Software Analysis and Design (CS:3004)

Domain Models

Course Instructor: Nida Munawar

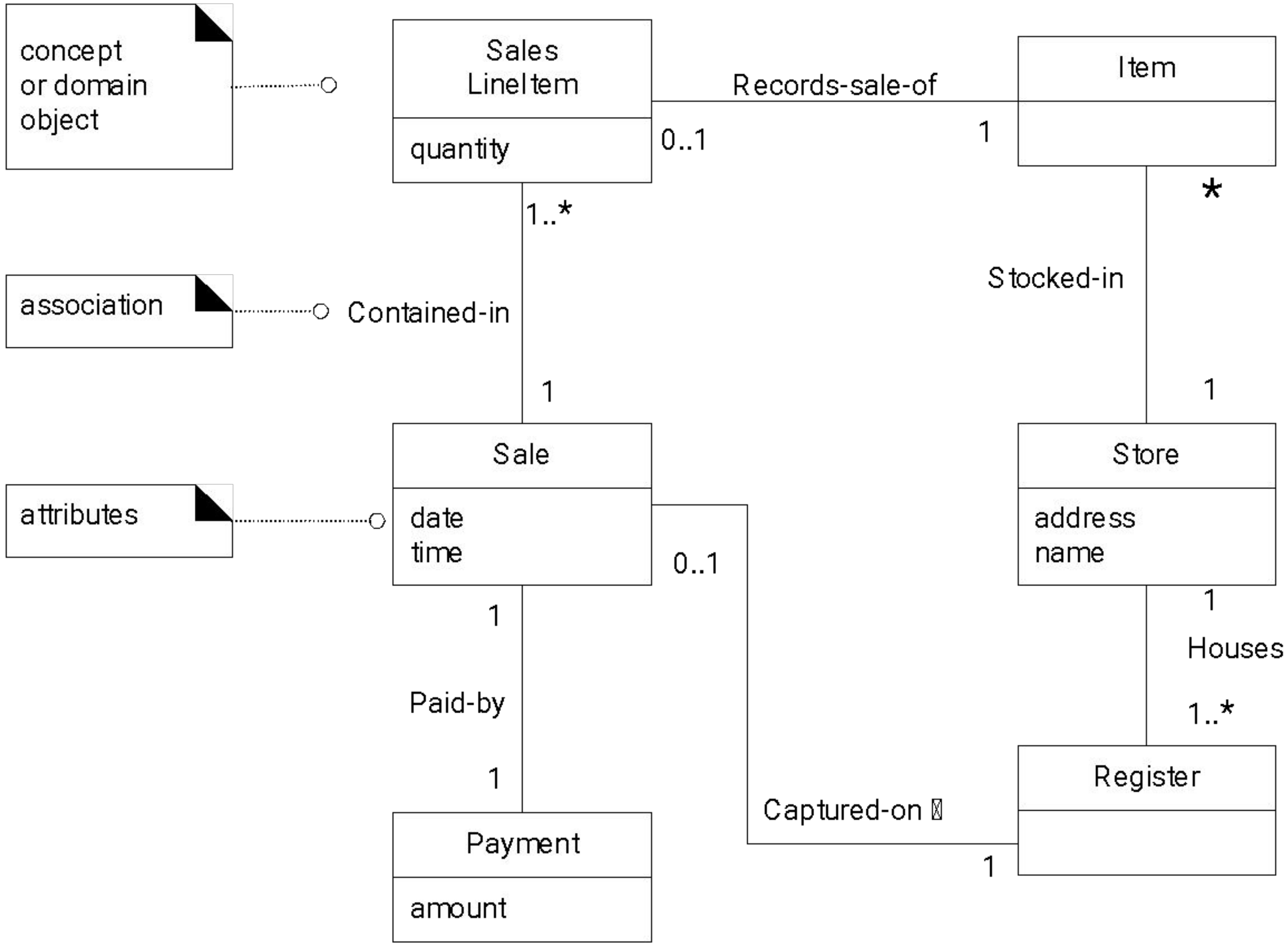
Email Address: nida.munawar@nu.edu.pk

Reference Book: Applying UML and Patterns (An introduction to Object-Oriented Analysis and Design And Iterative Development)

BY Craig Larman

Third Edition

Domain Model in UML Class Diagram Notation



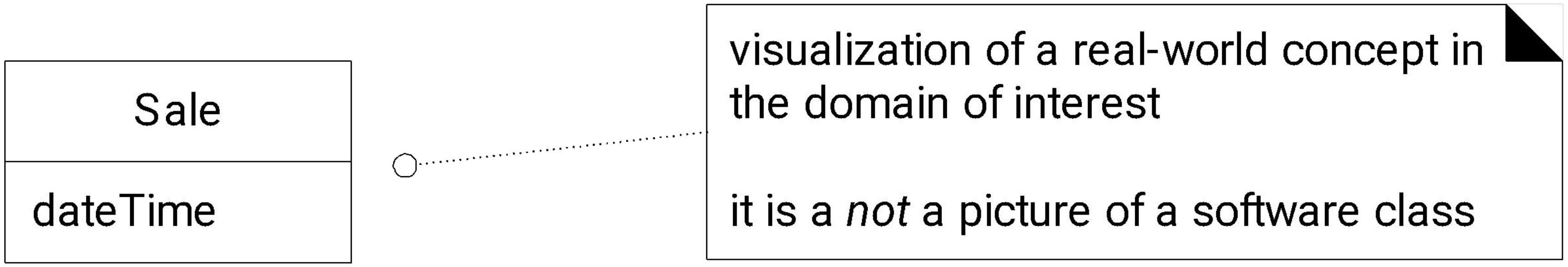
A “visual dictionary”

What is a Domain Model

- A **domain model** is a *visual* representation of conceptual classes or real-situation objects in a domain [MO95, Fowler96].

Domain models have also been called **conceptual models** , **domain object models**, and **analysis object models**.[2]

