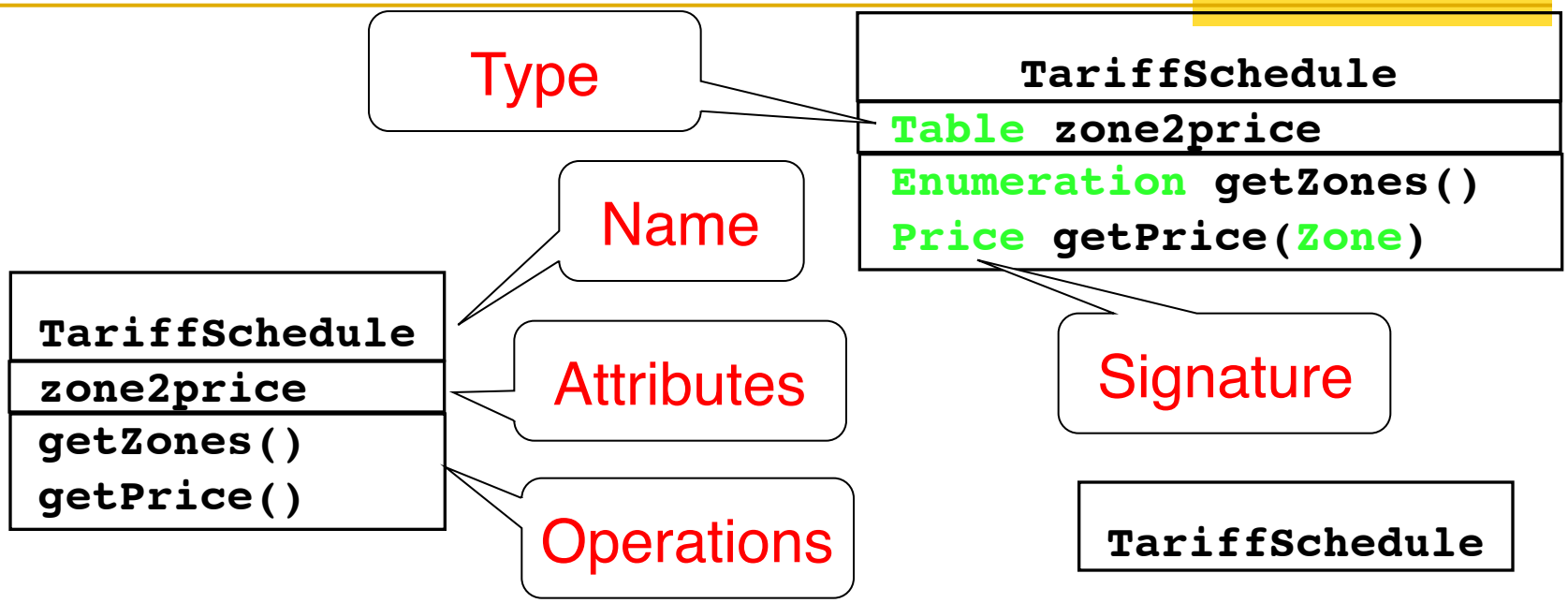
Modeling with UML (Class Diagrams)

Class Diagrams

- Class diagrams represent the structure of the system
- Used
  - during requirements analysis to model application domain concepts
  - during system design to model subsystems
  - during object design to specify the detailed behavior and attributes of classes.

Classes



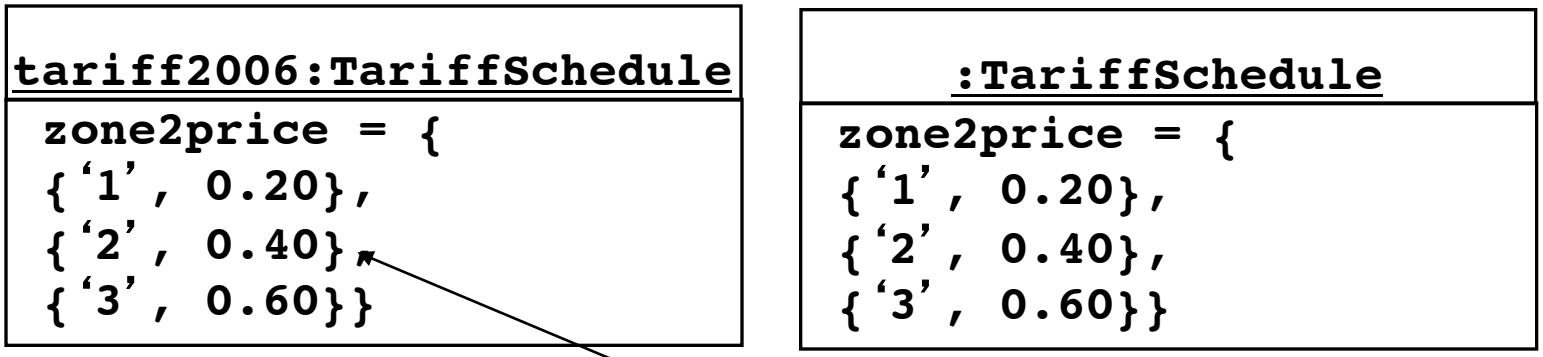
- A ***class*** represents a concept
- A class encapsulates state ***(attributes)*** and behavior ***(operations)***

