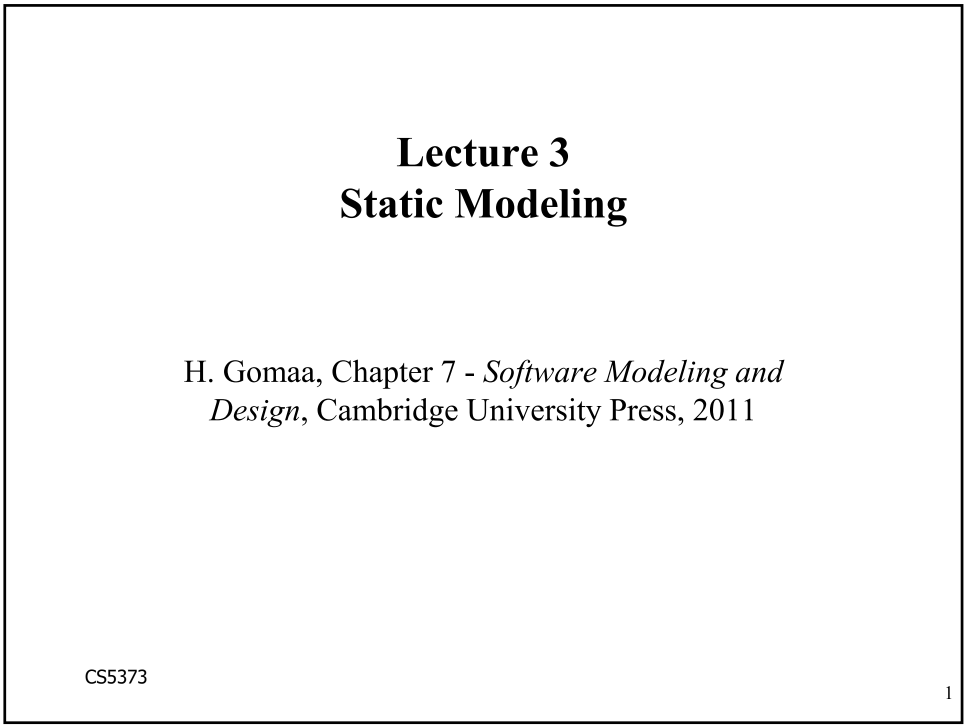
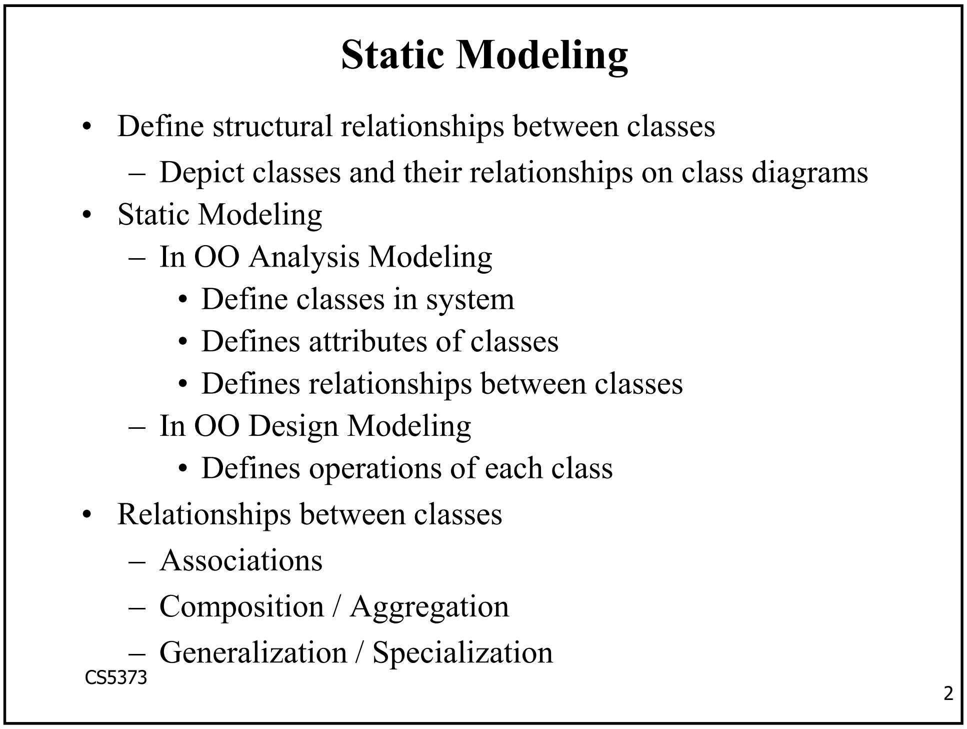


Lecture 3
Static Modeling

H. Gomaa, Chapter 7 - *Software Modeling and Design*, Cambridge University Press, 2011



Static Modeling

- Define structural relationships between classes
  - Depict classes and their relationships on class diagrams
- Static Modeling
  - In OO Analysis Modeling
    - Define classes in system
    - Defines attributes of classes
    - Defines relationships between classes
  - In OO Design Modeling
    - Defines operations of each class
- Relationships between classes
  - Associations
  - Composition / Aggregation
  - Generalization / Specialization

