Karlstad University
Department of Information Systems

Adapted for a textbook by Blaha M. and Rumbaugh J.

Object Oriented Modeling and Design

Pearson Prentice Hall, 2005

Development Process: Implementation and Testing

Remigijus GUSTAS

Phone: +46-54 700 17 65

E-mail: Remigijus.Gustas@kau.se

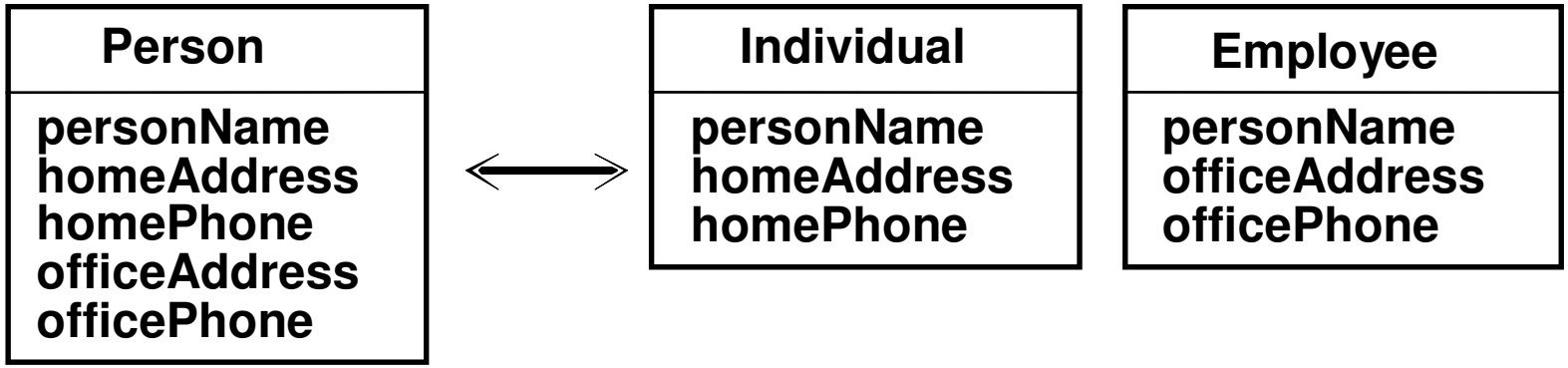
http://www.cs.kau.se/~gustas/

Implementation

- ✓ Implementation should be straightforward, almost mechanical, because all the difficult decisions are made during design.
- ✓ Implementation is final development phase that addresses specifics of programming languages.
- ✓ Small details can be added while writing code, but each one should affect only a small part of the program.
- ✓ Programmers can finally capitalize in implementation from good quality of analysis and design.

Partitioning a class

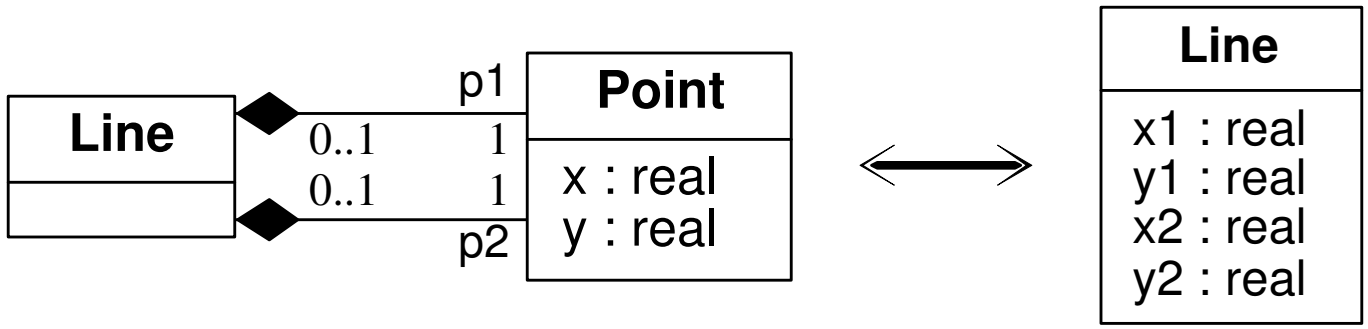
Sometimes it is helpful to partition of merge classes, dependent on attribute amounts.



Note: Relations between the partitioned classes can be introduced. Good quality of analysis and design should resolve some of these issues.

