SWE 621:
Software Modeling and Architectural Design

Lecture 3
Static Modeling

Hassan Gomaa

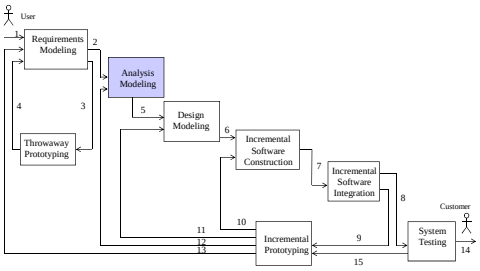
Dept of Computer Science
George Mason University
Fairfax, VA

Steps in Using COMET/UML

1 Develop Software Requirements Model
- Develop Use Case Model (Chapter 6)

2 Develop Software Analysis Model
- Develop static model of problem domain (Chapter 7)
- Structure system into objects (Chapter 8)
- Develop statecharts for state dependent objects (Chapter 10)
- Develop object interaction diagrams for each use case (Chapter 9, 11)

3 Develop Software Design Model



Static Modeling

Section 4

Hassan Gomaa

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

H. Gomaa, "Chapter 8 - Designing Concurrent, Distributed, and Real-Time Applications with UML", Addison Wesley Object Technology Series, July, 2000

Static Modeling

- Originated with Rumbaugh Object Modeling Technique (OMT)
- Represents static structure of system
- Based on information (Entity-Relationship) modeling
- 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

Design Concepts
Objects and Classes

- Objects represent “things” in real world
  - Provide understanding of real world
  - Form basis for a computer solution
- An Object (object instance) is a single “thing”
  - E.g., an account, an employee
- A Class (object class) is a collection of objects with the same characteristics
  - E.g., account, employee
- Attribute
  - Data value held by object in class
  - E.g., account number, balance