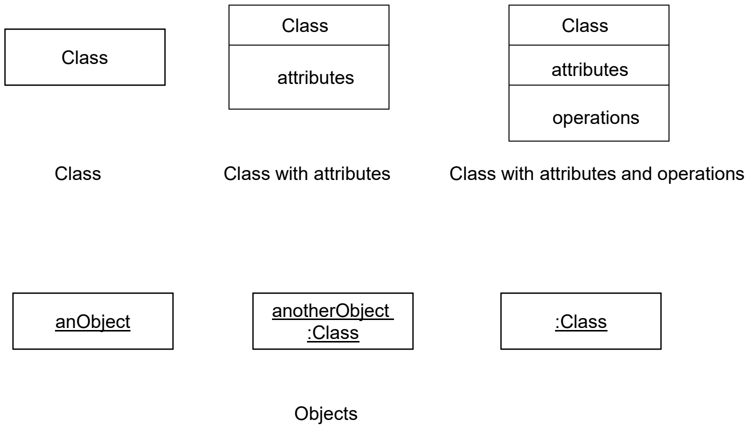
Static Modeling during Requirements Analysis Modeling

- Static Modeling
  - Software System Context Class Diagram
  - Static Modeling of Entity

Objects and Classes

- An Object (object instance) is a single “thing”
  - E.g., John’s car
  - Mary’s account
- A Class (object class) is a collection of objects with the same characteristics
  - E.g., account, employee, car, customer
- Attribute
  - Data value held by object
- Example of Attributes
  - E.g., account number, balance
- UML notation for objects & classes
- Example of classes and objects

Figure 2.2 UML notation for objects & classes



Class

Class
attributes

Class
attributes
operations

Class

Class with attributes

Class with attributes and operations

anObject

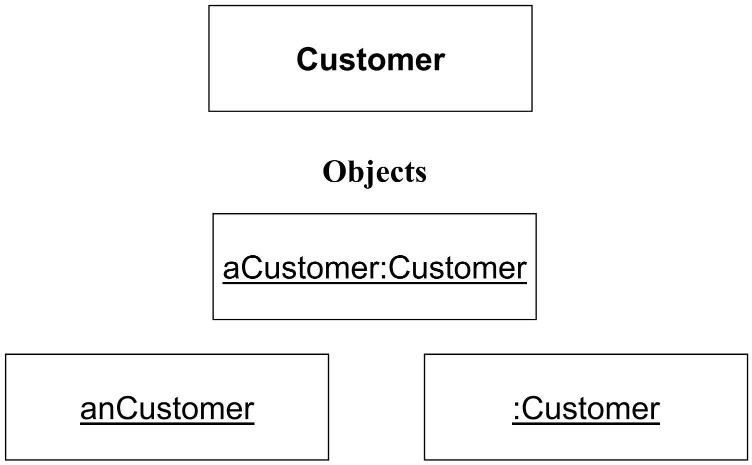
anotherObject :Class

:Class

Objects

Example of classes and objects

Class



Customer

Objects

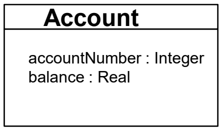
aCustomer:Customer

anCustomer

:Customer

Example of class with attributes

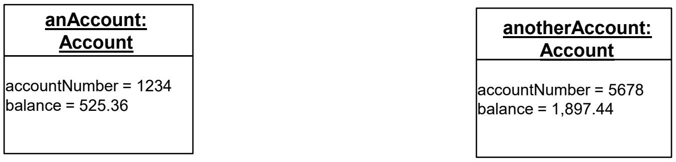
Class with attributes



Account

accountNumber : Integer
balance : Real

Objects with values



anAccount: Account

accountNumber = 1234
balance = 525.36

anotherAccount: Account

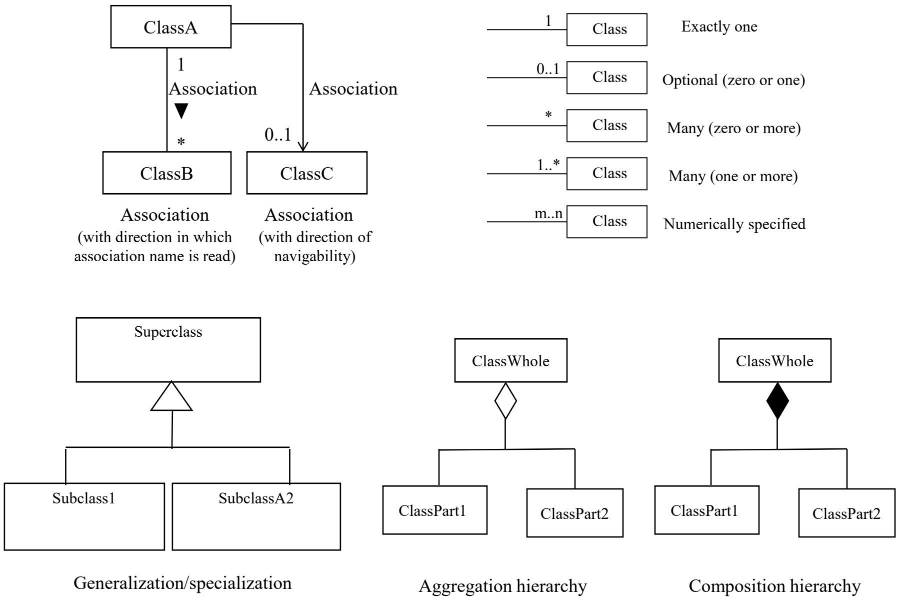
accountNumber = 5678
balance = 1,897.44

Associations

- Association is
  - Static, structural relationship between classes
  - e.g., Employee <u>works in</u> Department

