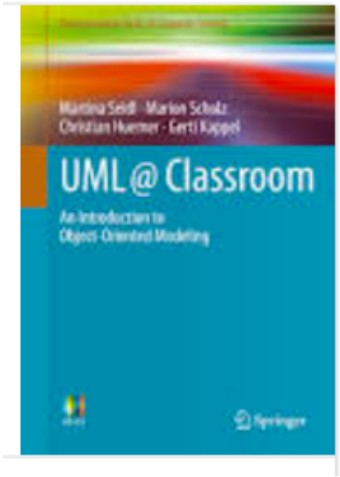
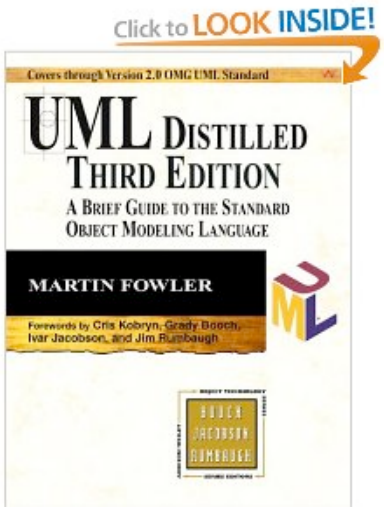
UML : The Unified Modeling Language

CSCI3081

Program Design and Development





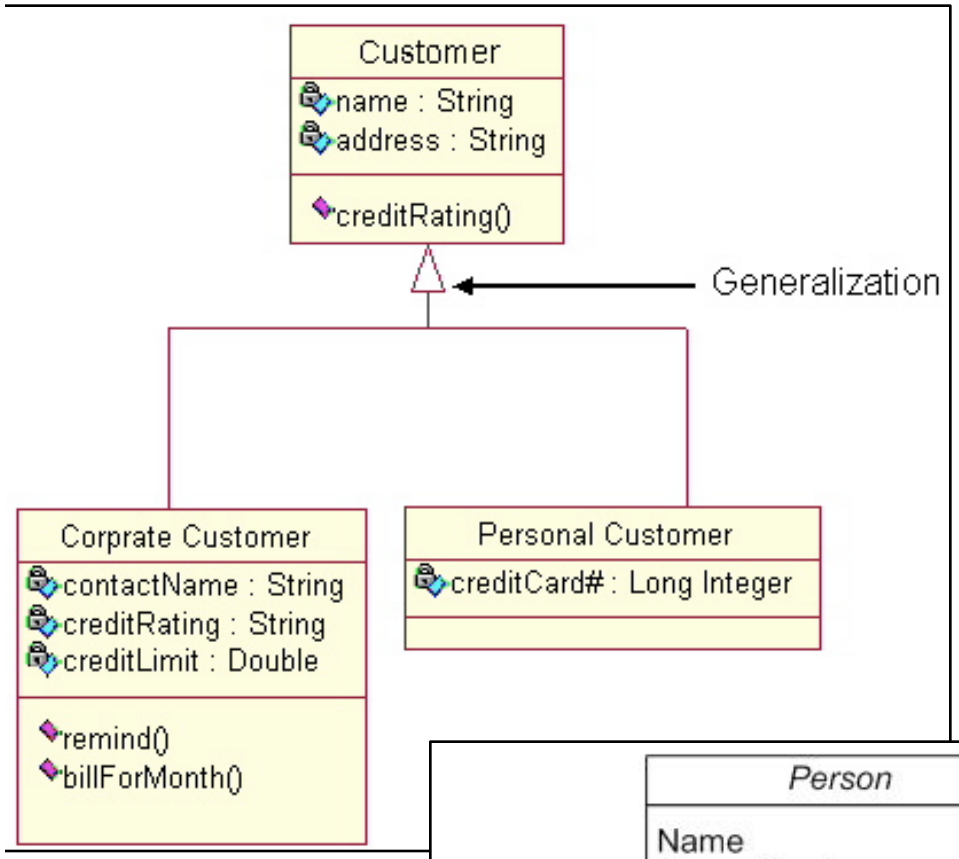
What is UML and What It Is Not.

