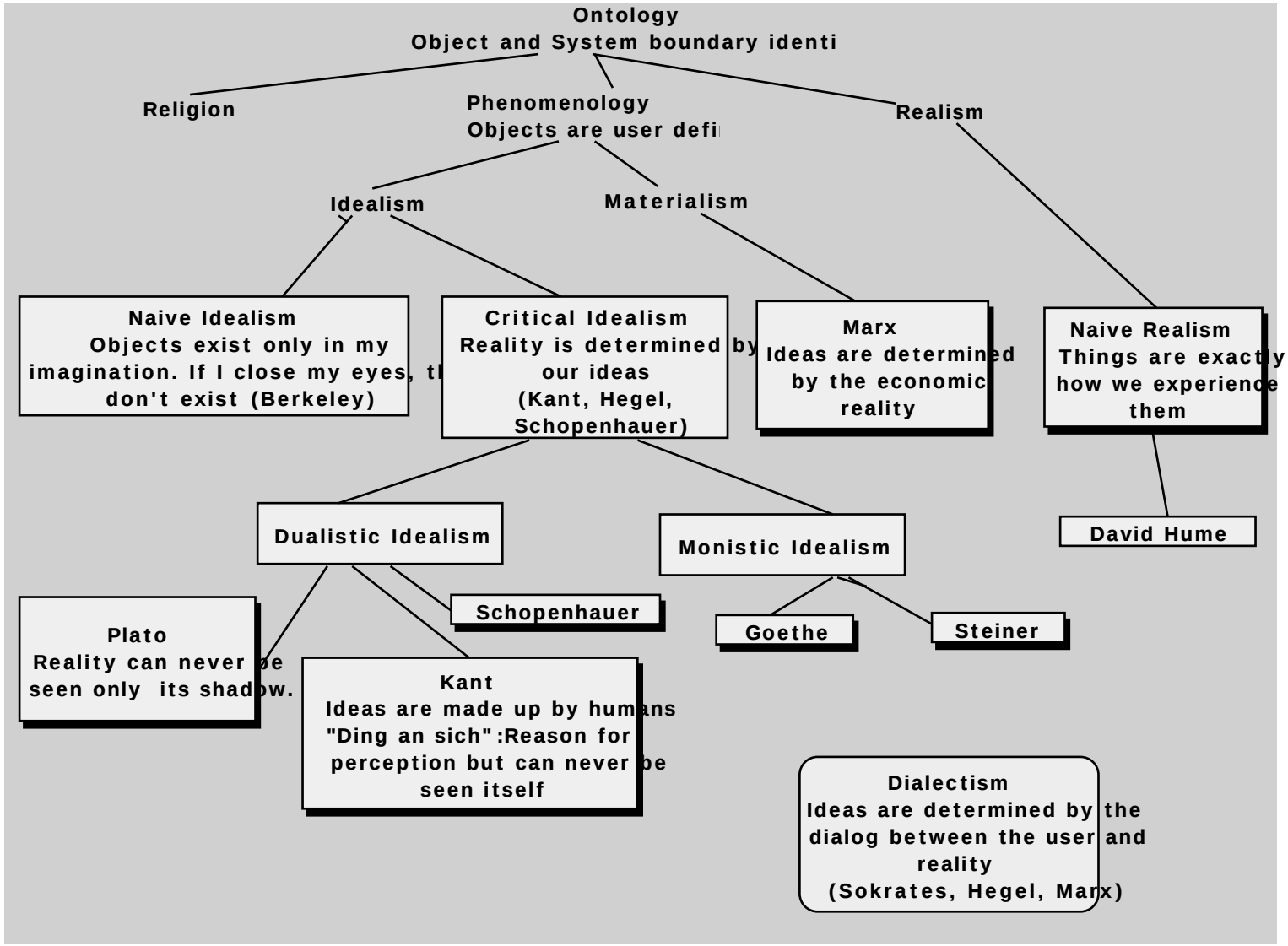
How do you find classes?

- Learn about problem domain analysis: Talk to or observe your client
- Find participating objects in use cases
- Apply general world knowledge and intuition
- Apply design patterns
- Try to establish a taxonomy
- Read scenario in problem statement:
  - **If a customer enters a store with the intention of buying a toy for a child, then advice must be available within a reasonable time concerning the suitability of the toy for the child. This will depend on the age range of the child and the attributes of the toy. If the toy is a dangerous item, then it is unsuitable.**
- Do a textual analysis of scenario (Abbott)
  - **Nouns are good candidates for classes**

Abbott's Textual Analysis

Part of speech	*Model component*	*Example*
Proper noun	object	Jim Smith
Improper noun	class	toy
Doing verb	method	buy
being verb	inheritance	is-a (kind-of)
having verb	aggregation	has an
modal verb	constraint	must be
adjective	attribute	3 years old
transitive verb	method	enter
intransitive verb	method (event)	depends on

Software Engineers are not the only System Analysts



Summary

- System modeling
  - **Object model**
  - **Dynamic model**
  - **Functional model**
- Object models
  - **Class diagrams and instance diagrams**
- UML constructs
  - **Classes , attributes, methods**
  - **Objects**
  - **Roles, Qualifiers**
  - **Links, associations**
- Class identification is a major activity of object modeling