- Multiplicity of Associations
  - Specifies how many instances of one class may relate to a single instance of another class
  - Examples
    - 1-to-1 association
    - 1-to-many association
    - Optional association (0 or 1)
    - Optional association (0, 1, or many)

Figure 2.3 UML notation for relationship on class diagram



One-to-One Association

- 1-to-1 association
  - Association between two classes is 1-to-1 in both directions

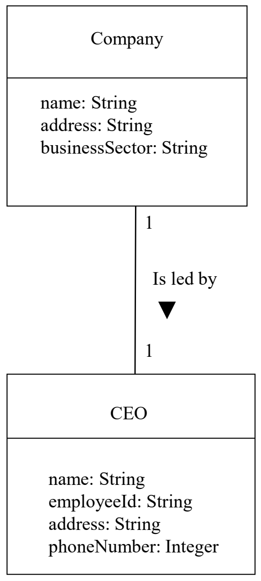
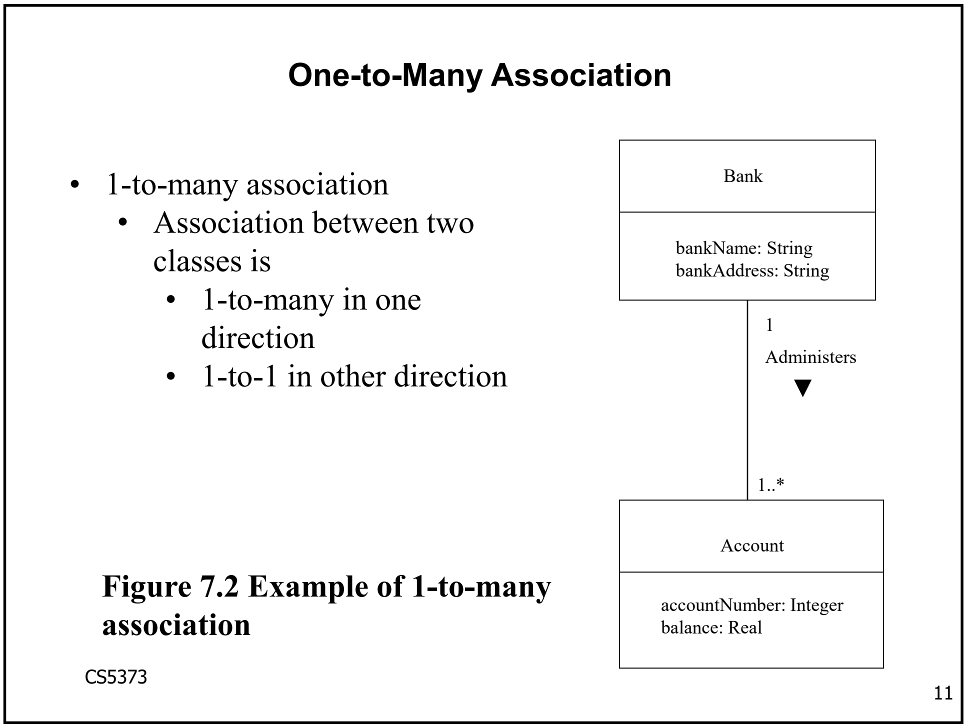


Figure 7.1 Example of 1-to-1 association

▶ Direction of Association



One-to-Many Association

- 1-to-many association
  - Association between two classes is
    - 1-to-many in one direction
    - 1-to-1 in other direction