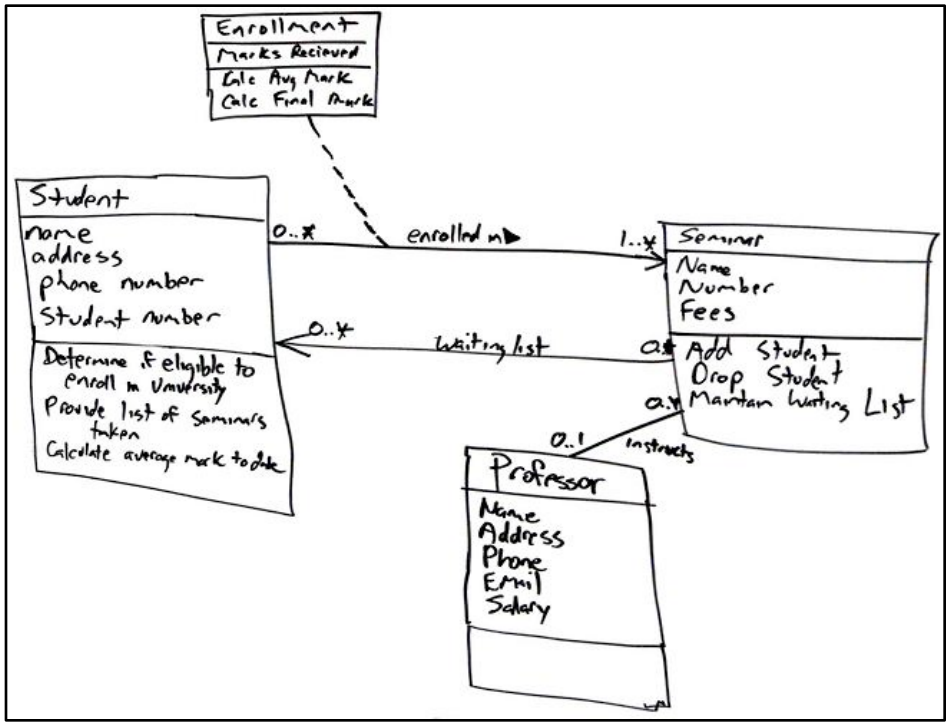
- What is UML
  - A graphical notation for capturing design information.
  - Used for designing and for conveying designs to others.

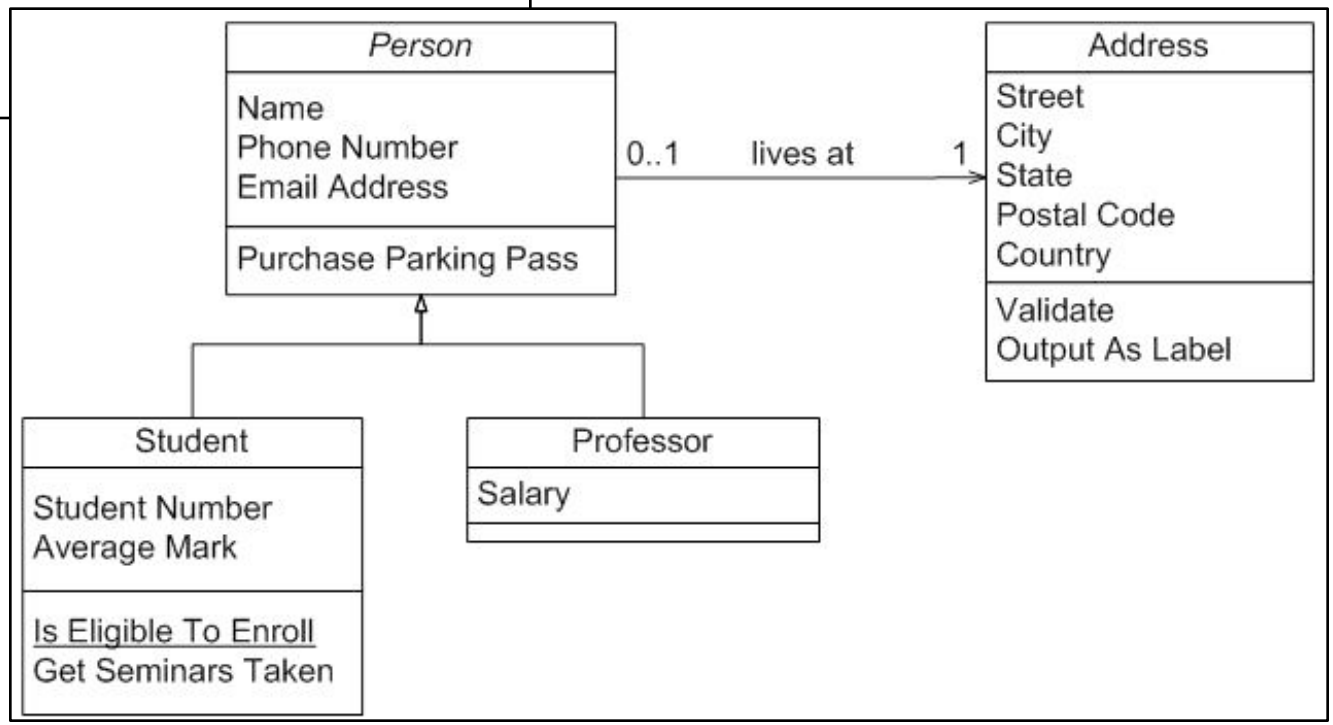
- No, really. What is UML?
  - A tool.
  - A language (syntax and semantics).
  - A syntax.
  - Jargon.

It is only as Unified as you make it!

- What it is not!
  - Not a Product you can buy (although many tools based on UML).
  - Not typically a Programming Language you can compile.
  - Not a Clearly Defined Function that maps model to implementation.


