Implementation Modeling

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.

Overview of Implementation

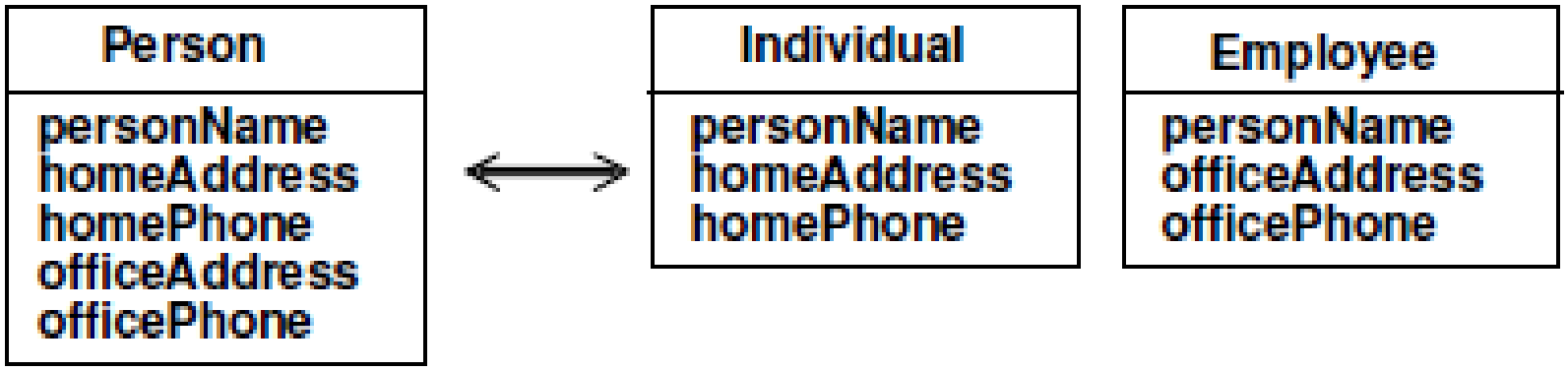
- Fine-tuning Classes
- Fine-tuning Generalizations
- Realize associations
- Prepare for testing

Fine-tuning Classes

- Partition a class
- Merge classes
- Partition/merge attributes
- Promote an attribute/demote a class

Partitioning a class

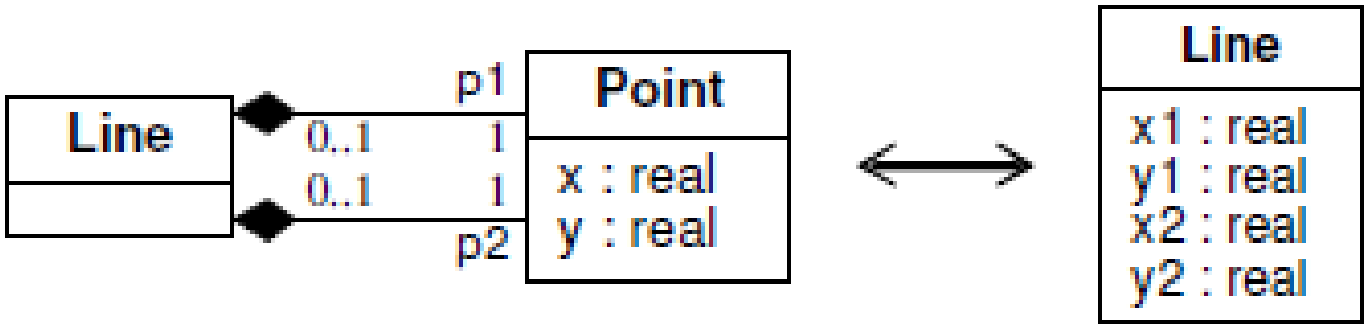
Sometimes it is helpful to partition of merge classes, dependent on attribute amounts.



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

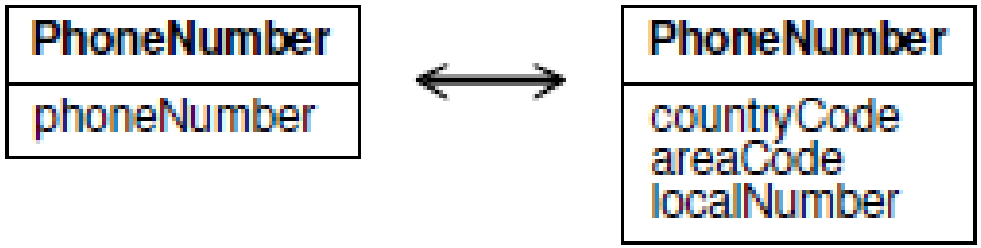
Merge classes

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.