Bank

bankName: String
bankAddress: String

1

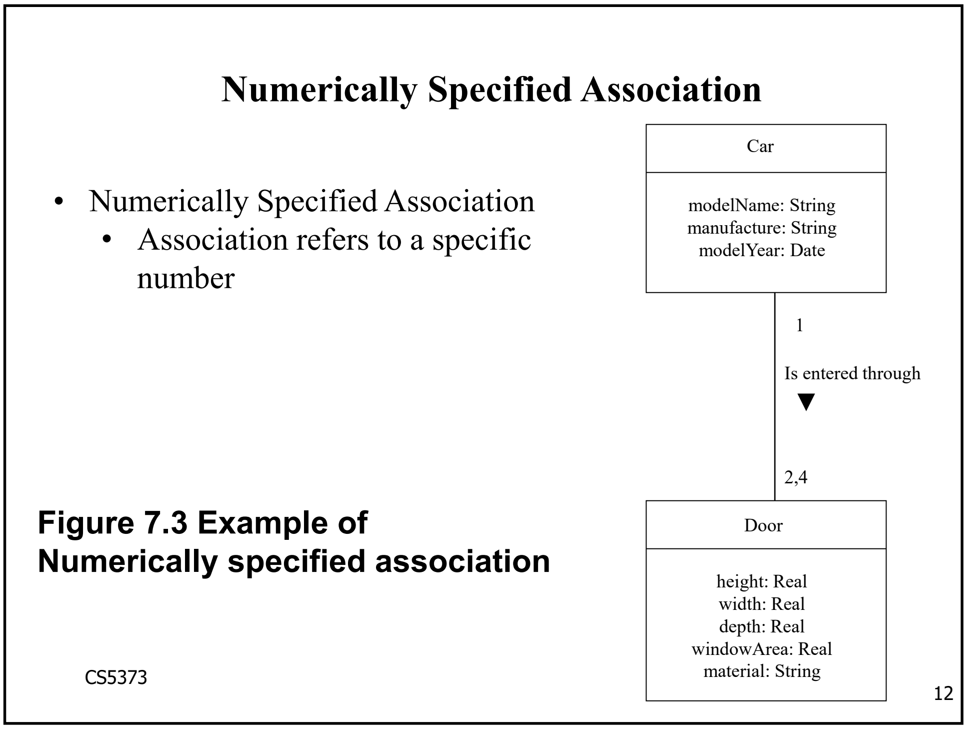
Administers ▼

1..*

Account

accountNumber: Integer
balance: Real

Figure 7.2 Example of 1-to-many association



Numerically Specified Association

- Numerically Specified Association
  - Association refers to a specific number

Car

modelName: String
manufacture: String
modelYear: Date

1

Is entered through ▼

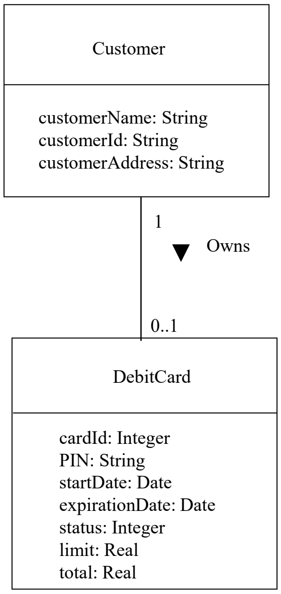
2,4

Door

height: Real
width: Real
depth: Real
windowArea: Real
material: String

Figure 7.3 Example of Numerically specified association

Figure 7.4 Optional (zero-or-one) association



Customer

customerName: String
customerId: String
customerAddress: String

1

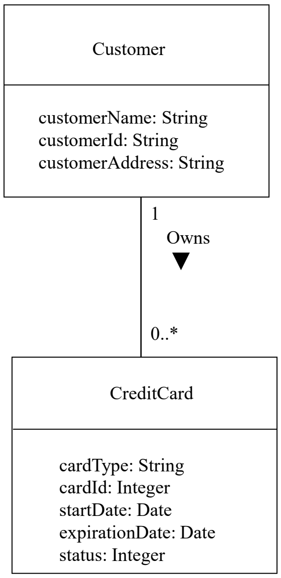
▼ Owns

0..1

DebitCard

cardId: Integer
PIN: String
startDate: Date
expirationDate: Date
status: Integer
limit: Real
total: Real

Figure 7.5 Optional (zero,one, or many) association



Customer

customerName: String
customerId: String
customerAddress: String

1

Owns
▼

0..*

CreditCard

cardType: String
cardId: Integer
startDate: Date
expirationDate: Date
status: Integer

Many-to-Many Association

- Many-to-Many association
  - 1-many association in each direction

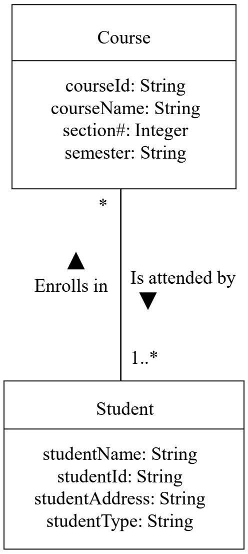
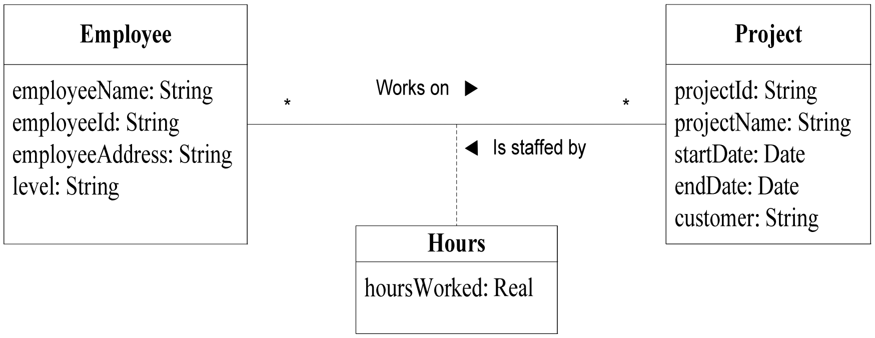


Figure 7.6 Many-to-many association

Association Class

- Modeling class association between two or more classes
  - Usually for many-to-many associations
  - Attributes of Association Class
    - Attributes of association
- E.g., Many-to-many association between
  - Project and Employee classes
    - Project Is staffed by Employee
    - Employee Works on project
  - Association Class - Hours
    - Attribute - Hours Worked

