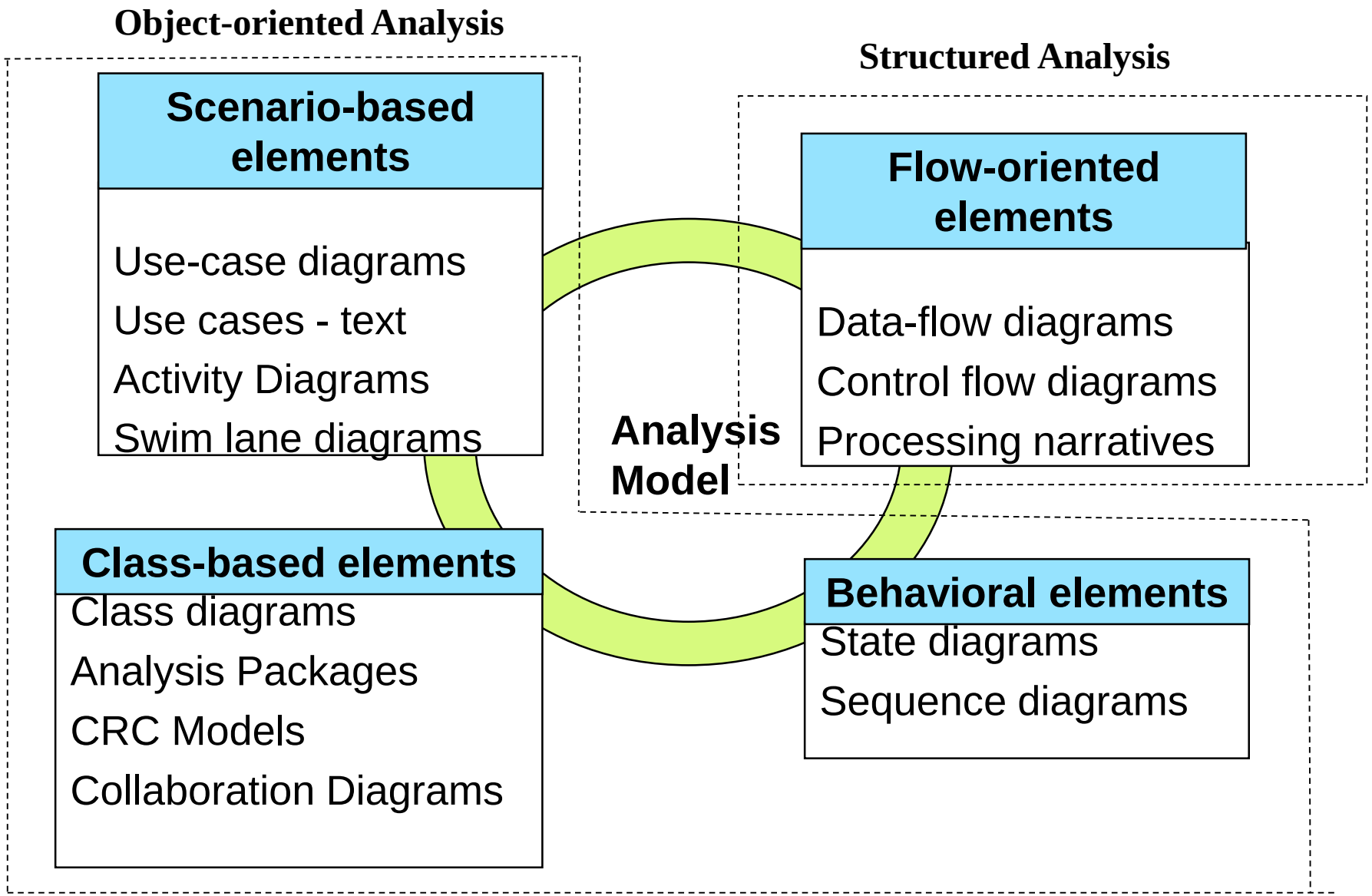
CSc 131 – Computer Software Engineering

Analysis Modeling (Continue)

Class-based modeling

Class-based Modeling

Elements of the Analysis Model



Elements of the Analysis Model

Class-Based Modeling

1. Identify analysis classes by examining the problem statement
2. Use a “grammatical parse” to isolate potential classes
3. Identify the attributes of each class
4. Identify operations that manipulate the attributes
5. Collaborations that occur between the classes that are defined.

Identifying Analysis Classes

1) Perform a grammatical parse of the problem statement or use cases
2) Classes are determined by underlining each noun or noun clause
3) A class required to implement a solution is part of the solution space
4) A class necessary only to describe a solution is part of the problem space
5) A class should NOT have an imperative procedural name (i.e., a verb)
6) List the potential class names in a table and "classify" each class according to some taxonomy and class selection characteristics
7) A potential class should satisfy nearly all (or all) of the selection characteristics to be considered a legitimate problem domain class

Potential classes	General classification	Selection Characteristics

Identifying Classes (Example)

Potential class	Classification	Accept / Reject
homeowner	role; external entity	reject: 1, 2 fail
sensor	external entity	accept
control panel	external entity	accept
installation	occurrence	reject
(security) system	thing	accept
number, type	not objects, attributes	reject: 3 fails
master password	thing	reject: 3 fails
telephone number	thing	reject: 3 fails
sensor event	occurrence	accept
audible alarm	external entity	accept: 1 fails
monitoring service	organizational unit; ee	reject: 1, 2 fail

Grammatical Parsing

Write an informal description of the problem. The customer requirements document is one such description.