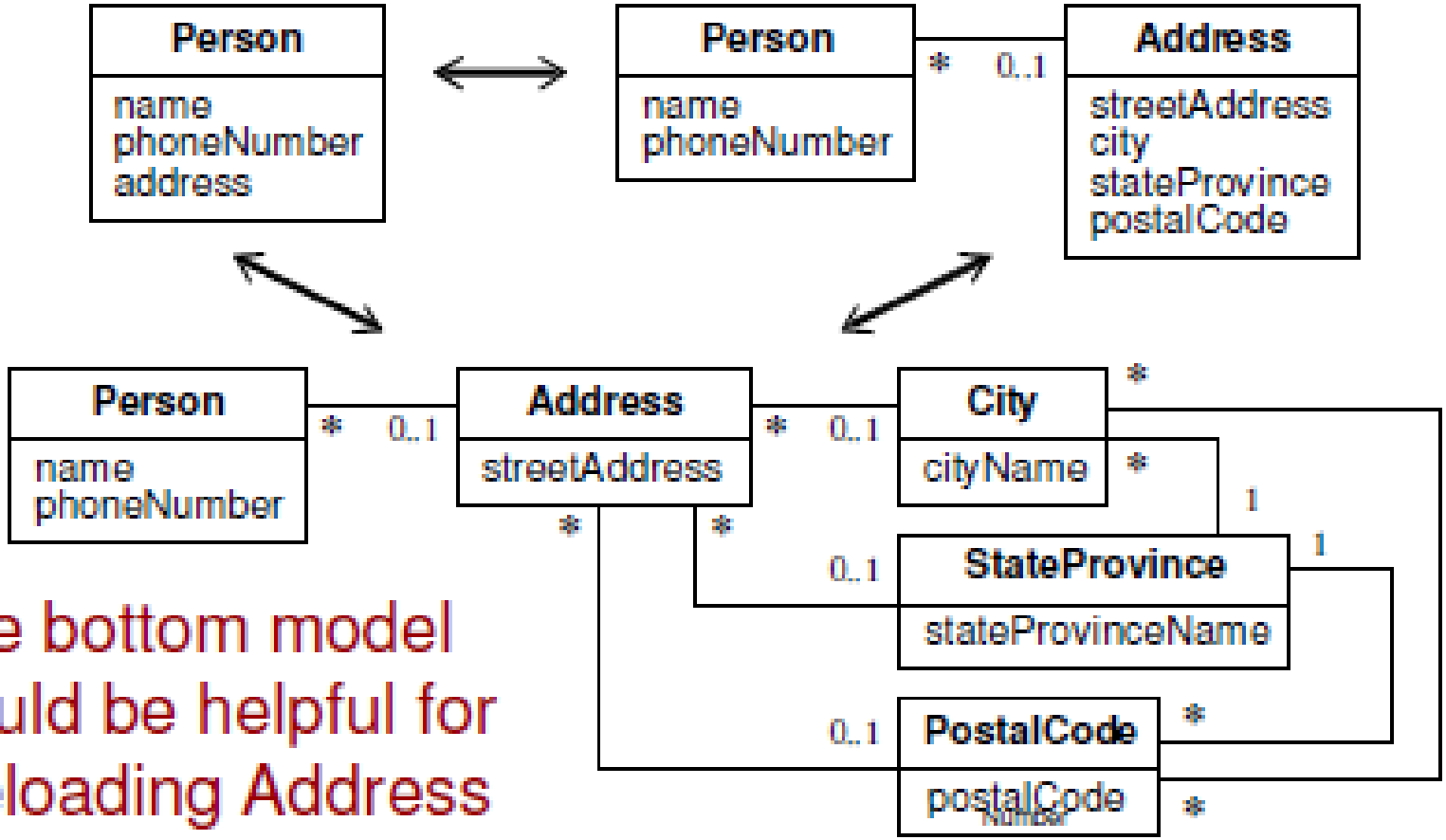
Decompose attributes or merge attributes



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.

