Software Engineering II

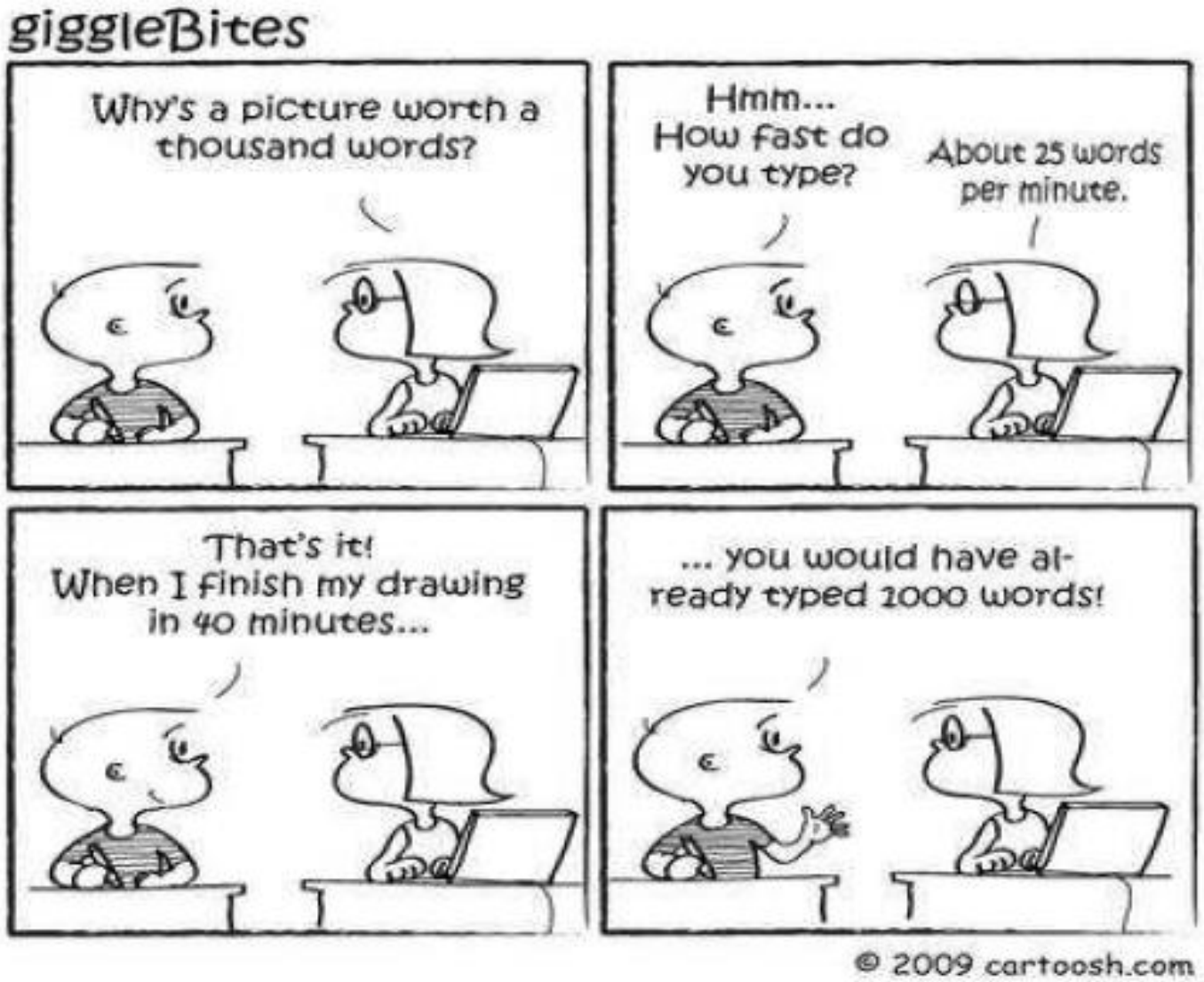
OO Modeling Using UML (Class Models)

Dr. Amr S. Ghoneim

Modelling

- A model is an abstraction of a system.
- Abstraction allows us to ignore unessential details
- Why building models?
  - To reduce complexity
  - To test the system before building it
  - To communicate with the customer
  - To document and visualize your ideas

Modelling


giggleBites
Why's a picture worth a thousand words?
Hmm... How fast do you type?
About 25 words per minute.
That's it! When I finish my drawing in 40 minutes...
... you would have already typed 1000 words!
© 2009 cartoosh.com

Some UML Diagrams

- Functional diagrams
  - Describe the functionality of the system from the user's point of view. It describes the interactions between the user and the system. It includes use case diagrams.
- Static diagrams
  - Describe the static structure of the system: Classes, Objects, attributes, associations.
- Dynamic diagrams:
  - **Interaction diagrams**
    - Describe the interaction between objects of the system
  - State diagrams
    - Describe the temporal or behavioral aspect of an individual object
  - Activity diagrams
    - Describe the dynamic behavior of a system, in particular the workflow.