  Each attribute has a ***type***

  Each operation has a ***signature***

The class name is the only mandatory information

Instances

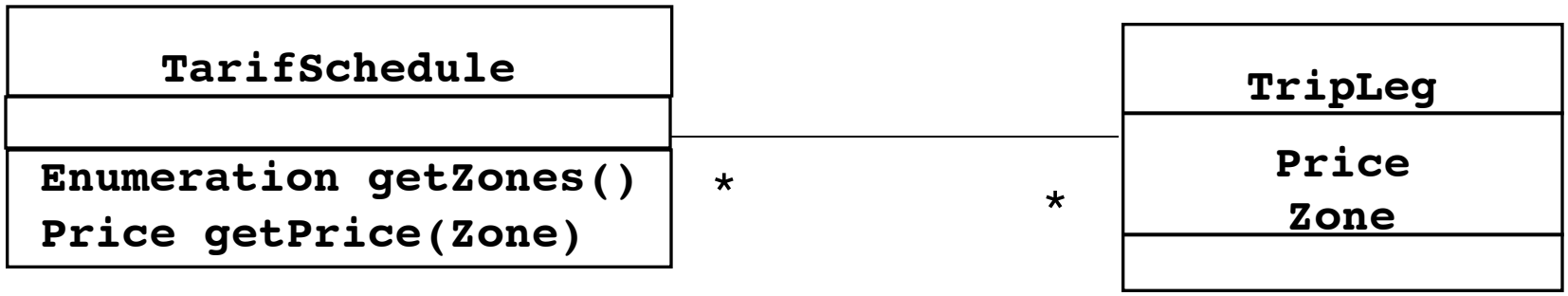


- An ***instance*** represents a phenomenon
- The attributes are represented with their ***values***
- The name of an instance is underlined
- The name can contain only the class name of the instance (anonymous instance)

Class vs. Object

- **Class**
  - An abstraction modeling an entity in the application or solution domain
  - The class is part of the system model (“Passenger”, “Ticket distributor”, “Server”, “TariffSchedule”)
- **Object**
  - A specific instance of a class (“Joe, the passenger who is purchasing a ticket from the ticket distributor”).

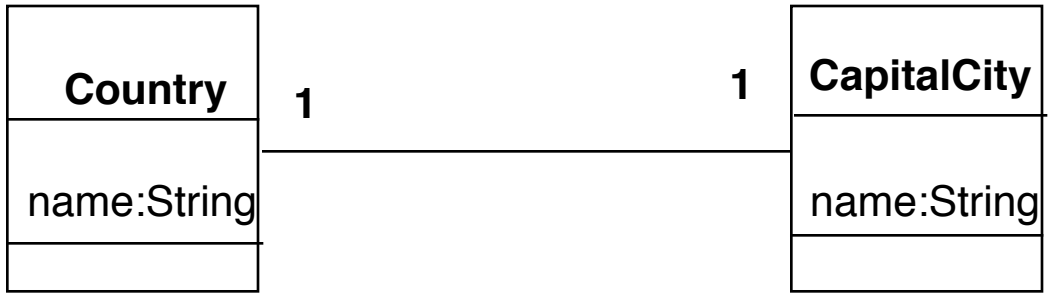
Associations



Associations denote relationships between classes.

The multiplicity of an association end denotes how many objects the instance of a class can legitimately reference.