Fine-Tuning Generalizations

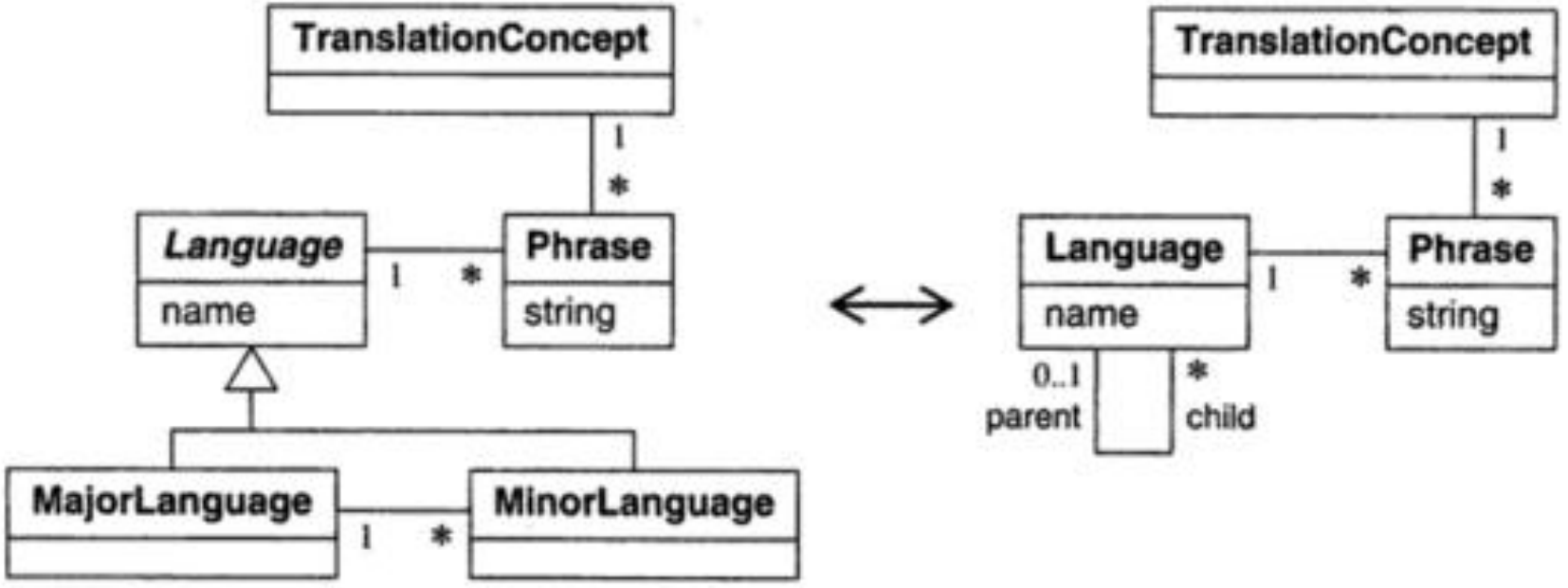


Figure 17.5 Removing / adding generalization. Sometimes it can simplify implementation to remove or add a generalization.

One-way Association

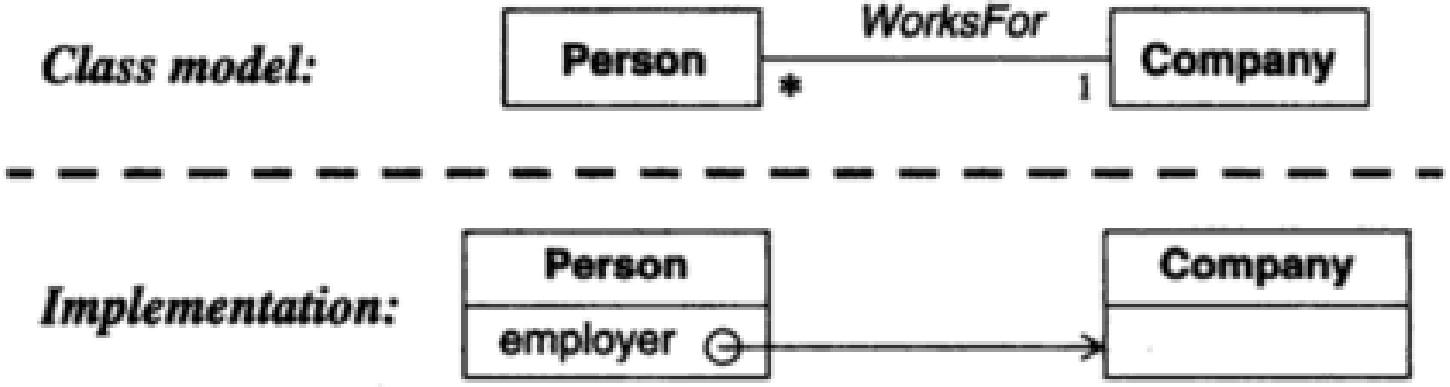


Figure 17.7 Implementing a one-way association with pointers. If an association is traversed only in one direction, you can implement it as a pointer.