Underline all nouns in the description

Decide which of these are really objects which the project requires and organize them in related clusters

Grammatical Parsing

University **Bank** will be opening in **Oxford, Mississippi**, in **January**, 2000. We plan to use a full service automated teller machine (**ATM**) system.The ATM system will interact with the **customer** through a **display screen, numeric and special input keys**, a **bankcard reader,** a **deposit slot**, and a **receipt printer**.Customers may make **deposits**, withdrawals, and balance inquires using the ATM machine, but the update to **accounts** will be handled through an **interface** to the **Accounts system**.Customers will be assigned a Personal Identification Number (**PIN**) and clearance level by the **Security system**. The PIN can be verified prior to any transaction.In the future, we would also like to support routine operations such as a **change of address** or **phone number** using the ATM

Grammatical Parsing

University Bank will be opening in Oxford, Mississippi, in January, 2000. We plan to use a full service automated teller machine (ATM) system.The *ATM system* will interact with the *customer* through a *display screen*, *numeric* and *special input keys*, a *bankcard reader*, a *deposit* slot, and a *receipt printer*.Customers may make *deposits*, *withdrawals*, and *balance inquires* using the ATM machine, but the update to *accounts* will be handled through an interface to the *Accounts system*.Customers will be assigned a *Personal Identification Number (PIN)* and *clearance level* by the *Security system*. The PIN can be verified prior to any transaction.In the future, we would also like to support routine operations such as a *change of address* or *phone number* using the ATM

Typical Classes (a reminder)

External entities - printer, user, sensor

Things - reports, displays, signals

Occurrences or events (e.g., interrupt, alarm)

Roles (e.g., manager, engineer, salesperson)

Organizational units (e.g., division, team)

Places (e.g., manufacturing floor or loading dock)

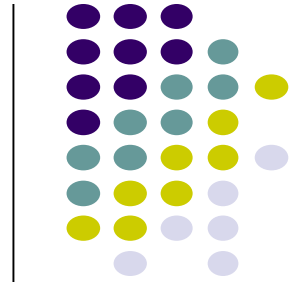
Structures (e.g., sensors, four-wheeled vehicles, or computers)

But, how do we select classes?

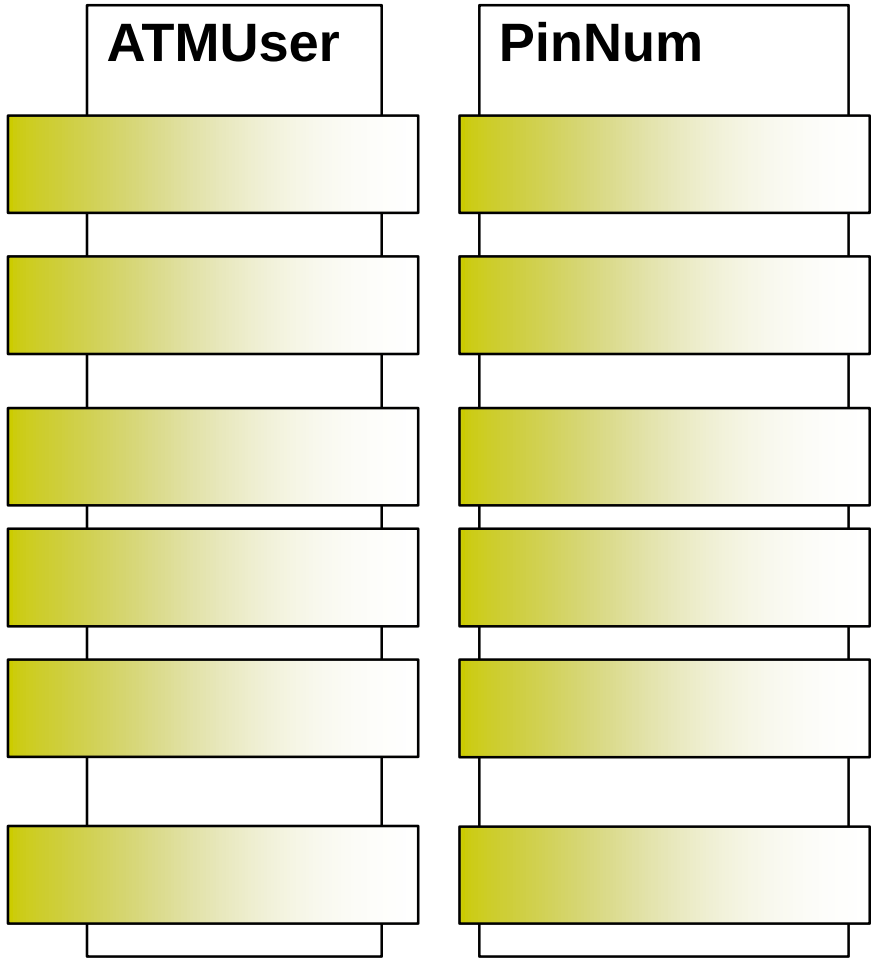
Selecting Classes—Criteria

- retained information – information about it must be remembered
- needed services – operations that change the attributes
- multiple attributes – if it is only one attribute,
  probably should be part of another class
- common attributes – common things for all instances of a class
- common operations – for all instances of the class
- essential requirements – appear in the PROBLEM space
  (remember we're doing analysis modeling!)

Selecting Classes—Example



- retained information
- needed services
- multiple attributes
- common attributes
- common operations
- essential requirements



Defining Attributes

Attributes describe a class that has been selected for inclusion in the analysis model.

