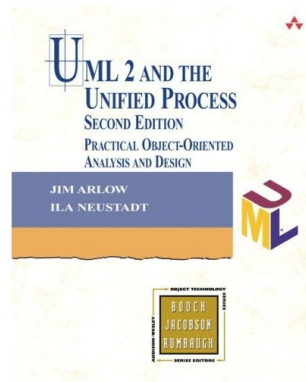


IMPORTANT NOTICE TO STUDENTS

These slides are **NOT** to be used as a replacement for student notes.
These **slides** are sometimes **vague and incomplete on purpose** to spark a class discussion

Introduction to Unified Modelling Language (UML) (part 2- Class diagram, Associations)

CS 446 / 646 ECE452
May 6th, 2011



*Material covered in this lecture is based on various chapters from **UML 2 and the Unified Process- 2nd Edition Practical Object Oriented Analysis & Design***

Objects

What are Objects

- “a discrete entity with a well-defined boundary that encapsulates state and behaviour;
an instance of a class” *UML Reference Manual, 2nd Edition-2004*

Observations

- an instance of a class
- state is maintained in fields
- behaviour is implemented via methods
- every object is uniquely identified

UML Object Notation

Representation

- box with two compartments
- object identifier
 - always underlined
 - anonymous objects
 - $:$ \rightarrow *instanceOf*
- attribute compartment
 - may omit some or all fields

object: class

accountNumber : String = "123"
owner : String = "Jim Andrew"
balance : double = 300.00

:class

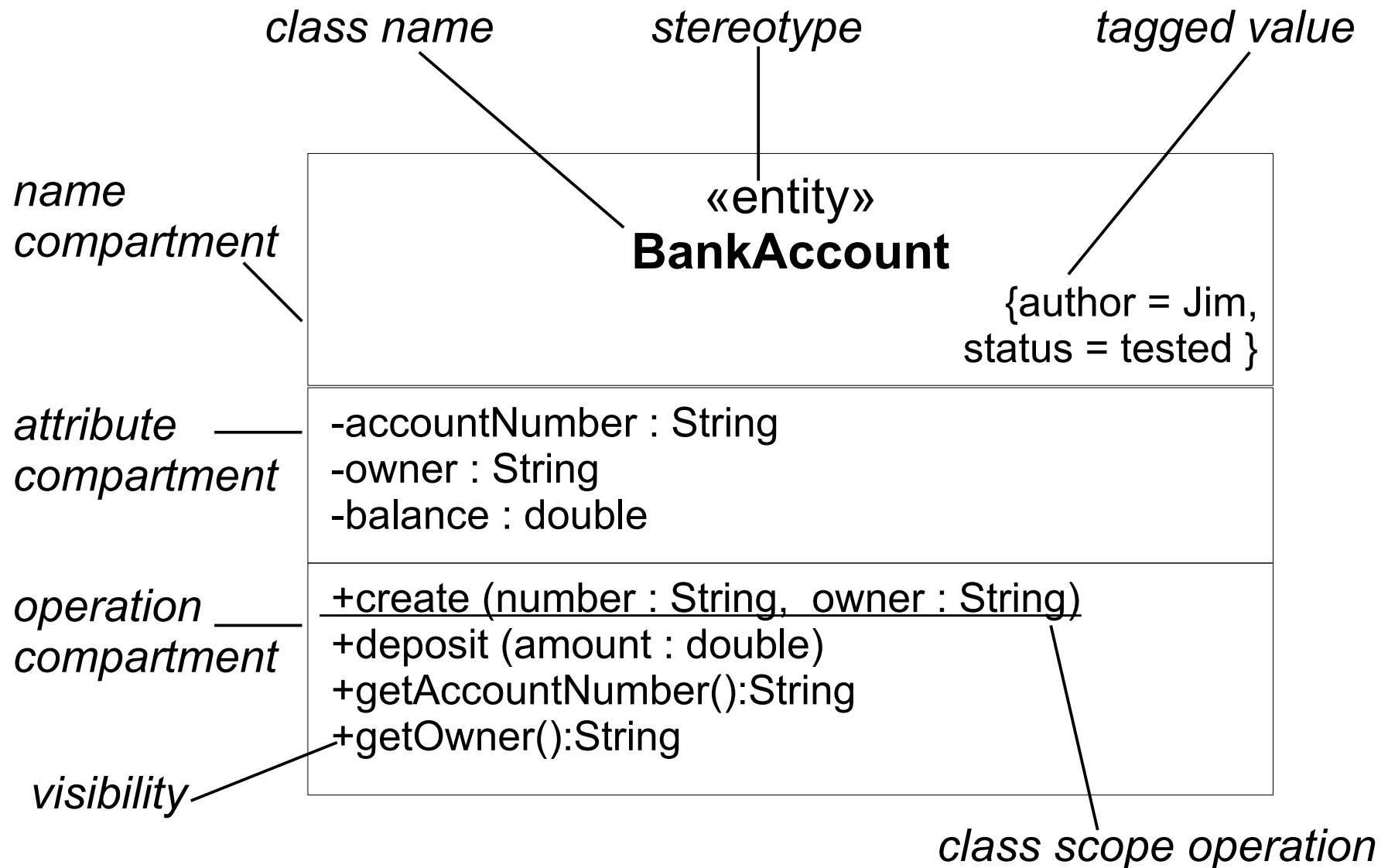
accountNumber : String = "123"
owner : String = "Jim Andrew"
balance : double = 300.00