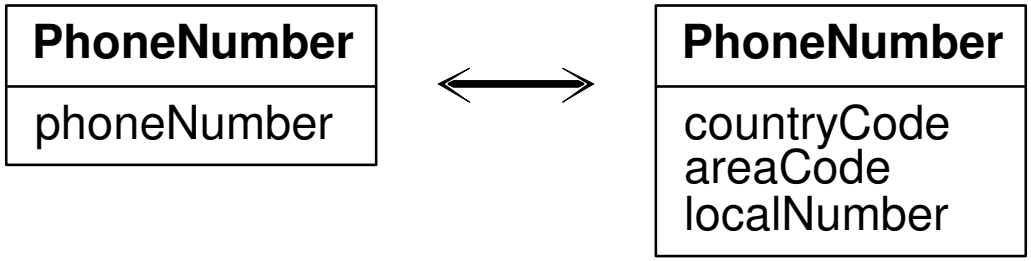
Merge classes

Converse to partitioning is to merge classes.



Note: Neither of representations is inherently superior, both are correct. Effects of introducing generalization and association must be considered.

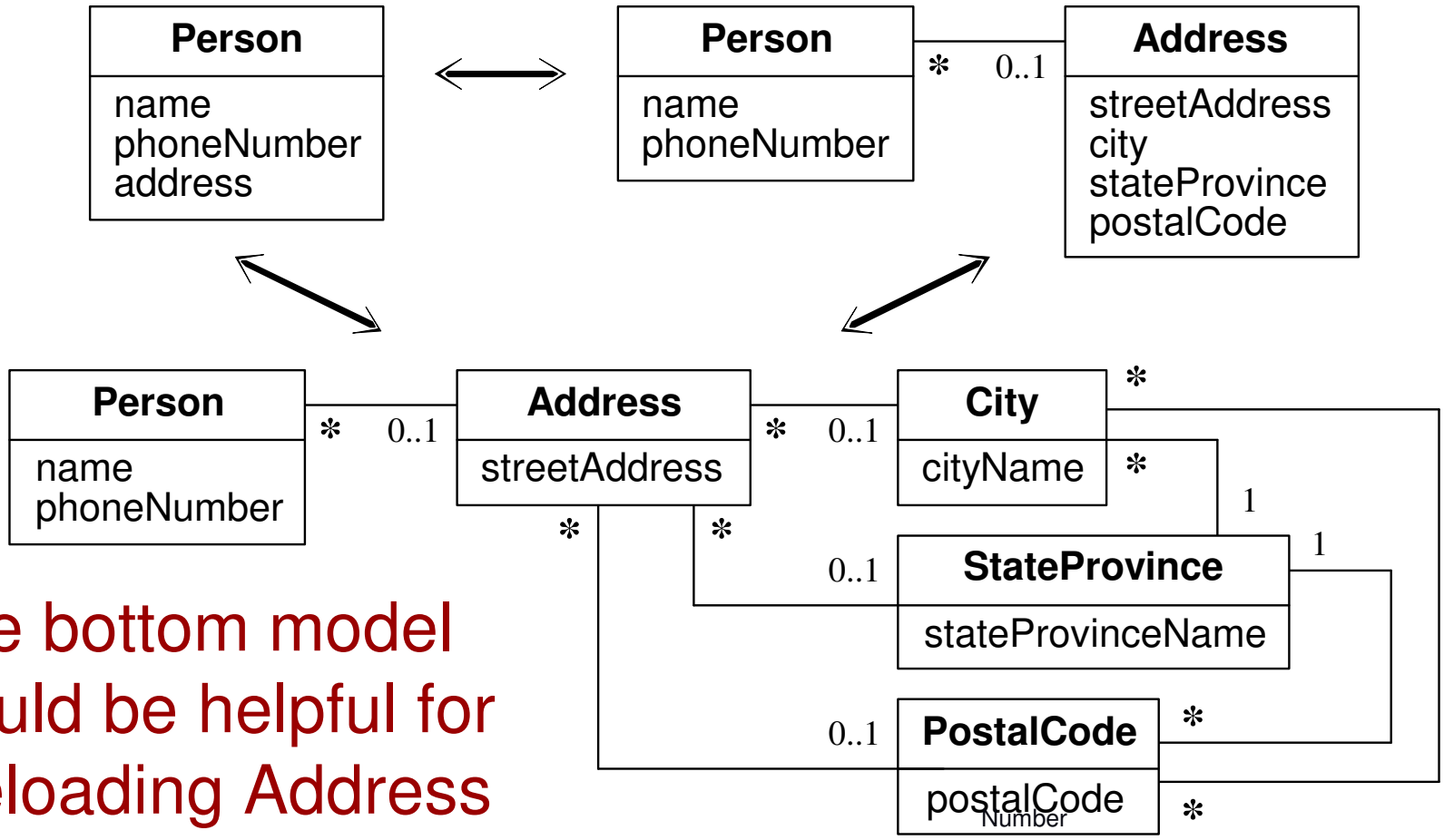
Decompose attributes or merge attributes



Note: Attributes are atomic (restriction of UML class diagrams).