UML Tools

- Some UML tools can generate code once UML diagram is completed.
- Some UML tools are sketching tools.
- Rational Rose is one of the most popular software for UML creation (IBM).
- Bouml is an open source s/w. It supports python, C++, java.
- Visio is a sketching tool.

Static Diagrams

- *Class diagrams : show the classes and their relations*

- *Object diagrams : show objects and their relations*

- *Package diagrams :shows how the various classes are grouped into packages to simplify complex class diagrams.*

Class Model

- A ***class model*** captures the static structure of the system by characterizing
  - the *classes and objects* in the system,
  - the *relationships* among the objects and
  - the *attributes* and *operations* for each class of objects
- Class models are the *most important* OO models
- In OO systems we build the system around objects not functionality

Objects

- Objects often appear as ***proper nouns*** in the problem description or discussion with the customer.
- Some object correspond to ***real world entities*** (Helwan University, MIT, Omar's car)
- Some objects correspond to ***conceptual entities*** (the formula for solving an equation, binary tree, etc.)
- The choice of objects depends on the analyst's judgment and the problem in hand. There can be *more than one correct representation*.

Class

- An object is an ***instance of*** a class
- A class describes a group of objects with the same
  - Properties (attributes)
  - Behavior (operations)
  - Kinds of relationships

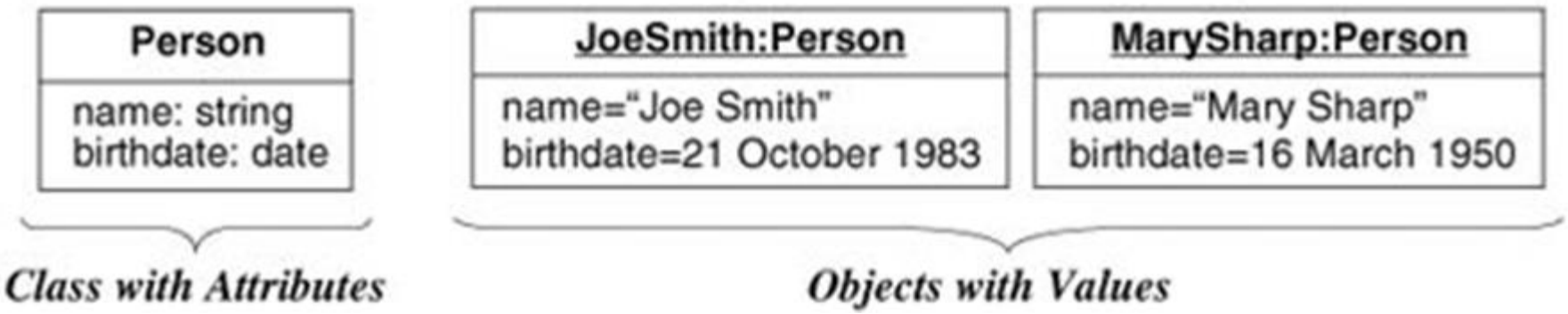
- *Person, Company and Window* are all classes
- Classes often appear as ***common nouns*** and ***noun phrases*** in problem description and discussion with customers or users

Class Model

- Provides a graphical notation for modeling classes and their relationships, thereby describing possible objects
- Class diagram is useful for:
  - *Designing and Implementing the programs*

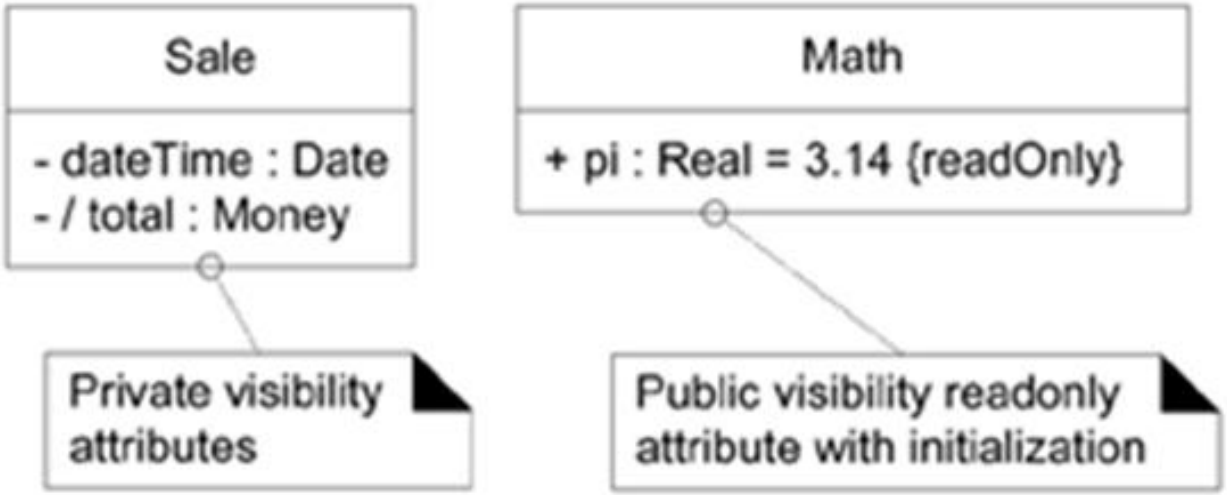
Figure 3.1 A class and objects. Objects and classes are the focus of class modeling.