Classes

What are Classes

- “the descriptor for a set of objects that share the same attributes, operations, methods, relationships, and behaviour” *UML Reference Manual, 2nd Edition-2004*
- template for objects with similar features

UML Class Notation



UML Class Notation

«exercise»
StudentToDo

+ reviewAndLearnMissingUMLClassNotations()

How Much to Show?

Scenarios

- *scenario 1*: inter-class relationship
- *scenario 2*: behaviour of a class or classes
- *scenario 3*: entity classes representing data entities

How Much to Show?

Scenarios

- *scenario 1*: inter-class relationship
 - class boxes are probably enough with name compartment only
- *scenario 2*: behaviour of a class or classes
 - class boxes with name & operations compartment
- *scenario 3*: entity classes representing data entities
 - ask the class?

Analysis Classes

What is an Analysis Class?

- represents an important concept of the problem domain
- maps on to real-world concepts
- an attempt to capture the big picture

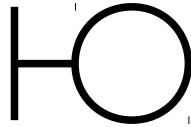
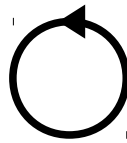
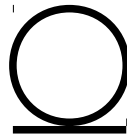
BankAccount
accountNumber owner balance
deposit () getAccountNumber() getOwner()

Q: what about savings
account sub-class of
BankAccount?

SavingAccount

Analysis Classes

Important Analysis Classes

- «boundary» 
 - a class that mediates interaction between the system (or subsystem) and its environment
- «control» 
 - a class that encapsulates use case specific behaviour
- «entity» 
 - a class that is used to model persistence information

Analysis Classes

Observations

- name reflects the intent of abstraction
- focus clearly defined/understood
- small & well defined set of responsibilities
- high cohesion & low coupling

Q: responsibility, cohesion & coupling?

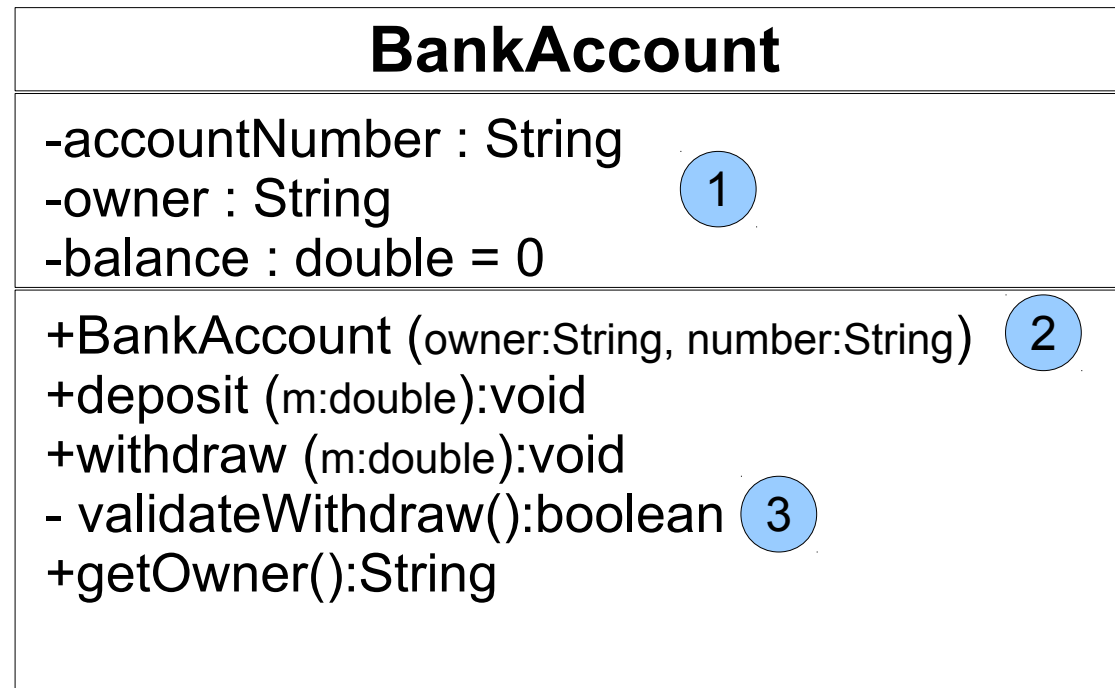
Q: can we use analysis classes for implementation?