Figure 7.11 Example of association class



Employee

employeeName: String
employeeId: String
employeeAddress: String
level: String

Works on ▶

◀ Is staffed by

* *

Project

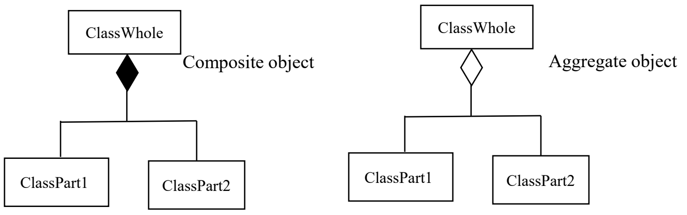
projectId: String
projectName: String
startDate: Date
endDate: Date
customer: String

Hours

hoursWorked: Real

Composition and Aggregation Hierarchies

- Whole/Part Relationships
  - Show parts of more complex class
  - *Is part of* Relationship
    - Between part classes and whole class
- Composition is stronger relationship than aggregation



ClassWhole

Composite object

ClassPart1 ClassPart2

ClassWhole

Aggregate object

ClassPart1 ClassPart2

Composition hierarchy

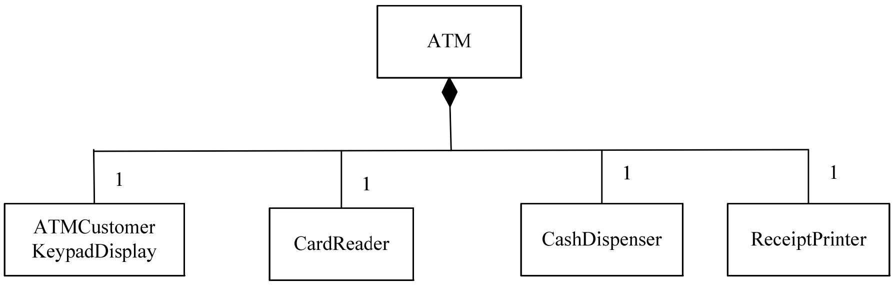
Aggregation hierarchy

Figure 2.3 UML notation for relationships on class diagrams

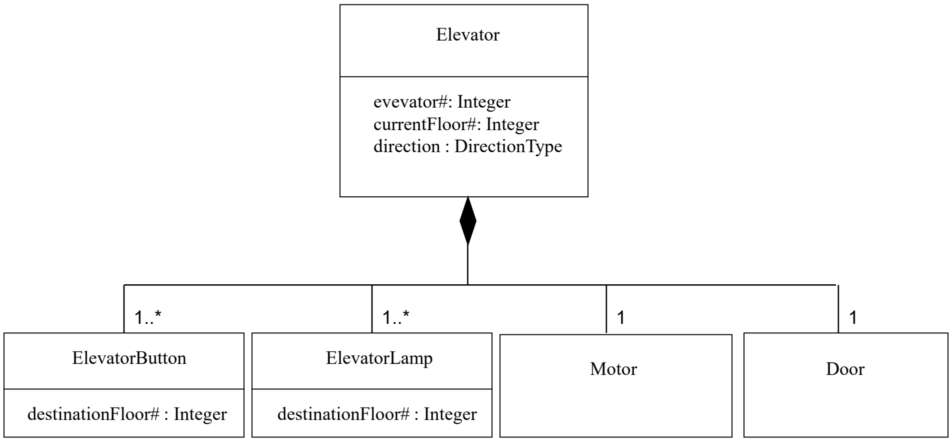
Composition Hierarchy

- Composition Hierarchy
  - Whole and part objects are created, live, die together
  - Often also has a physical association
  - Association between instances
- E.g., Composite class
    - ATM
  - Part classes

Figure 7.12 Example of composition hierarchy



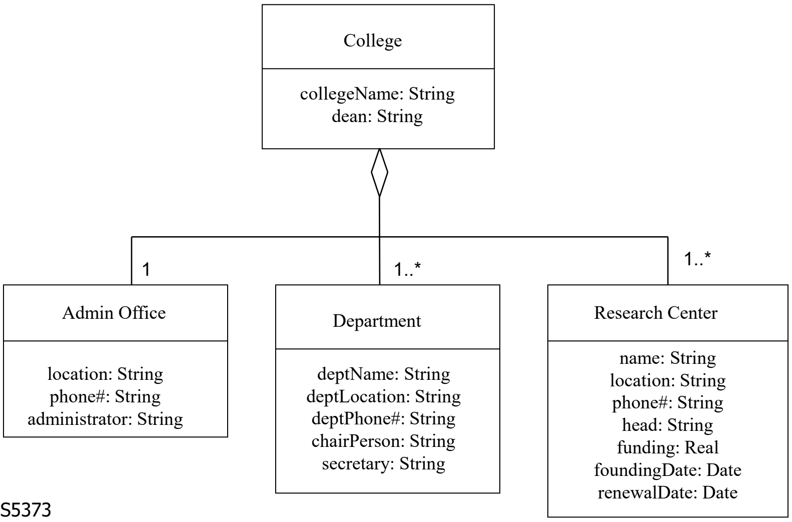
Example of composition hierarchy



Aggregation Hierarchy

- Aggregation Hierarchy
  - Part objects of aggregate object may be created and deleted independently of aggregate object
  - E.g., Aggregate class
    - College