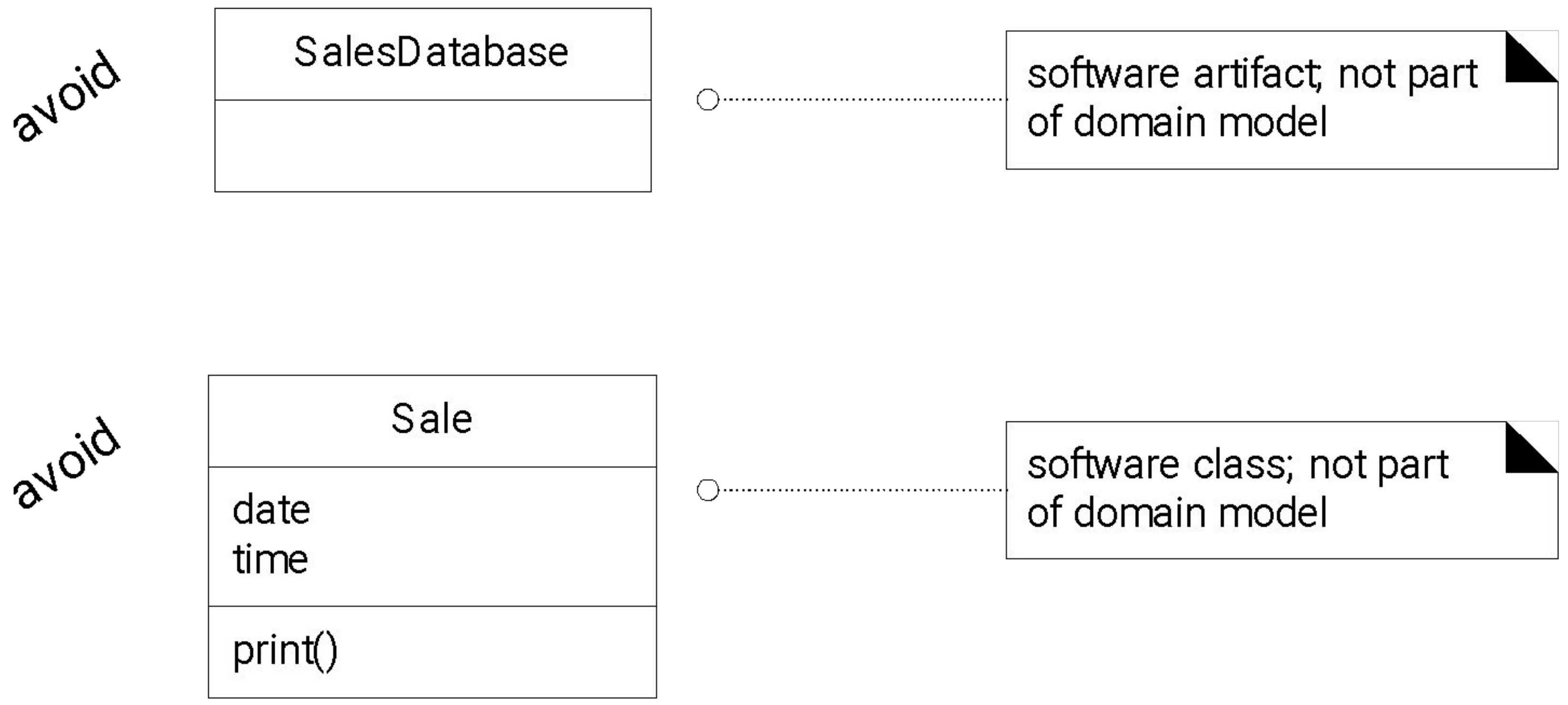
Fig. 9.3


Sale
dateTime
visualization of a real-world concept in the domain of interest
it is a *not* a picture of a software class

Domain Models

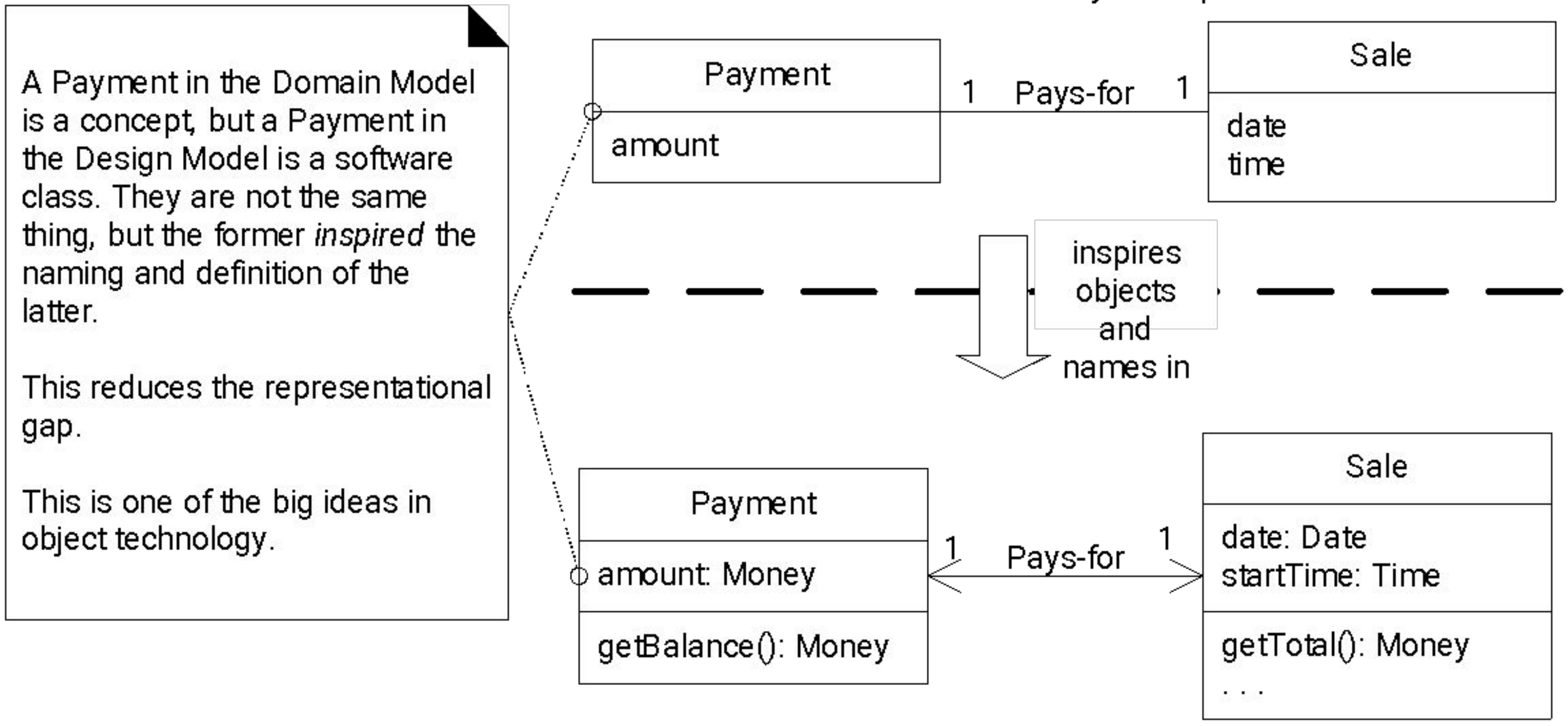
- Key object-oriented analysis step: Decompose domain into noteworthy concepts or objects
- UML class diagrams used to draw domain models
  - Conceptual perspective. Shows:
    - Domain objects (conceptual classes)
    - Associations between domain objects
    - Attributes of conceptual classes
- Domain model is NOT a model of software objects or our design
- The following should NOT be in a domain model
  - Software artifacts: Window, database, …
  - Responsibilities or methods

Fig. 9.4


avoid
SalesDatabase
software artifact; not part of domain model
avoid
Sale
date
time
print()
software class; not part of domain model

Software Class Names Inspired by Domain Model Objects

UP Domain Model
Stakeholder's view of the noteworthy concepts in the domain.



A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

This reduces the representational gap.

This is one of the big ideas in object technology.

UP Design Model
The object-oriented developer has taken inspiration from the real world domain in creating software classes.