BankAccount
accountNumber owner balance
deposit () withdraw() calculateInterest() clientPromotion()

Design Classes

What is a Design Class?

- Integrates the problem domain & solution domain
 - problem domain → refined analysis classes
 - solution domain → implementation specific classes
- are complete, ready to be implemented



Design Classes

Concerns

- responsibility
 - the public methods imply a contract
 - code evolution should be considered
- inheritance
 - major concern of design classes (why?)

Inheritance Under Review

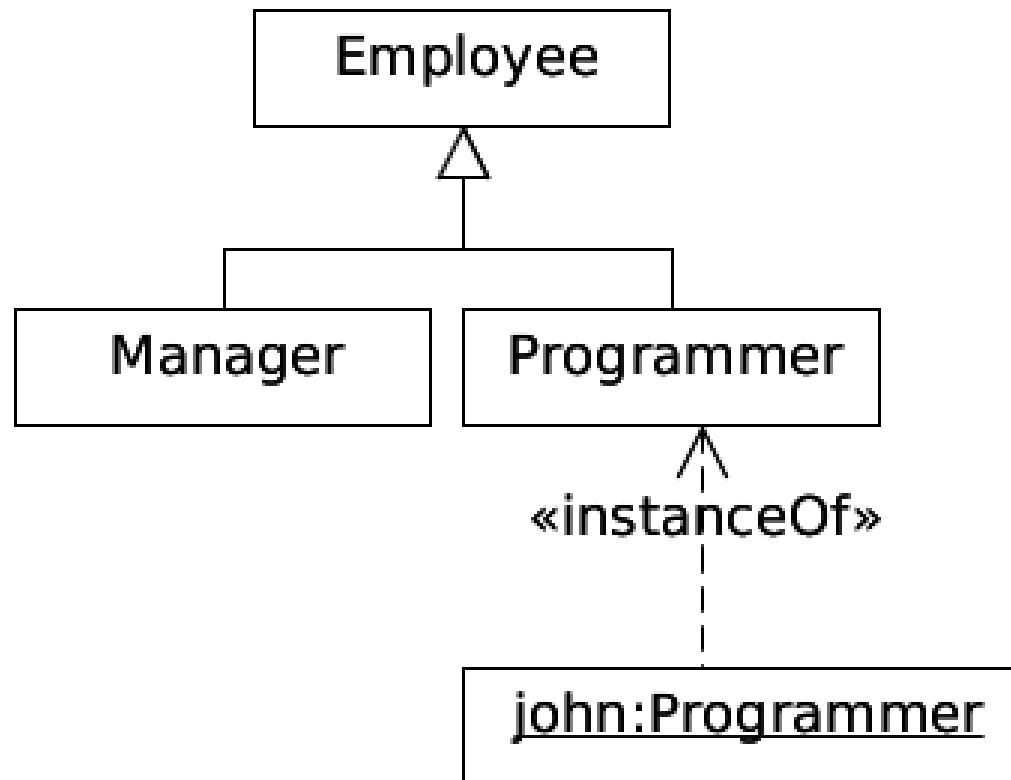
Concerns

- the strongest form of coupling between classes
- “fragile base class” problem
 - changes in the base class impact subclasses
- inflexible to change
 - fixed at runtime