Identifying Attributes

Attributes can be thought as a simple association with a value which is not an object

Examples: name, age, weight

Usually corresponding to nouns followed by possessive phrases

Examples: “color of the car”, “age of the donor”

Less likely to be fully described in the problem statement

Included in the class box diagram

A data object contains a set of attributes that act as an aspect, quality, characteristic, or descriptor of the object

Defining Operations

Do a grammatical parse of a processing narrative and look at the verbs

Operations can be divided into four broad categories:

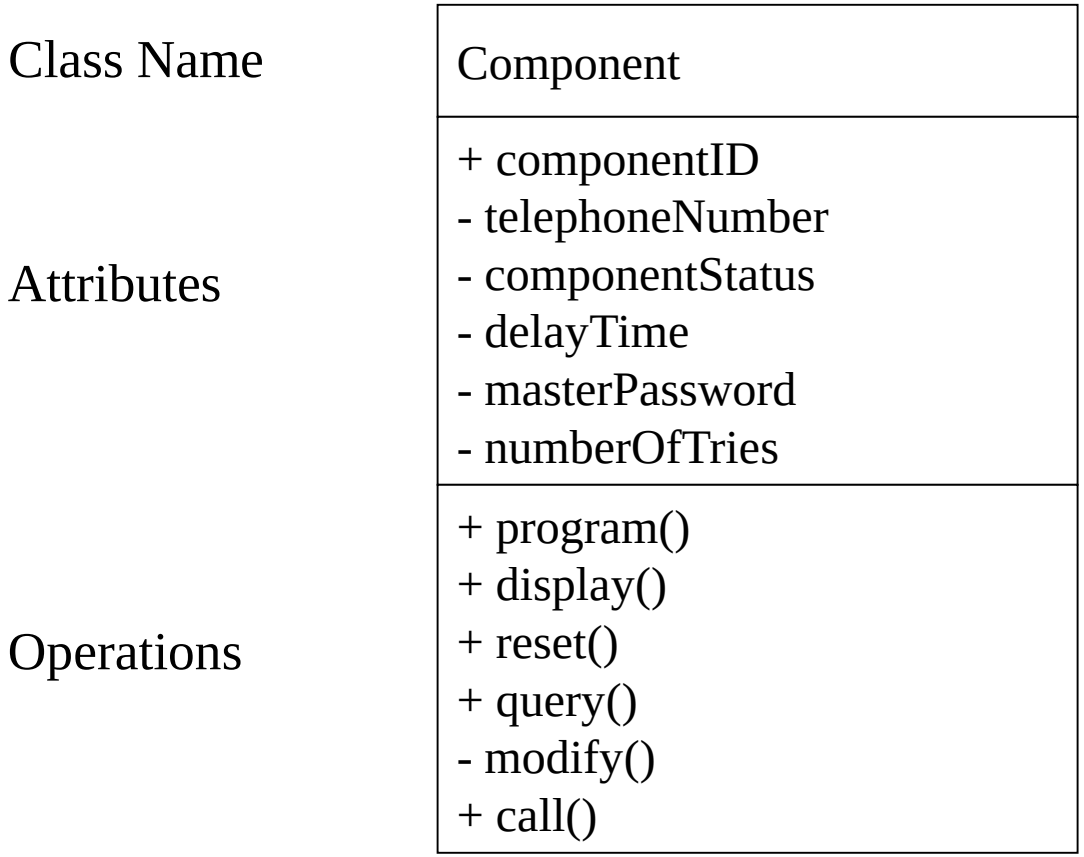
(1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting, i.e. **withdraw**)

(2) operations that perform a computation (i.e. **compute excess charge**)

(3) operations that inquire about the state of an object (i.e. **customer has sufficient fund** ?), and

(4) operations that monitor an object for the occurrence of a controlling event.

Example Class Box



Association, Generalization and Dependency (Ref: Fowler)

Association

- Represented by a solid line between two classes directed from the source class to the target class
- Used for representing (i.e., pointing to) object types for attributes
- May also be a part-of relationship (i.e., aggregation), which is represented by a diamond-arrow

Generalization

- Portrays inheritance between a super class and a subclass
- Is represented by a line with a triangle at the target end

Dependency

- A dependency exists between two elements if changes to the definition of one element (i.e., the source or supplier) may cause changes to the other element (i.e., the client)
- Examples
  - *One class calls a method of another class*
  - *One class utilizes another class as a parameter of a method*

UML Class Diagrams

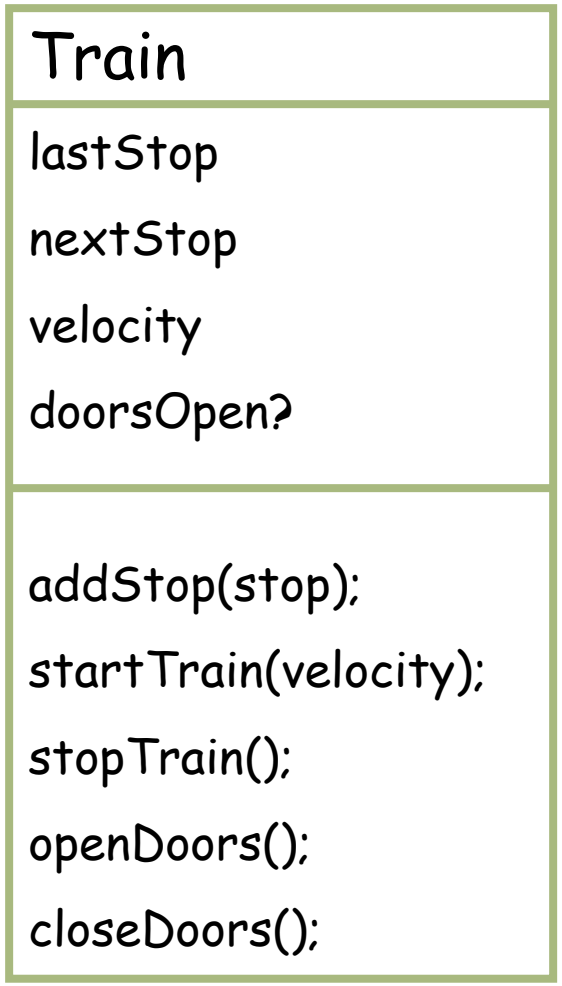
Describe classes

- In the OO sense

Class diagrams are static -- they display what interacts but not what happens when they do interact