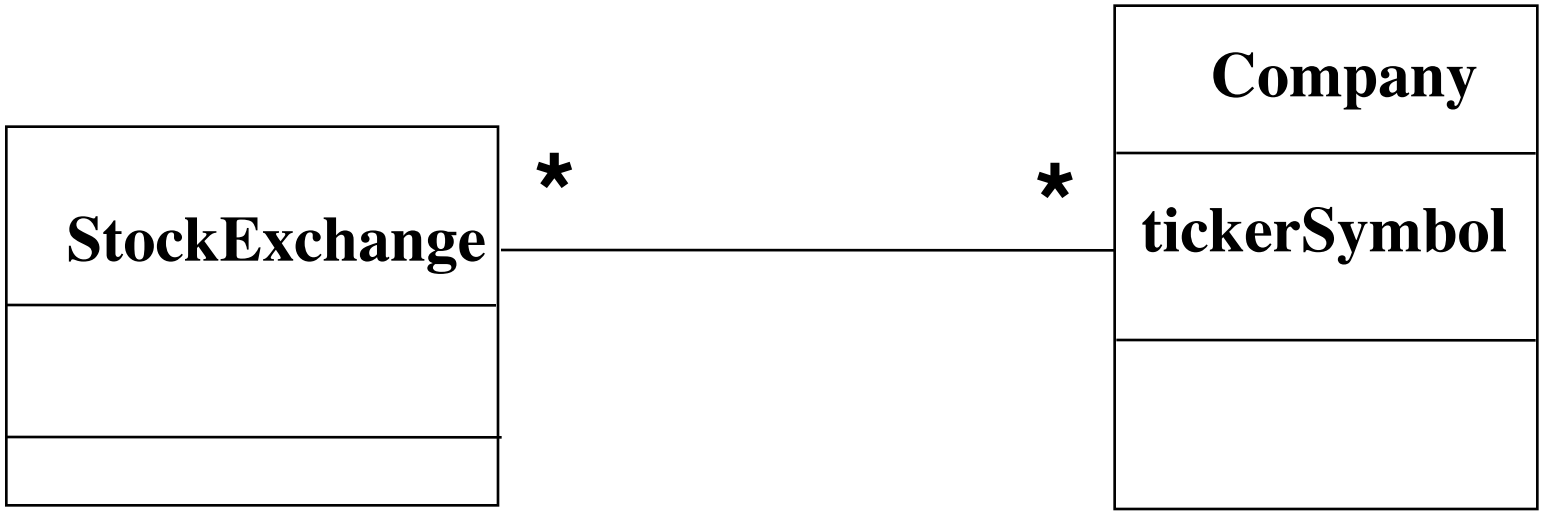
1-to-1 and 1-to-many Associations



1-to-1 association

1-to-many association

Many-to-Many Associations



Model-Driven Software Development

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

Analysis results in analysis object model (UML Class Diagram):

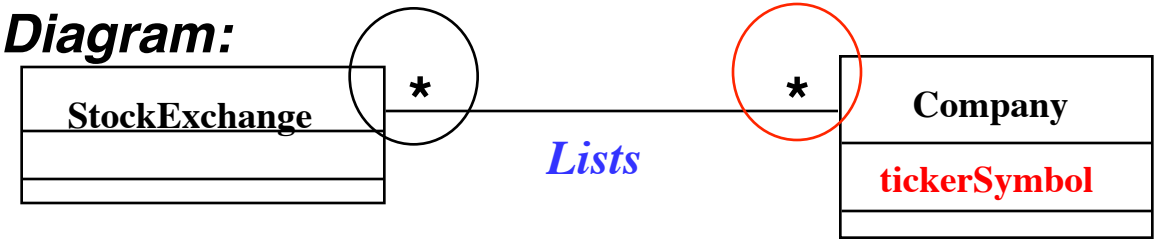
Implementation results in source code (Java):