Example: Ambitious John

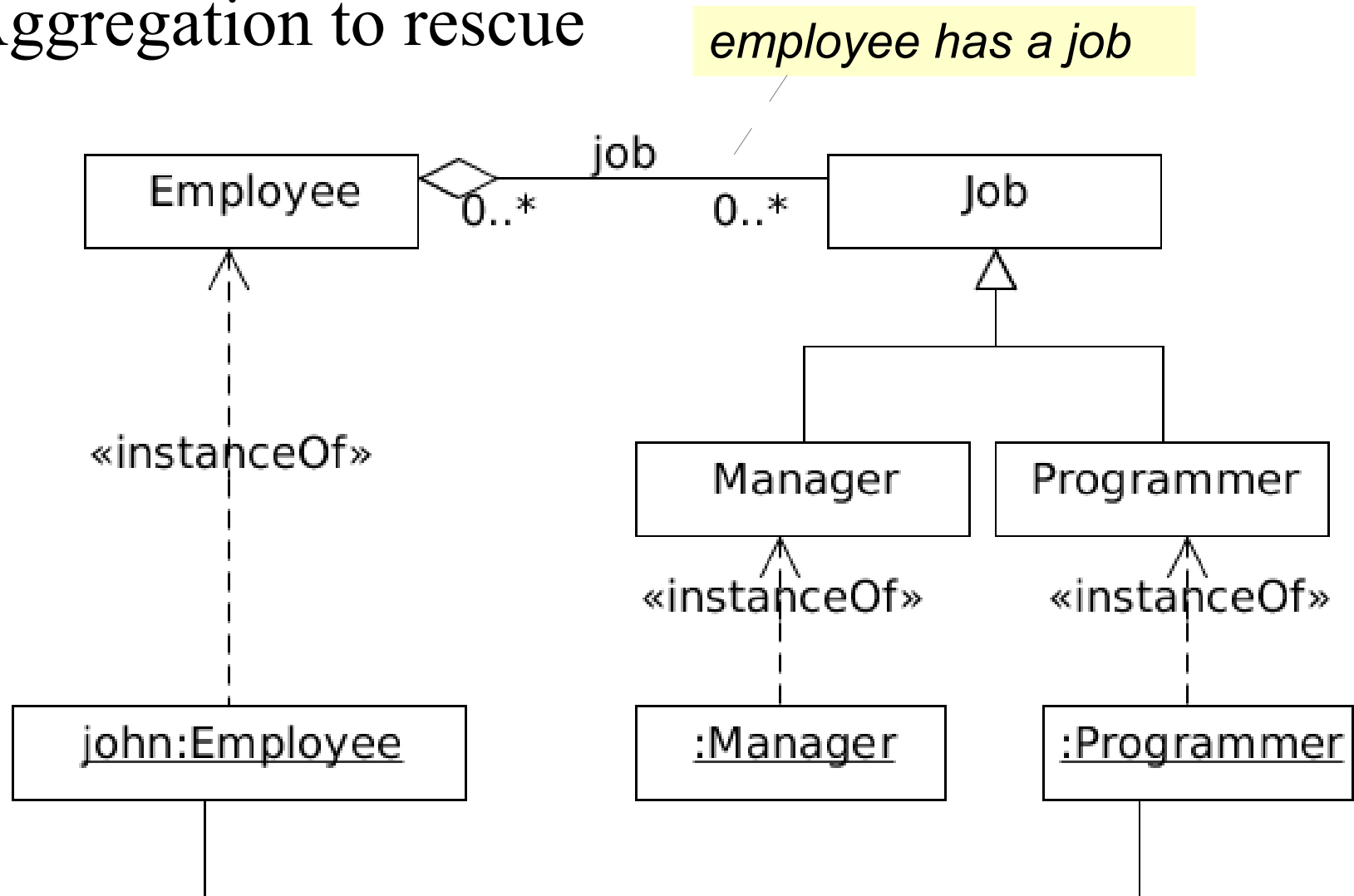
How to promote **john** to be a manager?

- `john:Programmer` vs. `john:Manager`



Example: Ambitious John

Aggregation to rescue



Inheritance or Interface

Inheritance

- interface & implementation
- concerns with reuse

Interface

- interface only
- concerns with functionality contract
 - “*design by contract*”
- concerns with reuse (**what?**)