Two-way Association

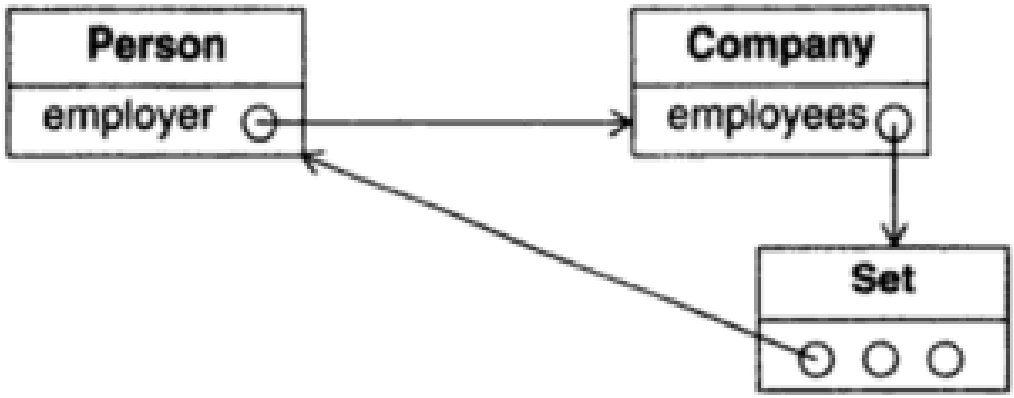


Figure 17.8 Implementing a two-way association with pointers. Dual pointers enable fast traversal of an association in either direction, but introduce redundancy, complicating maintenance.

Association as an Object

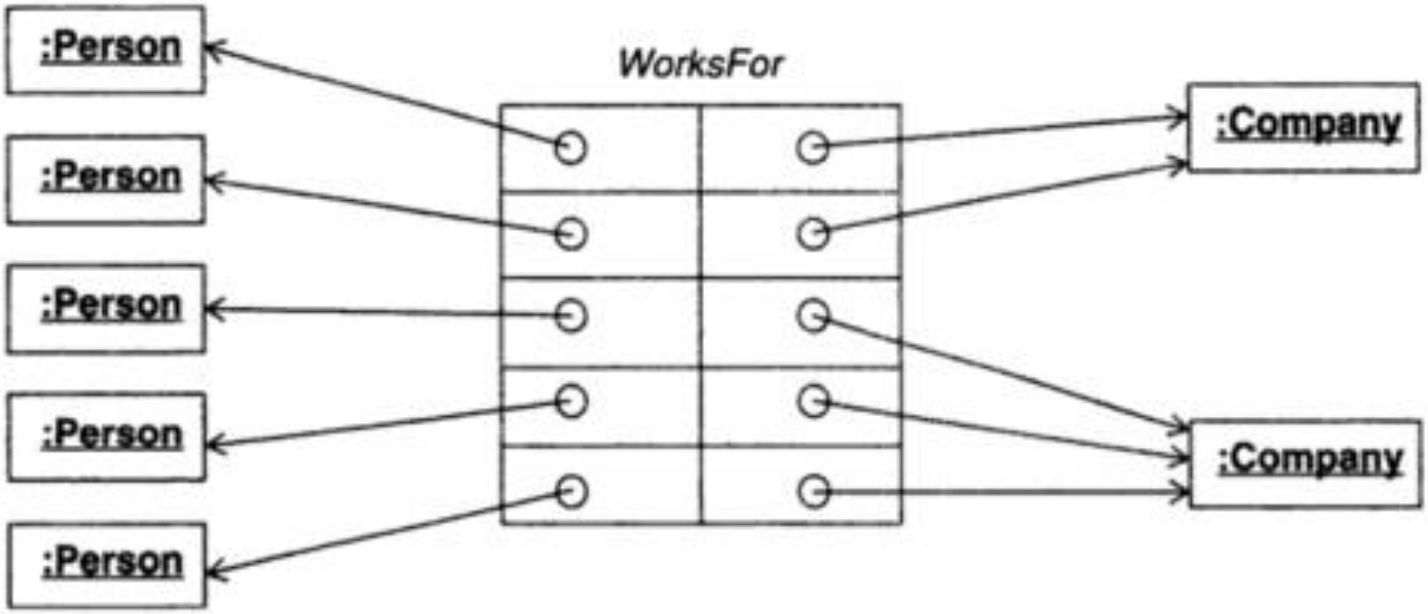


Figure 17.9 Implementing an association as an object. This is the most general approach for implementing associations but requires the most programming skill.

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.