Customer
name : String
address : String
creditRating()
Generalization
Corprate Customer
contactName : String
creditRating : String
creditLimit : Double
remind()
billForMonth()
Personal Customer
creditCard# : Long Integer


Enrollment
Marks Recieved
Calc Avg Mark
Calc Final Mark
Student
name
address
phone number
Student number
Determine if eligible to enroll in University
Provide list of Seminars taken
Calculate average mark to date
0..*
enrolled in
1..*
Seminar
Name
Number
Fees
Add Student
Drop Student
Maintain Waiting List
0..*
waiting list
0..1
instructs
Professor
Name
Address
Phone
Email
Salary


Person
Name
Phone Number
Email Address
Purchase Parking Pass
0..1
lives at
1
Address
Street
City
State
Postal Code
Country
Validate
Output As Label
Student
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
Professor
Salary

What Is It's Purpose In Life?

Miniaturization and Abstraction

"In modeling,
you must not search for absolute truth but for
adequacy for some purpose."
Blaha and Rumbaugh

- Why Model?
  - Test.
  - Communicate.
  - Improve understanding.
  - Visualize.

- Fowler Identifies 3 Uses of UML:
  - Sketch
  - Blueprint
  - Programming Language

How closely do UML diagrams correspond to a program's implementation?

- When are you making your model?
- How abstract is your model?
- Why are you making your model?

- Ways to Think About It
  - Design Tool
  - Aid to Communication with Programmers and Clients
  - Form of Documentation
  - Reduction of Complexity

Model Categories of UML

- **Class Model**
  - Class and Object Diagram
  - Conceptual design of the static objects of the system and how they relate.

- State Machine Model
  - State Diagram
  - Depiction of how the entire system and objects within change over time.

- Interaction Model
  - Use Case Diagram (Shows how people interact with the system.)
  - Sequence Diagram (Shows how data moves through the system.)
  - Activity Diagram (Shows the flow of control.)

Class Diagrams : The Most Frequently Used Component

Class Diagrams in UML

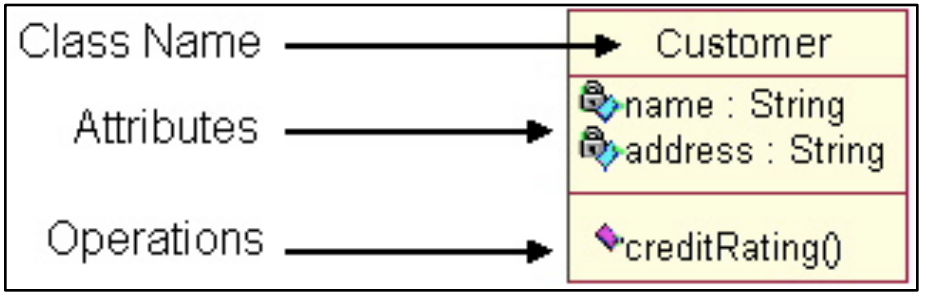
- describe and depict static objects
- describe and depict the static relationships among objects.
- tightly coupled to *OOP* and *classes* in OOP languages.

Class Diagram Elements

- Class / Attribute / Operation
- Association / Multiplicity / Association Class
- Inheritance / Composition / Dependencies
- Comments / Keywords / Constraints

Class Diagram Basic Elements

- Classes are composed of three things:
  - name
  - attributes
  - operations
  - (associations connect classes)



- Diagrams Represent Abstractions and Instantiations
  - Classes ➔ Object
  - Attributes ➔ Value
  - Operations ➔ Method
  - Associations ➔ Link

