DOMAIN MODEL: ADDING ASSOCIATIONS

Biju R Mohan

Lecture 13

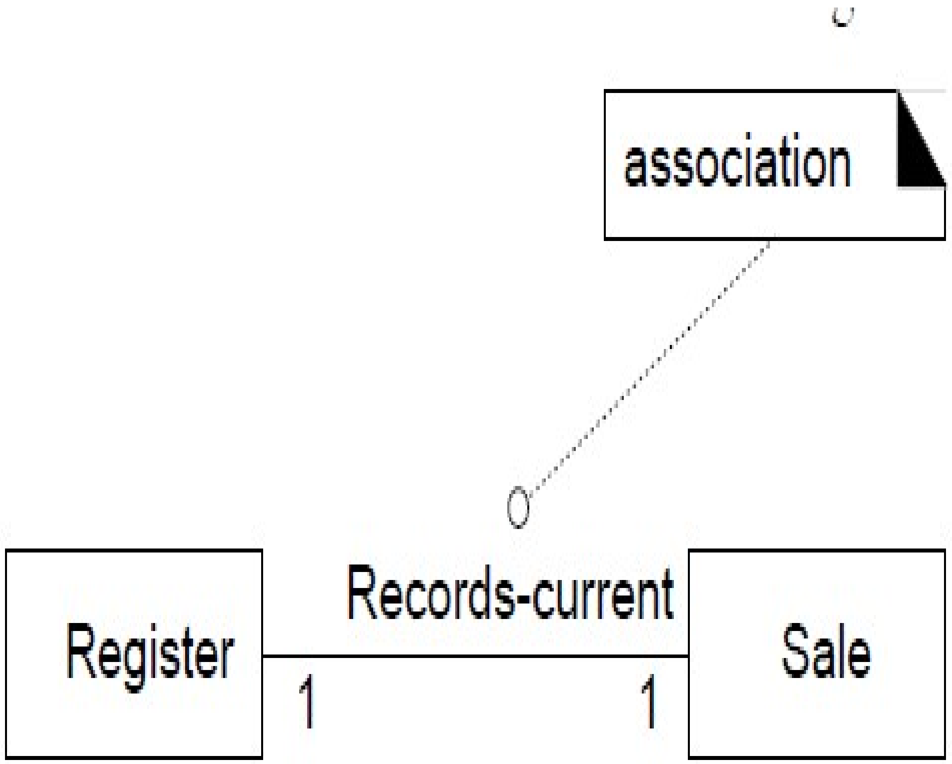
Agenda

- Identify associations for a domain model.
- Distinguish between need-to-know and comprehension-only associations.