Each box is a class

- List fields
- List methods



Class Diagrams: Relationships

Many different kinds of edges to show different relationships between classes

Mention just a couple

Association

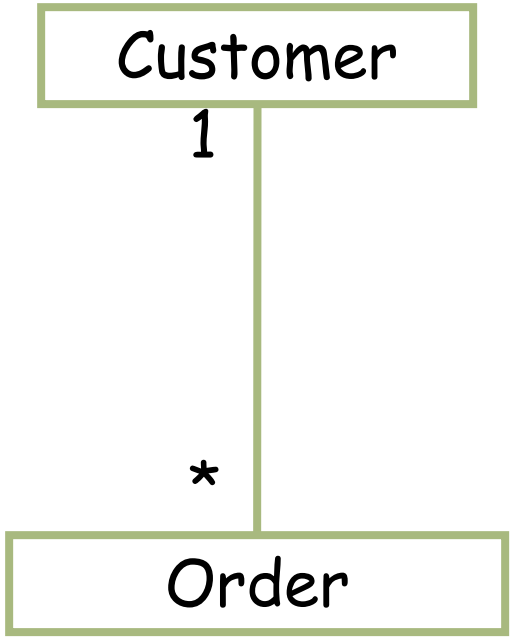
Association between two classes

if an instance of one class must know about the other in order to perform its work.

Label endpoints of edge with cardinalities

Use * for arbitrary

Can be directional (use arrows in that case)



Aggregation

An association in which one class belongs to a collection

- Shared: An object can exist in more than one collections
- No ownership implied

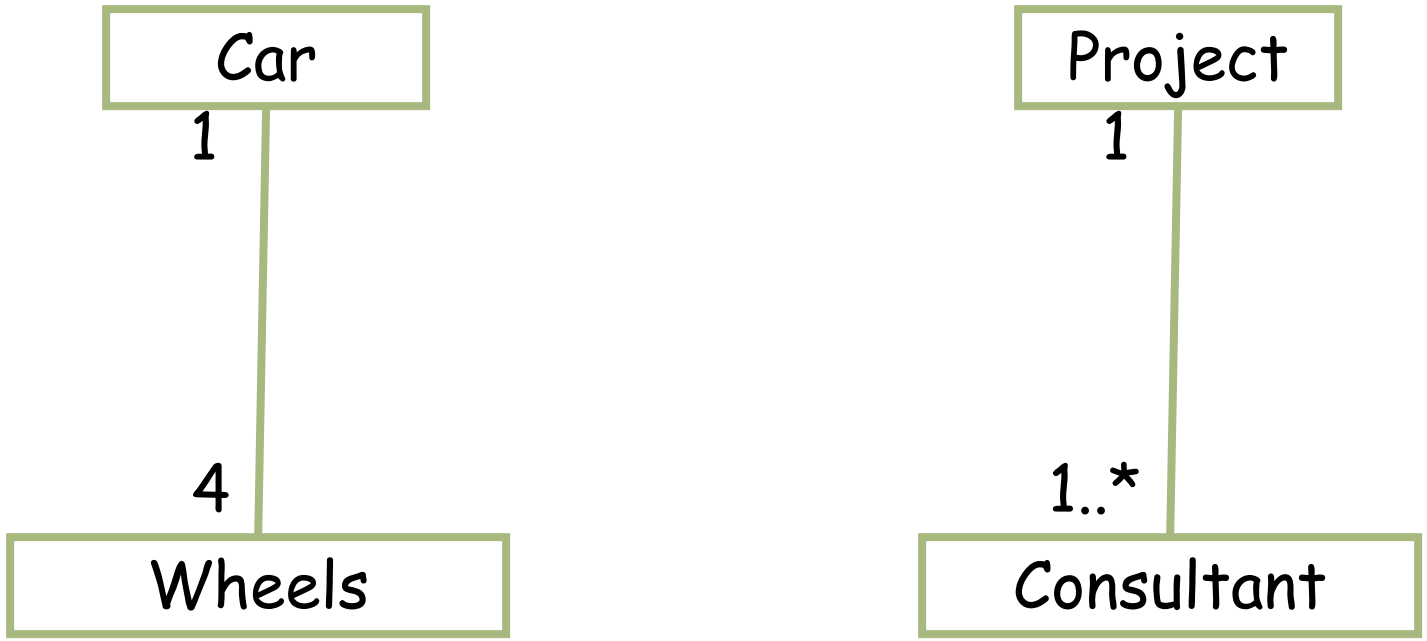
Denoted by hollow diamond on the “contains” side