Figure 2.2 UML notation for objects & classes

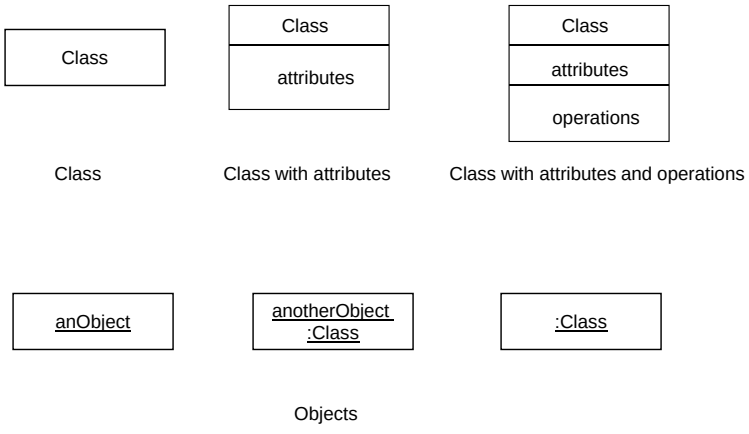


Figure 4.1 Example of classes and objects

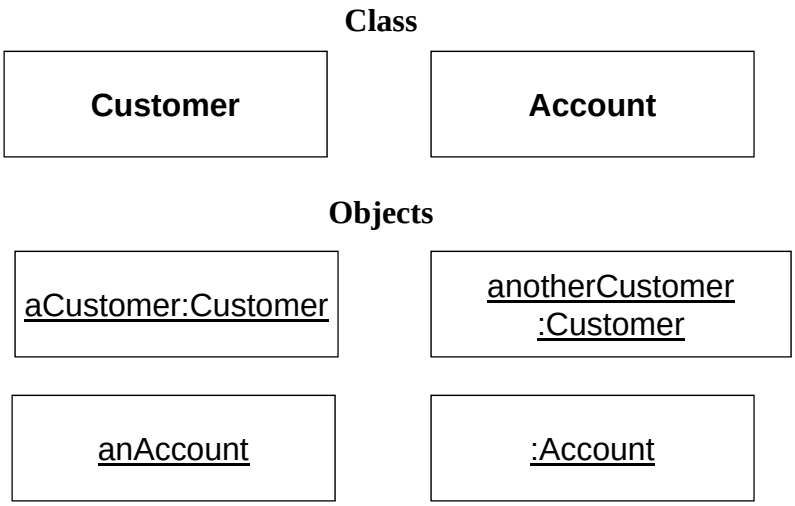
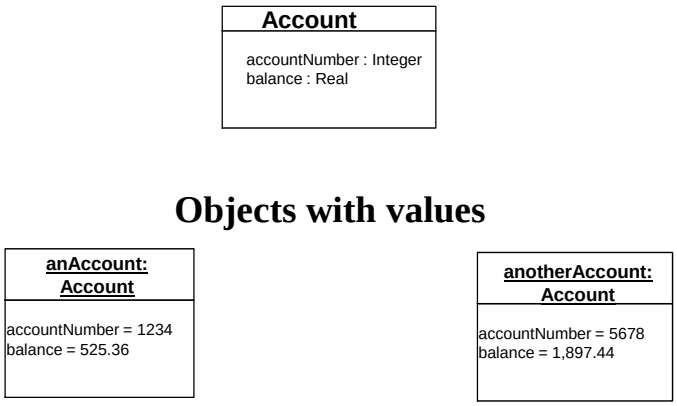


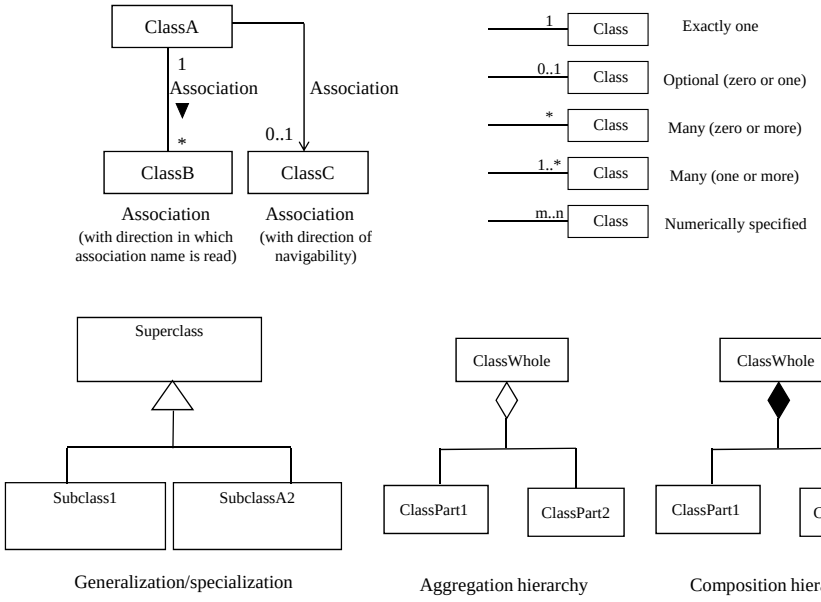
Figure 4.2 Example of class with attributes



Static Modeling

- Define structural relationships between classes
  - Depict classes and their relationships on class diagrams
- Relationships between classes
  - Associations
  - Composition / Aggregation
  - Generalization / Specialization
- Static Modeling during Analysis
  - System Context Class Diagram
    - Depict external classes and system boundary
  - Static Modeling of Entity classes
    - Persistent classes that store data

Figure 2.3 UML notation for relationship on class diagram



Associations

- Association is
  - static, structural relationship between classes
  - E.g, Employee works in Department
- Multiplicity of Associations
  - Specifies how many instances of one class may relate to a single instance of another class

Multiplicity of Associations

- 1-to-1 association
  - Company has President
- 1-to-many association
  - Bank manages Account
- Numerically specified association
  - Car has 2,4 Door
- Optional association (0 or 1)
  - Customer owns Debit Card
- Optional association (0, 1, or many)
  - Customer owns Credit Card
- Many-to-Many association
  - Course has Student
  - Student attends Course