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

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

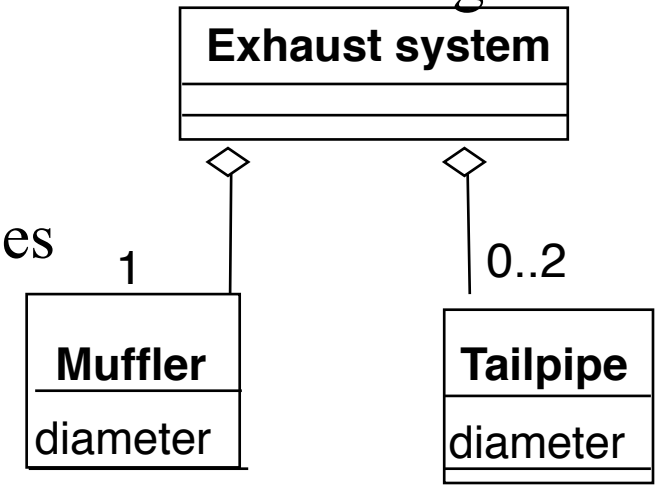
```
public class StockExchange
{
 private Vector m_Company = new Vector();
};

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

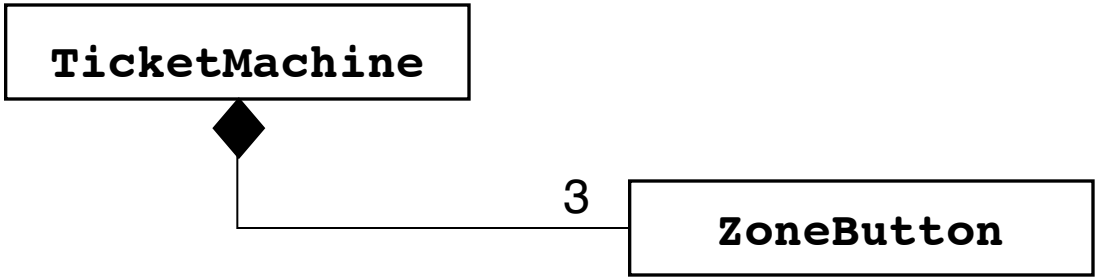
Associations are mapped to Attributes!

Aggregation

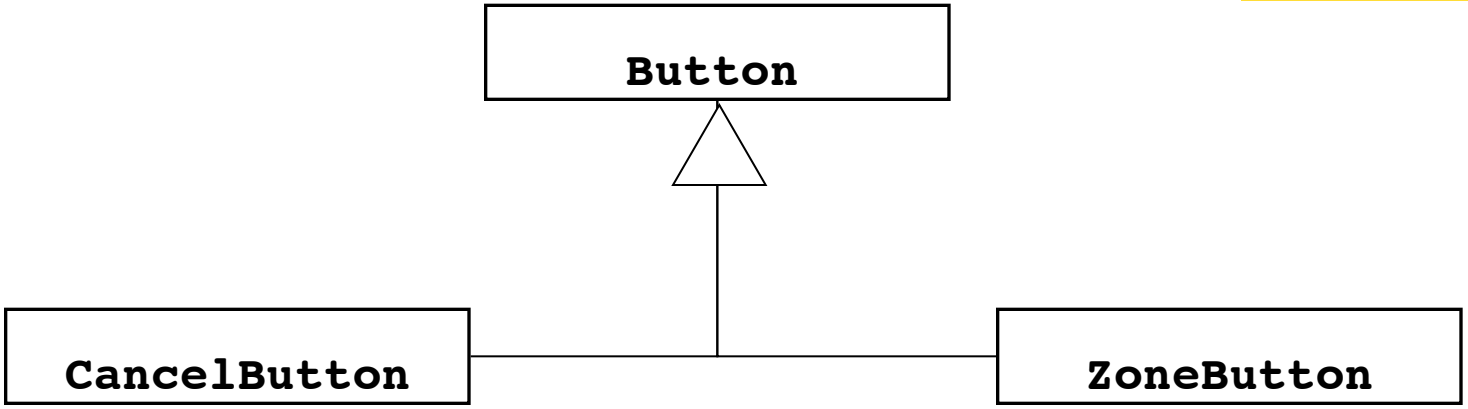
- An *aggregation* is a special case of association denoting a "consists-of" hierarchy
- The *aggregate* is the parent class, the components are the children classes