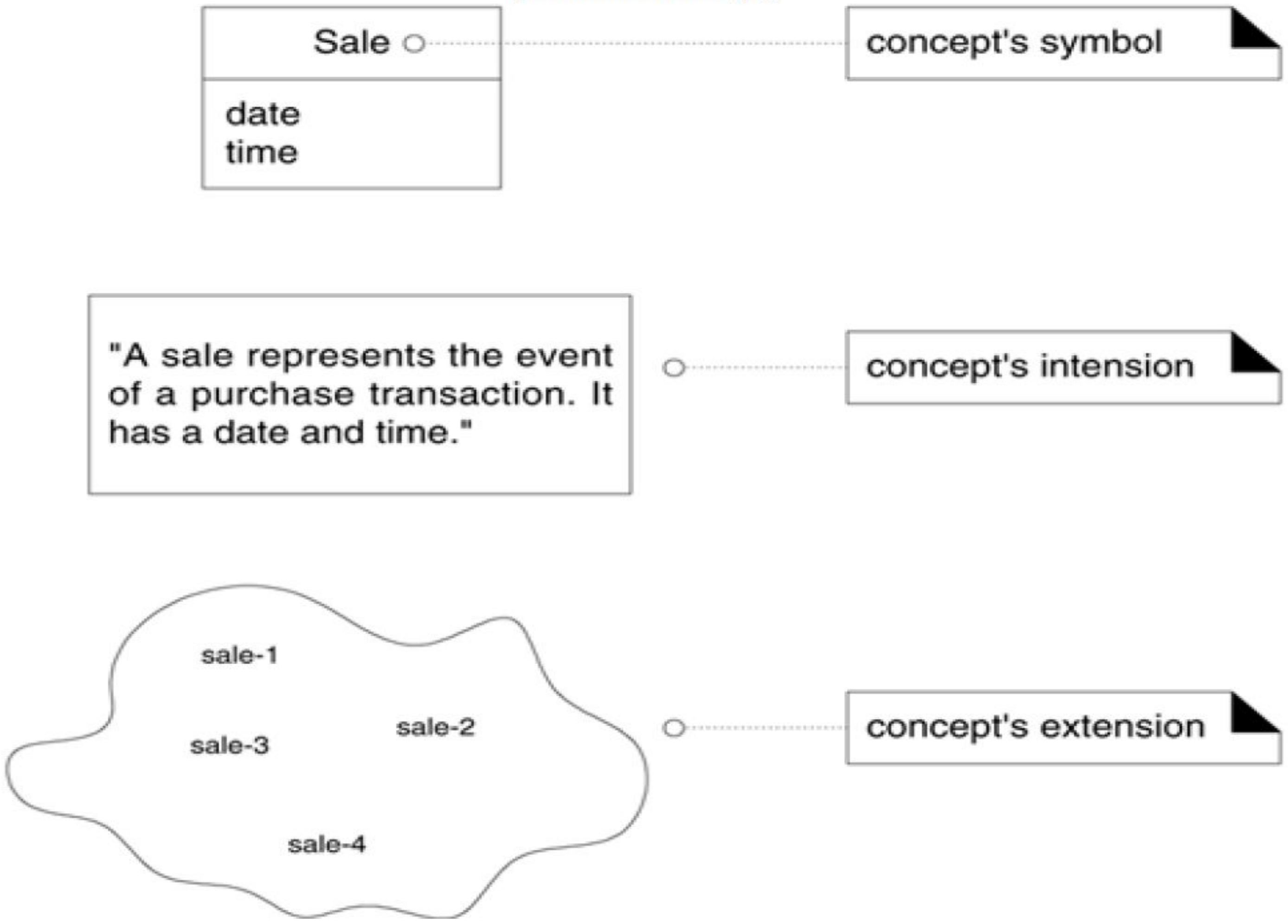
Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

Definition: What are Conceptual Classes?

- The domain model illustrates conceptual classes or vocabulary in the domain. Informally, a **conceptual class** is
- an idea, thing, or object. More formally, a conceptual class may be considered in terms of its symbol, intension, and extension
- **Symbol** words or images representing a conceptual class.
- **Intension** the definition of a conceptual class.
- **Extension** the set of examples to which the conceptual class applies.

Figure 9.5. A conceptual class has a symbol, intension, and extensio

[View full size image]



9.3. Motivation: Why Create a Domain Model?

- I'll share a story that I've experienced many times in OO consulting and coaching. In the early 1990s I was
- working with a group developing a funeral services business system in Smalltalk, in Vancouver (you should see
- the domain model!). Now, I knew almost nothing about this business, so one reason to create a domain model
- was so that I could start to understand their key concepts and vocabulary.
- We also wanted to create a **domain layer** of Smalltalk objects representing business objects and logic. So, we
- spent perhaps one hour sketching a UML-ish (actually OMT-ish, whose notation inspired UML) domain model,
- not worrying about software, but simply identifying the key terms. Then, those terms we sketched in the
- domain model, such as *Service* (like flowers in the funeral room, or playing "You Can't Always Get What You
- Want"), were also used as the names of key software classes in our domain layer implemented in Smalltalk.

How to create a domain model?

1. Find conceptual classes
   - How?
     - Method 1:Re-use or modify existing models(good but outside our scope)
     - Method 2:Use a category list
     - Method 3:Identify noun phrases
     - More on these later
2. Draw them as classes in a UML class diagram
3. Add associations and attributes
   - More on this later

Conceptual Class Category	Examples
physical or tangible objects	*Register* *Airplane*
specifications, designs, or descriptions of things	*ProductSpecification* *FlightDescription*
places	*Store* *Airport*
transactions	*Sale, Payment* *Reservation*
transaction line items	*SalesLineItem*
roles of people	*Cashier* *Pilot*
containers of other things	*Store, Bin* *Airplane*
things in a container	*Item* *Passenger*

other computer or electro-mechanical systems external to the system	*CreditPaymentAuthorizationSystem* *AirTrafficControl*
abstract noun concepts	*Hunger* *Acrophobia*
organizations	*SalesDepartment* *ObjectAirline*
events	*Sale, Payment, Meeting* *Flight, Crash, Landing*
processes (often *not* represented as a concept, but may be)	*SellingAProduct* *BookingASeat*
rules and policies	*RefundPolicy* *CancellationPolicy*
catalogs	*ProductCatalog* *PartsCatalog*

Conceptual Class Category	Examples
records of finance, work, contracts, legal matters	*Receipt, Ledger, EmploymentContract* *MaintenanceLog*
financial instruments and services	*LineOfCredit* *Stock*
manuals, documents, reference papers, books	*DailyPriceChangeList* *RepairManual*

Identifying nouns as candidates for domain objects

Main Success Scenario (or Basic Flow):

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale.**
3. **Cashier** enters **item identifier.**
4. System records **sale line item** and presents **item description, price,** and running **total.** Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment.**
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions)** and **Inventory** systems (to update inventory).
9. System presents **receipt.**
10. Customer leaves with receipt and goods (if any).

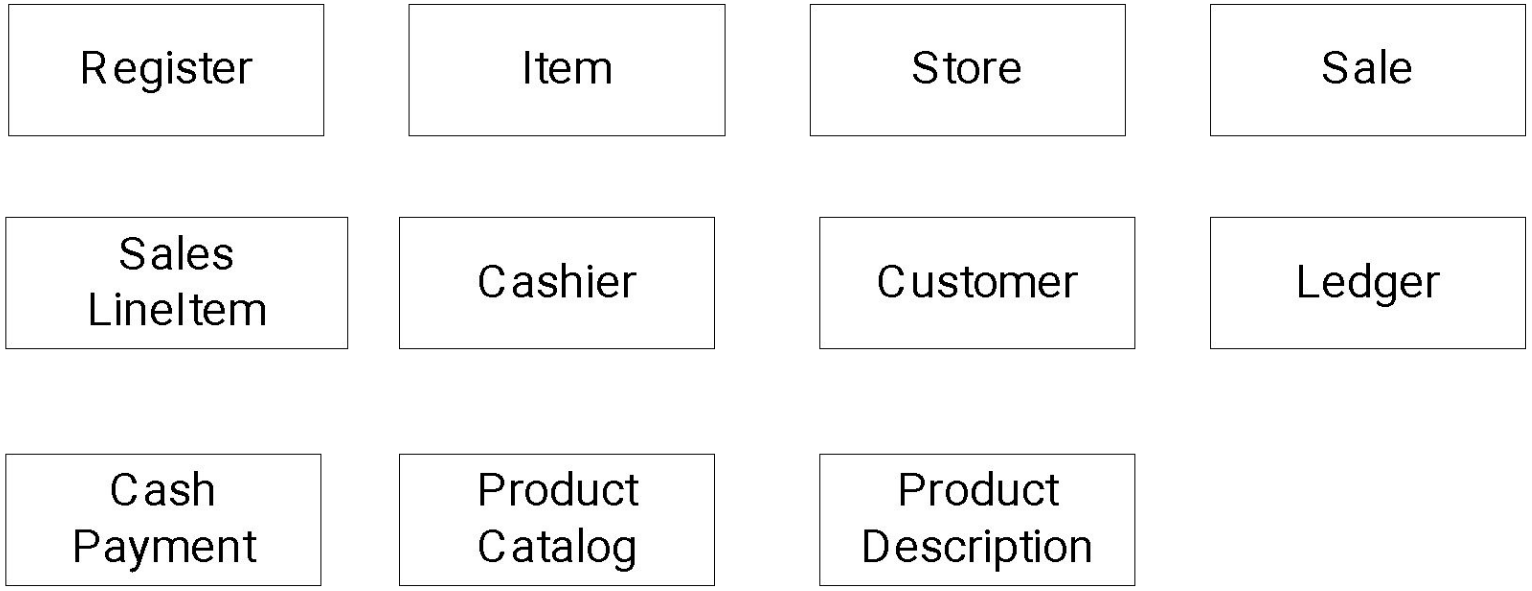
Extensions (or Alternative Flows):