Figure 7.1 Example of one-to-one association

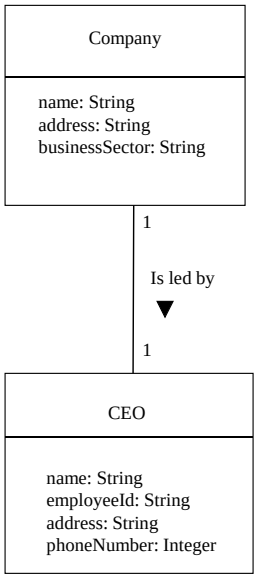


Figure 7.2 Example of one-to-many association

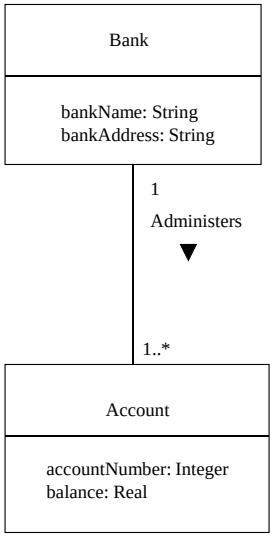


Figure 7.3 Numerically specified association

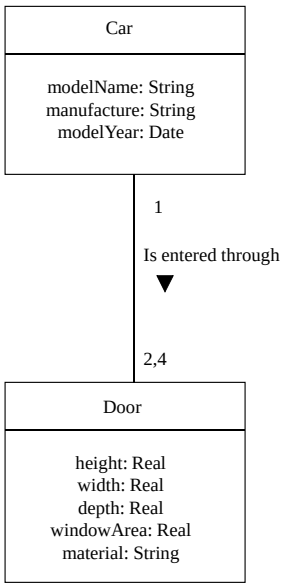


Figure 7.4 Optional (zero-or-one) association

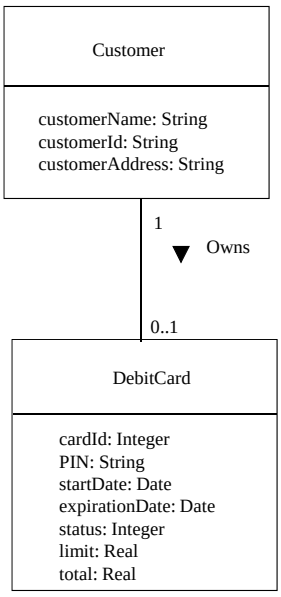


Figure 7.5 Optional (zero,one, or many) association

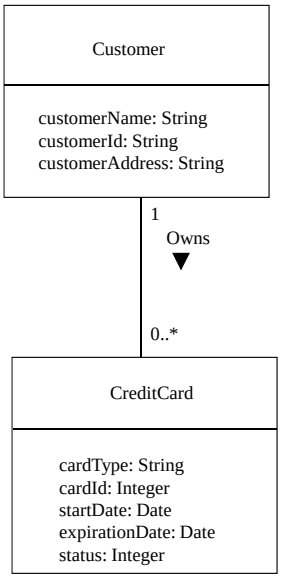
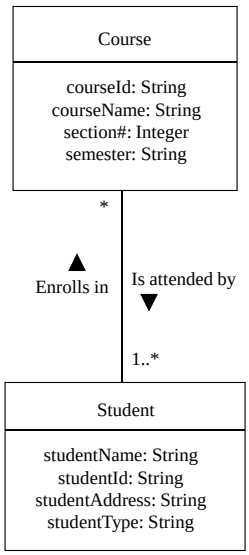


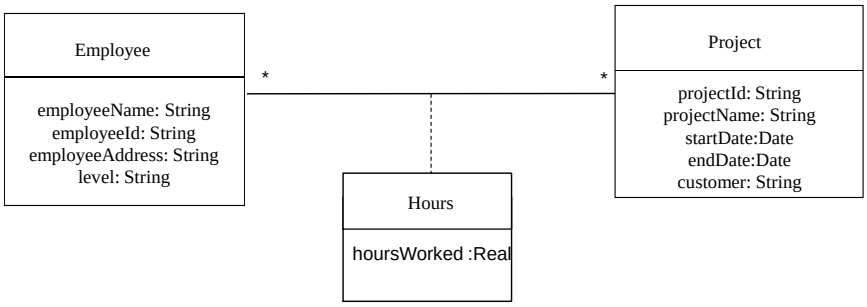
Figure 7.6 Many-to-many association



Association Class

- Class to model 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 has Employee
    - Employee works on project
  - Association Class - Hours
    - Attribute - Hours Worked