Further Considerations

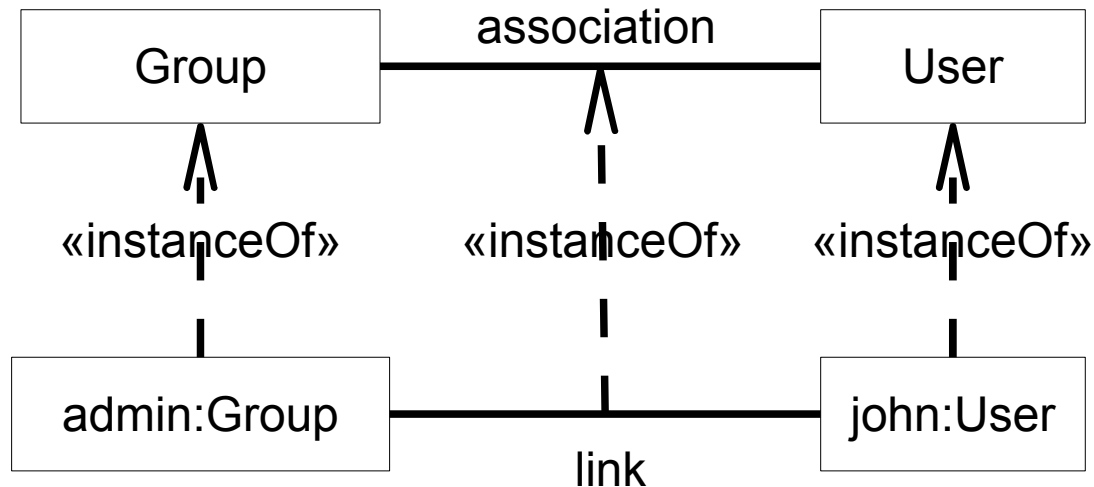
Student Self-Study

- Templates
- Nested classes
- Multiple Inheritance

Association Relationship

What is an Association?

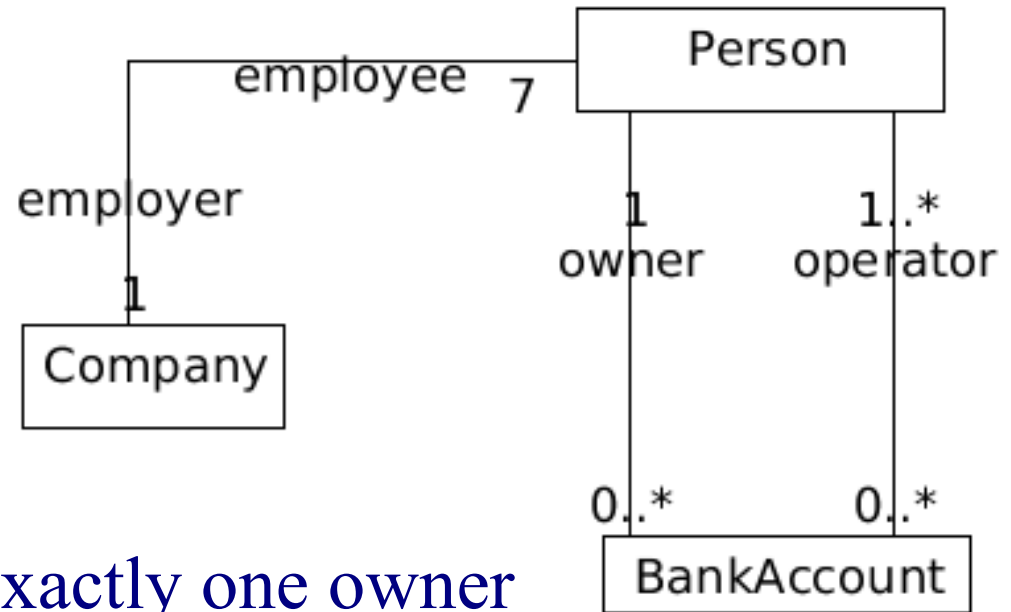
- connection between two things (usually classes)
- *“for a link between two object, there must be an association between the class of those objects”*
 - link is an instance of an association
 - link can be represented by dependency relationship



Example

Decipher this

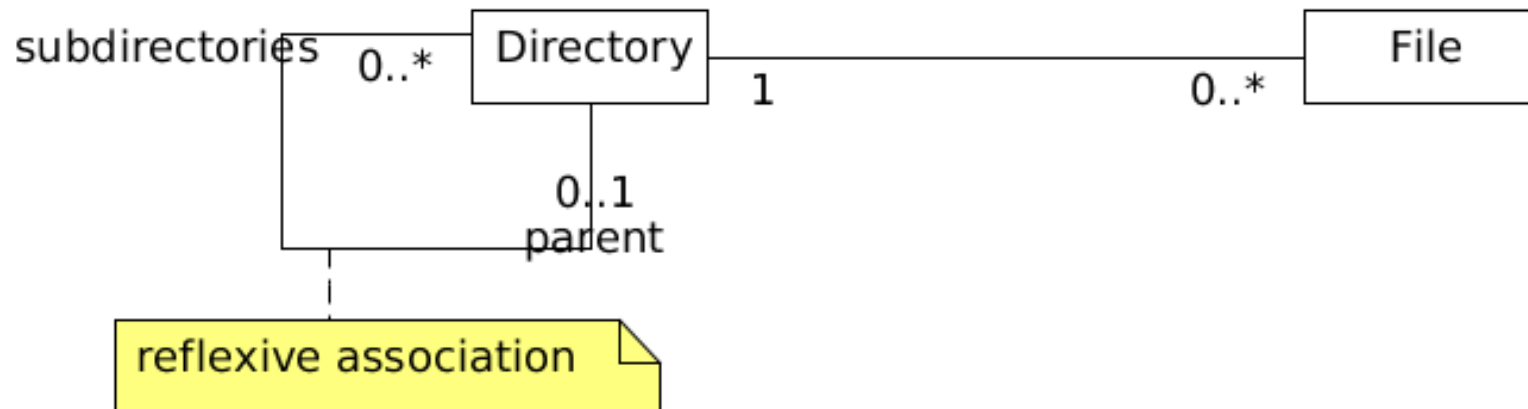
- a Company can have exactly seven employees
- a Person can be employed by exactly one Company
- a BankAccount can have exactly one owner
- a Person may have zero to many BankAccounts
- a BankAccount can have one or many operators
- a Person may operate zero to many BankAccounts



