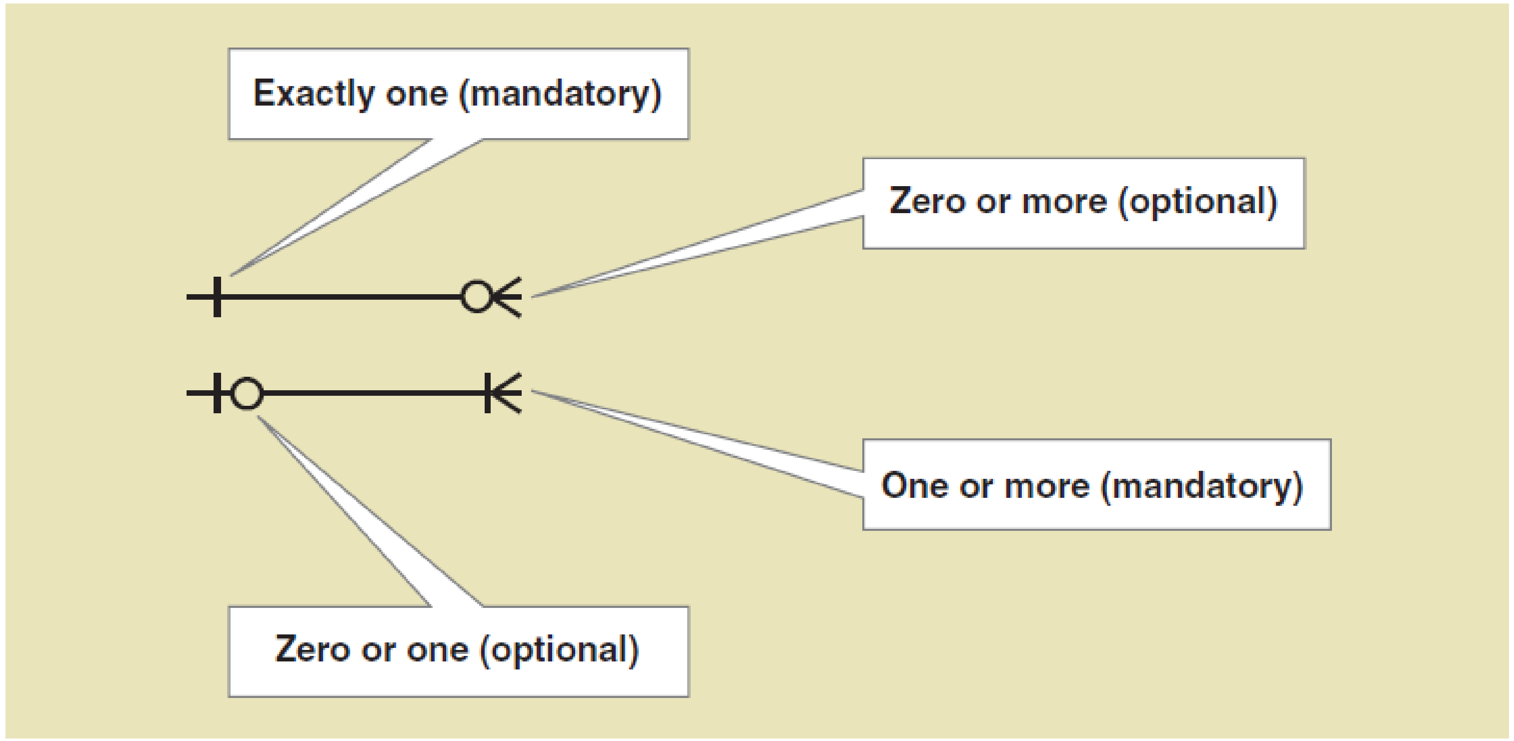
ER Model and UML notation

Other Notations

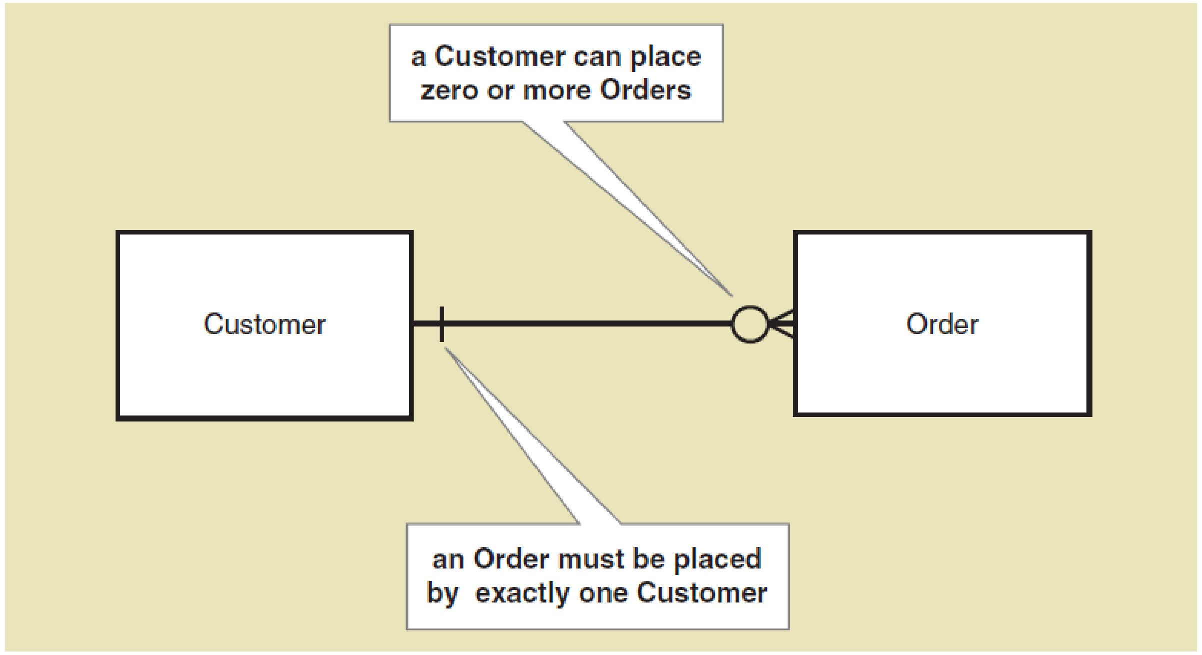
ERD Cardinality Symbols in tools

- Examples of crows feet notation for various cardinalities



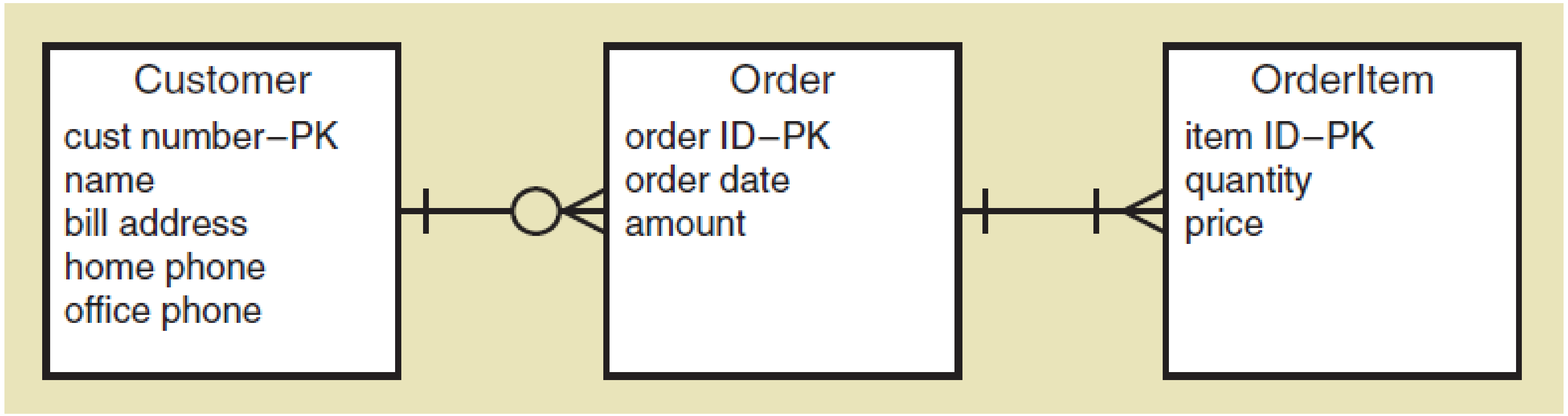
Example of ERD Notation

- ERD Models normally use “crows feet” notation to show cardinality


a Customer can place zero or more Orders
Customer
Order
an Order must be placed by exactly one Customer