Figure 7.11 Example of association class



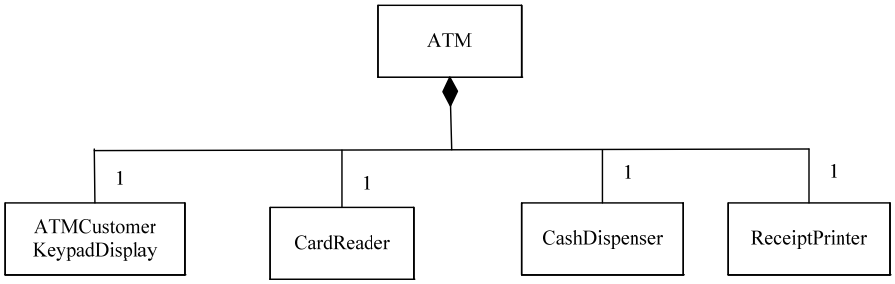
Composition/Aggregation Hierarchy

- Whole/Part Relationships
  - Show parts of more complex class
  - Composition is stronger relationship than aggregation
- IS PART OF Relationship
  - Between part classes and whole (composite or aggregate) class
- Composition is stronger relationship than aggregation

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

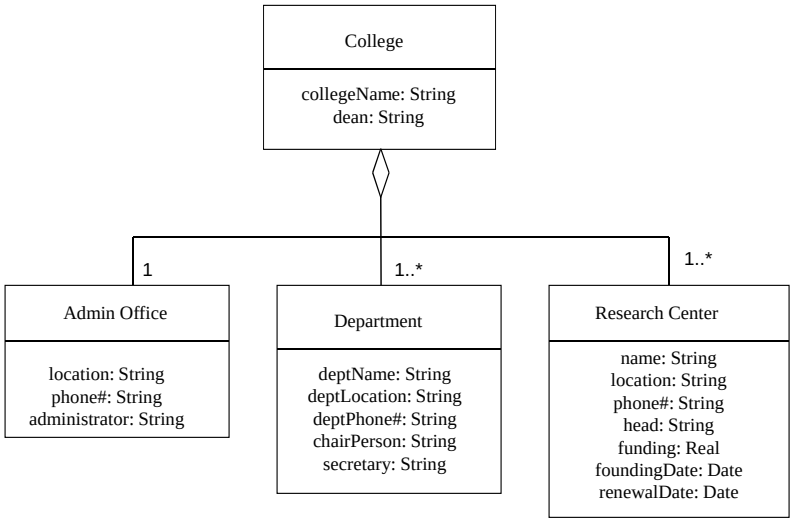
Figure 7.12 Example of composition hierarchy



Composition/Aggregation Hierarchy

- Aggregation Hierarchy
  - Part objects of aggregate object may be created and deleted independently of aggregate object
  - E.g., Aggregate class
    - College
  - Part classes
    - Admin Office IS PART OF College
    - Department IS PART OF College
    - Research Center IS PART OF College

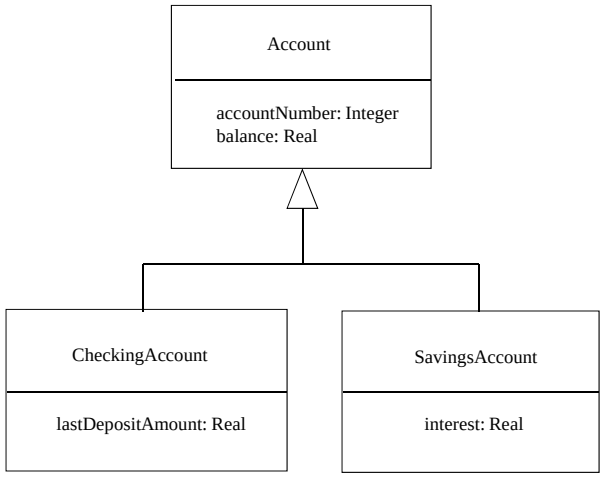
Figure 7.13 Example of aggregation hierarchy