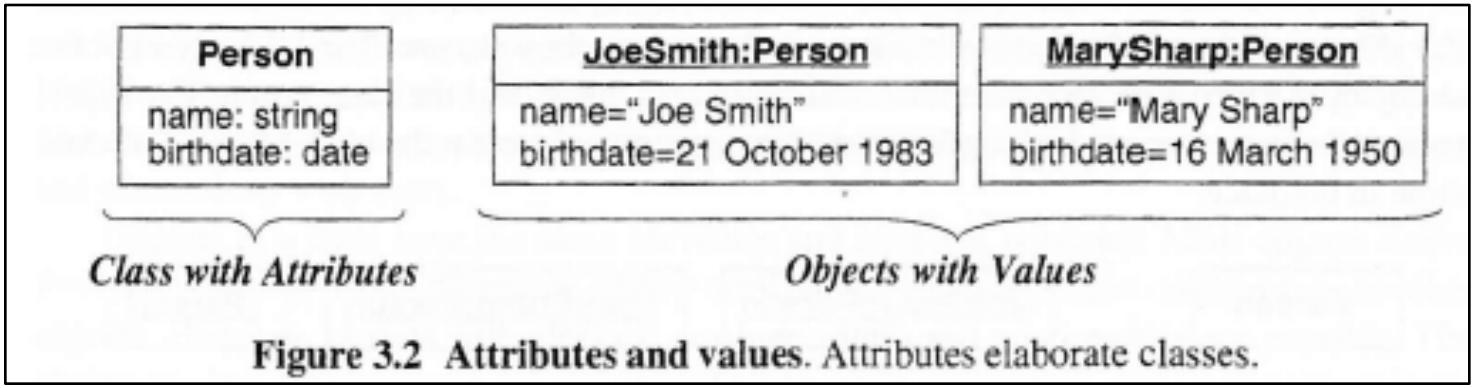
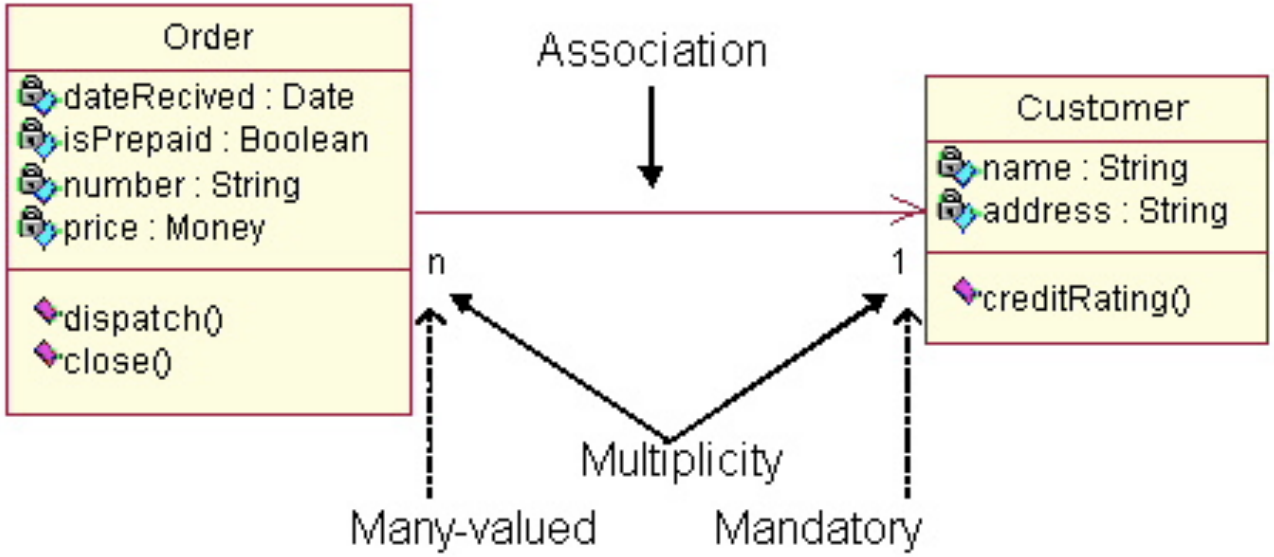


Figure 3.2 Attributes and values. Attributes elaborate classes.

Associations Among Classes

- Association
- Multiplicity
- Association Class
- Qualified Association

Indicator	Multiplicity
0..1	zero or one
1	one only (default)
0..*	zero or more
1..*	one or more
n	only n (where n > 1)
0..n	zero to n (where n > 1)
1..n	one to n (where n > 1)
*	zero or more