PhoneNumber attribute is expressed in two ways.

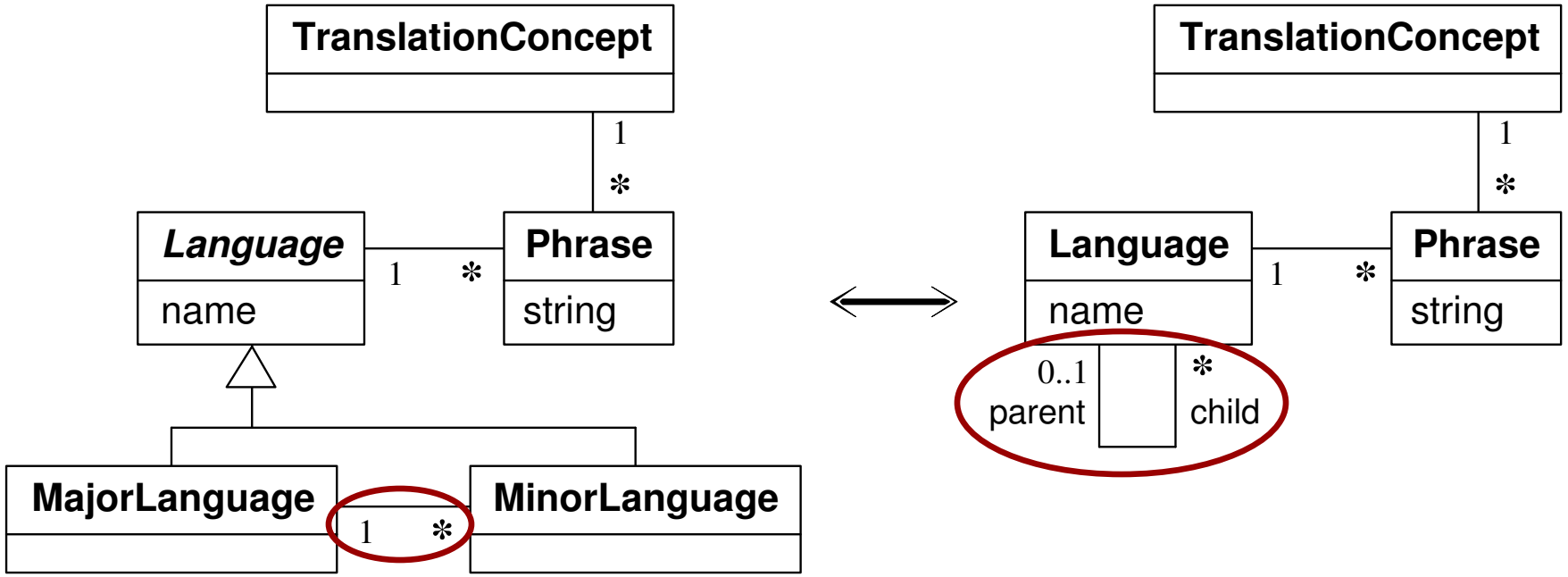
Promote an attribute to a class or demote a class



The bottom model would be helpful for preloading Address data.

Merging Generalization Hierarchies

Sometimes it is helpful to reconsider generalizations.