Generalization / Specialization Hierarchy

- Some classes are similar but not identical
  - Have some attributes in common, others different
- Common attributes abstracted into generalized class (superclass)
  - E.g., Account (Account number, Balance)
- Different attributes are properties of specialized class (subclass)
  - E.g., Savings Account (Interest)
- IS A relationship between subclass and superclass
  - Savings Account IS A Account

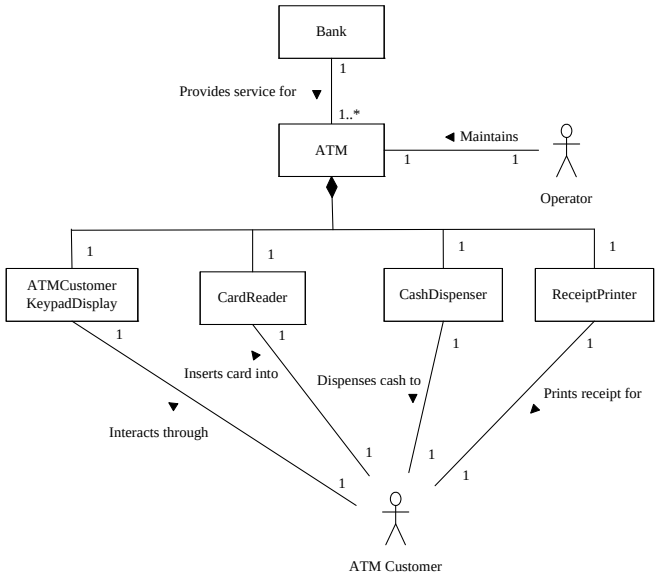
Figure 7.14 Generalization / specialization hierarchy



Static Modeling of Problem Domain

- During Analysis Modeling
  - Conceptual static model
  - Emphasizes real-world classes in the problem domain
  - Does not initially address software classes
  - Emphasis on
    - Physical classes
      - Have physical characteristics (can see, touch)
    - Entity classes
      - Data intensive classes

Figure 7.18 Conceptual static model for Banking System



Software System Context Class Diagram

- Defines boundary between system and external environment
  - May be depicted on Software System Context Class Diagram
- System
  - Depicted 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

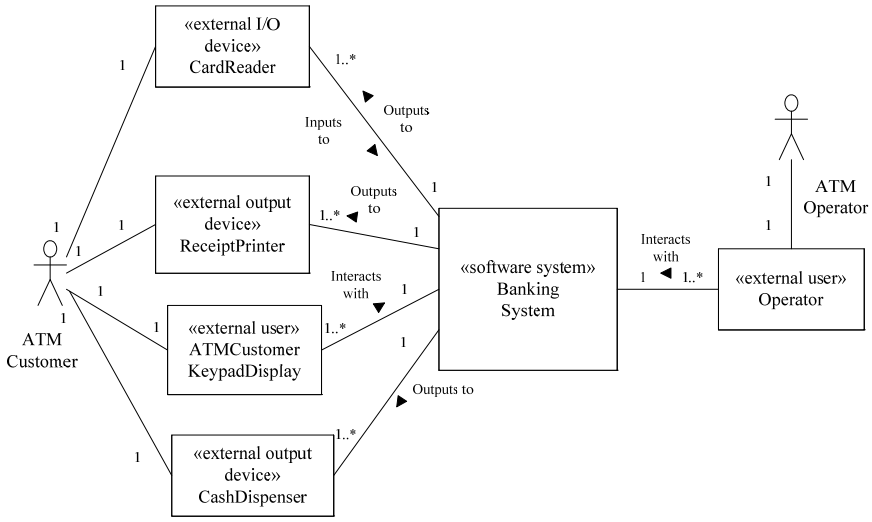
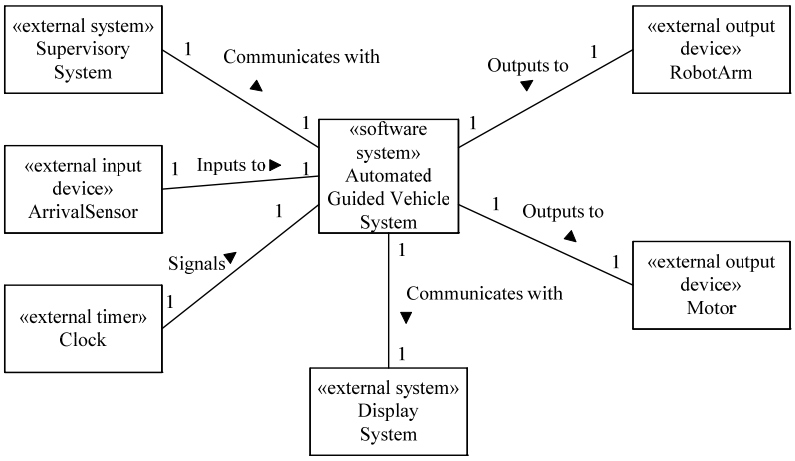