Associations

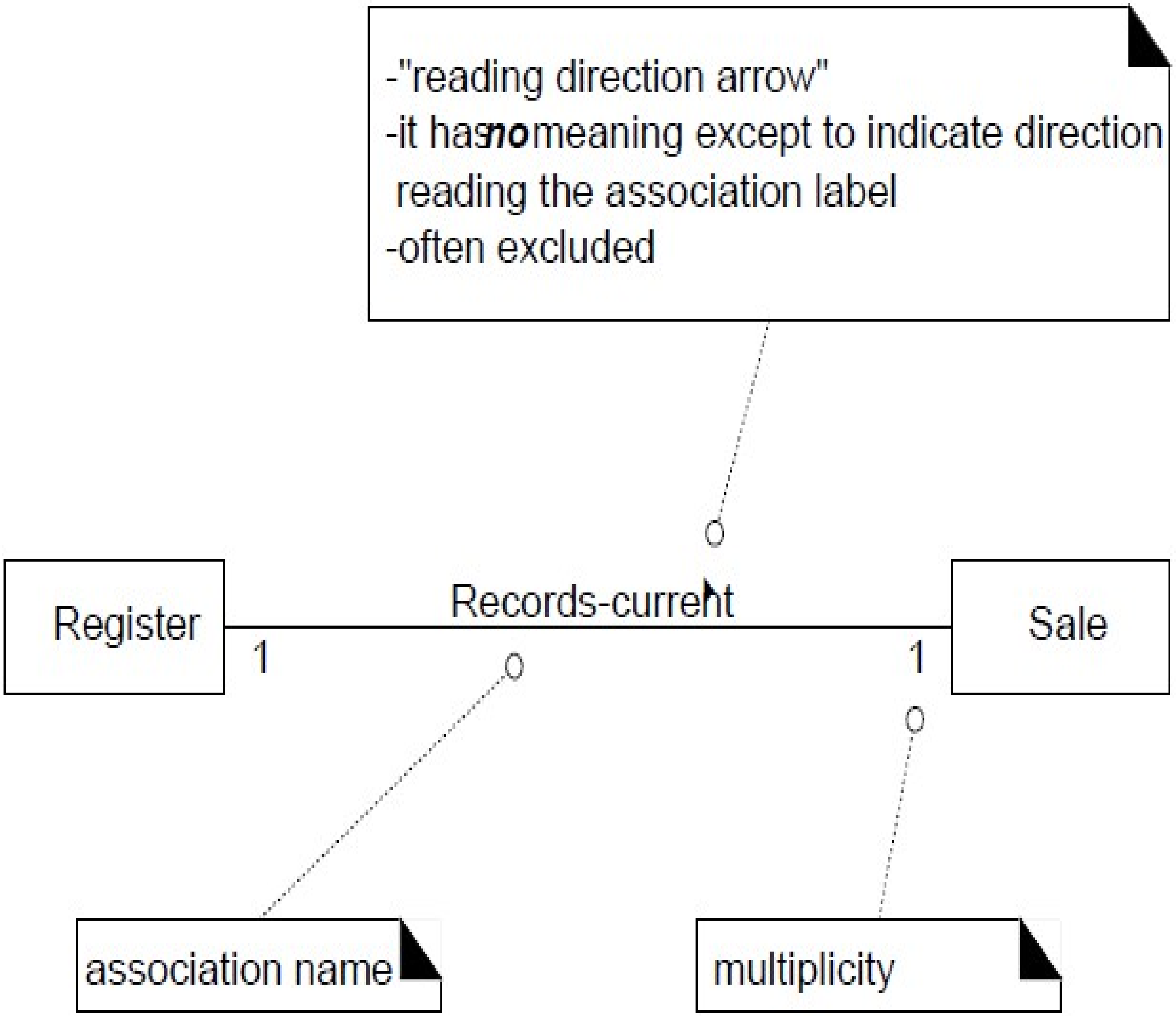
- An **association is a relationship between types (or more specifically, instances** of those types) that indicates some meaningful and interesting connection

UML Notation of Association



Types

- Consider including the following associations in a domain model:
  - Associations for which knowledge of the relationship needs to be preserved for some duration ("need-to-know" associations).
  - Associations derived from the Common Associations List.



-"reading direction arrow"
-it has***no***meaning except to indicate direction
reading the association label
-often excluded