Composition

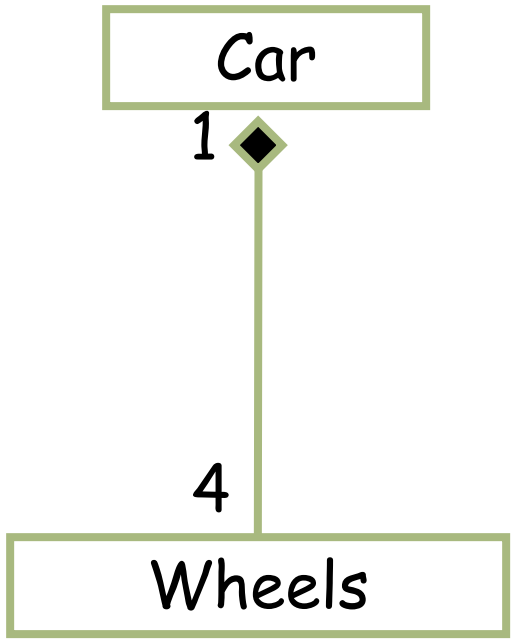
An association in which one class belongs to a collection

- No Sharing: An object cannot exist in more than one collections
- Strong “has a” relationship
- Ownership

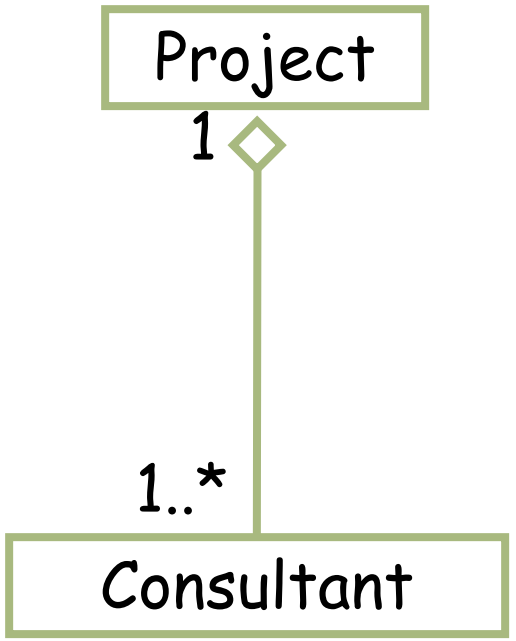
Denoted by filled diamond on the “contains” side