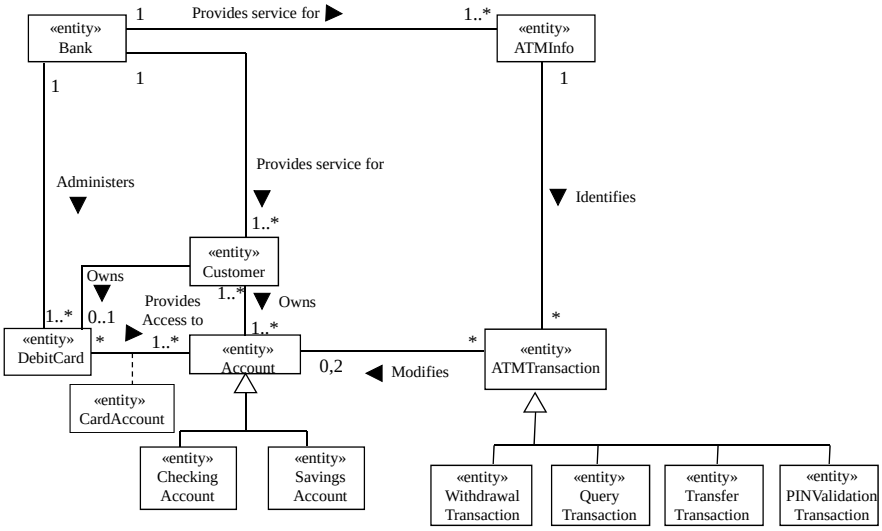
Figure 7.24 Factory Automation System software context class diagram



Static Modeling of Entity Classes

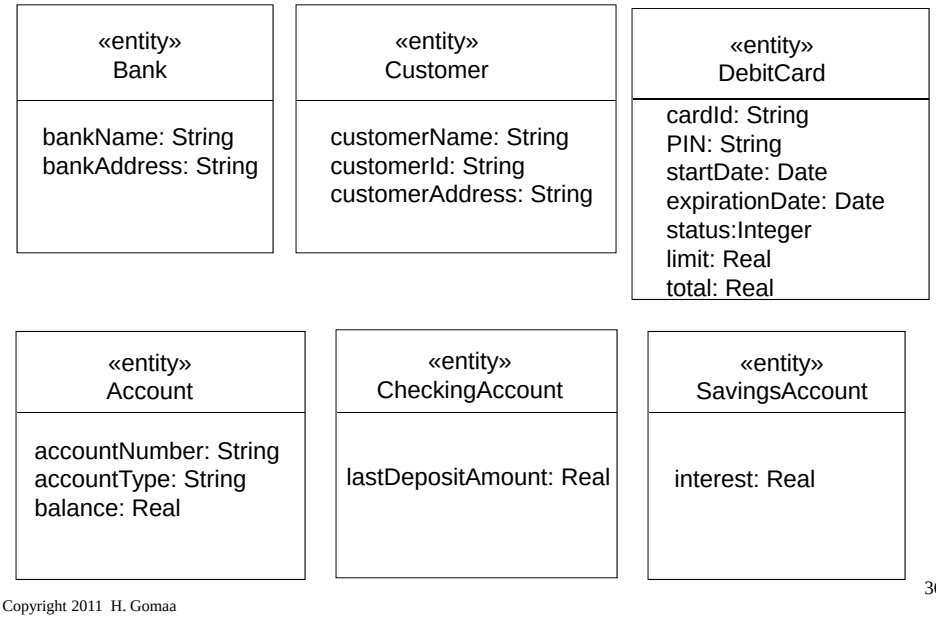
- Entity classes
  - Data intensive classes
  - Store long-living (persistent) data
  - Especially important for Information Systems
    - Many are database intensive
  - Also important for many real-time and distributed applications
- During analysis modeling
  - Model entity classes in the problem domain
  - Attributes
  - Relationships