Figure 7.13 Example of aggregation hierarchy



College

collegeName: String
dean: String

1

Admin Office

location: String
phone#: String
administrator: String

1..*

Department

deptName: String
deptLocation: String
deptPhone#: String
chairPerson: String
secretary: String

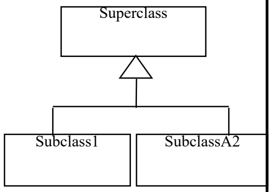
1..*

Research Center

name: String
location: String
phone#: String
head: String
funding: Real
foundingDate: Date
renewalDate: Date

Generalization / Specialization Hierarchy

- Also know as **inheritance**
- Some classes are similar but not identical
  - Have some attributes in common, others different
- Common attributes abstracted into generalized class (superclass)
- Different attributes are properties of specialized class (subclass)
- *Is a* relationship between subclass and superclass



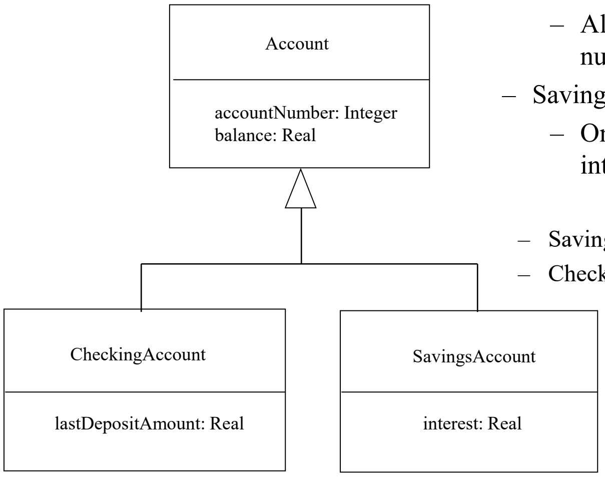
Superclass

Subclass1

SubclassA2

Generalization/specialization

Figure 7.14 Generalization / Specialization hierarchy



- Account (Account number, Balance)
  - All accounts have an account number and balance
- Savings Account (Interest)
  - Only savings accounts get interest

- Savings Account *Is a* Account
- Checking Account *Is a* Account

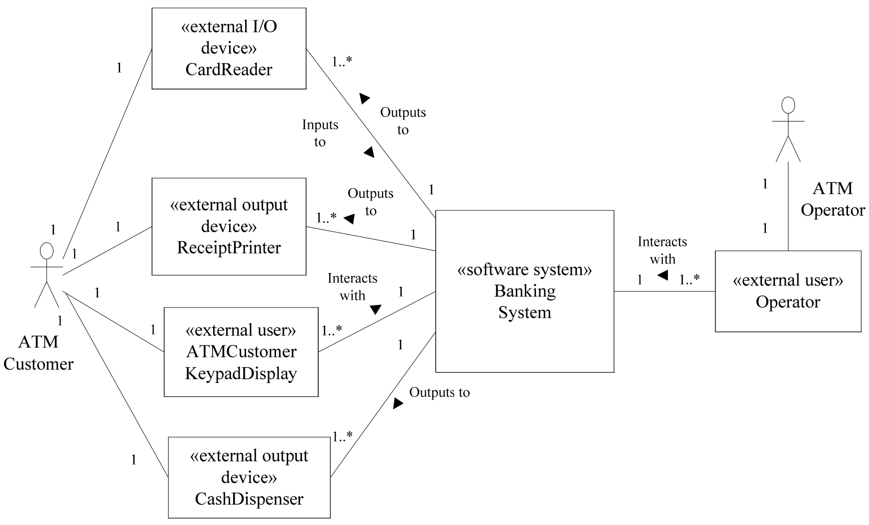
Software System Context Class Diagram

- Defines boundary between software system and external environment
- Software System
  - Depict as one aggregate «software system» class
- External environment
  - External classes that software system interfaces to
- Categories of external classes
  - «external I/O device»
  - «external user»
  - «external system»
  - «external timer»

Associations on Software System Context Class Diagram

- Software System Context Class Diagram shows
  - Association between software system and external class
  - Multiplicity of association (1 to 1, 1..* to 1, etc.)
- Associations have standard names
  - «external input device» *Inputs to* «software system»
  - «software system» *Outputs to* «external output device»
  - «external user» *Interacts with* «software system»
  - «external system» *Communicates with* «software system»
  - «external timer» *Signals* «software system»