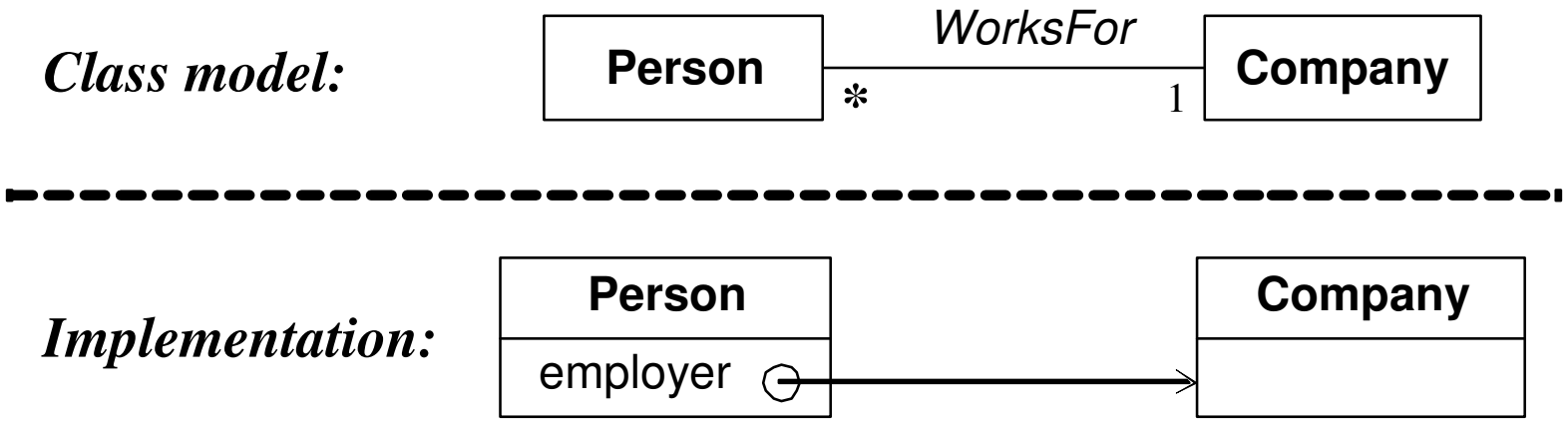
Semantics of association is preserved after merging.

Realizing Associations

- ✓ In system analysis and design phase we assume that associations are **bidirectional**.
- ✓ Software designer may optimize some associations in production phase.
- ✓ Associations can be implemented as:
  - One-way associations,
  - Two-way associations with pointers,
  - Two-way associations as an object.

One-way Associations

- ✓ Traversed only in one direction.
- ✓ Implemented by using pointer (attribute that contains an object reference).
- ✓ A simple (multiplicity 'one') or a set of pointers (multiplicity 'many').



Two-way Associations

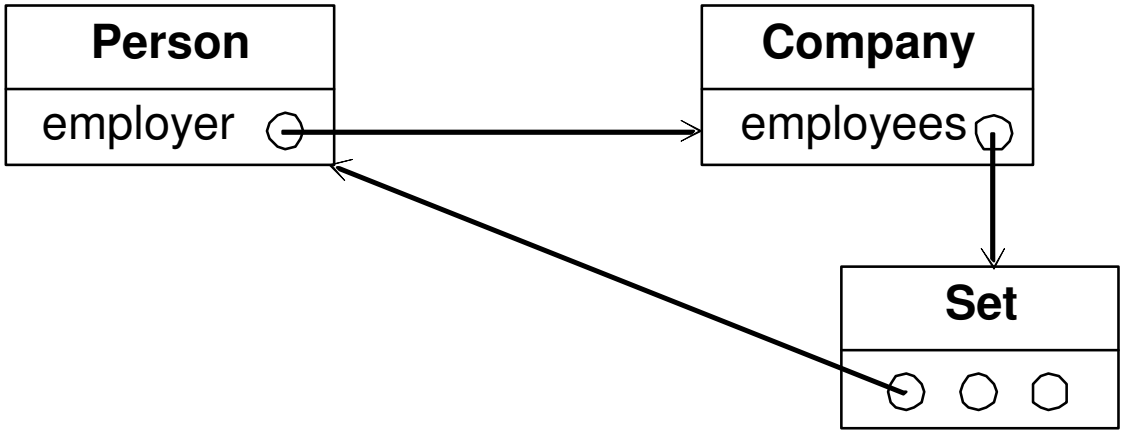
Can be implemented as:

- ✓ One-way Associations
  - (-) Slower access where there is no pointer
  - (+) Fast to add and to remove associations
  - (+) Space efficient
  - All instances would accessed and selected via Person objects (Query for previous diagram: 'Who **WorksFor** IBM?')