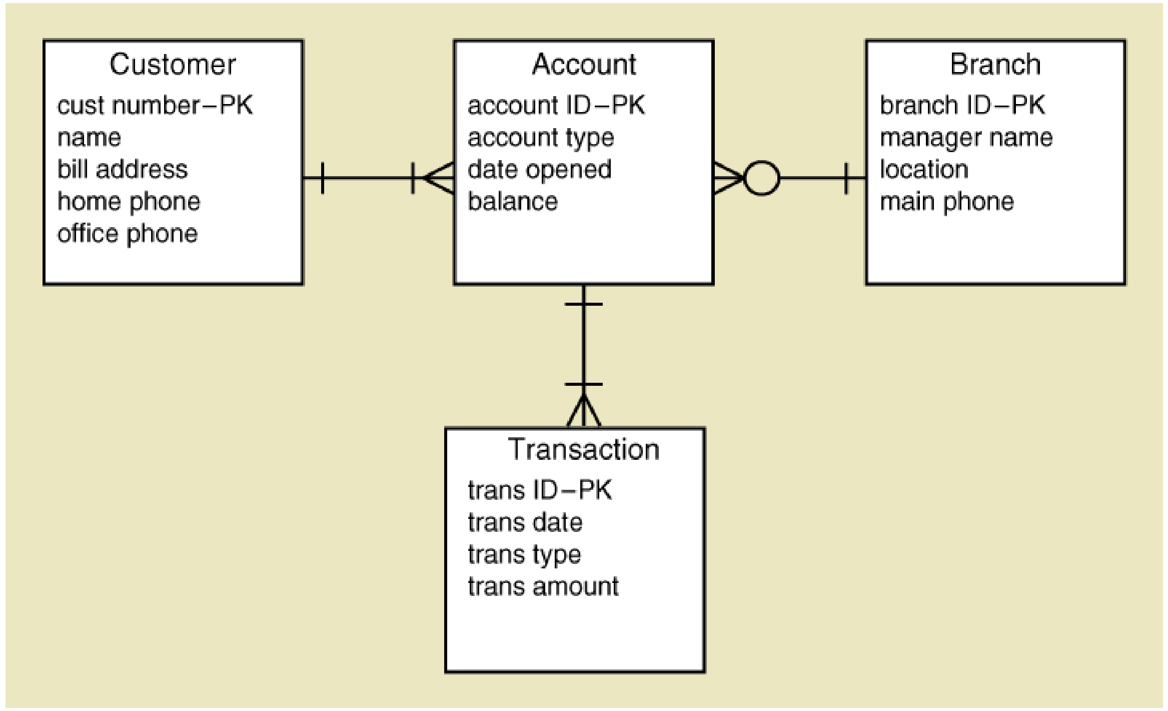
Expanded ERD with Attributes

- ERD with cardinalities and attributes
- There are several different notation methods for attributes in ERD models
- This notation places attributes within data entities



An ERD for a Bank

- What are the key fields?
- How many accounts can a customer have?
- How many branches can a customer be assigned to?
- How many customers can a branch have?

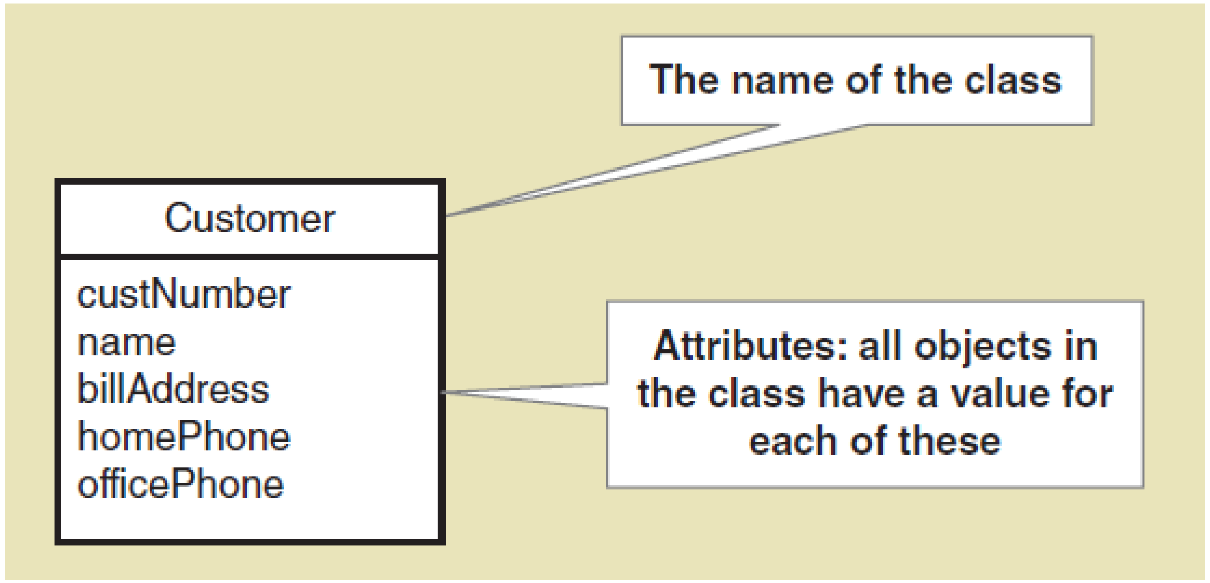


The Domain Model Class Diagram

- Class
  - A type of classification used to describe a collection of objects
- Domain Class
  - Classes that describe objects in the problem domain
- Class Diagram
  - A UML diagram that shows classes with attributes and associations (plus methods if it models software classes)
- Domain Model Class Diagram
  - A class diagram that only includes classes from the problem domain, not software classes so no methods