Multiplicity

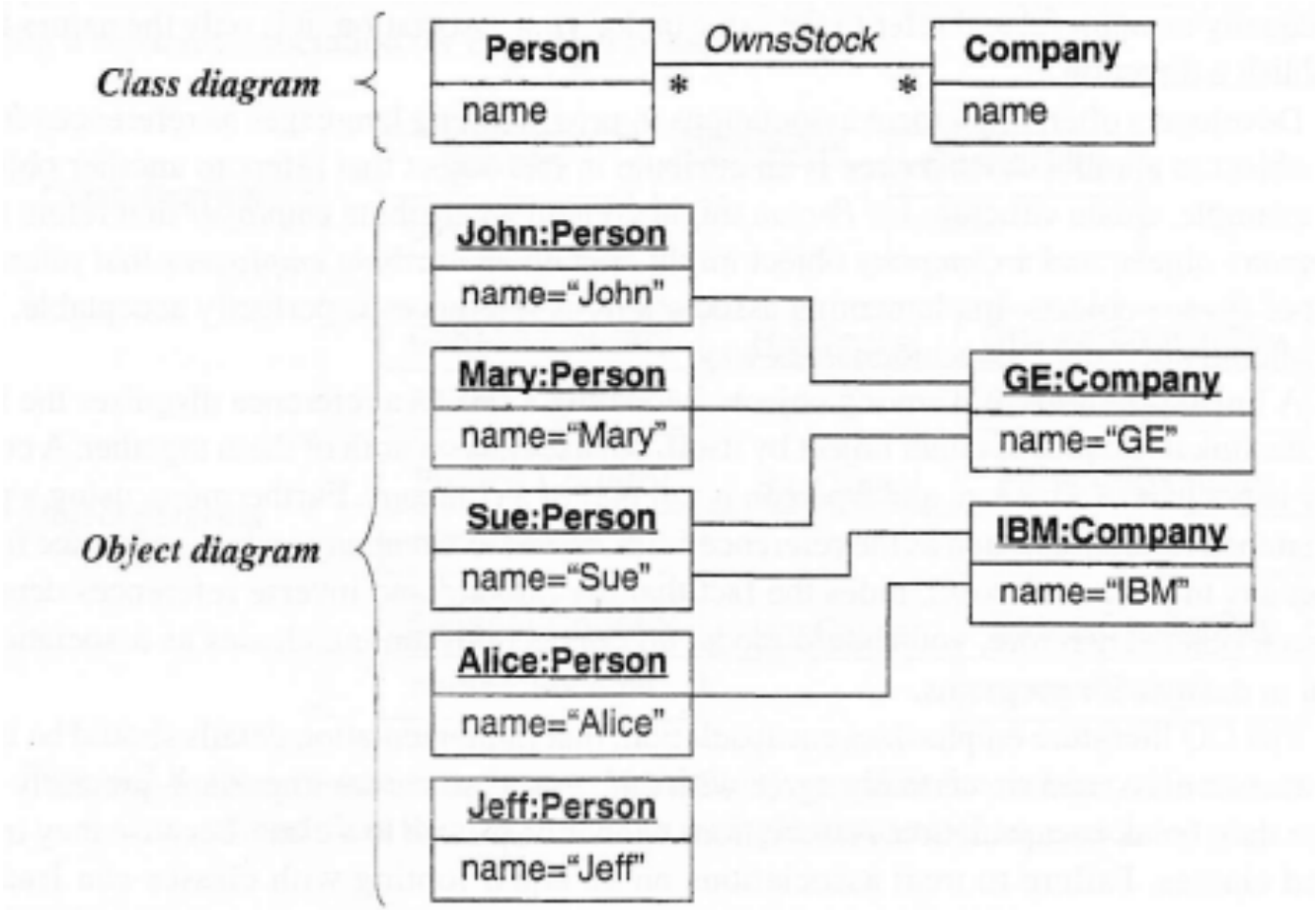
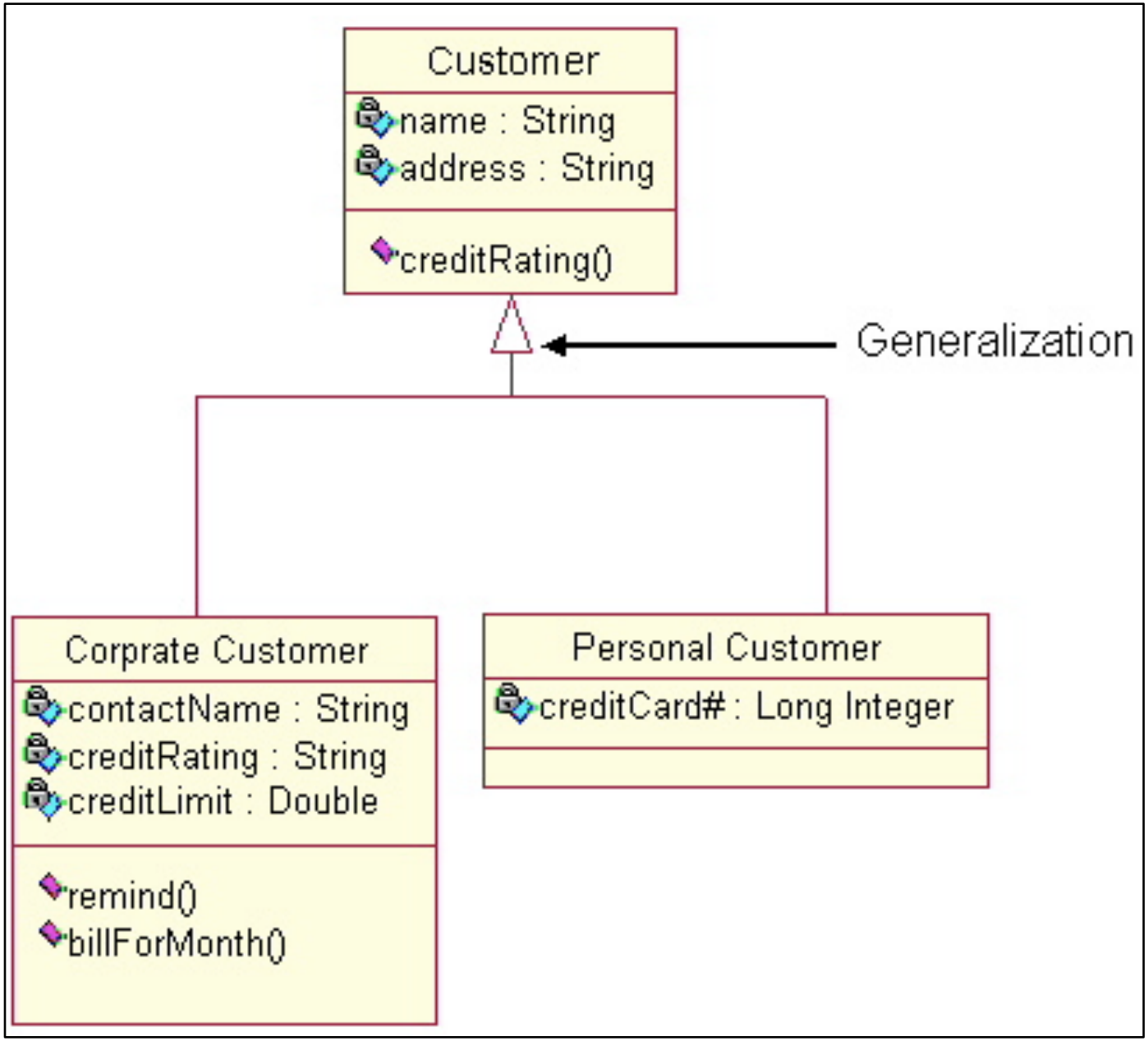