7a. Paying by cash:

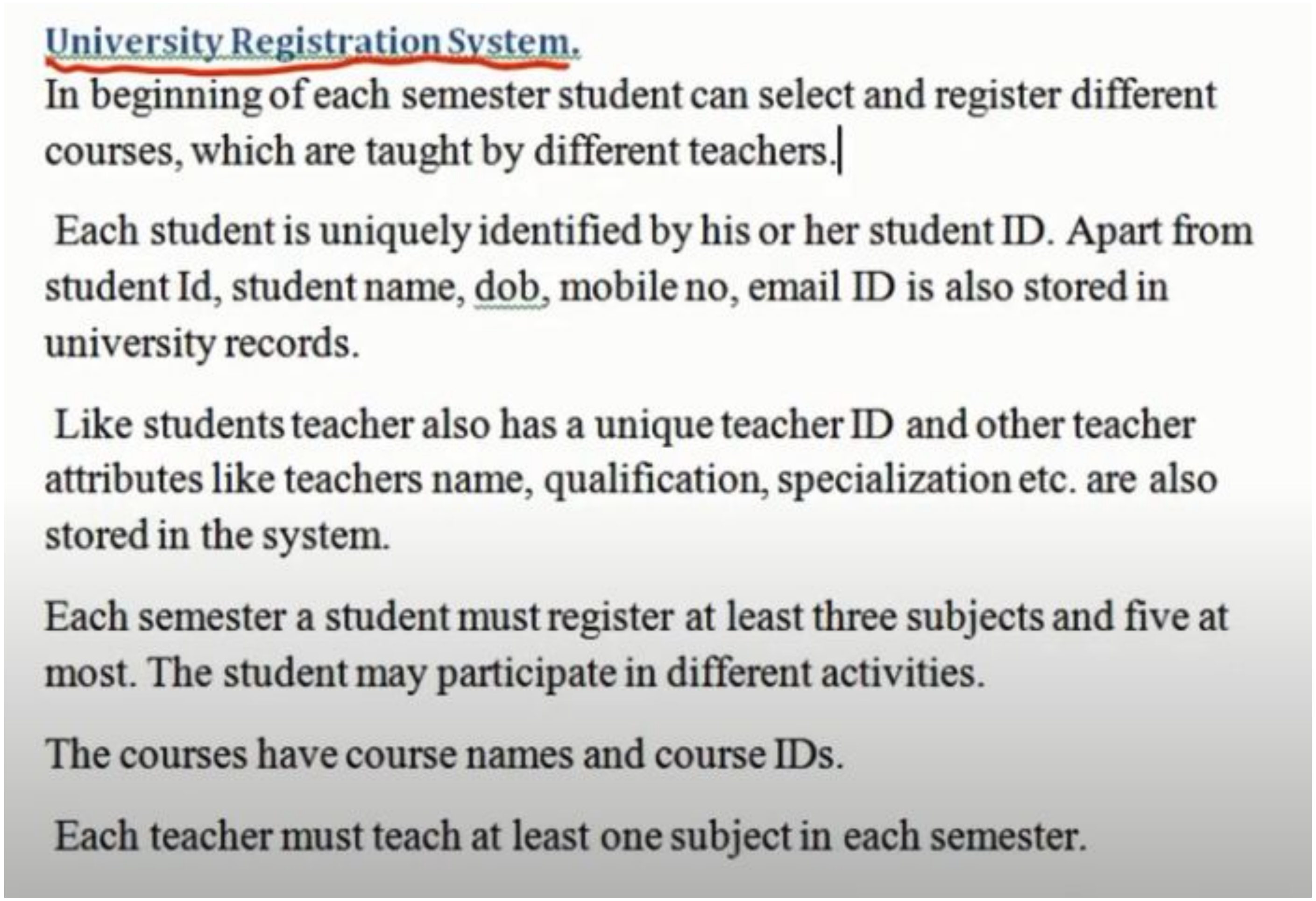
1. Cashier enters the cash **amount tendered.**
2. System presents the **balance due,** and releases the **cash drawer.**
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

Sale	*Cashier*
CashPayment	*Customer*
SalesLineItem	*Store*
Item	*ProductDescription*
Register	*ProductCatalog*
Ledger	

Candidate Conceptual Classes: Process Sale, Iteration 1



Activity: identify the nouns



University Registration System.

In beginning of each semester student can select and register different courses, which are taught by different teachers.

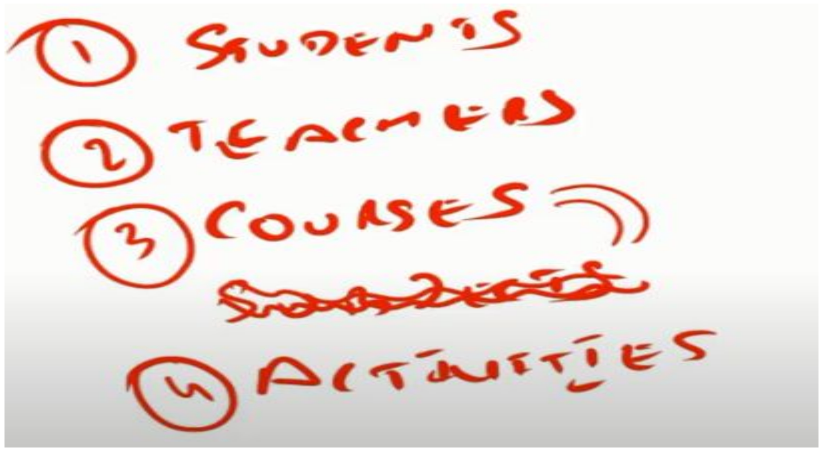
Each student is uniquely identified by his or her student ID. Apart from student Id, student name, dob, mobile no, email ID is also stored in university records.

Like students teacher also has a unique teacher ID and other teacher attributes like teachers name, qualification, specialization etc. are also stored in the system.

Each semester a student must register at least three subjects and five at most. The student may participate in different activities.

The courses have course names and course IDs.