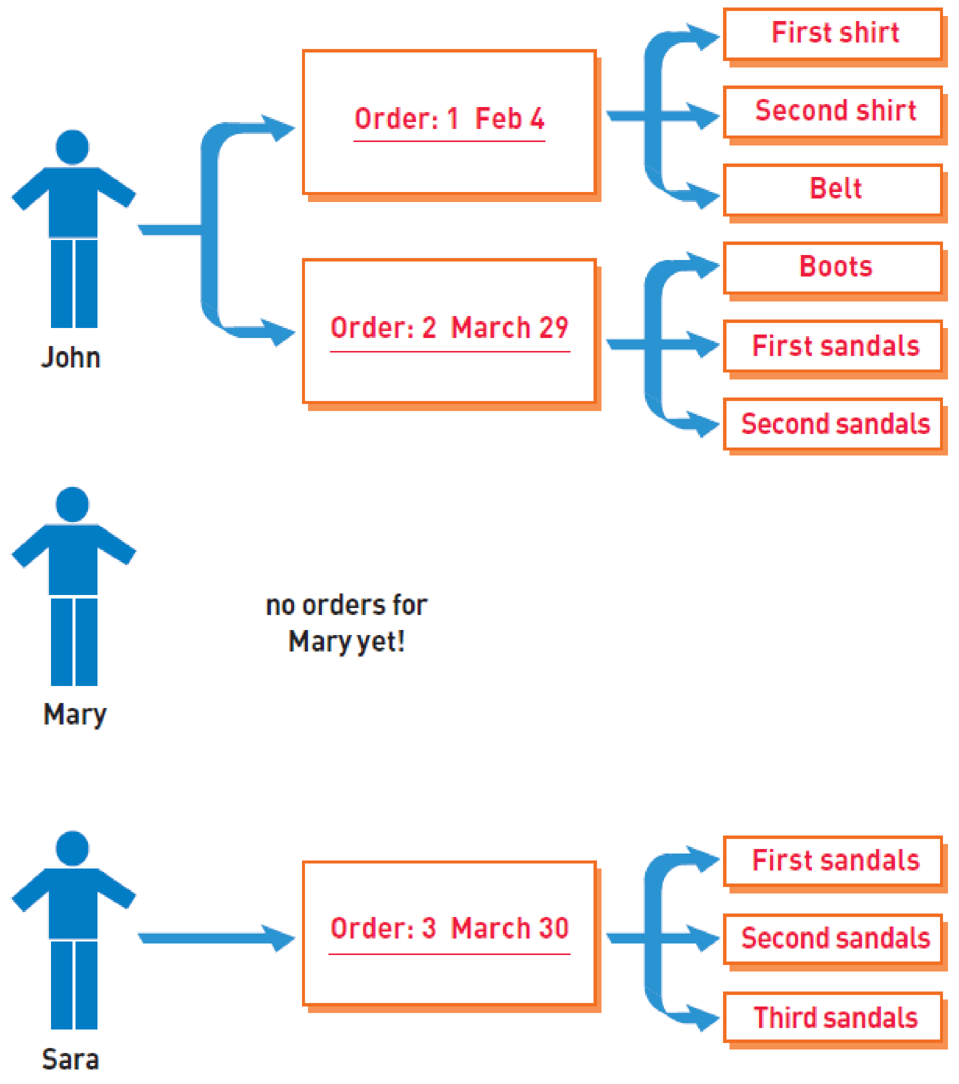
UML Domain Class Notation

- Domain class a name and attributes (no methods)
- Class name is always capitalized
- Attribute names are not capitalized and use **camelback notation** (words run together and second word is capitalized)
- Compound class names also use camelback notation

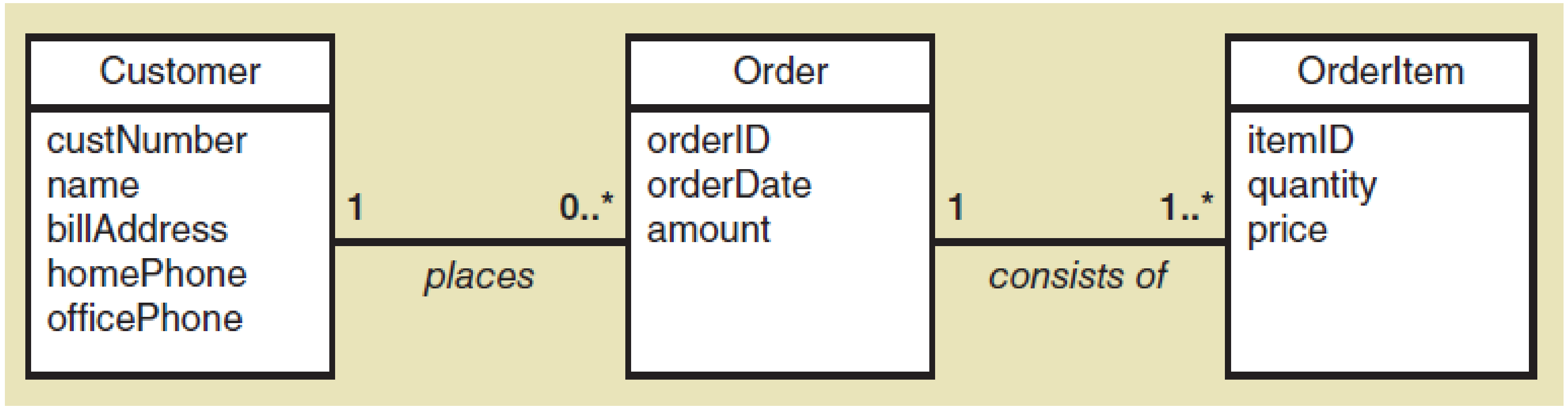


Example

- What are the classes?
- How many relationships?
- What are min and max cardinalities?
- What type of relationships are they?