A solid diamond denotes *composition*: A strong form of aggregation where the *life time of the component instances* is controlled by the aggregate. That is, the parts don't exist on their own ("the whole controls/destroys the parts")



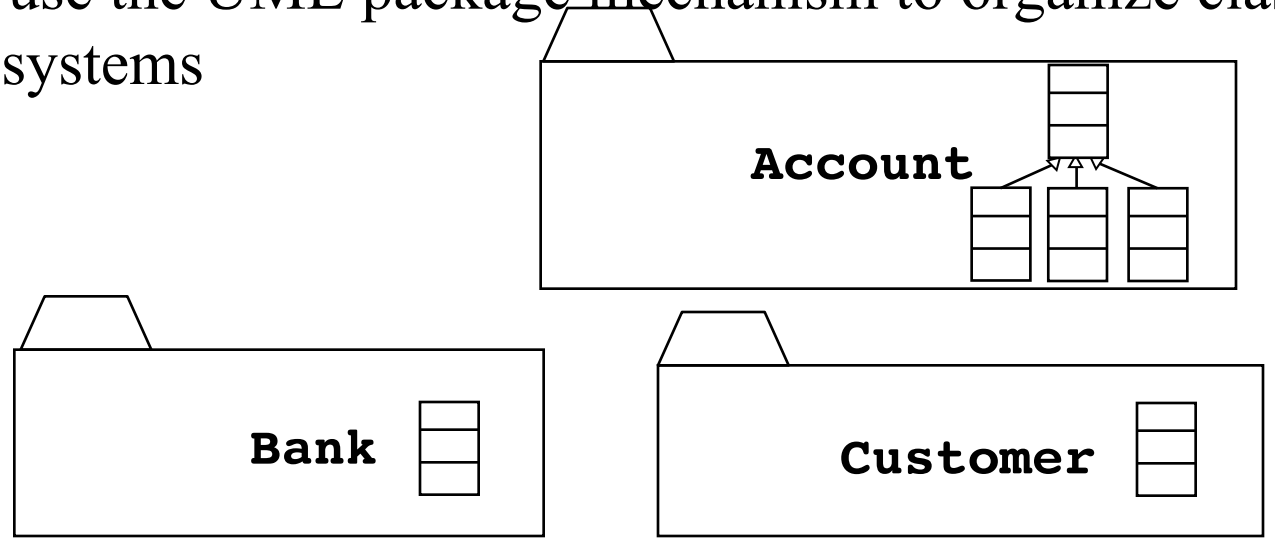
Inheritance



- *Inheritance* is another special case of an association denoting a "kind-of" hierarchy
- Inheritance simplifies the analysis model by introducing a taxonomy
- The **children classes** inherit the attributes and operations of the **parent class.**