Register

Records-current

Sale

1

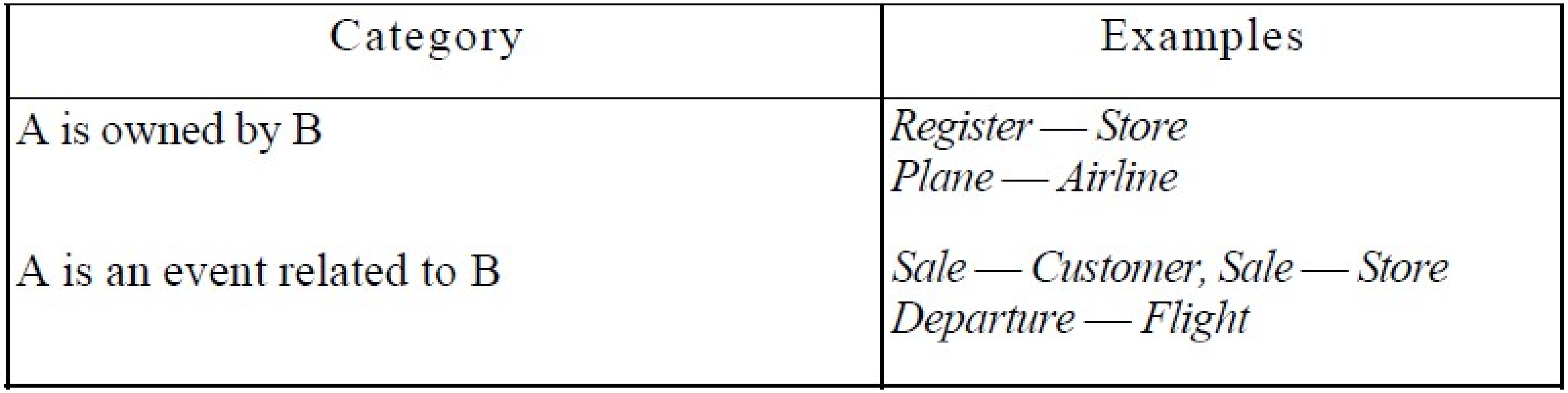
1

association name

multiplicity

Category	Examples
A is a physical part of B	*Drawer — Register (or more specifically, a POST)* *Wing — Airplane*
A is a logical part of B	*SalesLineItem — Sale* *FlightLeg—FlightRoute*
A is physically contained in/on B	*Register — Store, Item — Shelf* *Passenger — Airplane*
A is logically contained in B	*ItemDescription — Catalog* *Flight— FlightSchedule*
A is a description for B	*ItemDescription — Item* *FlightDescription — Flight*
A is a line item of a transaction or report B	*SalesLineItem — Sale* *Maintenance Job — Maintenance-Log*
A is known/logged/recorded/reported/captured in B	*Sale — Register* *Reservation — FlightManifest*