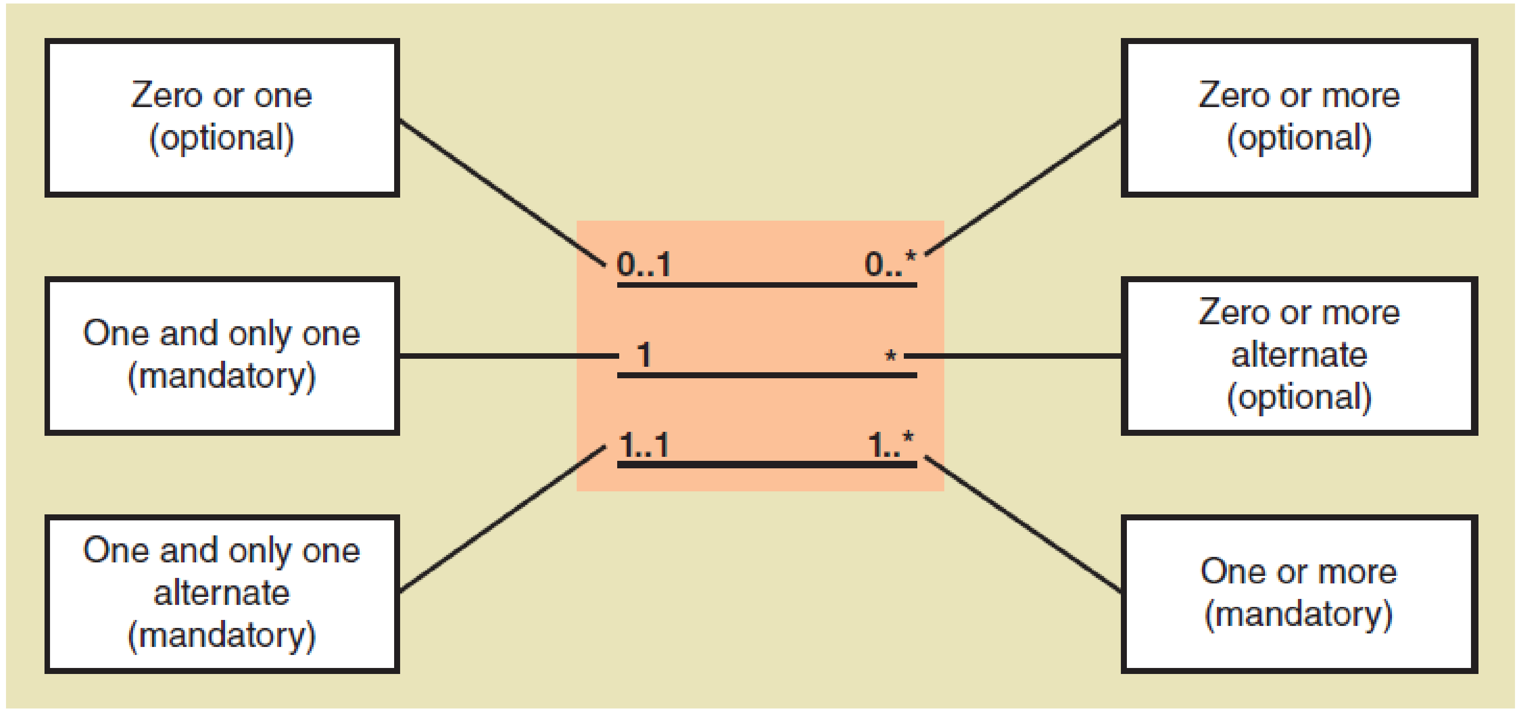


A Simple Domain Model Class Diagram

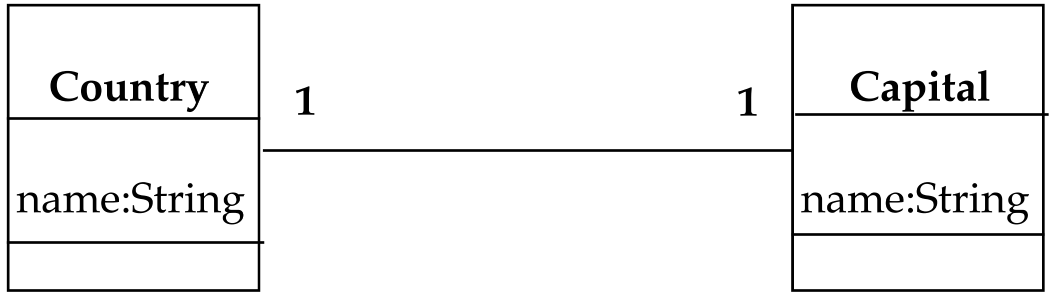


- Note: This diagram matches the semantic net shown previously
  - A customer places zero or more orders
  - An order is placed by exactly one customer
  - An order consists of one or more order items
  - An order item is part of exactly one order