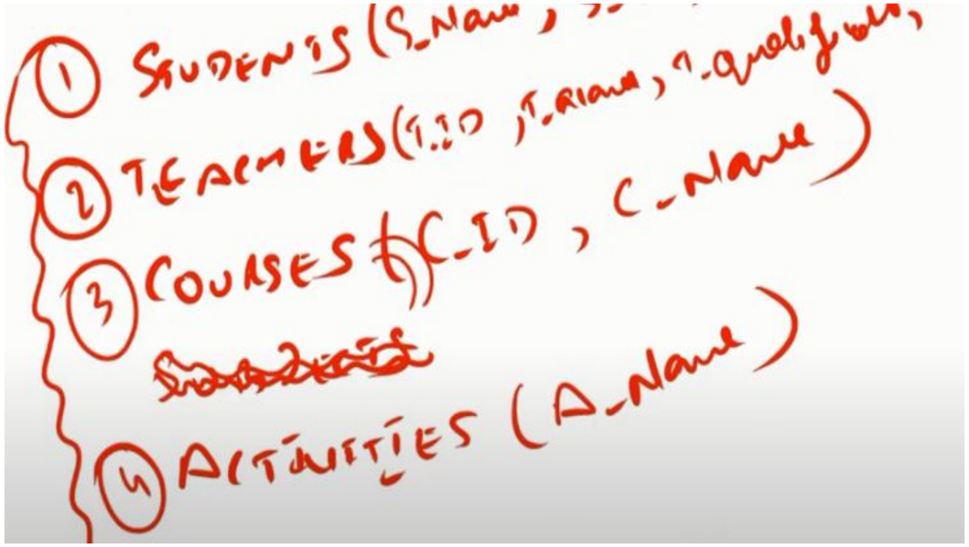
Each teacher must teach at least one subject in each semester.



1. STUDENTS
2. TEACHERS
3. COURSES

~~[illegible]~~

4. ACTIVITIES

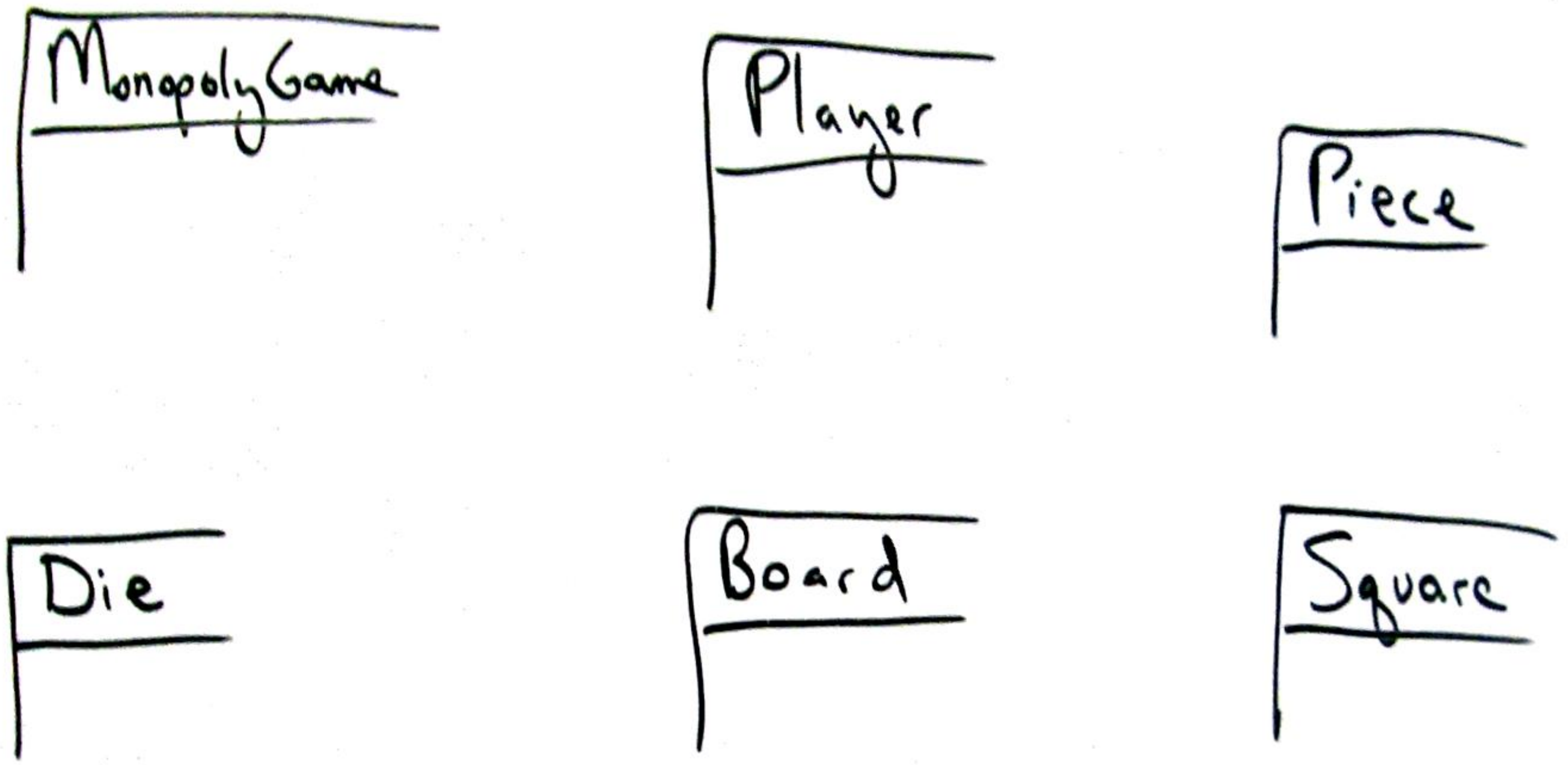


1. STUDENTS (S_Name, [illegible]
2. TEACHERS (T_ID, T_Name, T_qualification,
3. COURSES (C_ID, C_Name)

~~[illegible]~~

4. ACTIVITIES (A_Name)

Case Study: Monopoly Domain



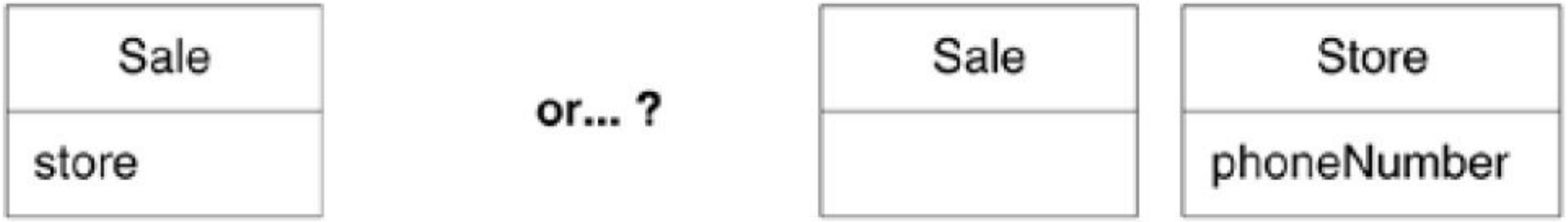
A Common Mistake with Attributes vs. Classes

- Should “something” be
  - an attribute of a class? or
  - a separate conceptual class?