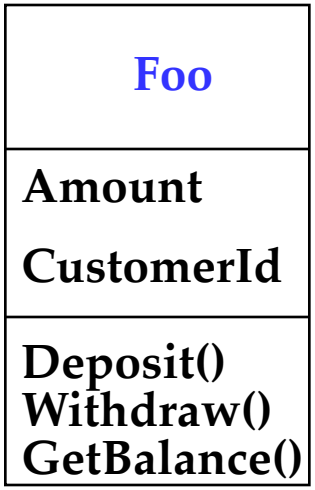
Packages

- Packages help you to organize UML models to increase their readability
- We can use the UML package mechanism to organize classes into subsystems



- Any complex system can be decomposed into subsystems, where each subsystem is modeled as a package.

Object Modeling in Practice



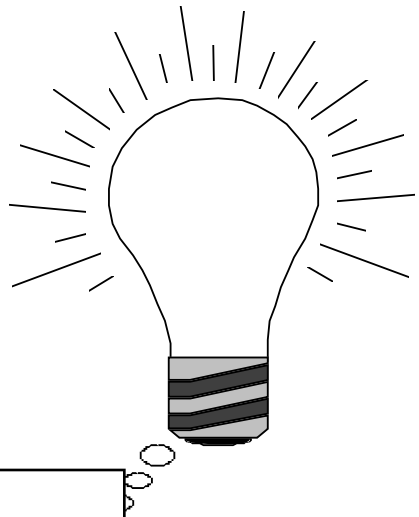
Class Identification: Name of Class, Attributes and Methods

Is Foo the right name?

Object Modeling in Practice: Brainstorming

"Data"
Amount CustomerId
Deposit() Withdraw() GetBalance()

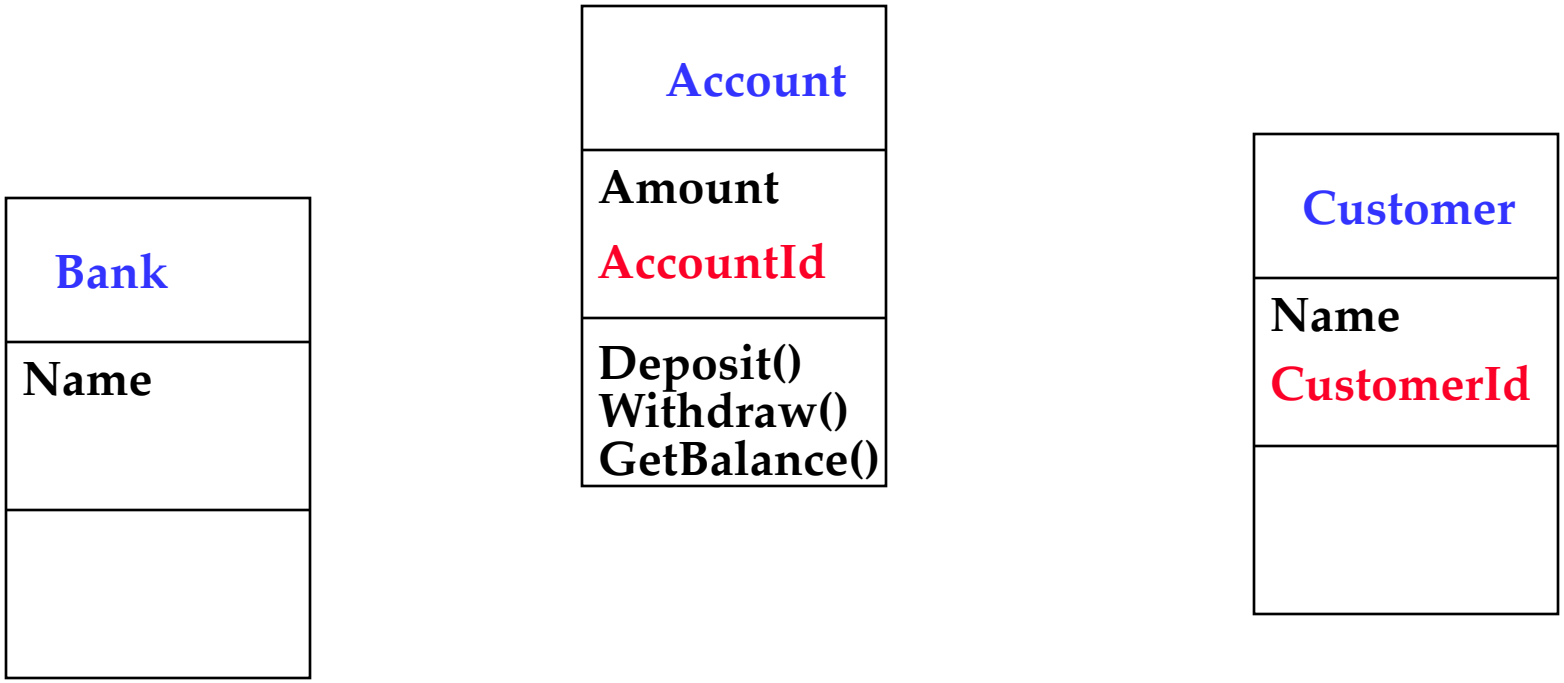