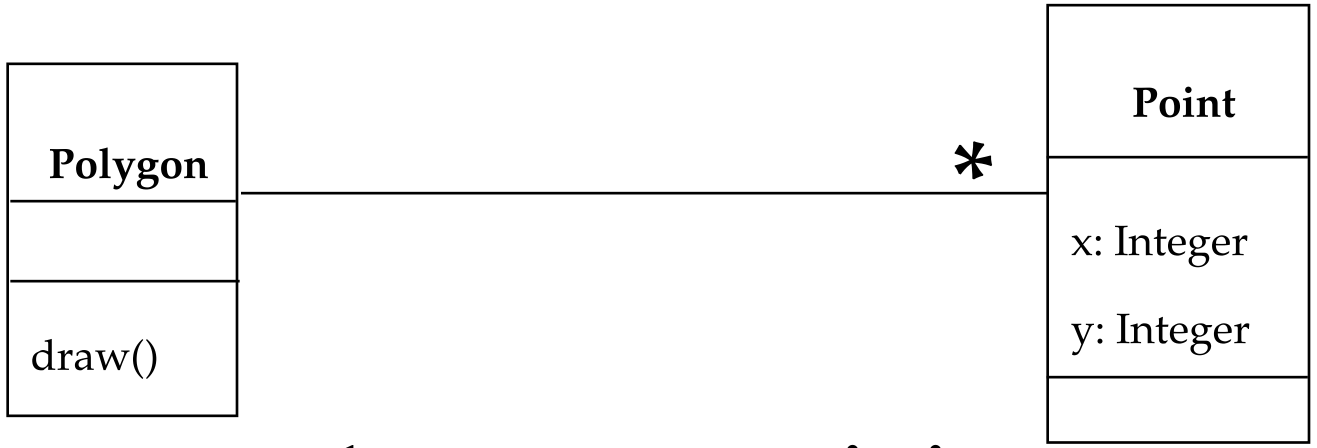
UML Notation for Multiplicity



1-to-1 and 1-to-many Associations



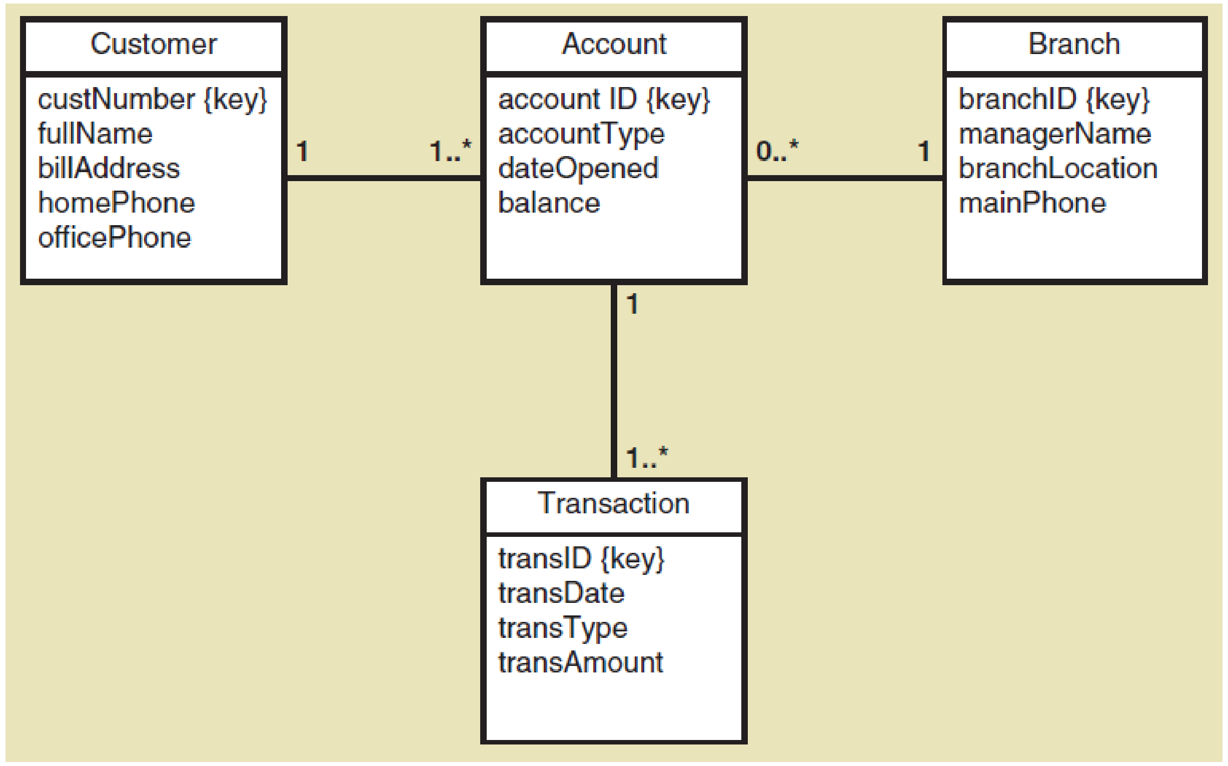
1-to-1 association



1-to-many association

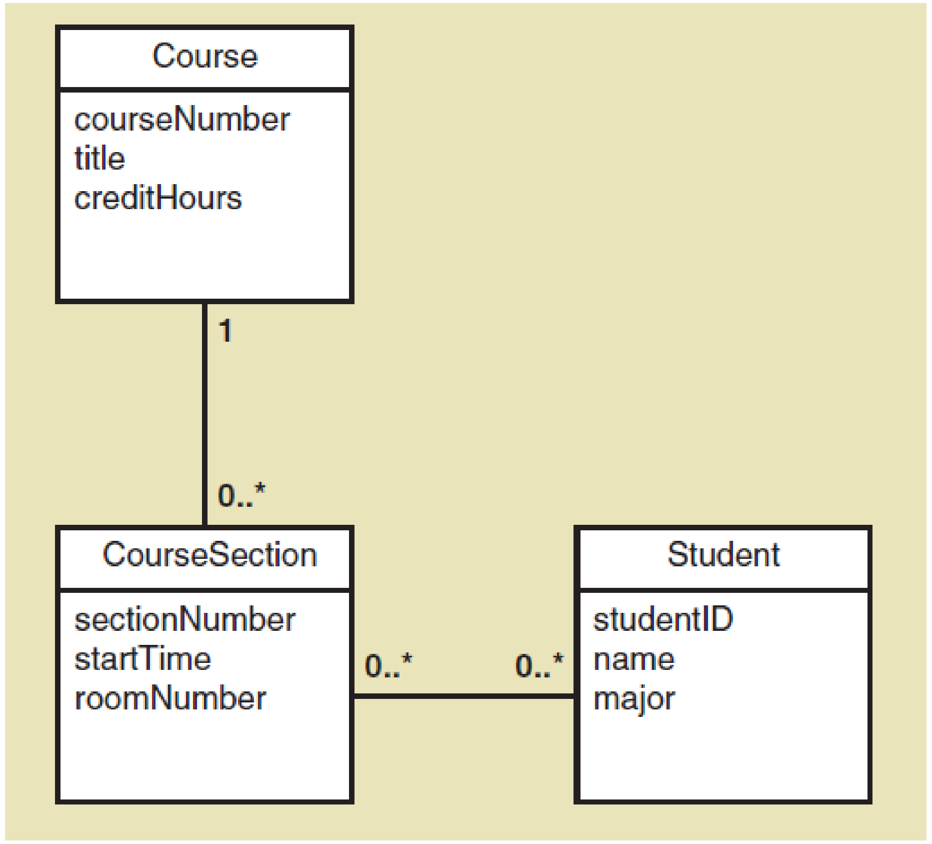
Domain Model Class Diagram

- Bank with many branches
- Note notation for the key
    - Note the precise notation for the attributes (camelback)
    - Note the multiplicity notation



Domain Model Class Diagram

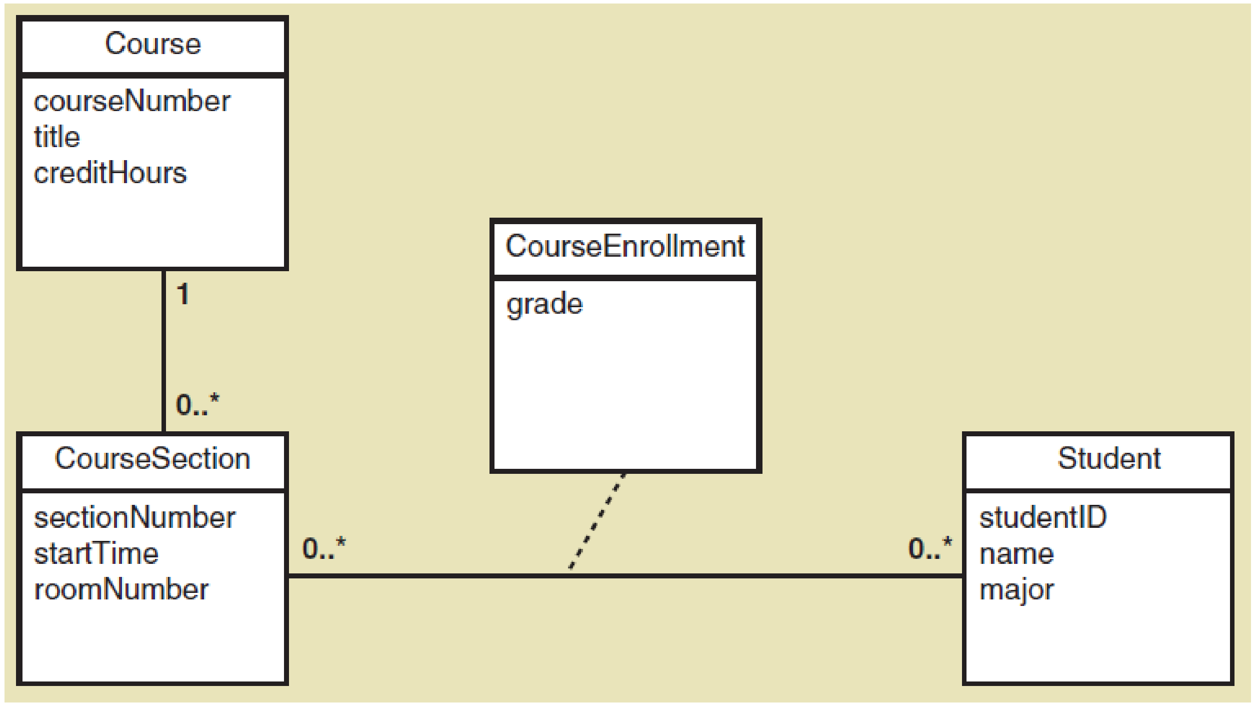
- Course Enrollment at a University
- A Course has many CourseSections
- A CourseSection has many Students and a Student is registered in many CourseSections
- Problem
  - How/where to capture student grades?



Refined Course Enrollment Model

with an Association Class CourseEnrollment

- **Association class**— an association that is treated as a class in a many to many association because it has attributes that need to be remembered (such as grade)

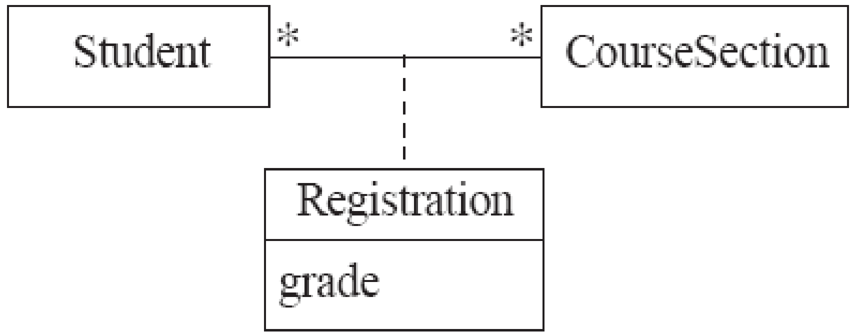


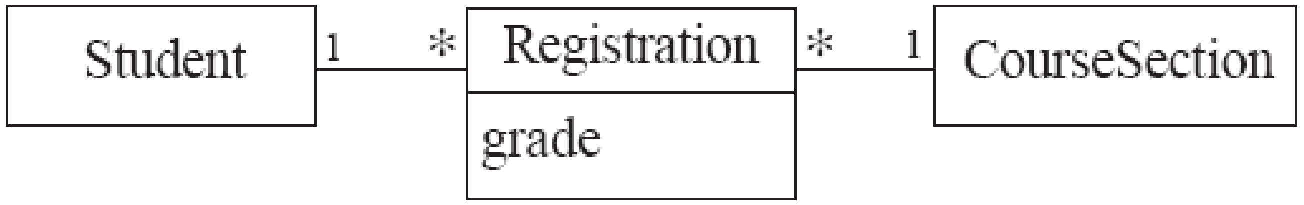
Association Class Properties

- The association class **is** the same "thing" as the association itself
- The unique identifier (key) for the association class is the concatenation of the keys of the attached classes
  - In the previous example the key for CourseSection is CourseNumber+SectionNumber
  - Hence the key for CourseEnrollment is CourseNumber+SectionNumber+StudentID
  - Note: If more information is required to uniquely identify instances of the association class, then the model is incorrect, i.e., if the key cannot be formed by the concatenation of the endpoint keys, it is in error.