Reflexive Association

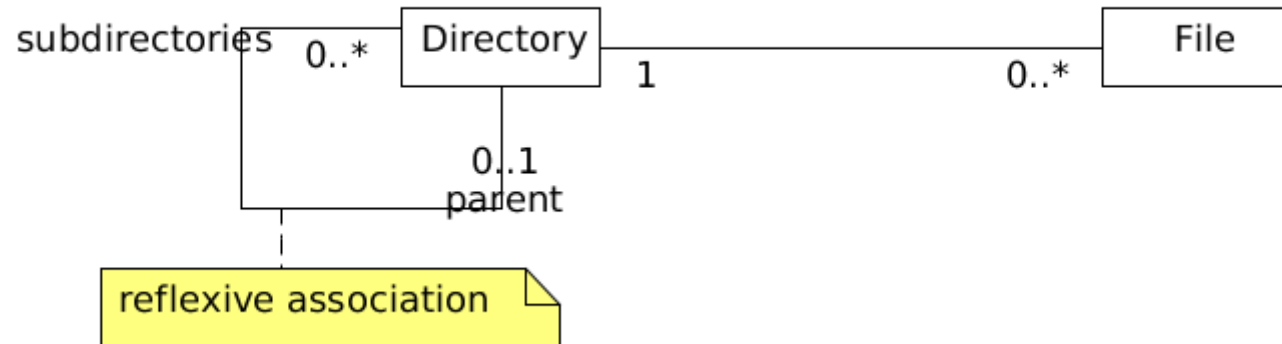
Definition

- A class has an association with itself

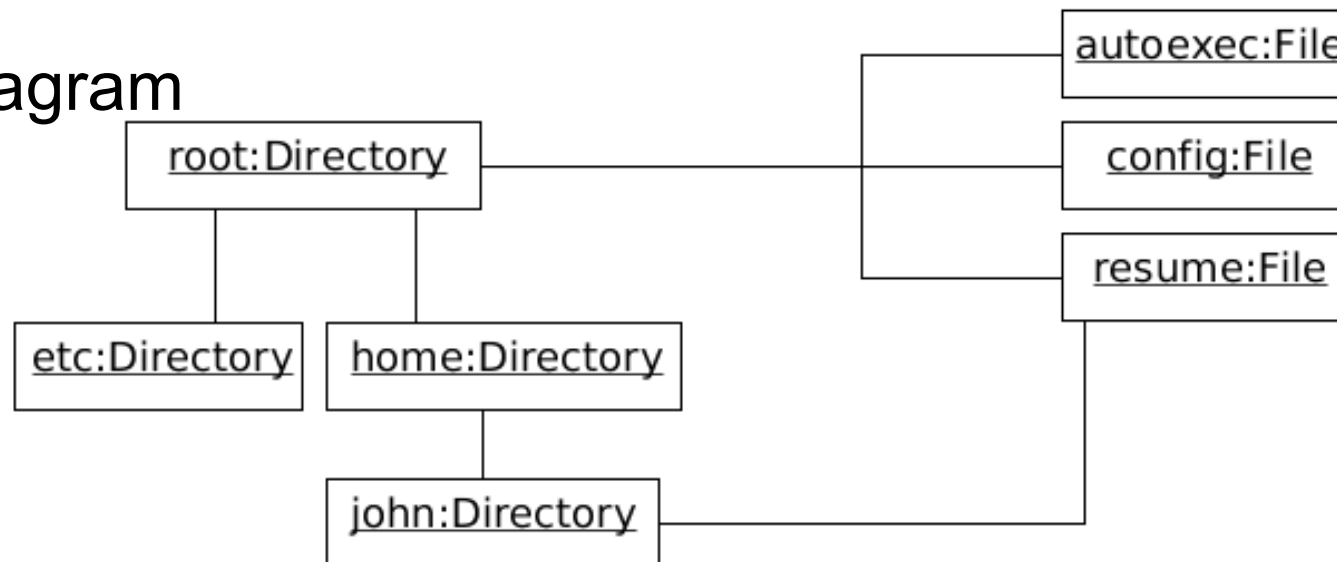


Reflexive Association

Class Diagram



Object Diagram



Navigation

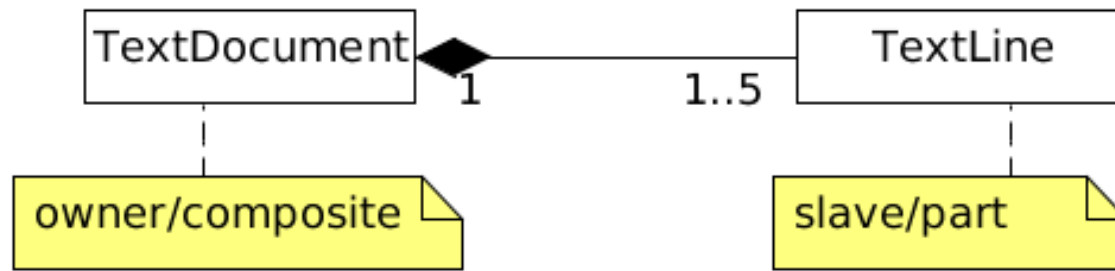
Review the following

- navigation notation & rules