Foo
Amount CustomerId
Deposit() Withdraw() GetBalance()



Account
Amount CustomerId
Deposit() Withdraw() GetBalance()

Is Foo the right name?

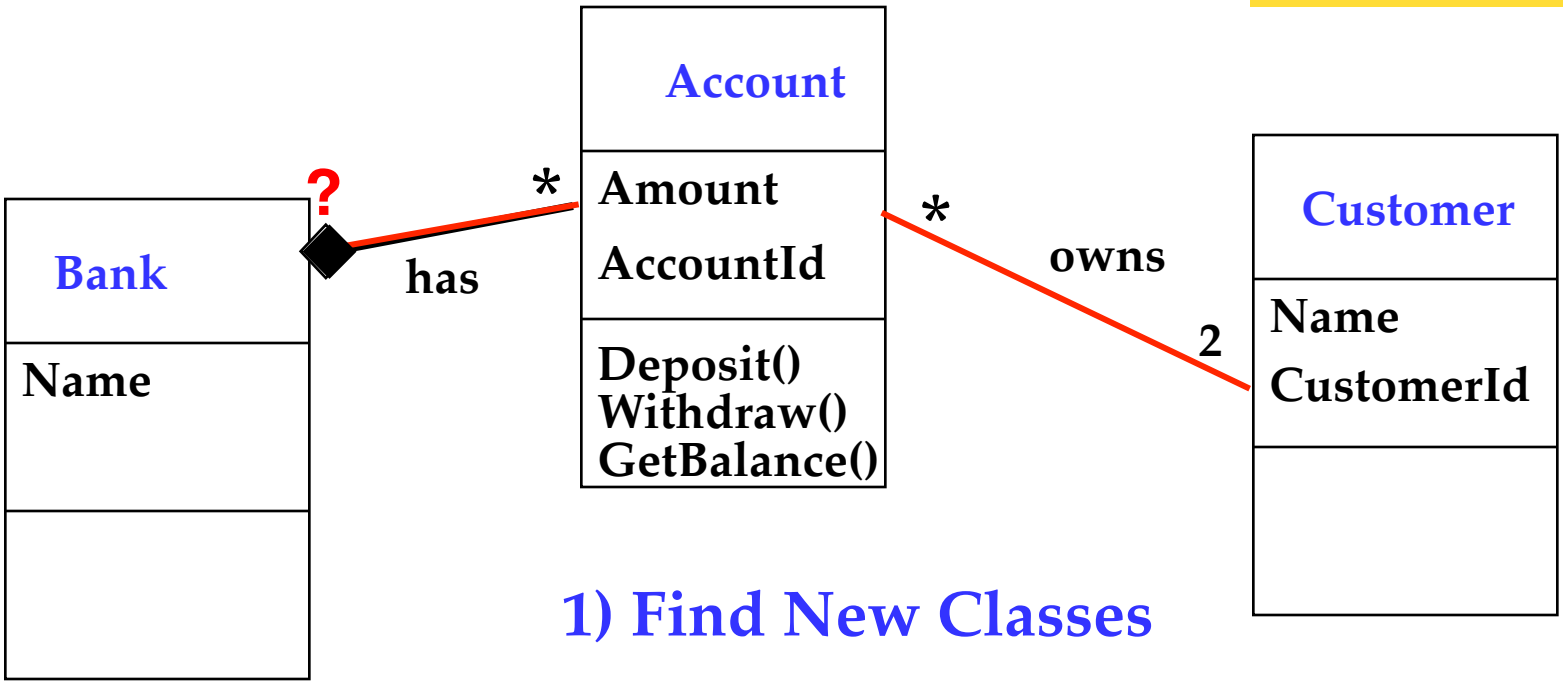
Object Modeling in Practice: More classes



1) Find New Classes

2) Review Names, Attributes and Methods

Object Modeling in Practice: Associations



1) Find New Classes

2) Review Names, Attributes and Methods

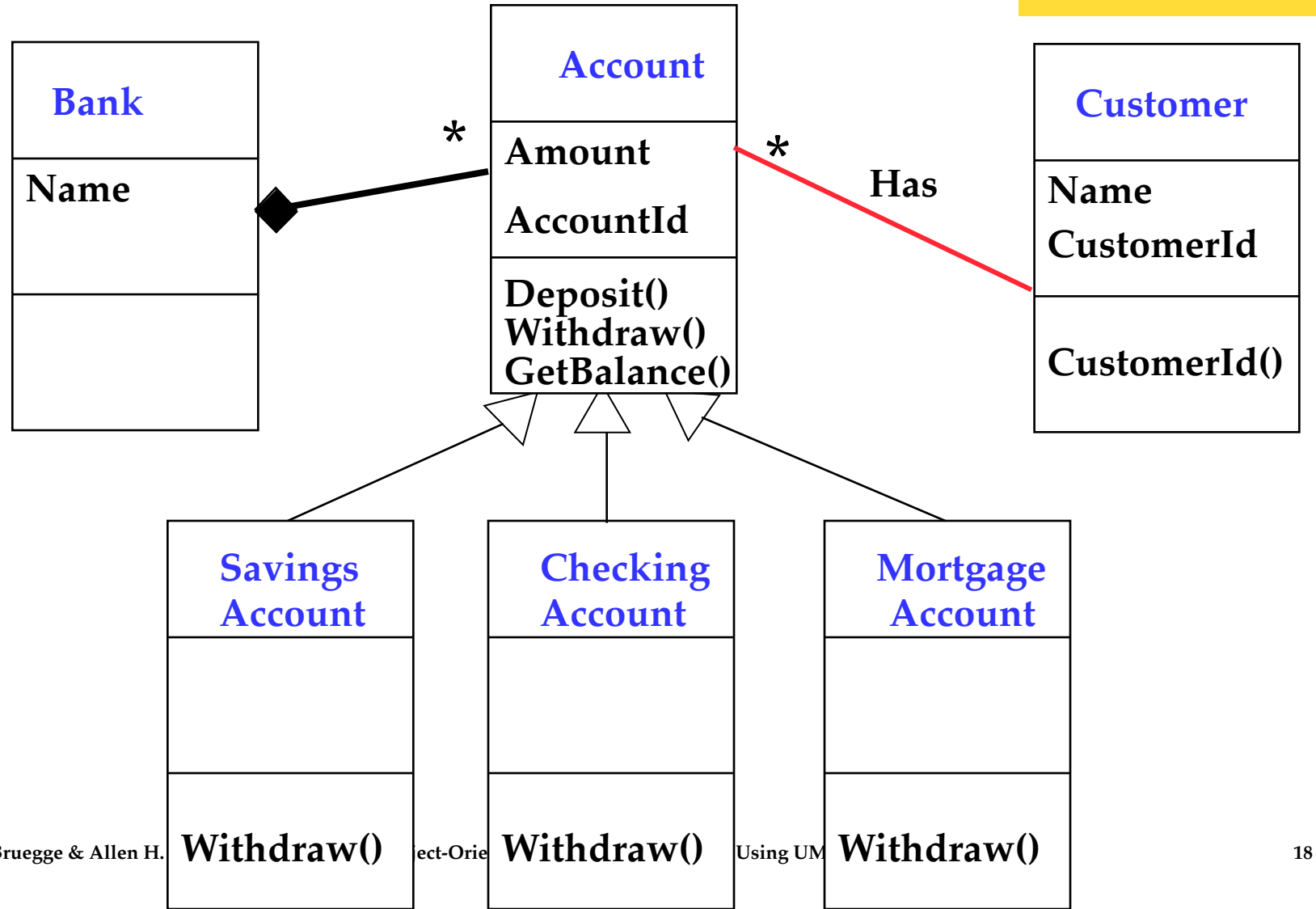
3) Find Associations between Classes

4) Label the generic associations

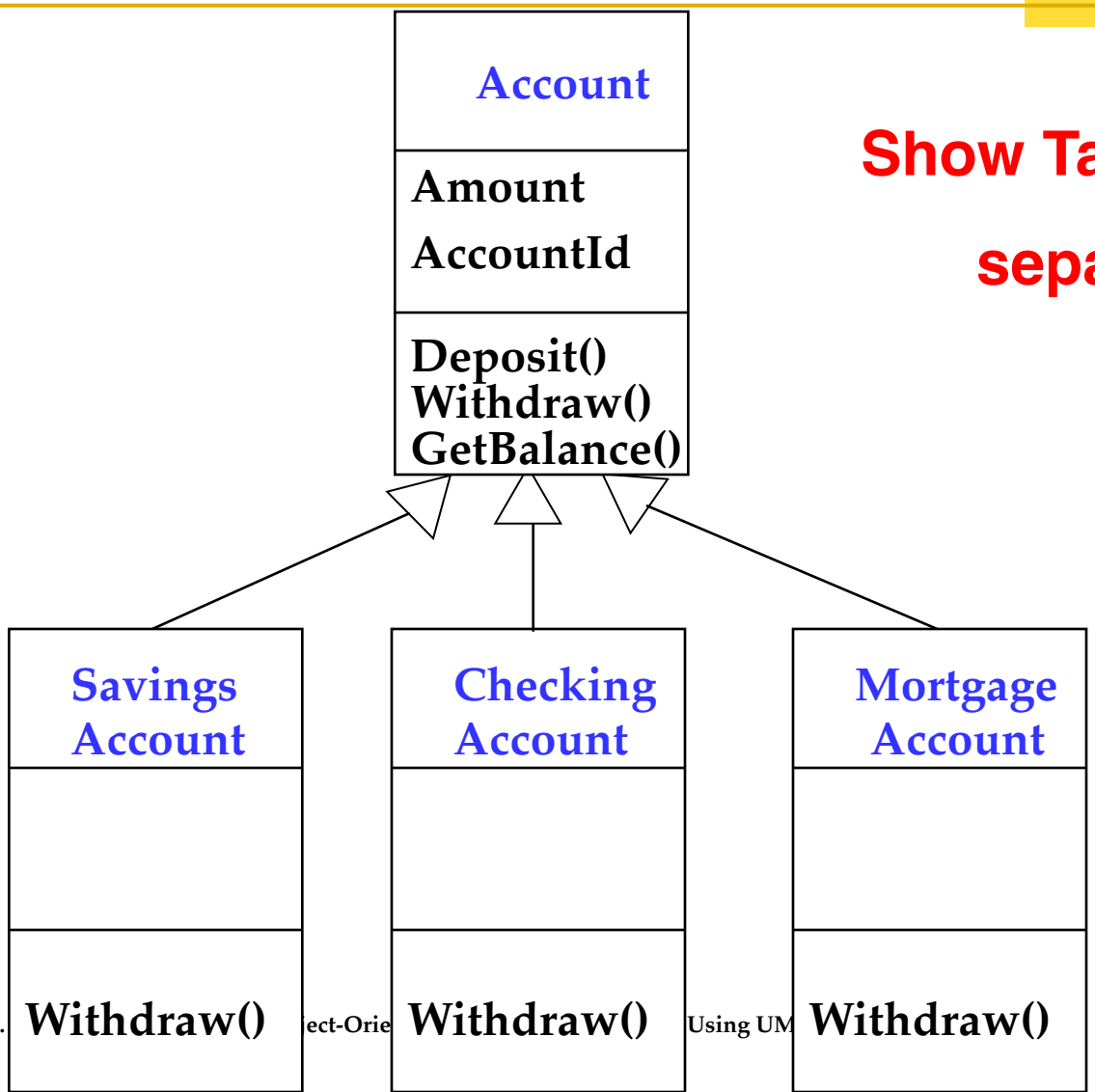