Association classes

- Sometimes, an attribute that concerns two associated classes cannot be placed in either of the classes
- The following are equivalent

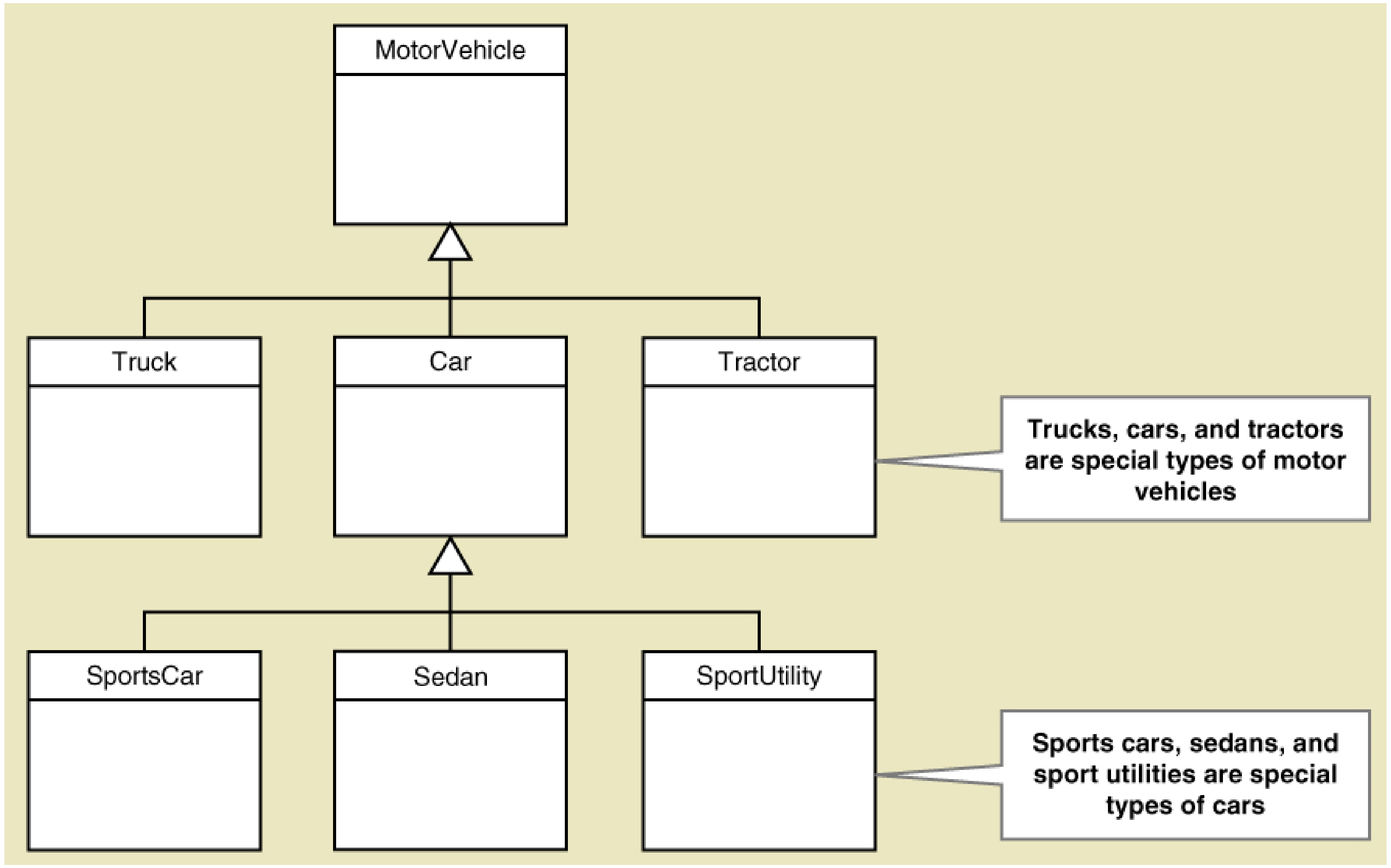




More Complex Issues about Classes:
Generalization/Specialization Relationships

- Generalization/Specialization
  - A hierarchical relationship where subordinate classes are special types of the superior classes. Often called an Inheritance Hierarchy
- Superclass
  - the superior or more general class in a generalization/specialization hierarchy
- Subclass
  - the subordinate or more specialized class in a generalization/specialization hierarchy
- Inheritance
  - the concept that subclasses classes inherit characteristics of the more general superclass

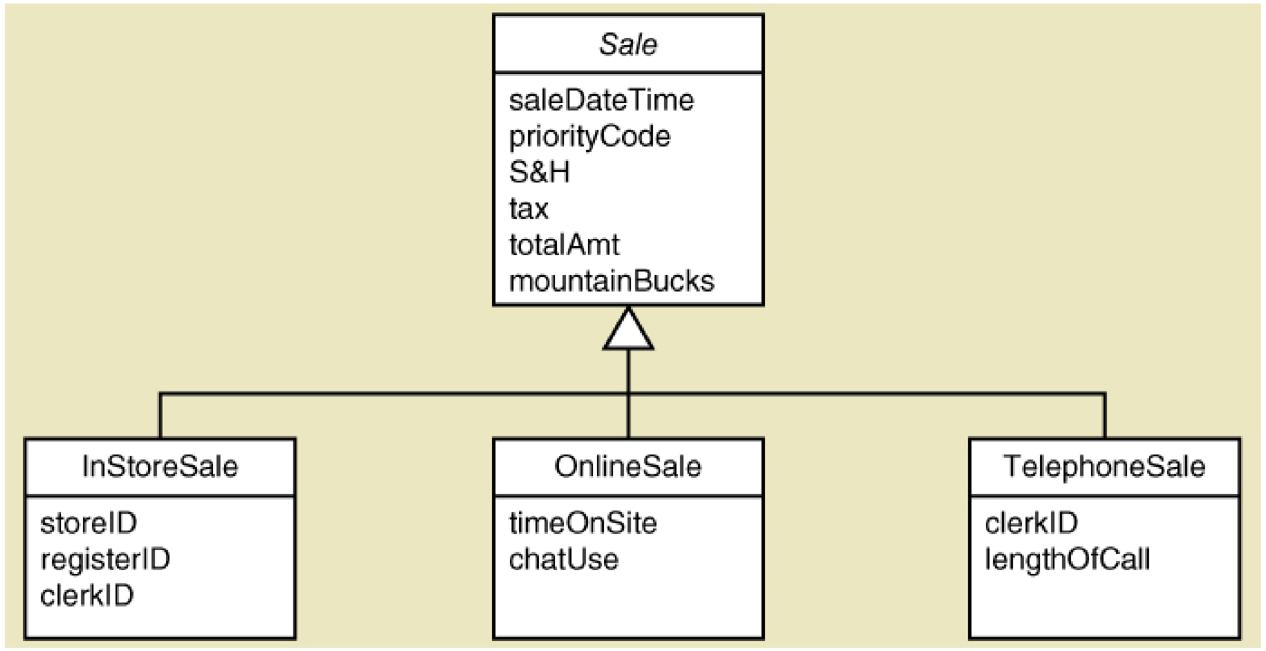
Generalization/Specialization


MotorVehicle
Truck
Car
Tractor
SportsCar
Sedan
SportUtility
Trucks, cars, and tractors are special types of motor vehicles
Sports cars, sedans, and sport utilities are special types of cars