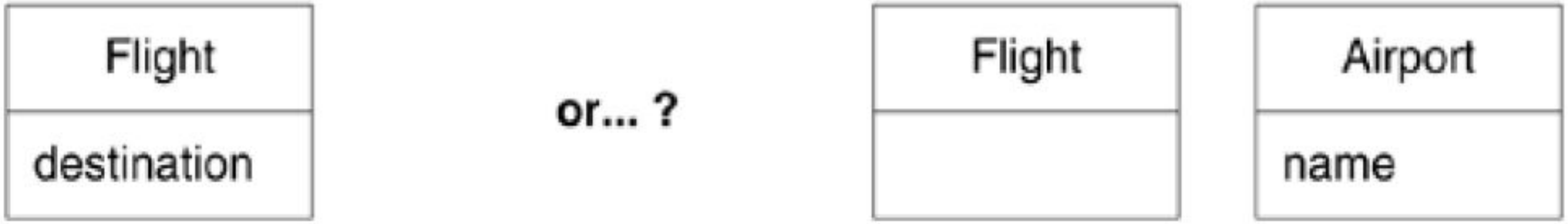
- Guideline
- If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute.

example

Store: An attribute of Sale or separate class



Flight destination: An attribute or a separate class



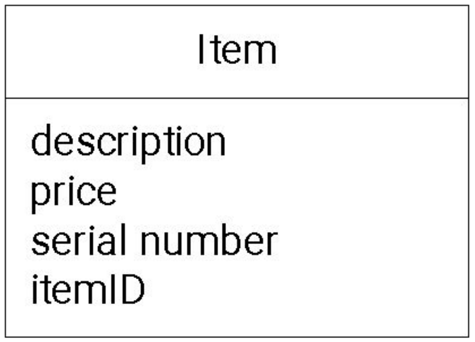
A Common Mistake with Attributes vs. Classes

- Examples:
  - Store: An attribute of Sale or separate class
    - Store: Not a number or text
    - Should be separate conceptual class
  - Flight destination: An attribute or a separate class
    - Destination is an airport, not a number
    - Should be separate conceptual class
- Guideline:
  - If we thing of something as simply a number or text in real life, it should be an attribute.
  - Otherwise it should be a conceptual class.

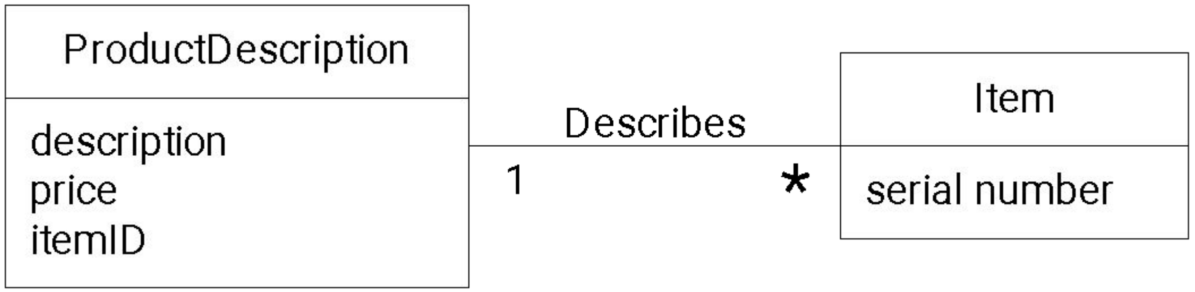
“Description” Classes

- A description class contains information that describes something else
  - Example: ProductDescription
  - Records price, picture and text description of an item
- Why use them? Why not include all that information in the Product class?
  - We need this information stored and represented even though there are no objects of that particular product type.
  - Don’t want to duplicate product information for each duplicate product object
    - Serial number belongs with product object
    - Picture of product belongs with product description
- Need objects that are “specifications” or “descriptions” of other objects