Figure 21.4 Conceptual static model for Banking System - entity classes



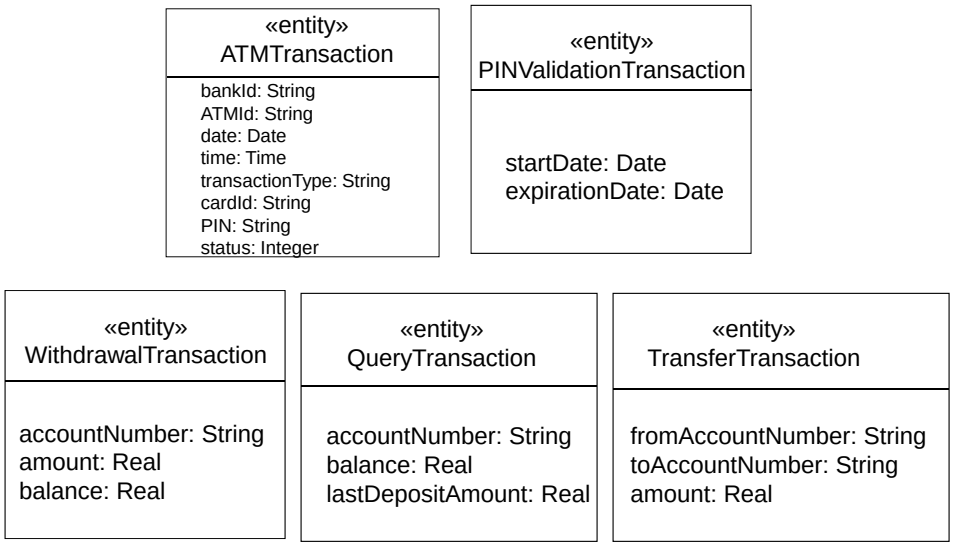
35

Figure 21.5 Conceptual static model for Banking System



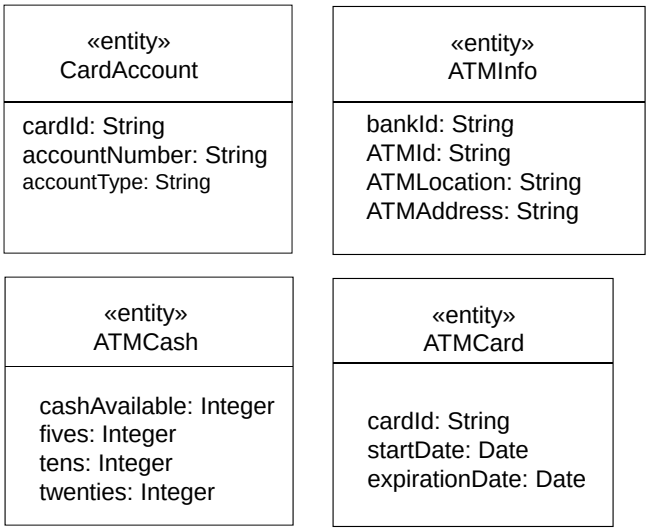
36

Figure 21.6 Conceptual static model for Banking System



37

Figure 21.7 Conceptual static model for Banking System



38

Steps in Using COMET/UML

1 Develop Software Requirements Model
   - Develop Use Case Model (Chapter 6)

2 Develop Software Analysis Model
   - Develop static model of problem domain (Chapter 7)
   - Structure system into objects (Chapter 8)
   - Develop statecharts for state dependent objects (Chapter 10)
   - Develop object interaction diagrams for each use case (Chapter 9, 11)

3 Develop Software Design Model