Values and Attributes



- An ***attribute*** is a named property of class that describes a value held by each object of that class.
- *Attribute name* is unique per class.
- Several classes may have the same attribute name.
- A ***value*** is a piece of data assigned to an attribute.

More on Attributes



Operations and Methods

- An ***operation*** is a function or procedure that may be applied *to* or *by* objects of a class.
- A ***method*** is the implementation of an operation for a class.
- An operation is ***polymorphic*** if it takes different forms in different classes.
- All objects of the same class have the same operations.

Financial Asset	
-type:	int
-age:	float
-currentValue:	float
-. . .	
+getCurrentValue():	int
+printDetails():	void
+	

Summary of Class Notation

- The attribute and operation compartments are optional.
- You may show them or not depending on the level of abstraction you want.
- A missing attribute compartments means that the attributes are not specified yet.
- But empty compartment means that the attributes are specified but there are none.

Summary of Basic Class Notation

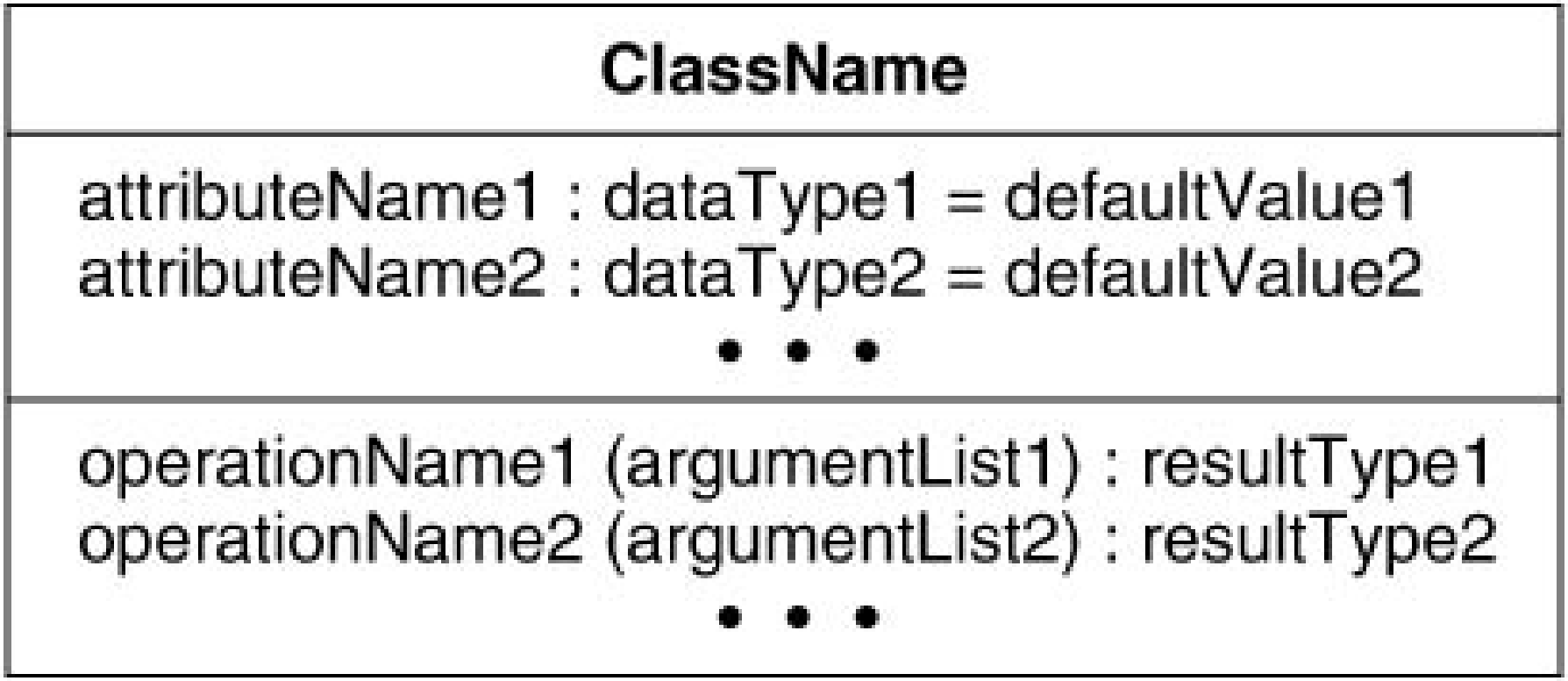


Figure 3.5 Summary of modeling notation for classes. A box represents a class and may have as many as three compartments.