- uni-direction & bi-direction associations

Types of Association

Composition

- strong whole/part (owner/slave) relationship between classes

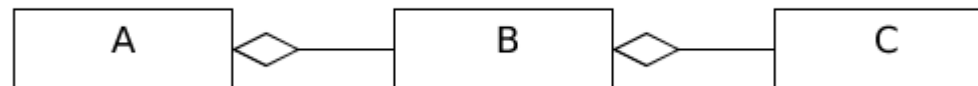
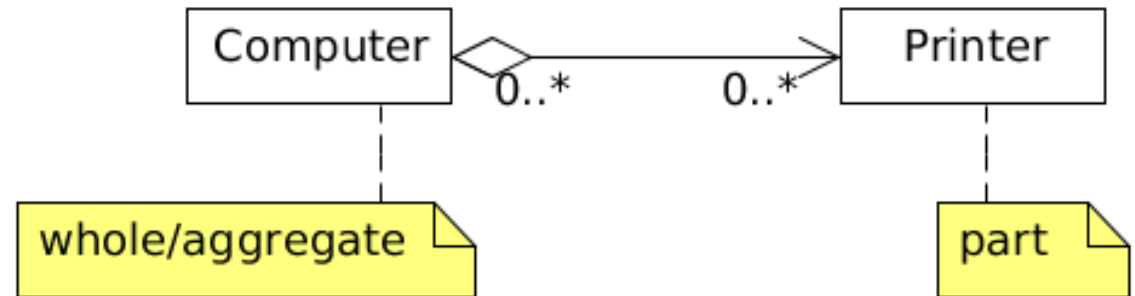


- can we have more than “1” parent?
- what happens when the parent is destroyed?