Description class example

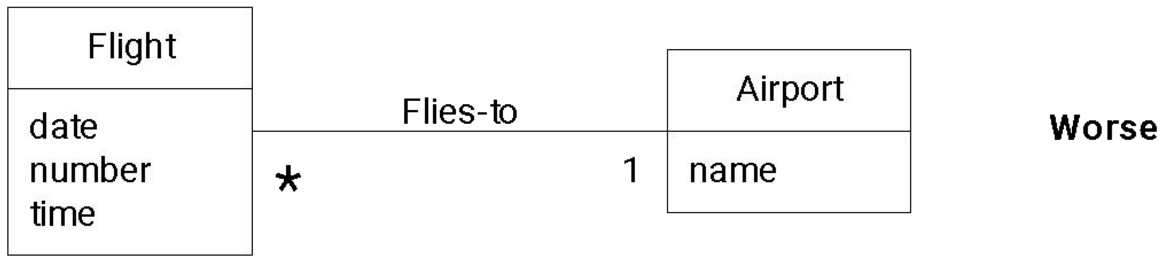


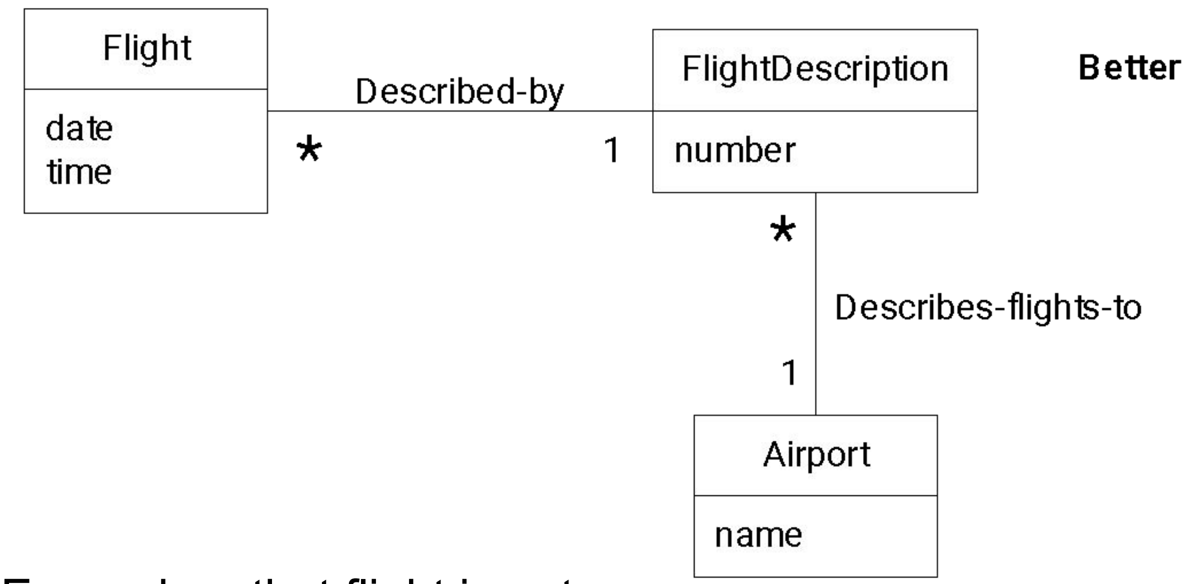
Worse



Better

Description class example

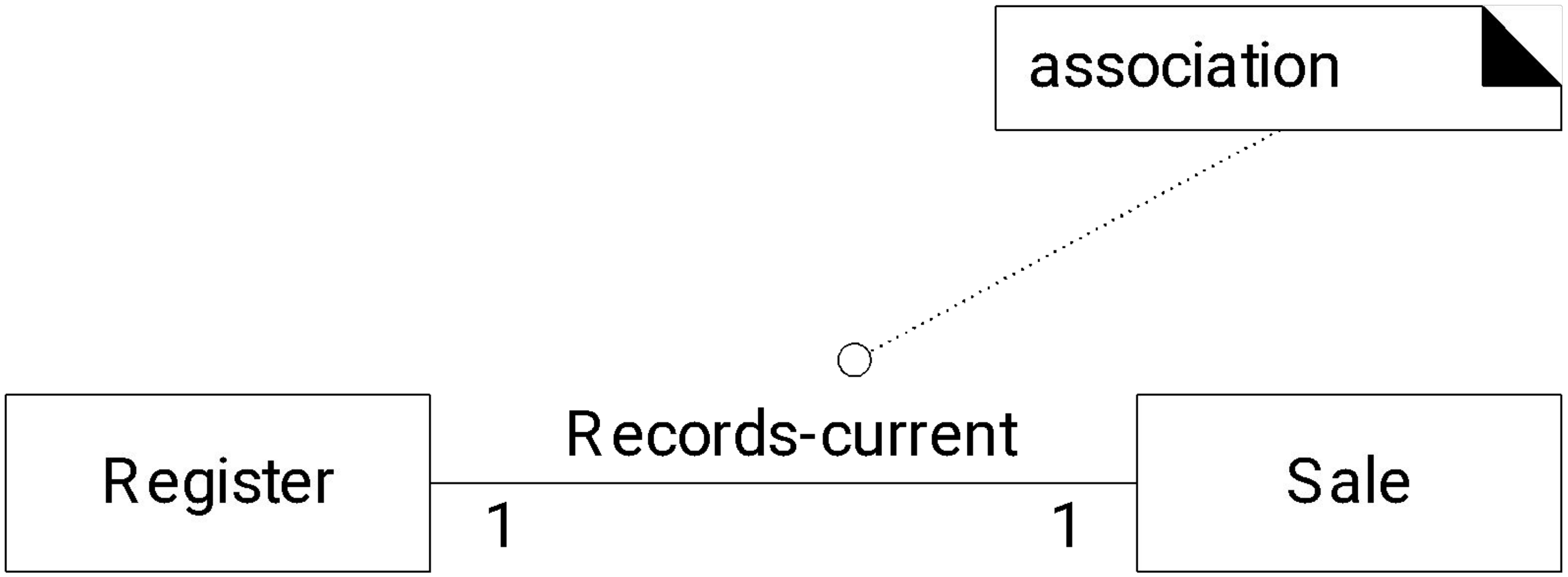




Even when that flight is not
scheduled that day, the flight description exists.

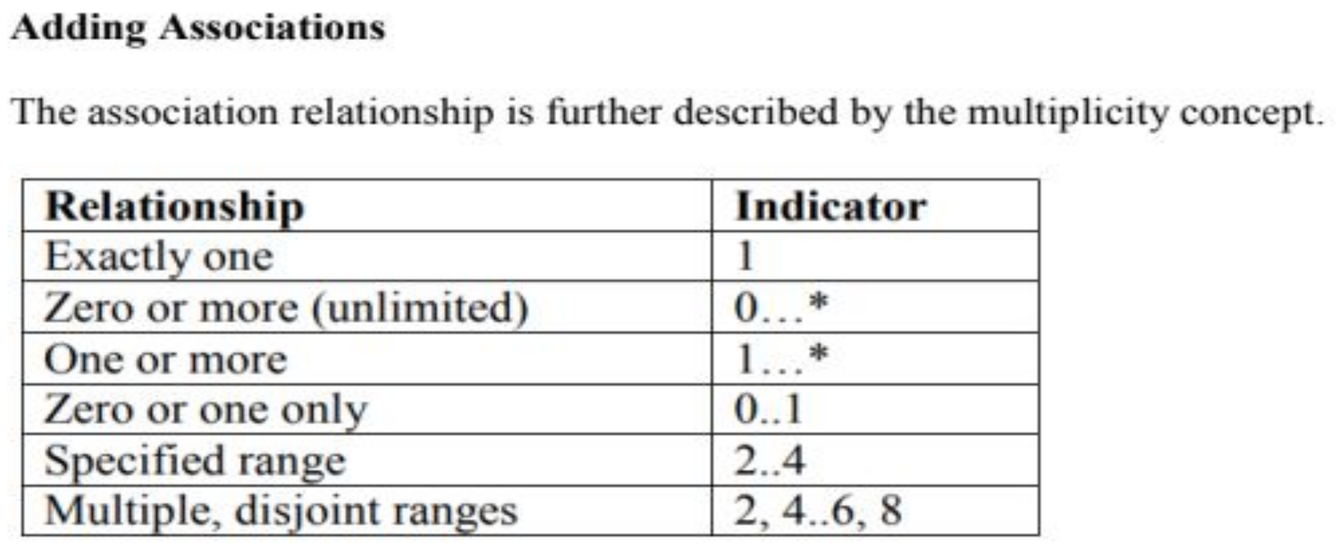
Associations

- Association: A relationship between (instances of) classes that indicates some meaningful and interesting connection.
- Used to show relationships that need to be remembered and preserved for some duration



Association

Association is a relationship between two objects. In other words, association defines the multiplicity between objects.

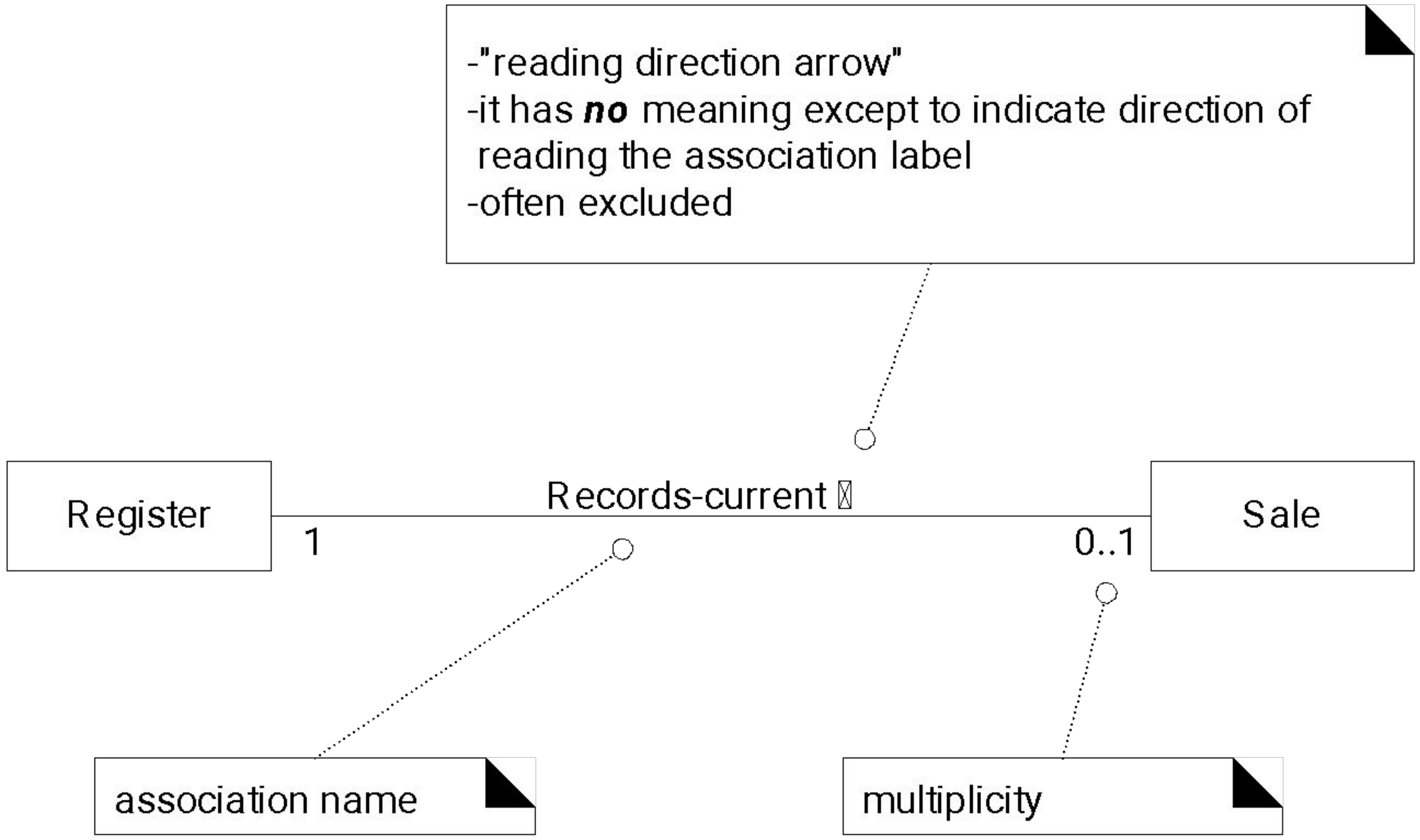


Adding Associations

The association relationship is further described by the multiplicity concept.