A sample class model

- Model the classes in a system that represents flights. Each city has at least an airport. Airlines operate flights from and to various airports. A flight has a list of passengers, each with a designated seat. Also a flight uses one of the planes owned by the operating airline. Finally a flight is run by a pilot and a co-pilot.

A sample class model

- Model the classes in a system that represents ***flights***. Each ***city*** has at least an ***airport***. ***Airlines*** operate flights from and to various airports. A flight has a ***list*** of ***passengers***, each with a designated ***seat***. Also a flight uses one of the ***planes*** owned by the operating airline. Finally a flight is run by a ***pilot*** and a ***co-pilot***.

A sample class model

- ***Flights*** • ***city*** • ***Airlines***
- ***list*** of ***passengers*** • ***Seat*** • ***planes***
- ***pilot*** and a ***co-pilot***.

City

Airline

Pilot

Plane

Passenger

Airport

Flight

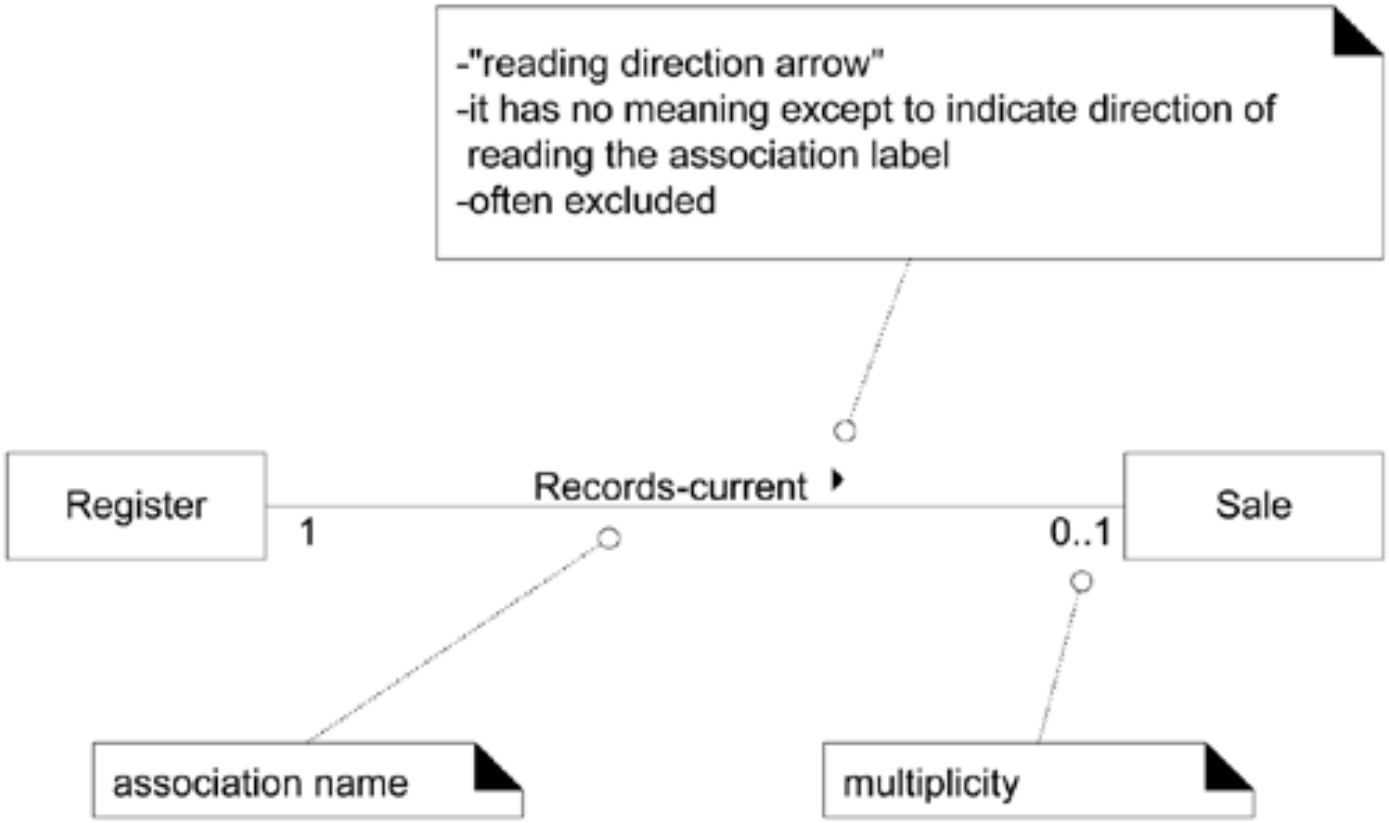
Seat

Associations

- An association is a relationship between classes that indicate some meaningful and interesting relationship.
- It's represented by a line with a name.
- Properly naming associations is important to enhance understanding: use verb phrase.