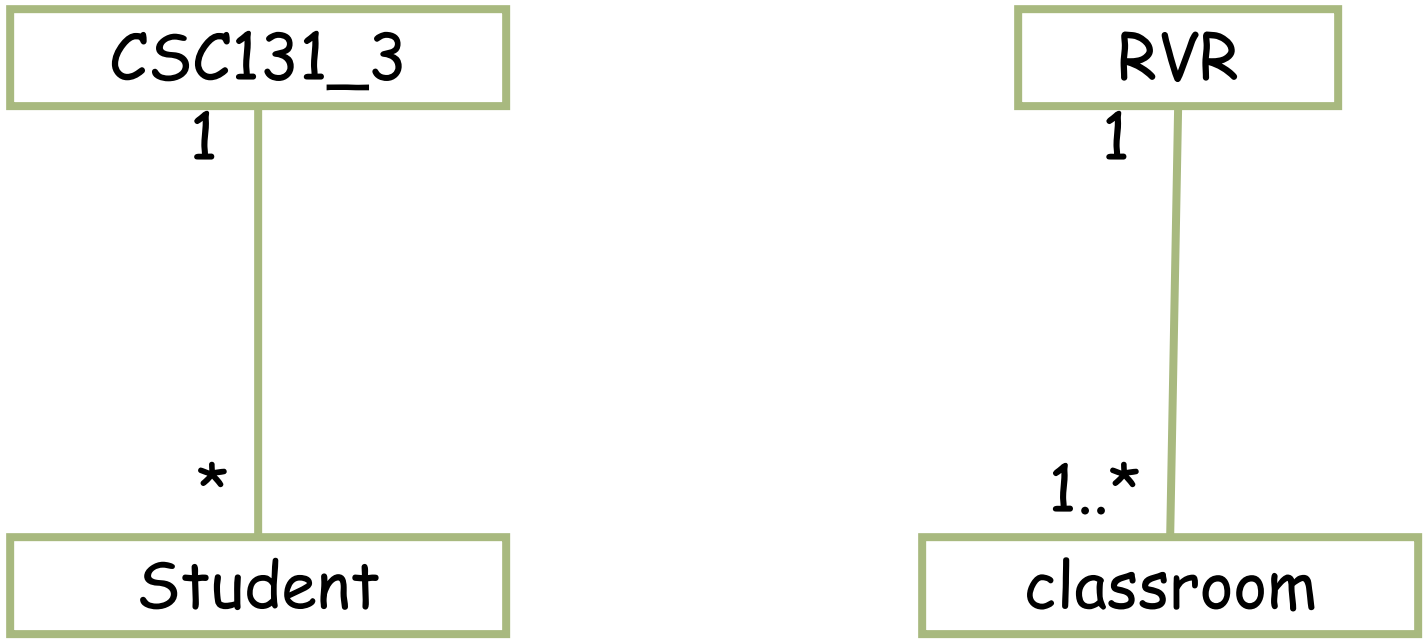

Car
1
4
Wheels
Project
1
1..*
Consultant

Composition

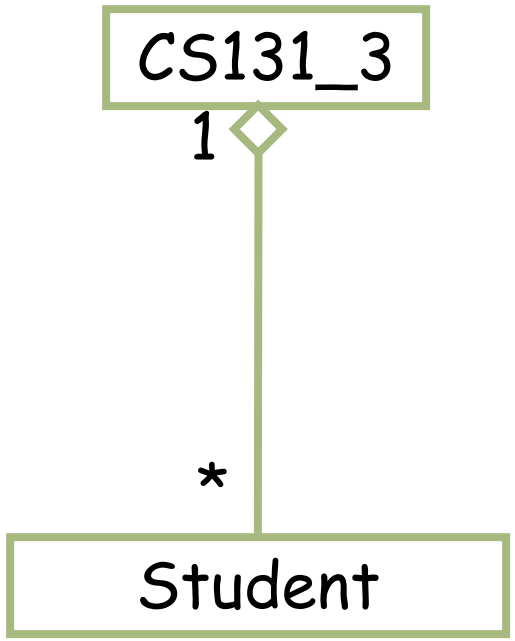


Aggregation

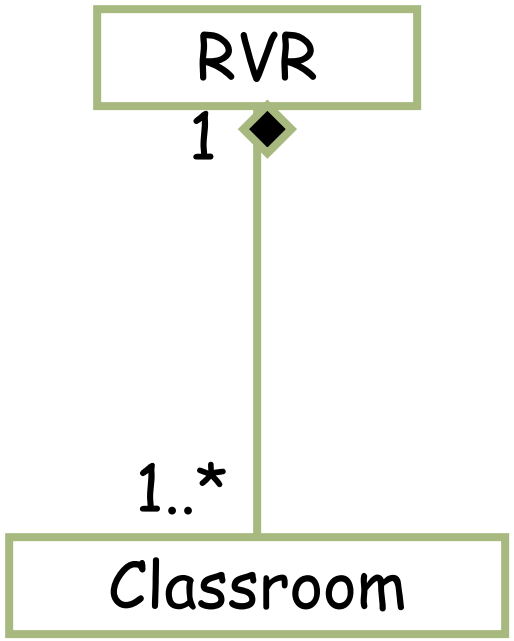



CSC131_3
1
*
Student
RVR
1
1..*
classroom

Aggregation