- ✓ Two-way Associations with Pointers
  - (+) Fast access in both directions
  - (-) Extra time is required to maintain referential integrity
  - (-) Extra space is required for pointers in both directions

Two-way Associations with Pointers

- ✓ This approach is useful if accesses outnumber updates

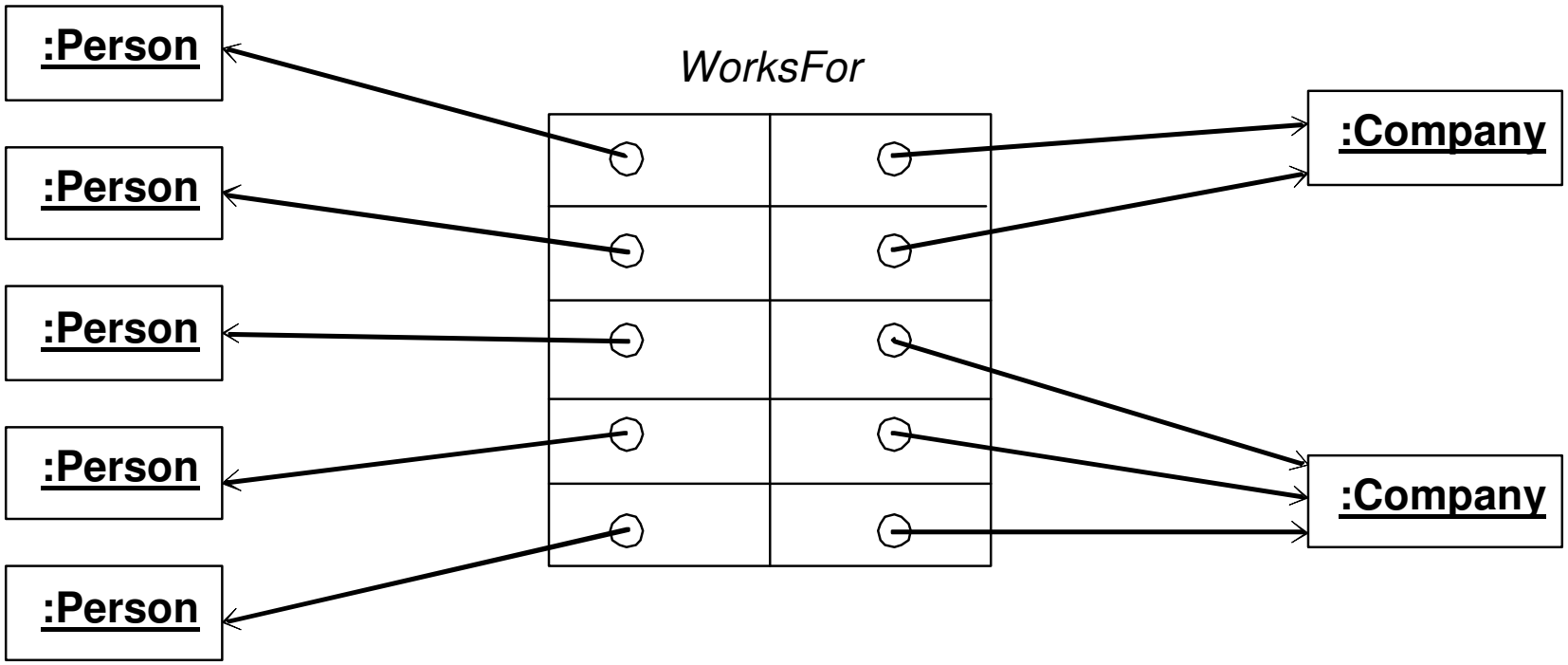


Dual pointers enable fast access in either direction, but introduce redundancy, complicating maintenance.

Two-way Associations as an Object

Most general approach, but requires some programming skill.

- Access is slightly slower than with pointers, but implementation is independent of either class.



Testing

- ✓ Careful analysis and design will **reduce errors** in software and need less testing.
- ✓ **Testing** is a quality assurance mechanism for catching residual errors.
- ✓ Testing provides an independent measure of the software quality (number of bugs).
  - Records of **bugs** and **customer complaints**.
- ✓ Testing is necessary at every development step:
  - The domain model against user requirements,
  - The system architecture is tested during design,
  - The actual code is tested during implementation.

Types of Testing

- ✓ Unit testing (your own classes and methods).
  - Developers should try to cover all paths and cases, by using special values of arguments.
  - Preconditions, postconditions and invariants can be used to trap errors.
- ✓ Integration testing (how other classes and methods fit together).
  - Its is recommended formal reviews, where developers present their work and receive comments.
- ✓ System testing
  - **Alpha testing:** A separate team should carry out system testing.
  - **Beta testing:** Once alpha testing is complete, customers perform beta tests.