5) Determine the multiplicity of the associations

6) Review associations

Practice Object Modeling: Find Taxonomies



Practice Object Modeling: Simplify, Organize

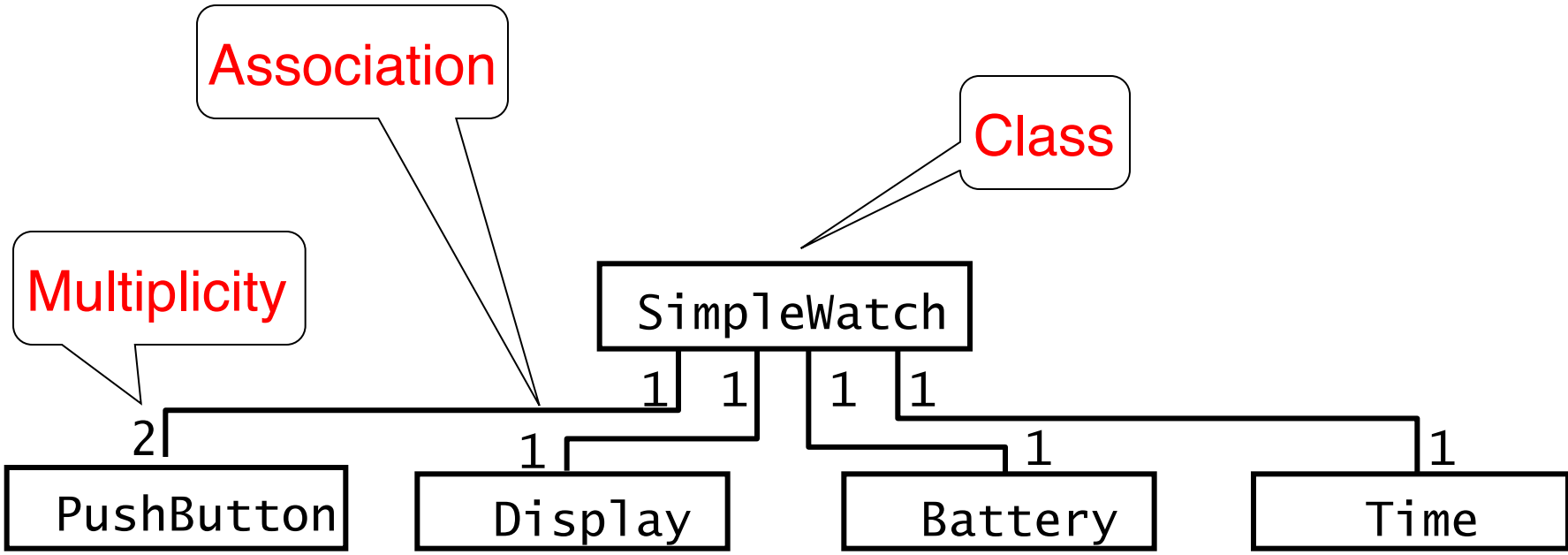


Show Taxonomies separately

Exercise: Class diagrams

- Draw a class diagram for a SimpleWatch that has 2 push buttons to set the time, a LCD display to view the time, has a battery, and shows current time.

Exercise: Class diagrams



Class diagrams represent the structure of the system

Exercise: Class diagrams

Class diagrams represent the structure of the system