Figure 3.7 Many-to-many association. An association describes a set of potential links in the same way that a class describes a set of potential objects.

Combining and Relating Classes

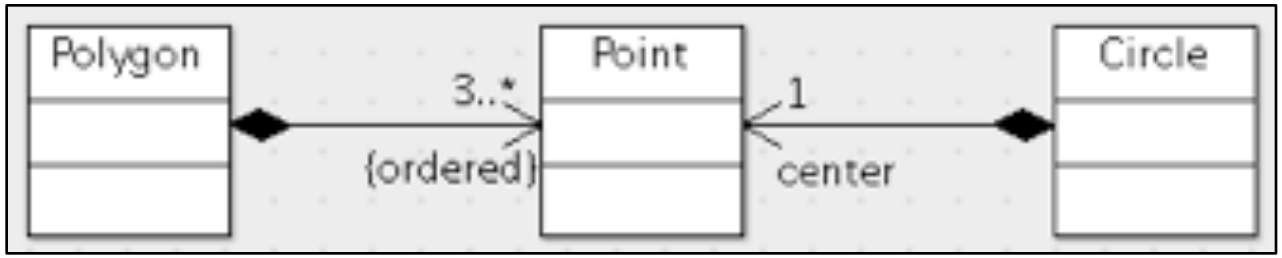
- Generalization (Inheritance)
  - Subclass inherits implementation.
  - Subtype inherits interface << interface >>
- Composition
  - A single instantiation of a class is part of another class.
  - The elemental part does not exist if the whole does not exist.
- Aggregation
  - Multiple instantiations of a class create an instance of another class.
  - The elemental parts are independent of the aggregate.
- Dependencies
  - Depiction of a coupling among classes.
  - The attributes and operations of one class (user) depend on another (supplier).
  - Modifications of the supplier require a modification to the user.