Types of Association

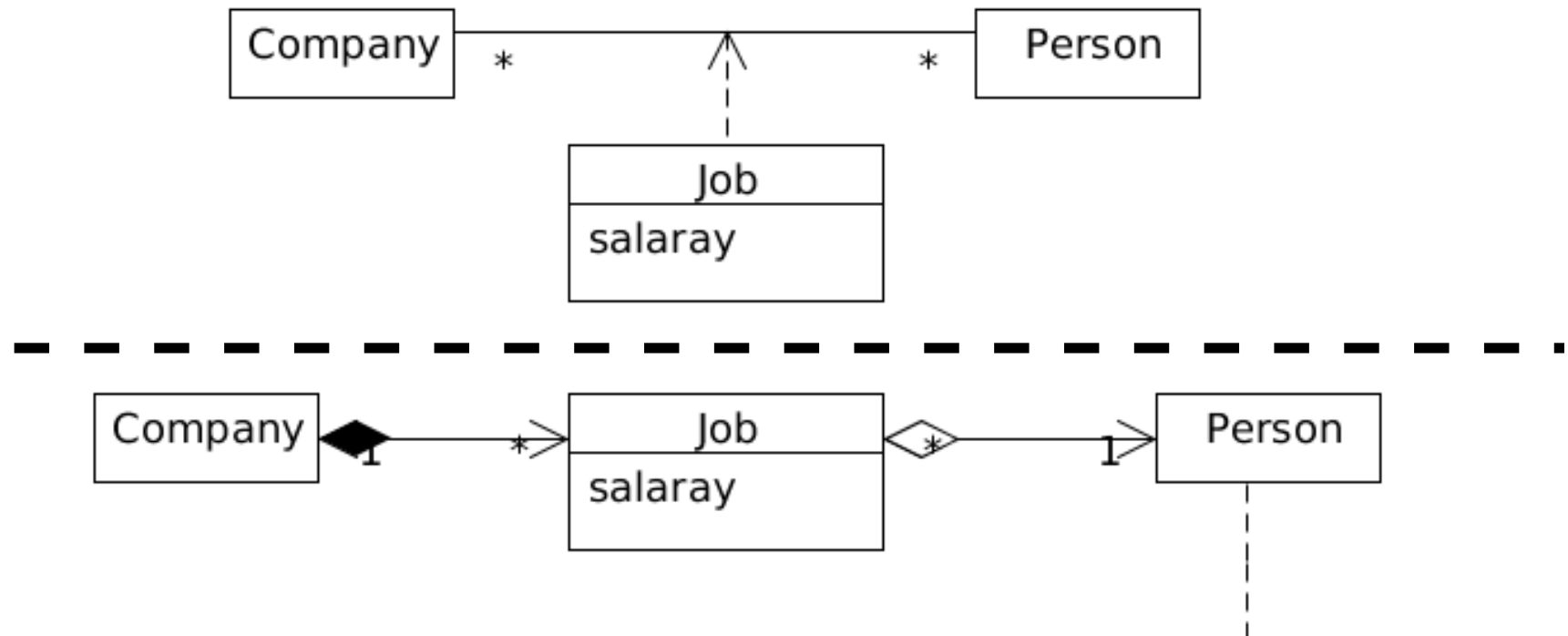
Aggregation

- weaker whole/part relationship between classes
 - classes can exist on their own
- observations
 - a computer may be attached to zero or more printers
 - many computers can use a printer
 - printer may exist even if there are no computers
 - aggregation is transitive

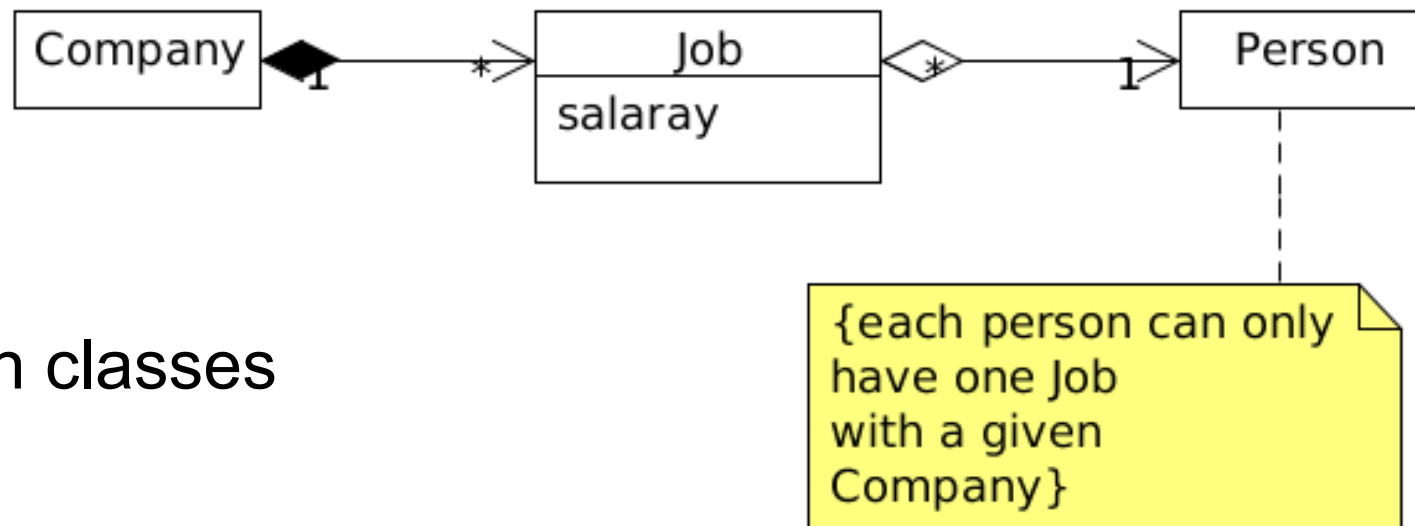


Association Classes

Analysis classes



Design classes



Association Classes

Observations

- association classes are purely analysis classes
- convert association class into normal design class
 - use the various relationships to clarify
 - association (aggregation, composition, dependency)
 - additional constraints to refine the model
 - navigation, multiplicity

Student Study

Compulsory Reading

- Sequence diagram
- State diagram
- Activity diagram