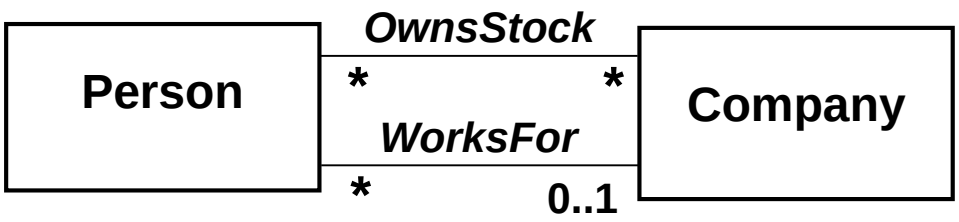
Associations

- An optional "reading direction arrow" indicates the direction to **read** the association name; *it does not indicate direction of visibility or navigation*



Associations

- There may be more than one associations between classes (this is not uncommon).



Associations

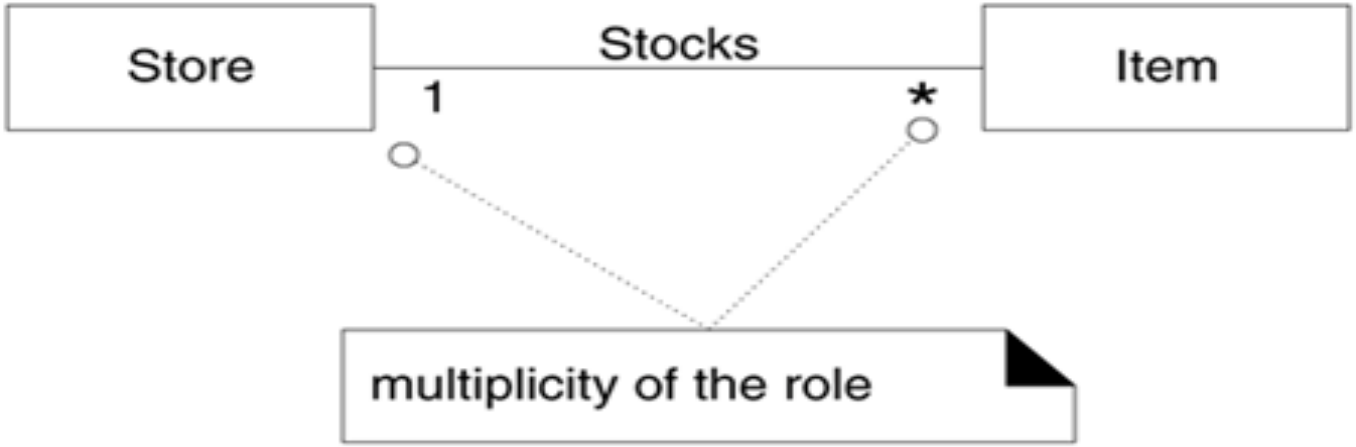
- Associations are usually implemented by a ***reference*** from an object to another.

- Associations are inherently bidirectional. They can be traversed in either direction. A person *WorksFor* a company and a company *Employs* a person.

- Associations could be unidirectional.

Multiplicity

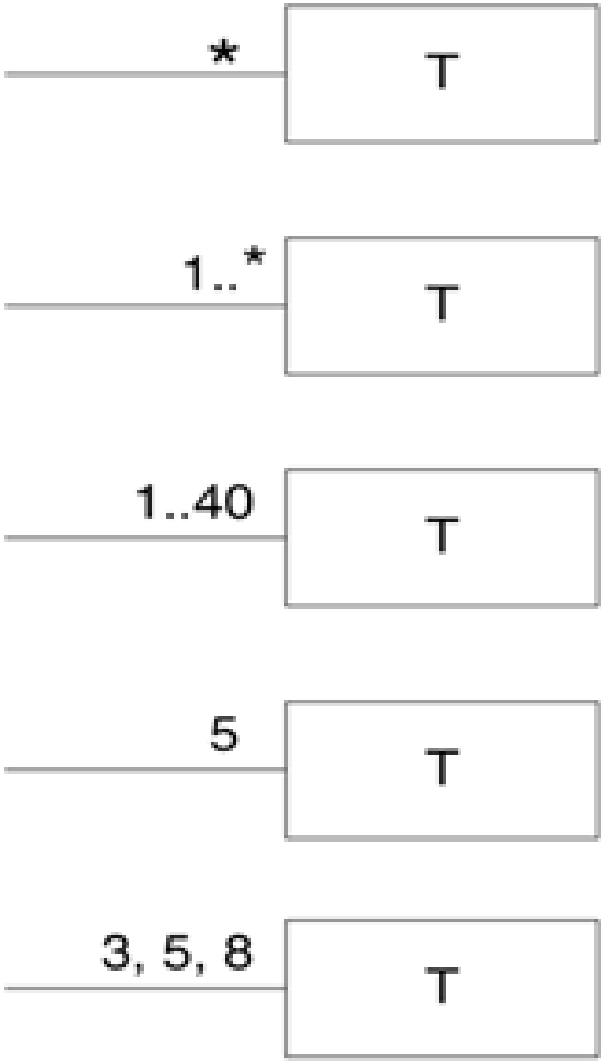
- ***Multiplicity*** specifies the number of instances of one class that may relate to a single instance of an associated class.