Generalization, Aggregation, and Composition



Aggregation, Composition, Dependency

Composition



Aggregation

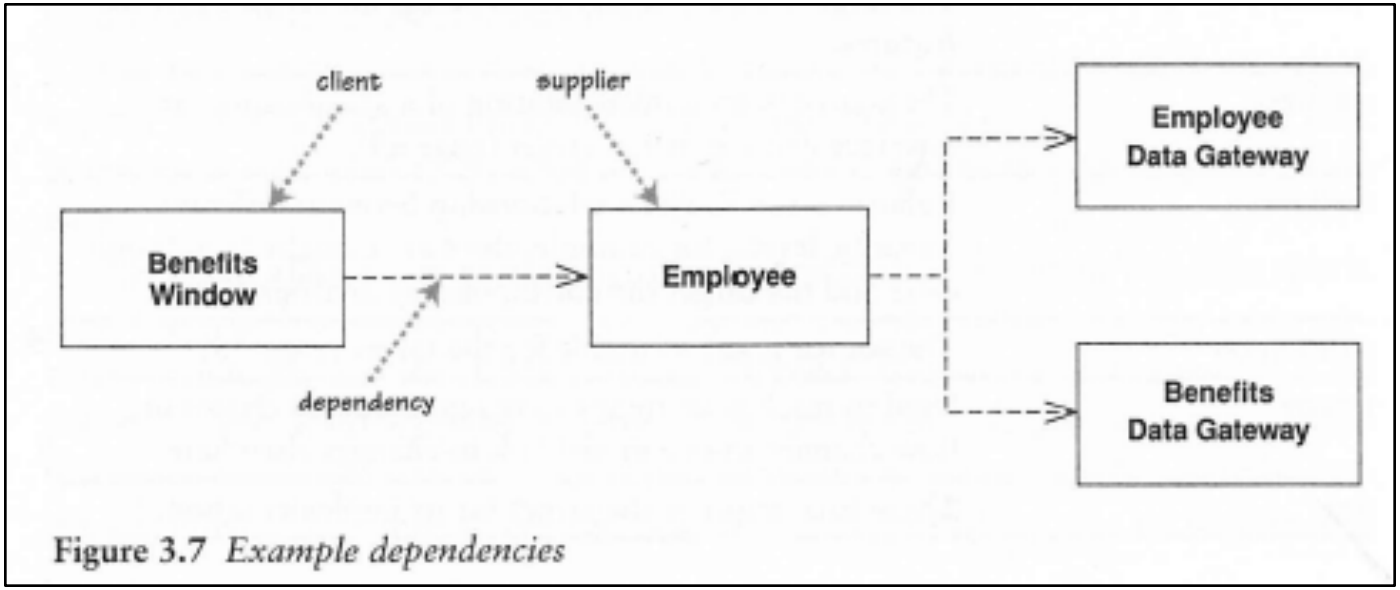


Figure 3.7 *Example dependencies*

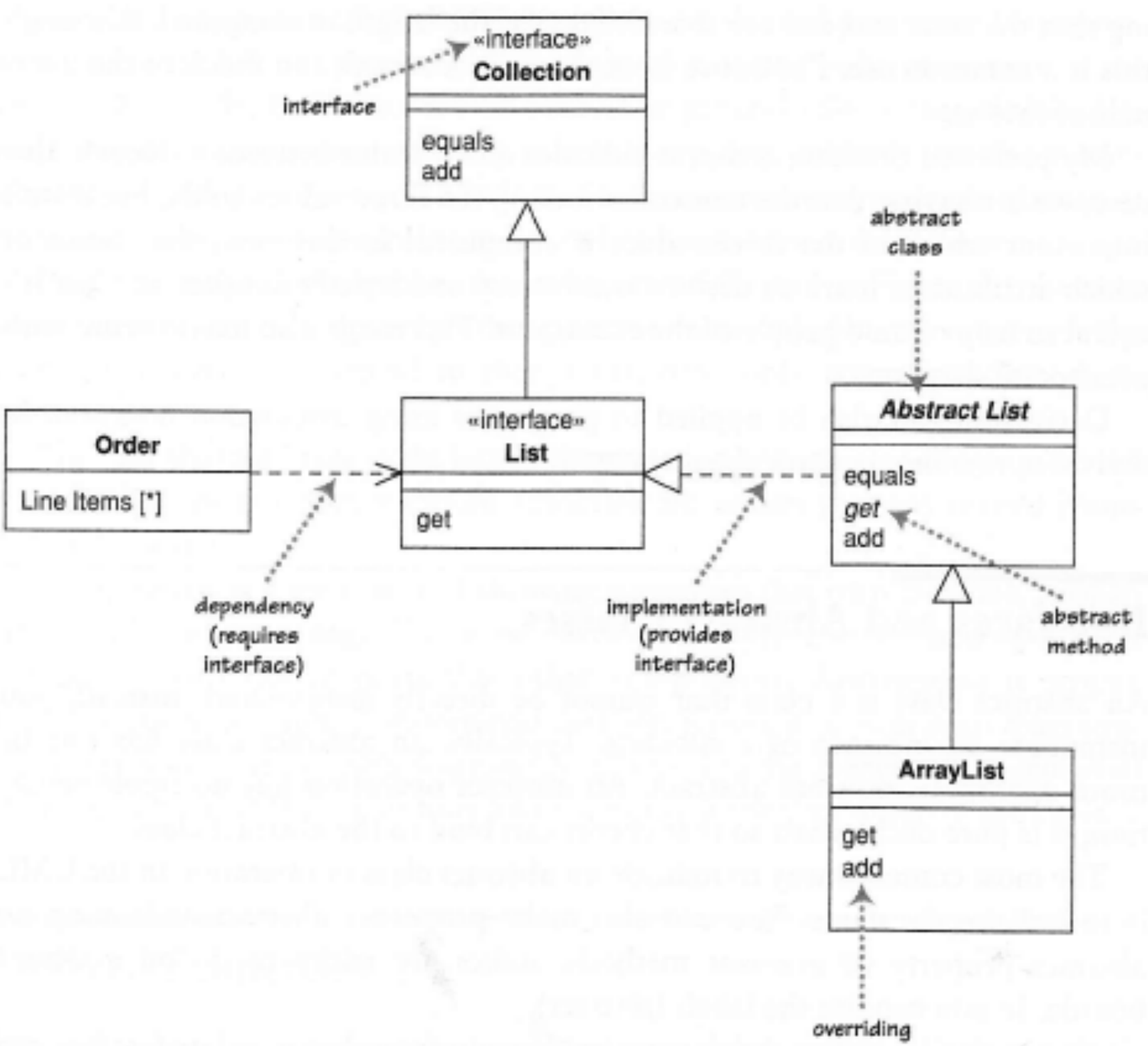
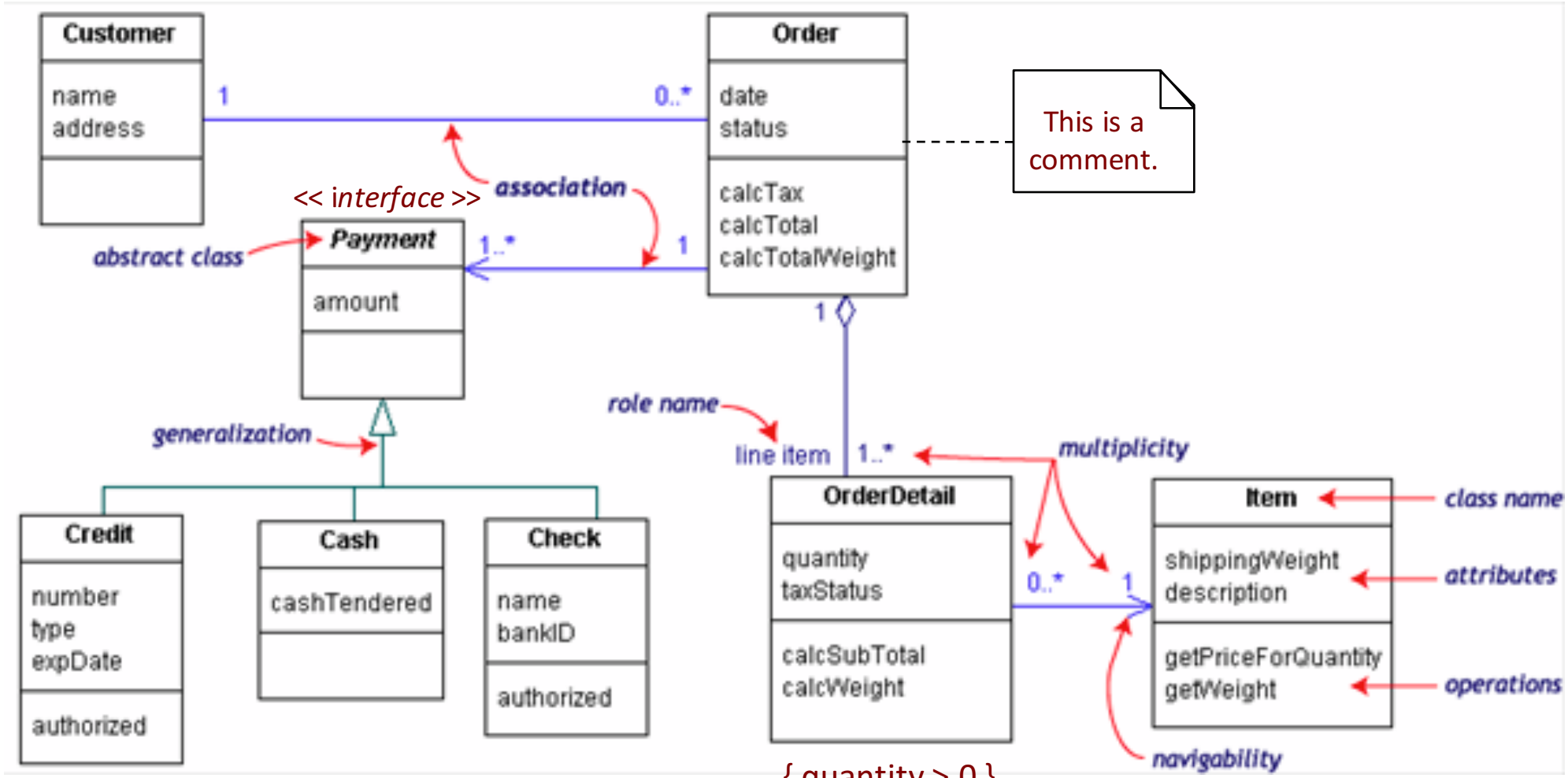