A is a member of B	*Cashier — Store* *Pilot — Airline*

A is a member of B	*Cashier — Store* *Pilot — Airline*
A is an organizational subunit of B	*Department — Store* *Maintenance — Airline*
A uses or manages B A communicates with B	*Cashier — Register* *Pilot — Airplane* *Customer — Cashier* *Reservation Agent — Passenger*
A is related to a transaction B	*Customer — Payment* *Passenger — Ticket*
A is a transaction related to another transaction B	*Payment — Sale* *Reservation — Cancellation*
A is next to B	*SalesLineItem — SalesLineItem* *City— City*



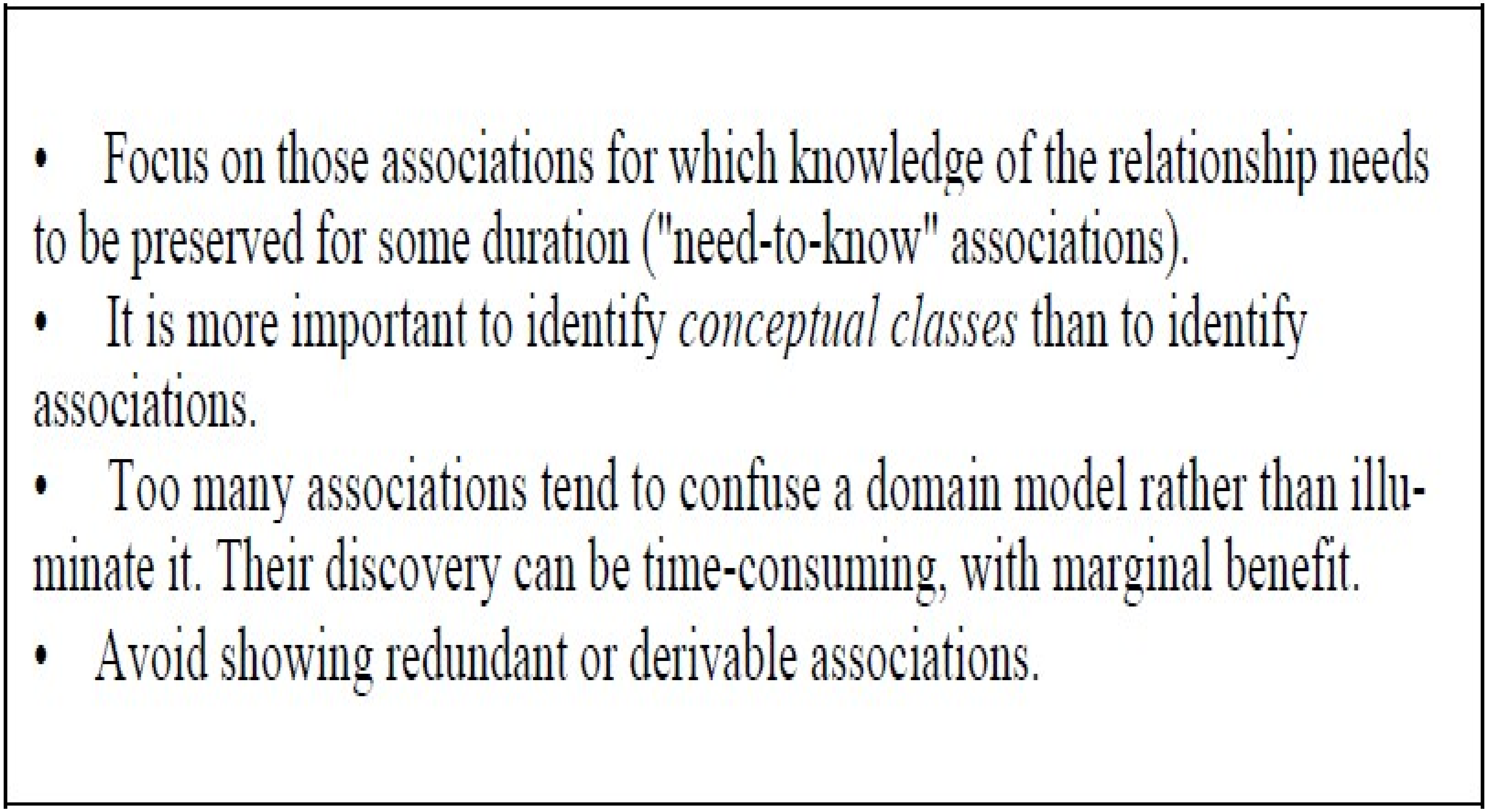
Category	Examples
A is owned by B	*Register — Store* *Plane — Airline*
A is an event related to B	*Sale — Customer, Sale — Store* *Departure — Flight*