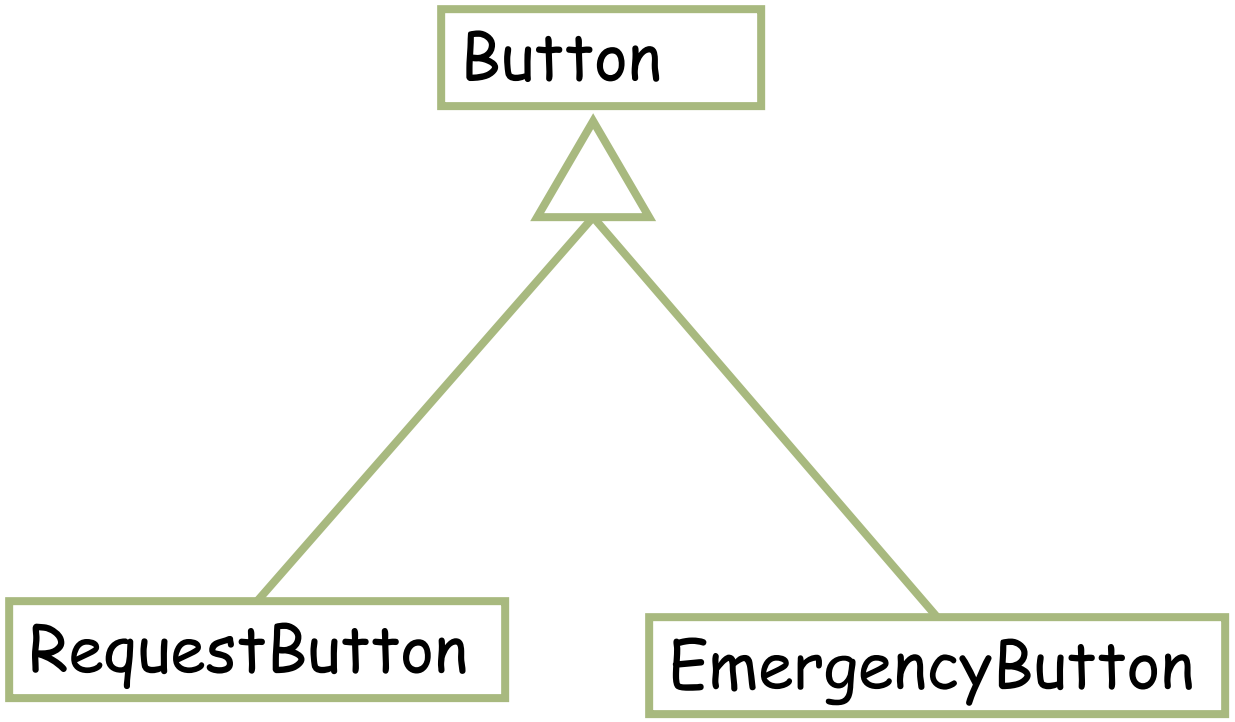
Composition



Generalization

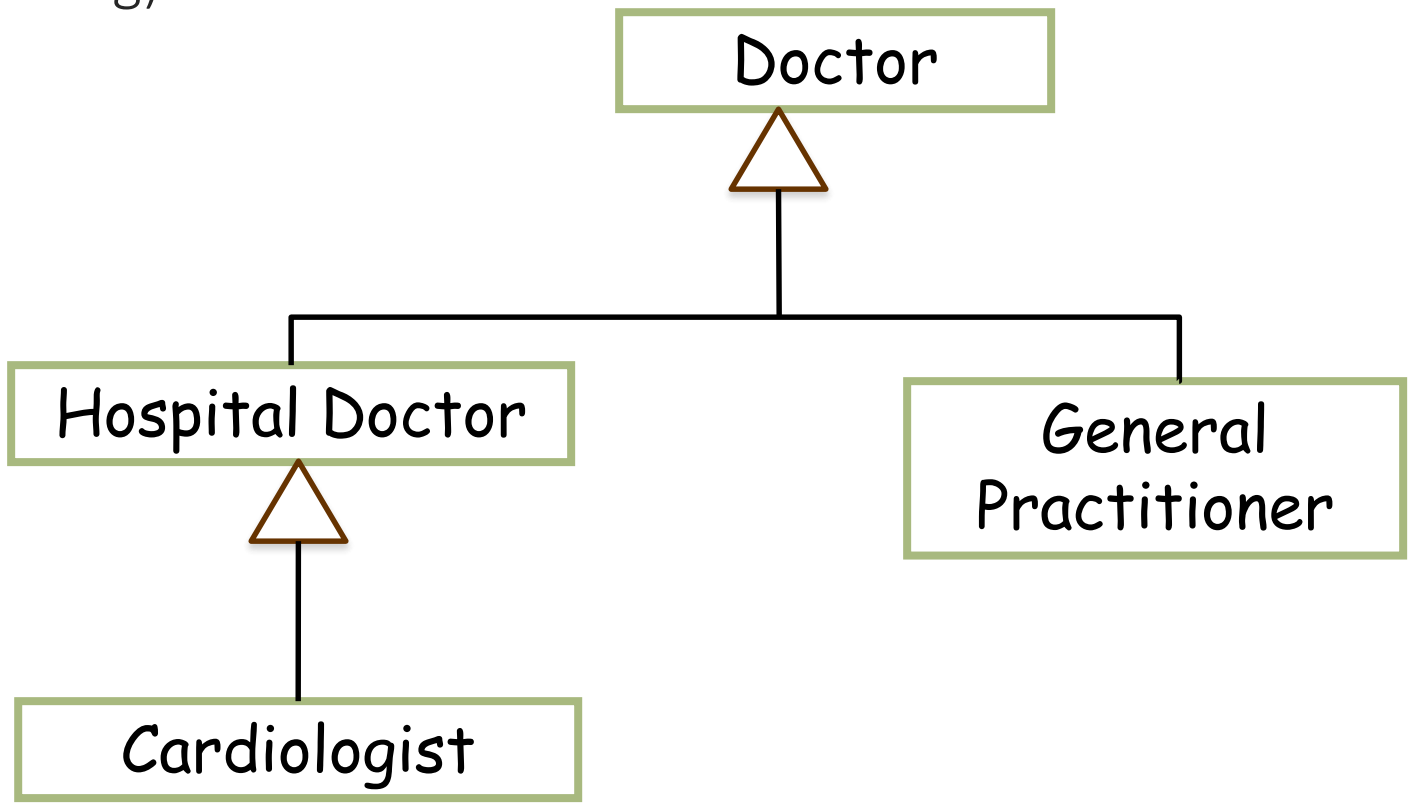
Inheritance between classes

Denoted by open triangle



Generalization

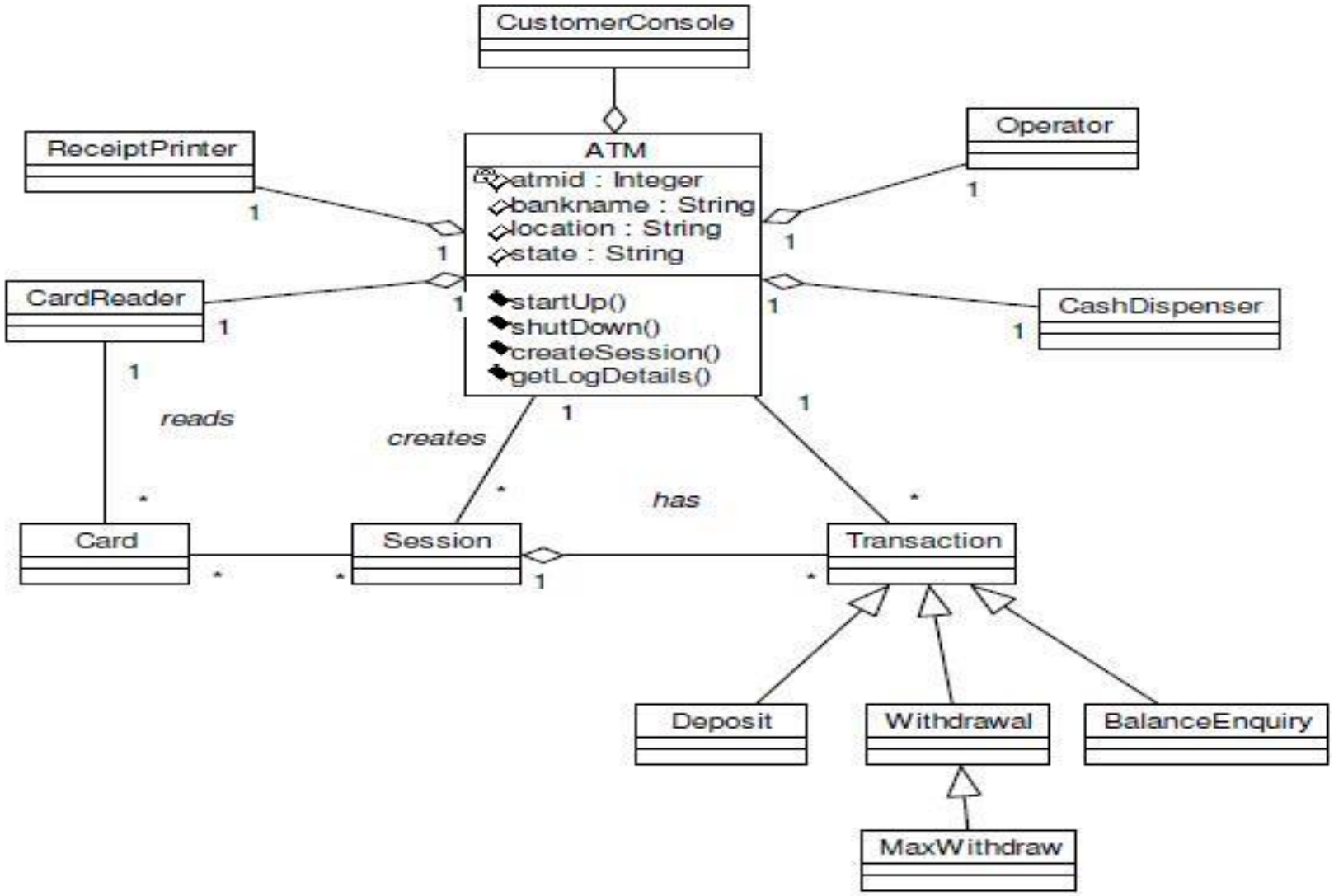
(Think subclassing)