Figure 5.6 *A Java example of interfaces and an abstract class*

Abstraction and <<interface>>

Putting It All Together

Comments
<< Keywords >>
{ Constraints }



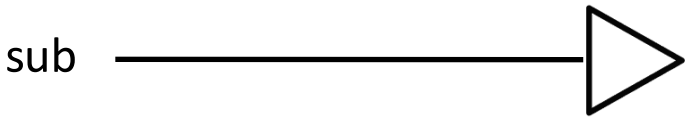
Symbol Table

Bidirectional Association

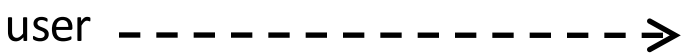
Bidirectional Association

Unidirectional Association

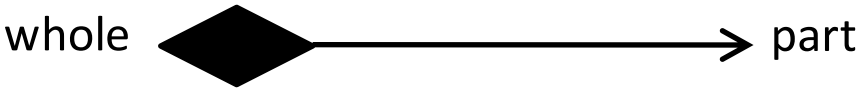
Association Class



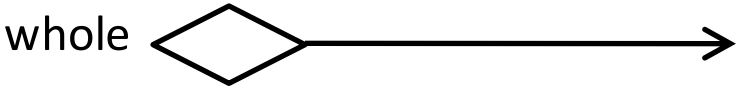
Generalization (Inheritance)



Dependency



Composition



Aggregations

Try This

Cars

- 3 or 4 wheels
- Human and/or autonomous control
- Sensors: Distance (for speed) and Camera (for direction)
- 2, 4, or all-wheel drive (independent speed control of wheel)

- Speed of wheels control ...
  - Speed of car (including 0).
  - Turning of car.
- Speed controlled by ..
  - Distance Sensor (object avoidance)
  - Cruise Control or Human Input
  - Steering Input from Camera or human