High-Priority Associations

Here are some high-priority association categories that are invariably useful to include in a domain model:

- A is a *physical or logical part* of B.
- A is *physically or logically contained* in/on B.
- A is *recorded in* B.

Association Guidelines



- Focus on those associations for which knowledge of the relationship needs to be preserved for some duration ("need-to-know" associations).
- It is more important to identify *conceptual classes* than to identify associations.
- Too many associations tend to confuse a domain model rather than illuminate it. Their discovery can be time-consuming, with marginal benefit.
- Avoid showing redundant or derivable associations.

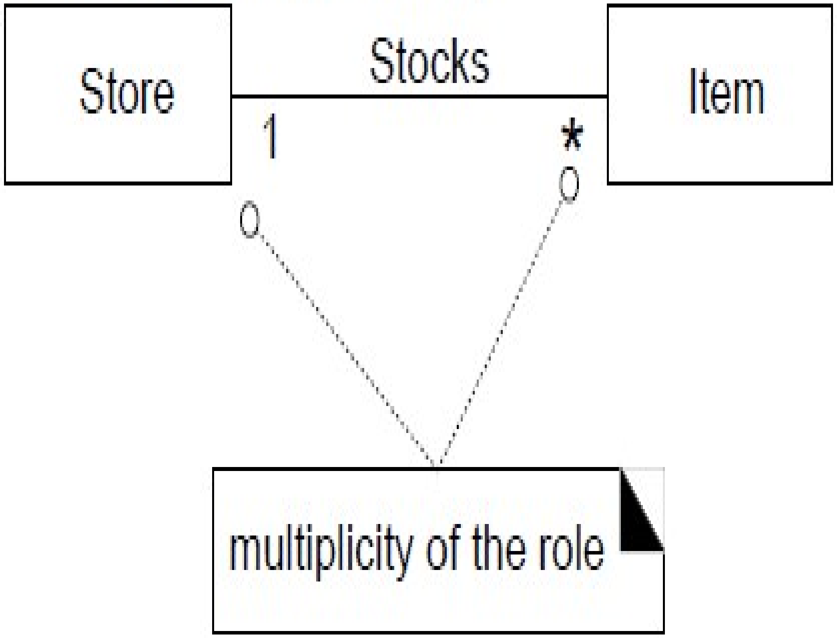
Roles

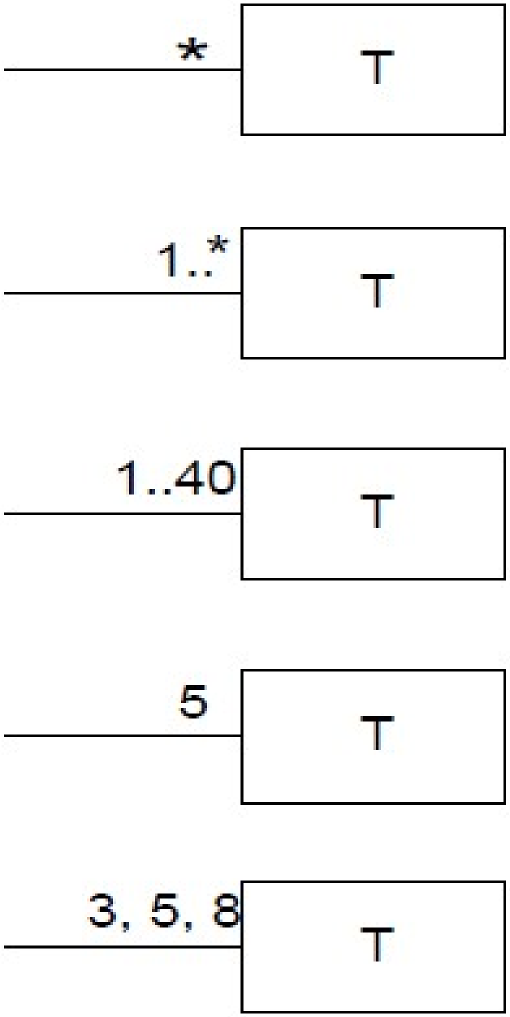
Each end of an association is called a **role.** Roles may optionally have:

- name
- multiplicity expression
- navigability

Multiplicity

Multiplicity defines how many instances of a class *A* can be associated with one instance of a class *B* (see Figure 11.3).





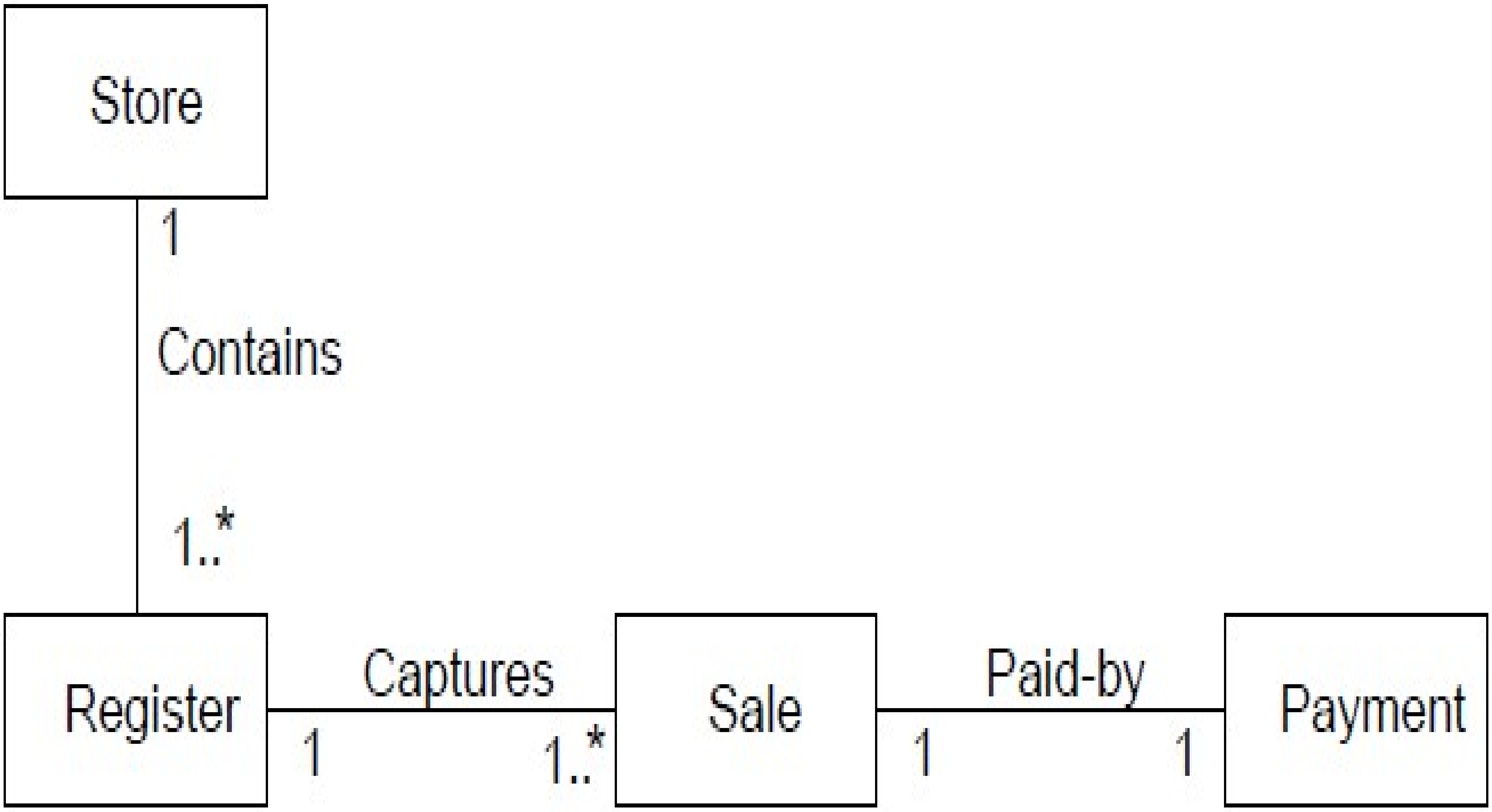
zero or more;
"many"