Multiplicity

- Multiplicity exposes hidden assumptions in the model
- For example, if a person ***WorksFor*** a company, can he work for more than one company? In other words, is it ***one-to-one*** or ***one-to-many*** association?

Multiplicity Values



zero or more; "many"

one or more

one to 40

exactly 5

exactly 3, 5, or 8

Links and Associations

- A ***link*** is a physical or conceptual connection among objects
  - For example John works for GE company.

- An ***association*** is a relationship between classes and represents group of links.
  - For example, a person *WorksFor* a company.

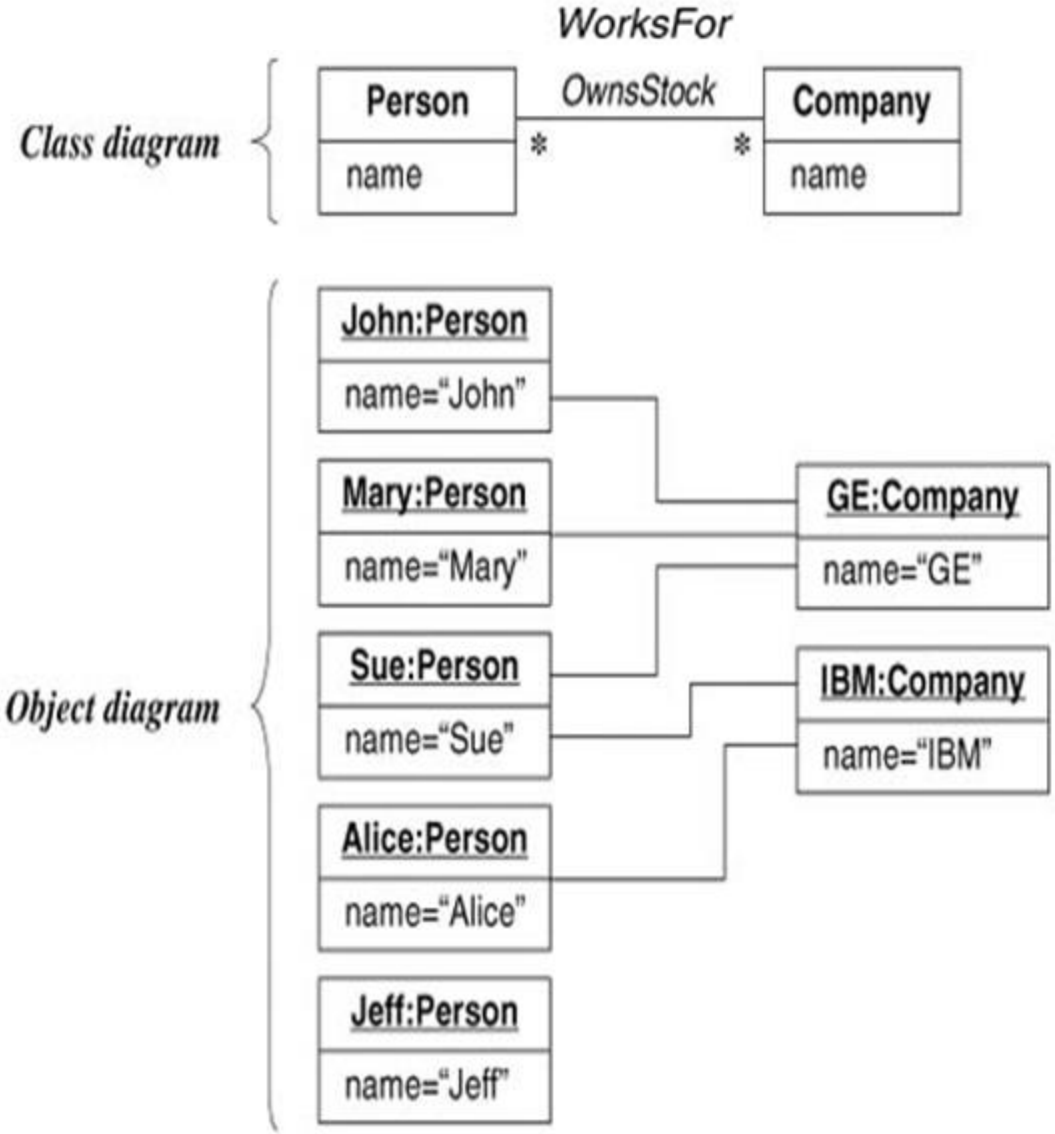


Figure 3.7 Many-to-many association. An association describes a set of potential links in the same way that a class describes a set of potential objects.