Generalization/Specialization

Inheritance for RMO Three Types of Sales

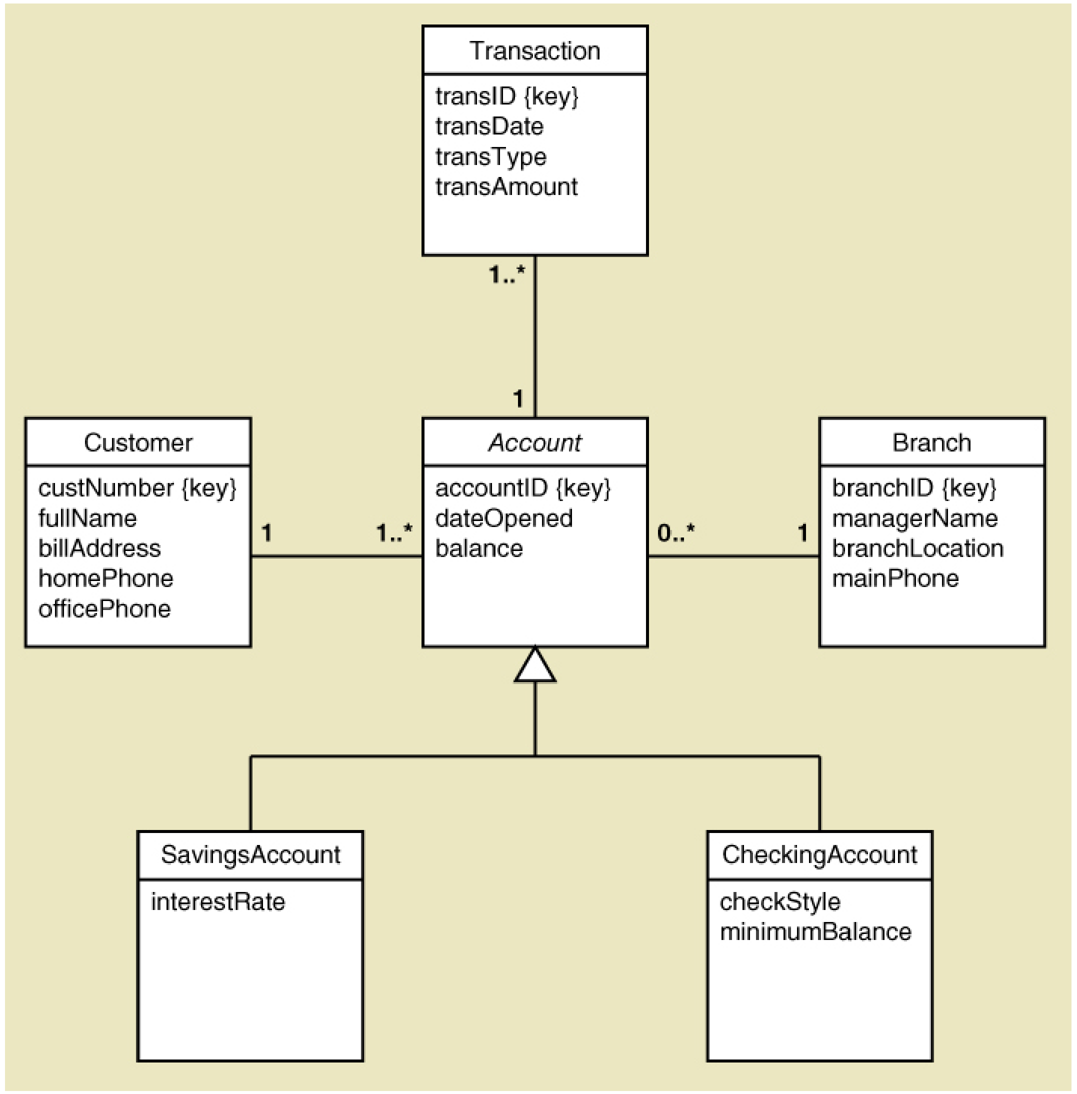
- Abstract class— a class that allow subclasses to inherit characteristics but never gets instantiated. In Italics (*Sale*)
- Concrete class— a class that can have instances
- Inheritance – Attributes of OnlineSale are:
    - timeOnSite, chatUse, saleDateTime, priorityCode, S&H, tax, totalAmt…



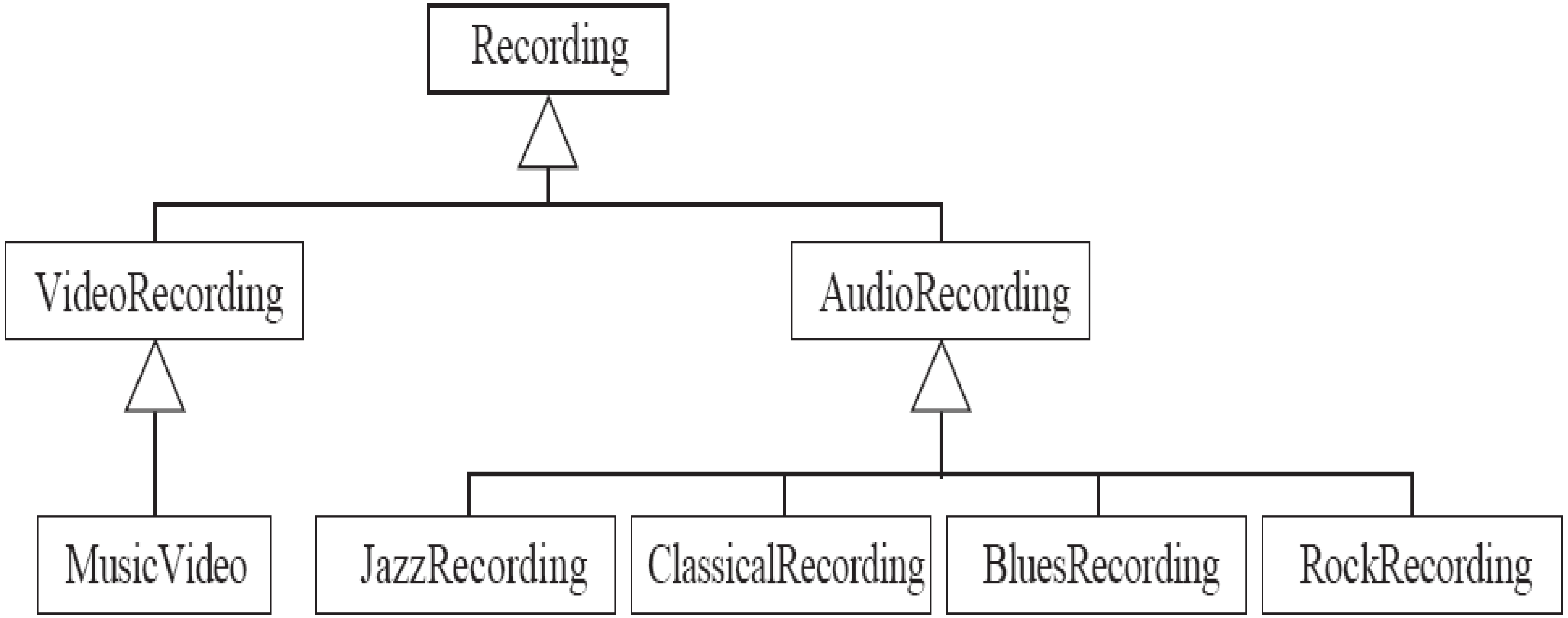
Generalization/Specialization

Inheritance for the Bank with Special Types of Accounts

- A SavingsAccount has 4 attributes
- A CheckingAccount has 5 attributes
- Note: the subclasses inherit the associations too



Avoiding unnecessary generalizations

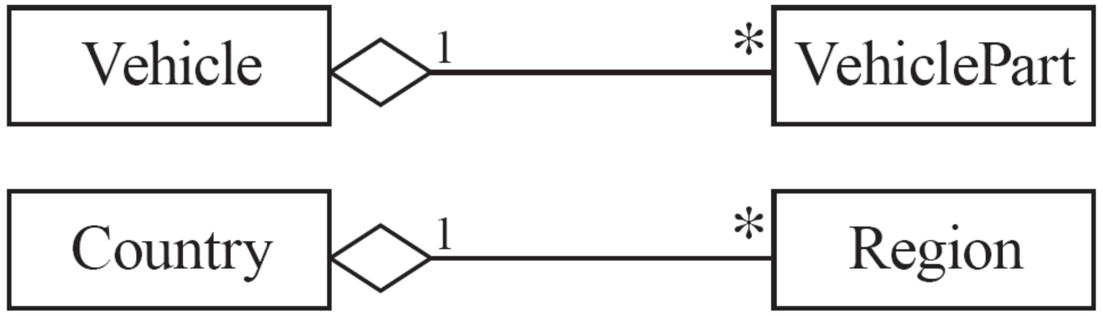


More Complex Issues about Classes:
Whole Part Relationships

- Whole-part relationship— a relationship between classes where one class is part of or a component portion of another class
- Aggregation— a whole part relationship where the component part exists separately and can be removed and replaced (UML diamond symbol on next slide)
  - Computer has disk storage devices (storage devices exist apart from computer)
  - Car has wheels (wheels can be removed and still be wheels)
- Composition— a whole part relationship where the parts cannot be removed (filled in diamond symbol)
  - OrderItem on an Order (without the Order, there are no OrderIterms)
  - Chip has circuits (without the chip, there are no circuits)