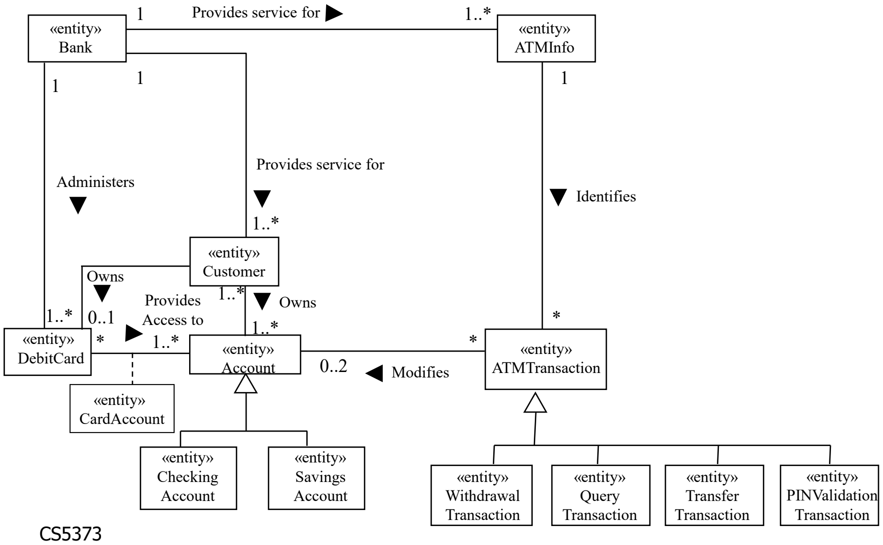
Figure 7.23 Banking System software context class diagram


«external I/O device» CardReader
«external output device» ReceiptPrinter
«external user» ATMCustomer KeypadDisplay
«external output device» CashDispenser
«software system» Banking System
«external user» Operator
ATM Customer
ATM Operator
Inputs to
Outputs to
Outputs to
Interacts with
Outputs to
Interacts with
1
1..*
1..*
1..*
1..*
1..*

Static Modeling of Entity Classes

- Entity classes
  - Data-intensive classes
  - Store long-living (persistent) data
- During analysis modeling
  - Model entity classes in the problem domain
  - Attributes
  - Relationships

Figure 21.4 Conceptual static model for Banking System - entity classes



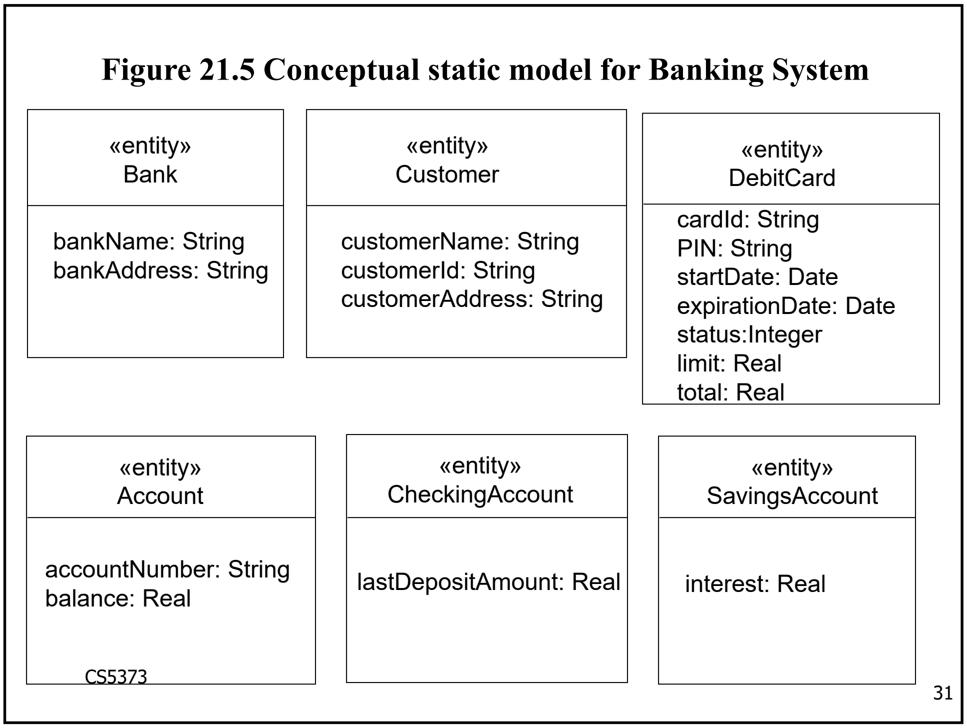


Figure 21.5 Conceptual static model for Banking System

«entity»
Bank

bankName: String
bankAddress: String

«entity»
Customer

customerName: String
customerId: String
customerAddress: String

«entity»
DebitCard

cardId: String
PIN: String
startDate: Date
expirationDate: Date
status:Integer
limit: Real
total: Real