Links and Associations

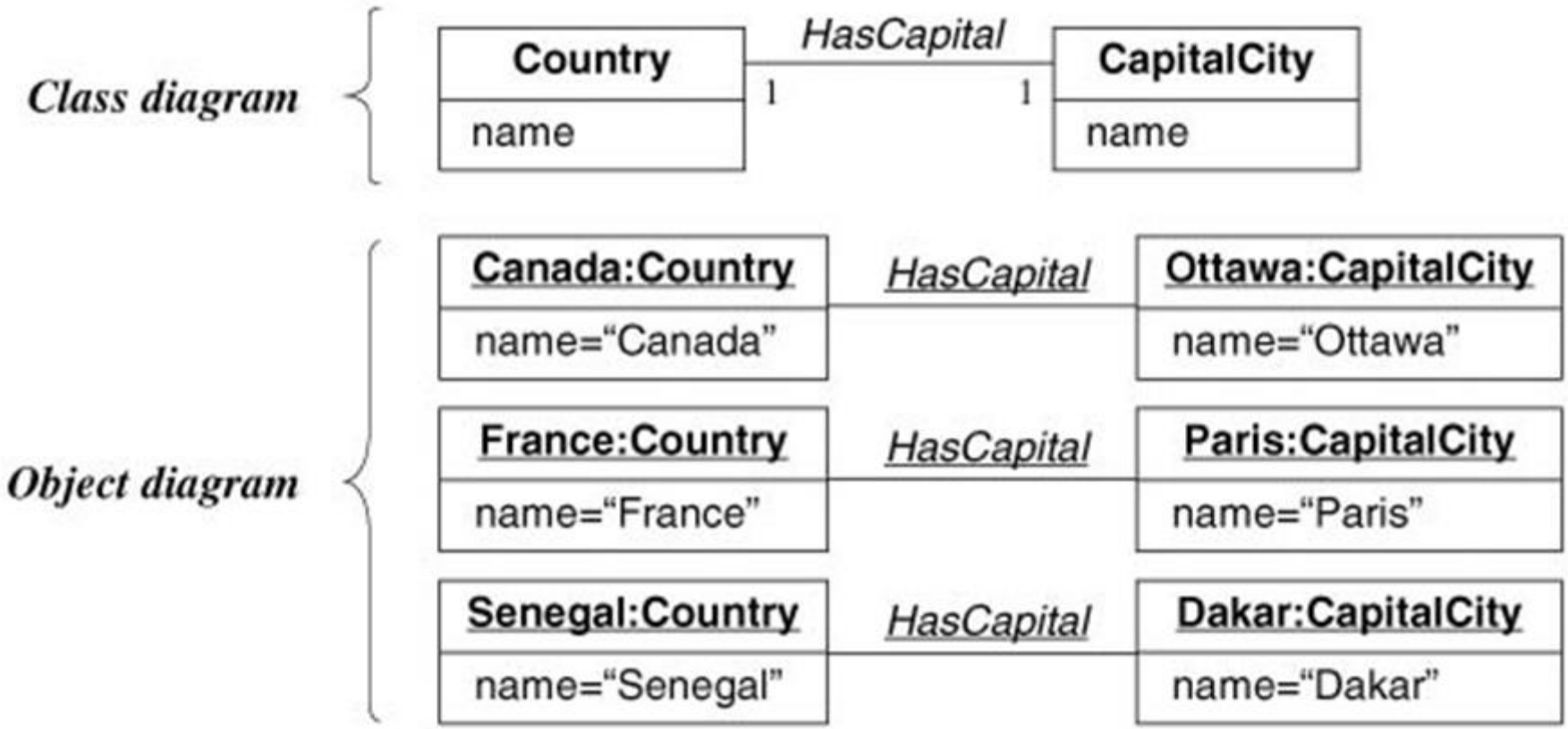


Figure 3.8 One-to-one association. Multiplicity specifies the number of instances of one class that may relate to a single instance of an associated class.

Links and Associations

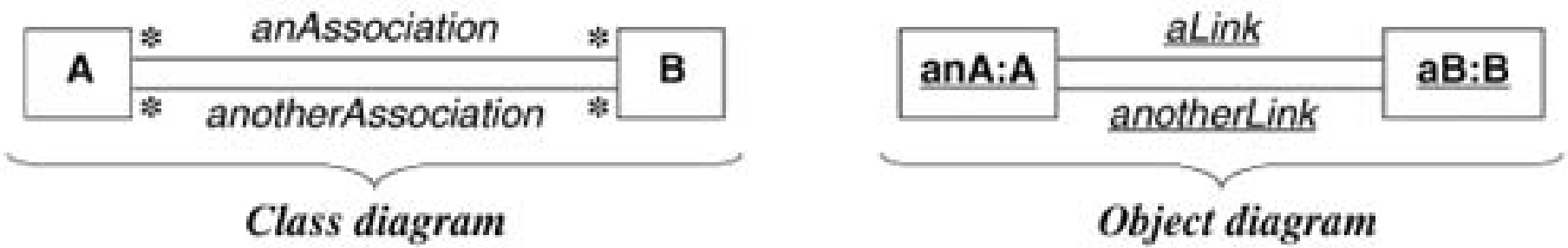


Figure 3.11 Association vs. link. You can use multiple associations to model multiple links between the same objects.