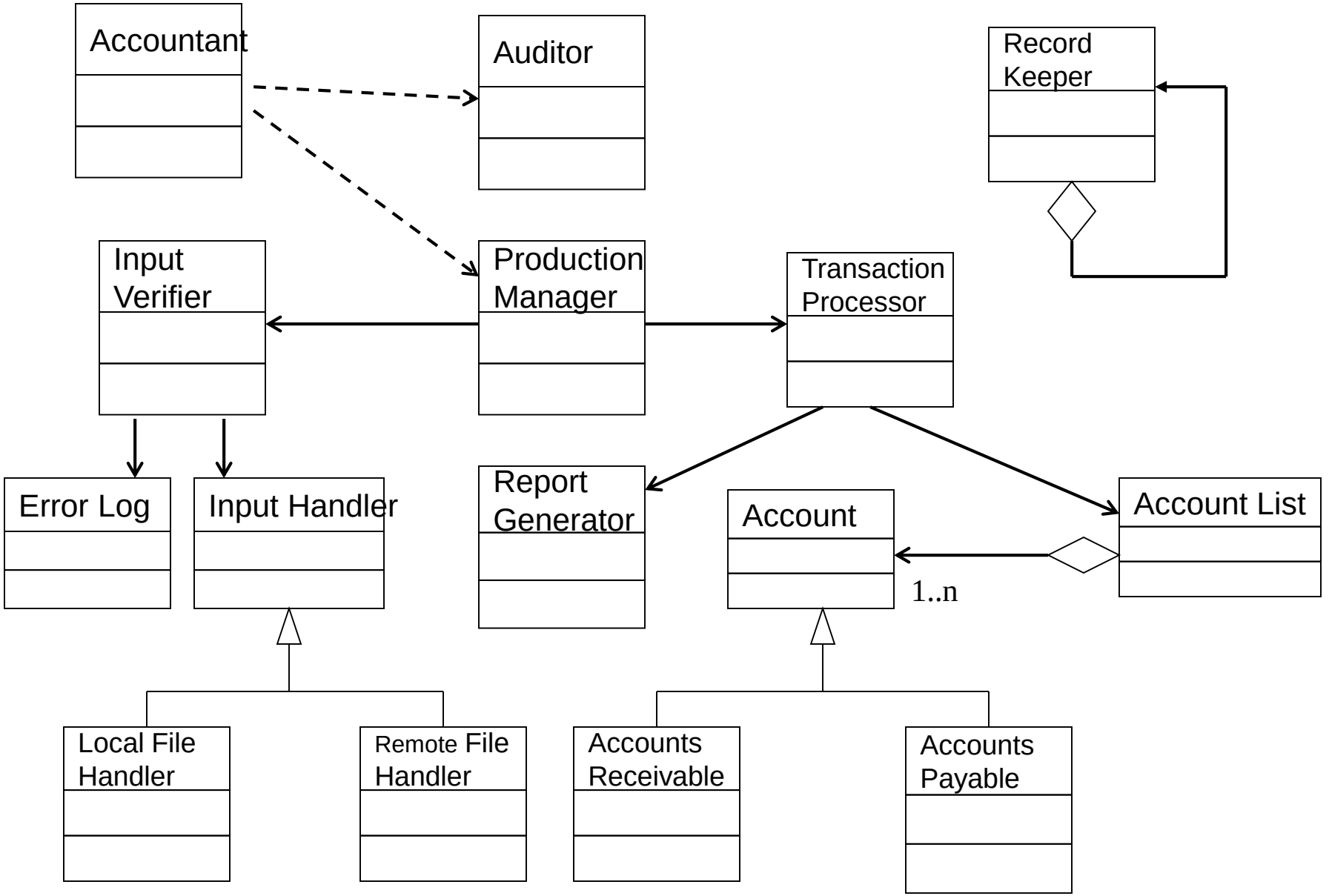
Symbol	Meaning
0	None
1	One
m	An integer value
0..1	Zero or one
m, *n*	*m* or *n*
m..*n*	At least *m*, but not more than *n*
*	Any nonnegative integer (zero or more)
0..*	Zero or more (identical to *)
1..*	One or more

Fig. 25.8 | Multiplicity types.

Example Class Diagram (1)



Example Class Diagram (2)



Unified Model Language

UML is the industry standard for documenting OO models

Class Diagrams	visualize *logical structure* of system in terms of *classes, objects and relationships*
Use Case Diagrams	show external *actors and use cases* they participate in
Sequence Diagrams	visualize *temporal message ordering* of a *concrete scenario* of a use case
Collaboration (Communication) Diagrams	visualize *relationships* of objects exchanging messages in a *concrete scenario*
State Diagrams	specify the *abstract states* of an object and the *transitions* between the states