«entity»
Account

accountNumber: String
balance: Real

«entity»
CheckingAccount

lastDepositAmount: Real

«entity»
SavingsAccount

interest: Real

CS5373

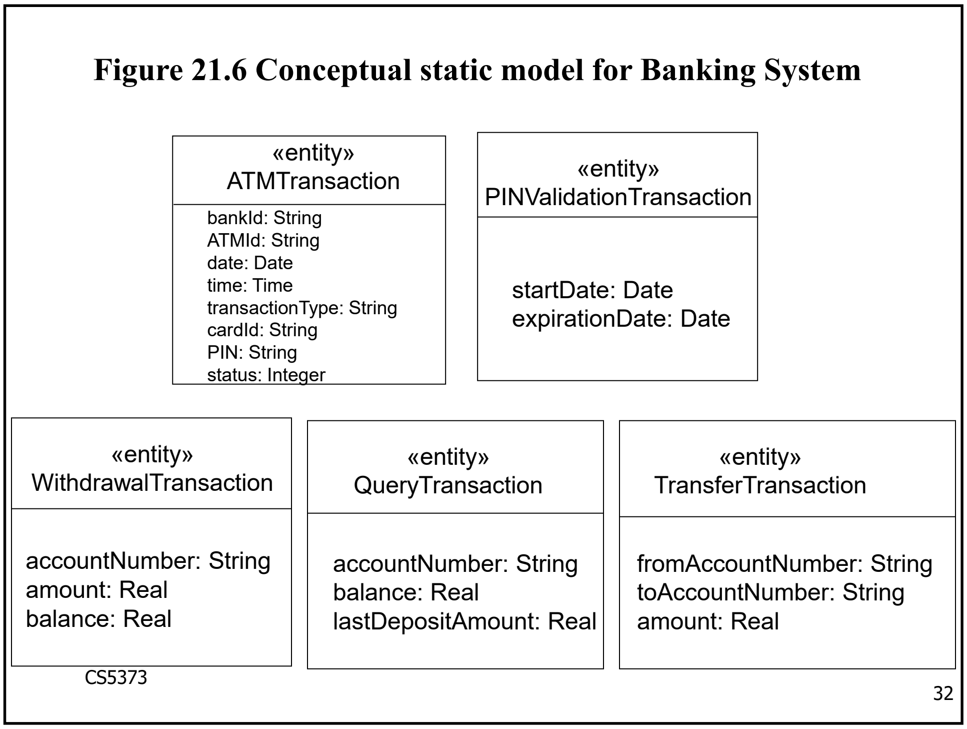


Figure 21.6 Conceptual static model for Banking System

«entity»
ATMTransaction

bankId: String
ATMId: String
date: Date
time: Time
transactionType: String
cardId: String
PIN: String
status: Integer

«entity»
PINValidationTransaction

startDate: Date
expirationDate: Date

«entity»
WithdrawalTransaction

accountNumber: String
amount: Real
balance: Real

«entity»
QueryTransaction

accountNumber: String
balance: Real
lastDepositAmount: Real

«entity»
TransferTransaction

fromAccountNumber: String
toAccountNumber: String
amount: Real

CS5373

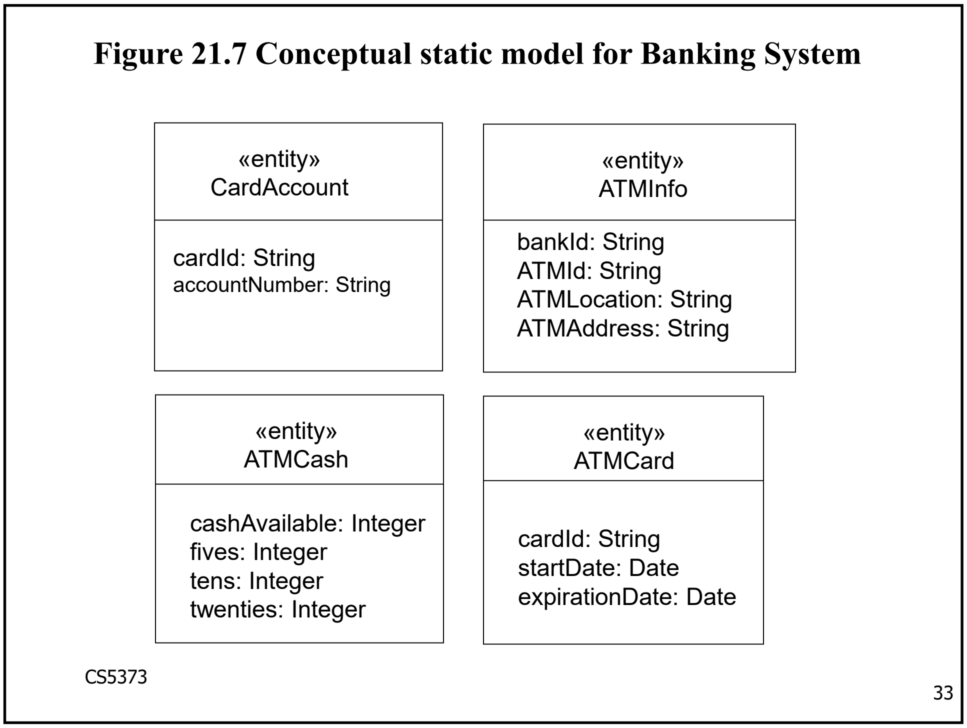


Figure 21.7 Conceptual static model for Banking System

«entity»
CardAccount

cardId: String
accountNumber: String

«entity»
ATMInfo

bankId: String
ATMId: String
ATMLocation: String
ATMAddress: String

«entity»
ATMCash

cashAvailable: Integer
fives: Integer
tens: Integer
twenties: Integer

«entity»
ATMCard

cardId: String
startDate: Date
expirationDate: Date