Roles

- Each association End can be labeled by a role.
- This makes understanding associations easier.
- They are especially important for ***Self-associations*** between objects of the same class.

employee	employer
Joe Doe	Simplex
Mary Brown	Simplex
Jean Smith	United Widgets

Figure 3.12 Association end names. Each end of an association can have a name.

Self associations

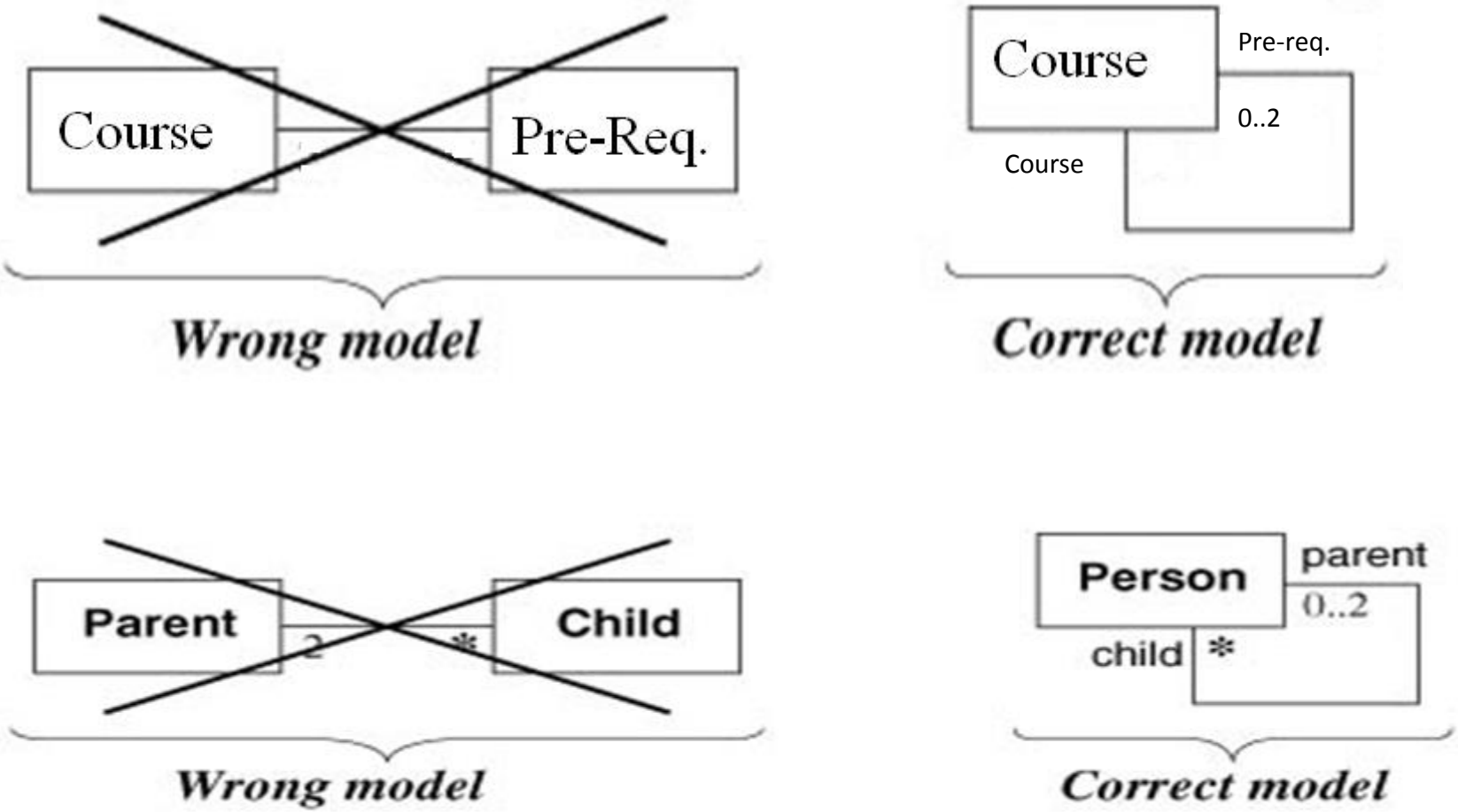


Figure 3.14 Association end names. Use association end names to model multiple references to the same class.