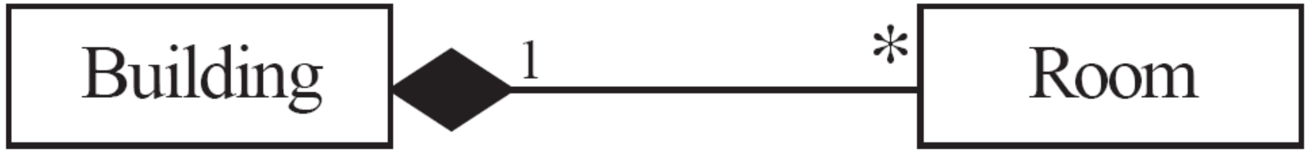
Aggregation

- Aggregations are special associations that represent 'part-whole' relationships.
  - The 'whole' side is often called the *assembly* or the *aggregate*
  - This symbol is a shorthand notation association named `isPartOf`



Composition

- A *composition* is a strong kind of aggregation
  - if the aggregate is destroyed, then the parts are destroyed as well



More on UML Relationships

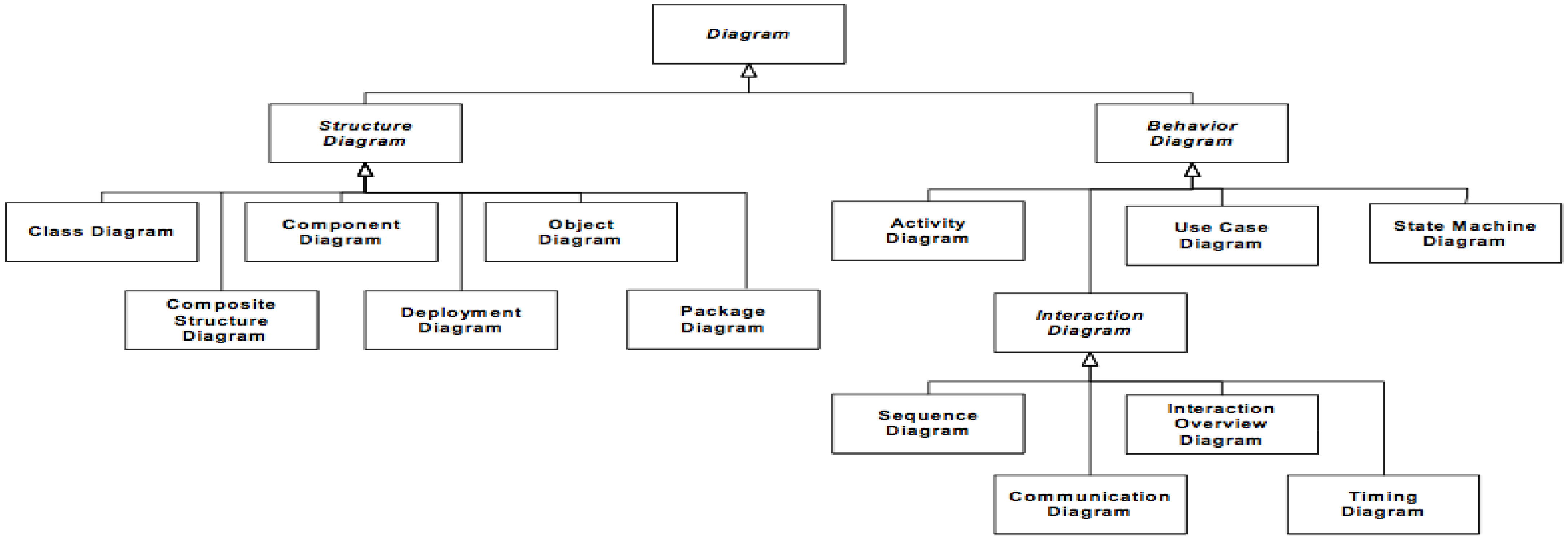
- There are actually three types of ***relationships*** in class diagrams
  - Association Relationships
    - These are associations discussed previously, just like ERD relationships
  - Whole Part Relationships
    - One class is a component or part of another class
  - Generalizations/Specialization Relationships
    - Inheritance
- Try not to confuse relationship with association

We use Models to describe Software Systems

- System model: Object model + functional model + dynamic model

- Object model: What is the structure of the system?
- Functional model: What are the functions of the system?
- Dynamic model: How does the system react to external events?

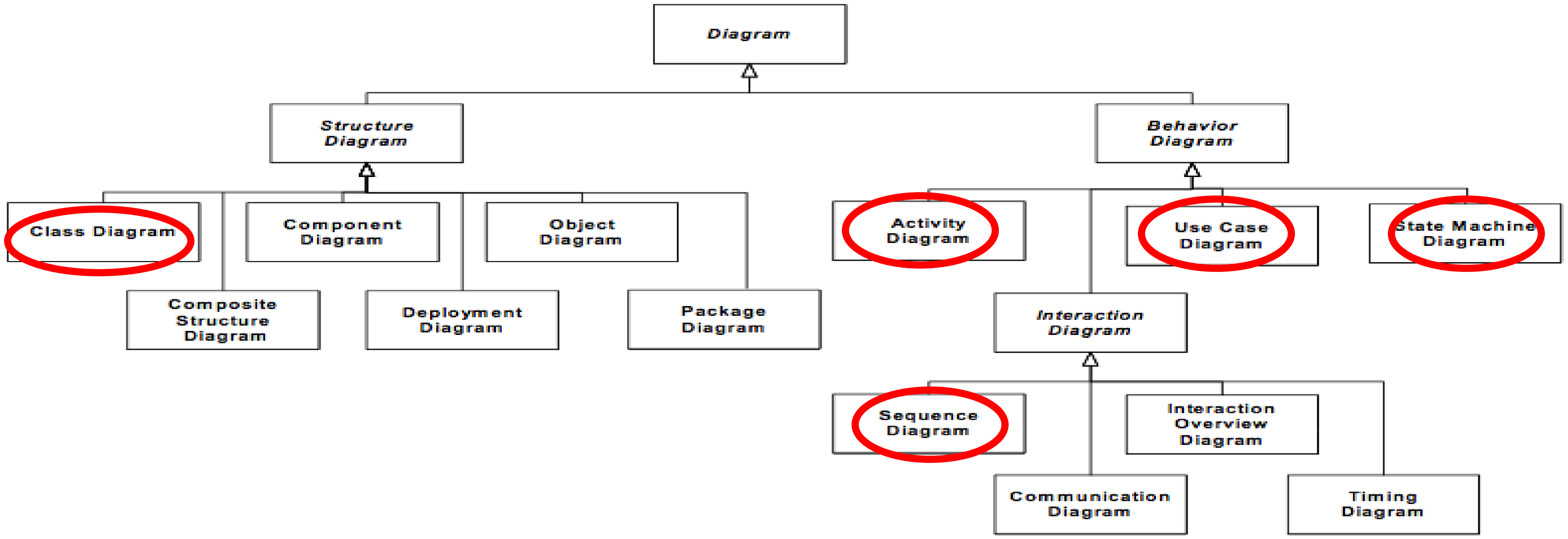
Another view on UML Diagrams



UML First Pass

- Use case diagrams
  - Describe the functional behavior of the system as seen by the user
- Class diagrams
  - Describe the static structure of the system: Objects, attributes, associations
- Sequence diagrams
  - Describe the dynamic behavior between objects of the system
- Statechart diagrams
  - Describe the dynamic behavior of an individual object
- Activity diagrams
  - Describe the dynamic behavior of a system, in particular the workflow.

Another view on UML Diagrams



We use Models to describe Software Systems

- System model: Object model + Functional model + Dynamic model

- Object model: What is the structure of the system?
  - UML Notation: Class diagrams
- Functional model: What are the functions of the system?
  - UML Notation: Use case diagrams
- Dynamic model: How does the system react to external events?
  - UML Notation: Sequence, State chart and Activity diagrams