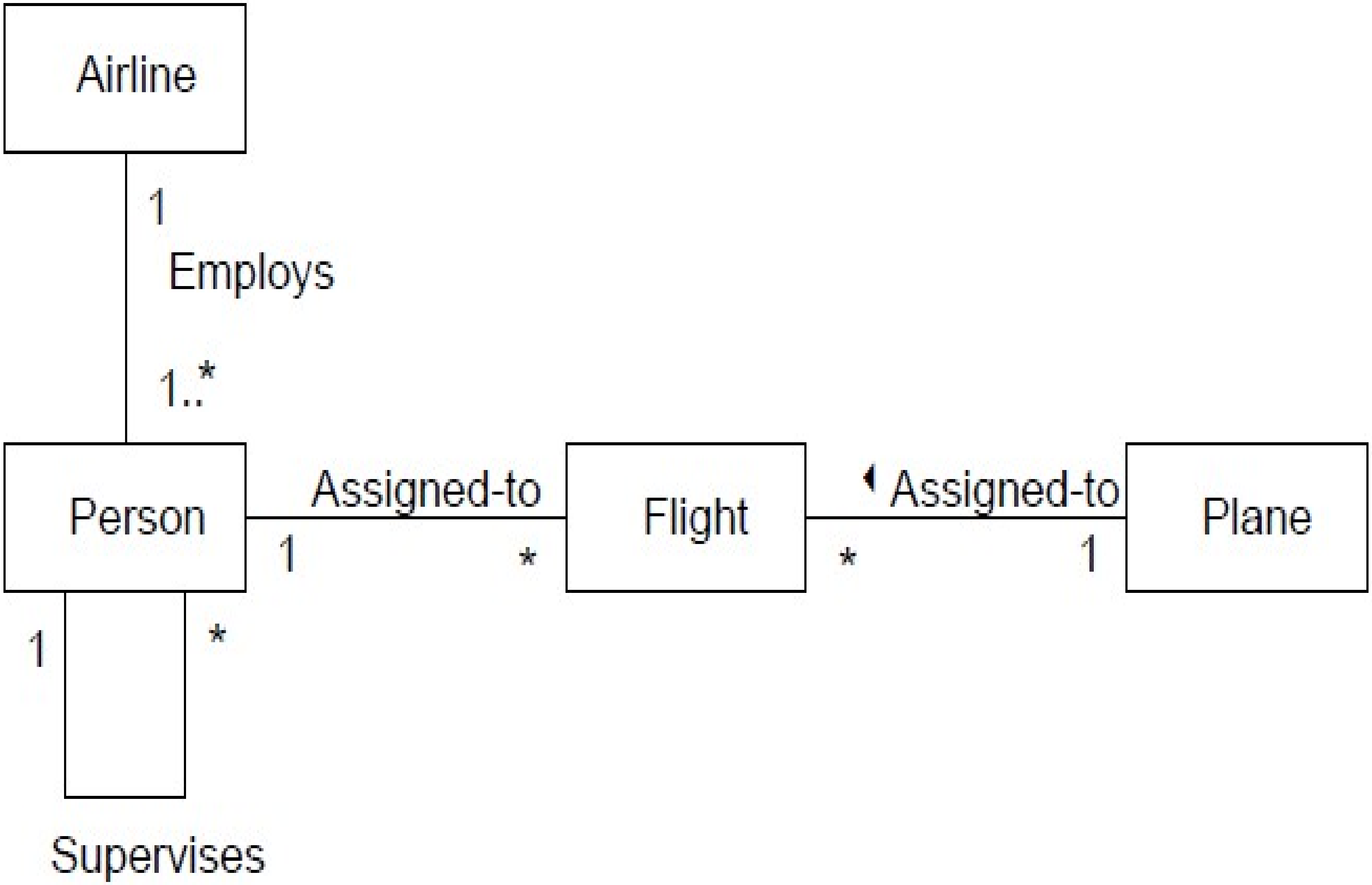
one or more

one to 40

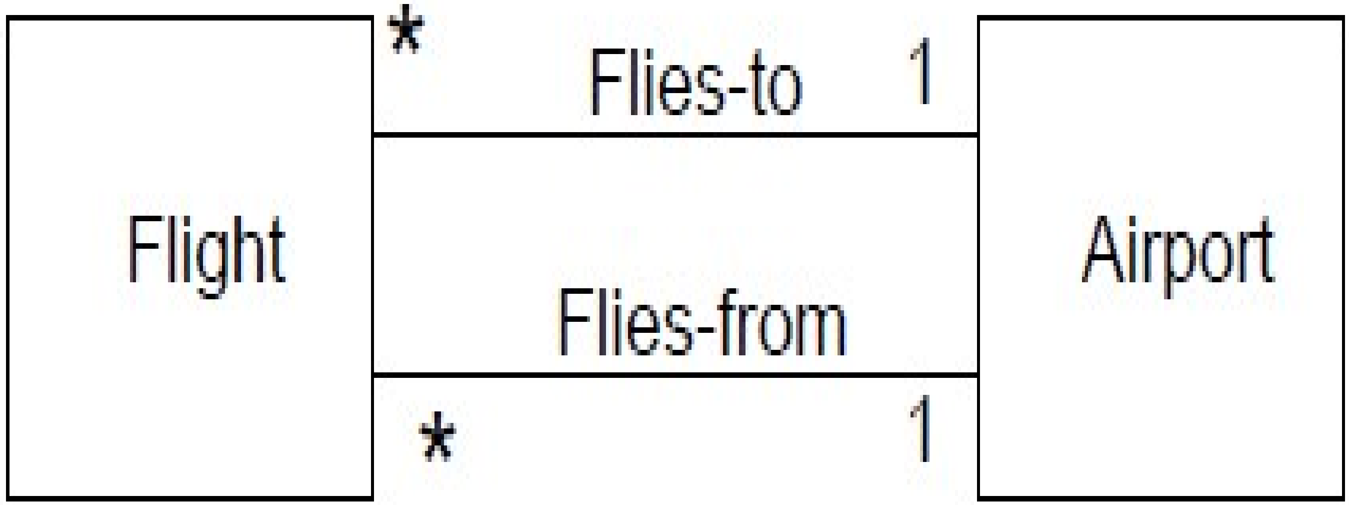
exactly 5

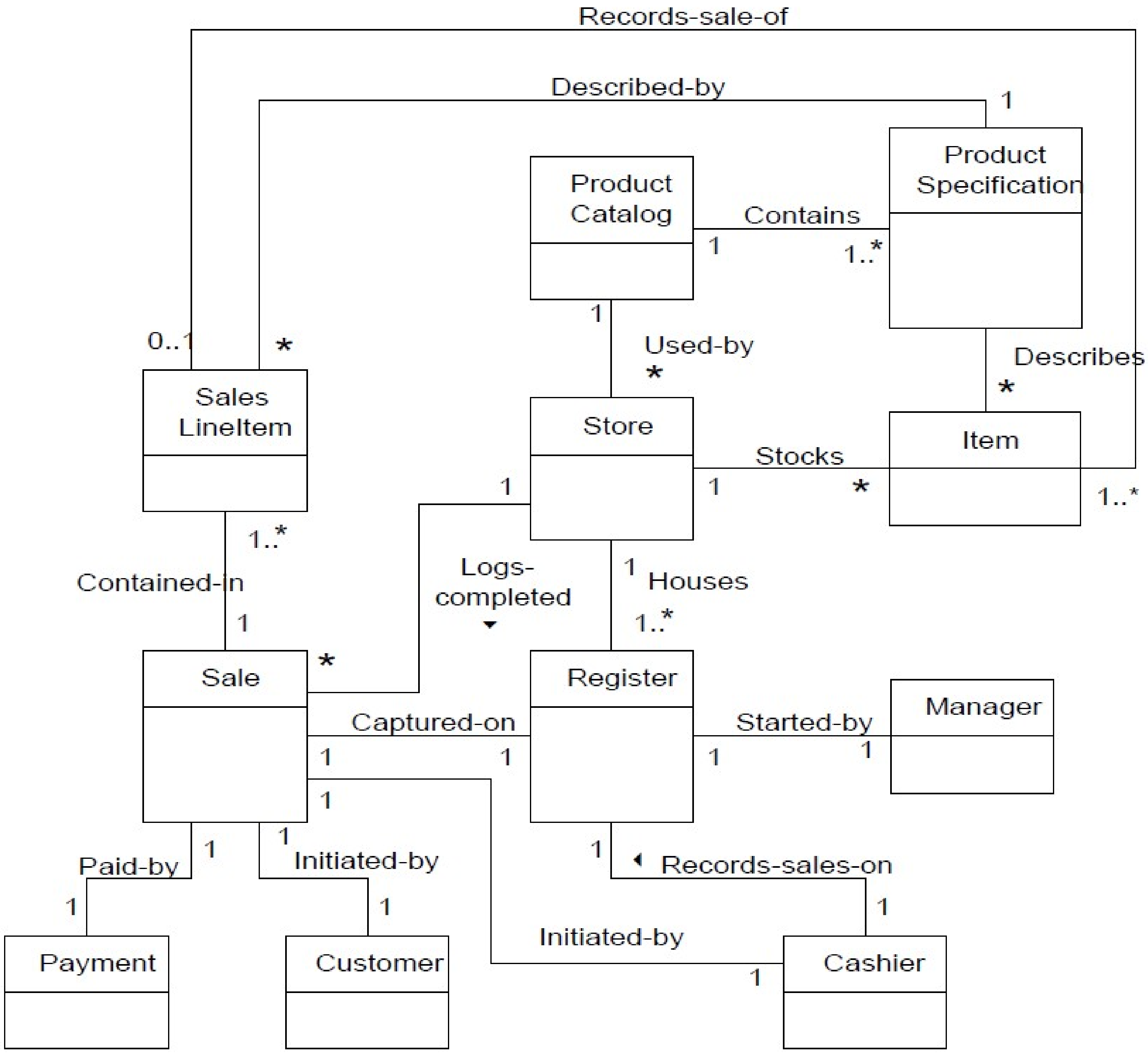
exactly 3, 5, or


Store
1
Contains
1..*
Register
Captures
1
1..*
Sale
Paid-by
1
1
Payment


Airline
1
Employs
1..*
Person
Assigned-to
1
*
Flight
◂ Assigned-to
*
1
Plane
1
*
Supervises

Multiple Associations Between Two Types




Records-sale-of
Described-by
1
Product Specification
Product Catalog
Contains
1
1..*
1
Used-by
*
Describes
*
0..1
*
Sales LineItem
Store
Item
Stocks
1
*
1..*
1
1..*
Contained-in
Logs-completed
1
Houses
1..*
1
*
Sale
Register
Manager
Captured-on
Started-by
1
1
1
1
1
1
1
Paid-by
Initiated-by
1
Records-sales-on
1
1
1
Initiated-by
1
Payment
Customer
Cashier

Reference

- Chapter 11 **Applying UML Patterns (Applying UML Patterns: An Introduction To Object-Oriented Analysis And Design) Craig L**