Relationship	**Indicator**
Exactly one	1
Zero or more (unlimited)	0…*
One or more	1…*
Zero or one only	0..1
Specified range	2..4
Multiple, disjoint ranges	2, 4..6, 8

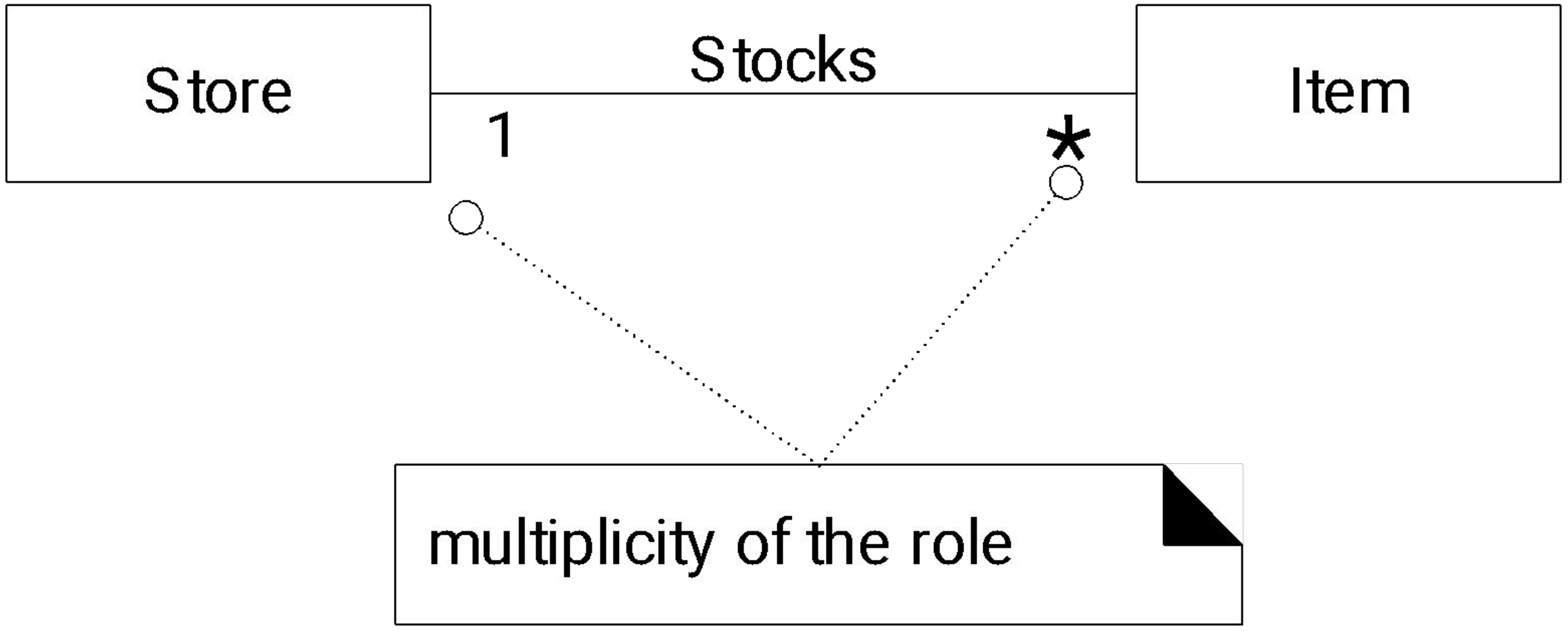
Association Directionalities


-"reading direction arrow"
-it has ***no*** meaning except to indicate direction of reading the association label
-often excluded
Register
Records-current ▶
Sale
1
0..1
association name
multiplicity

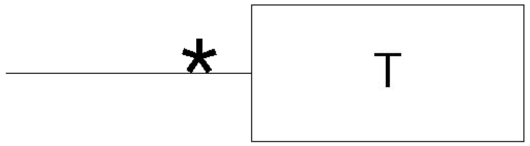
How to name an association?

- Pattern:
  - Class name – verb phrase – class name
  - Readable and meaningful
- Try to avoid “has” or “uses”. These give no extra information

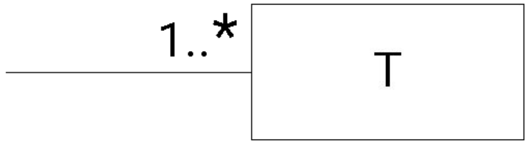
Association Multiplicities



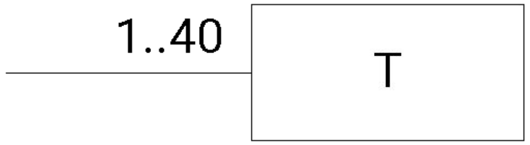
Multiplicities



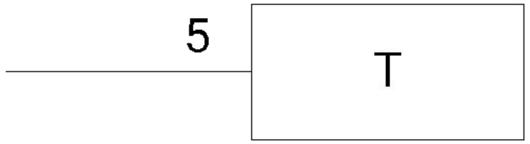
zero or more;
"many"



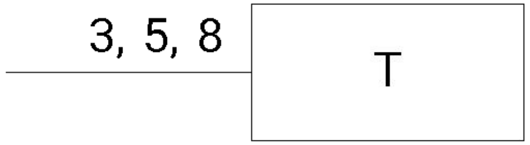
one or more



one to 40

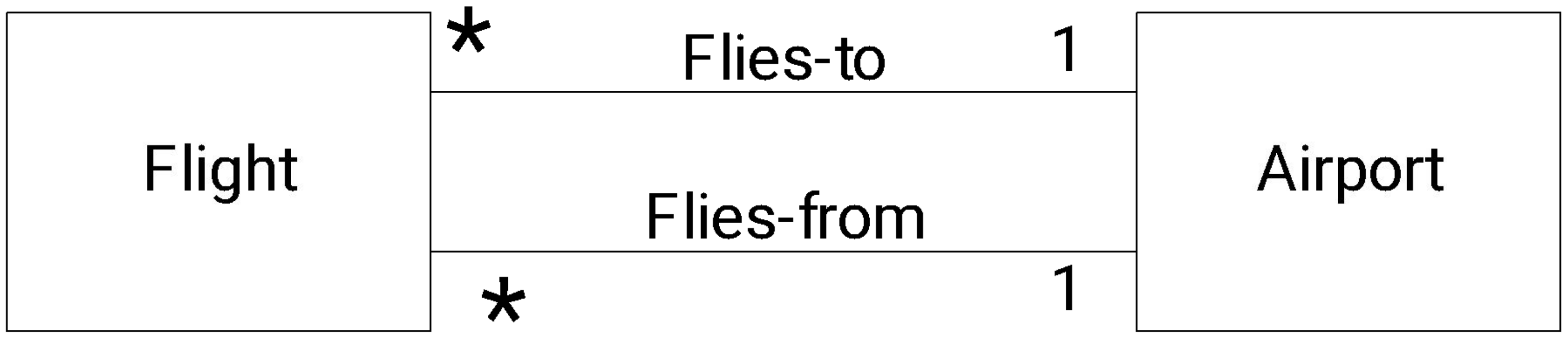


exactly 5



exactly 3, 5, or 8

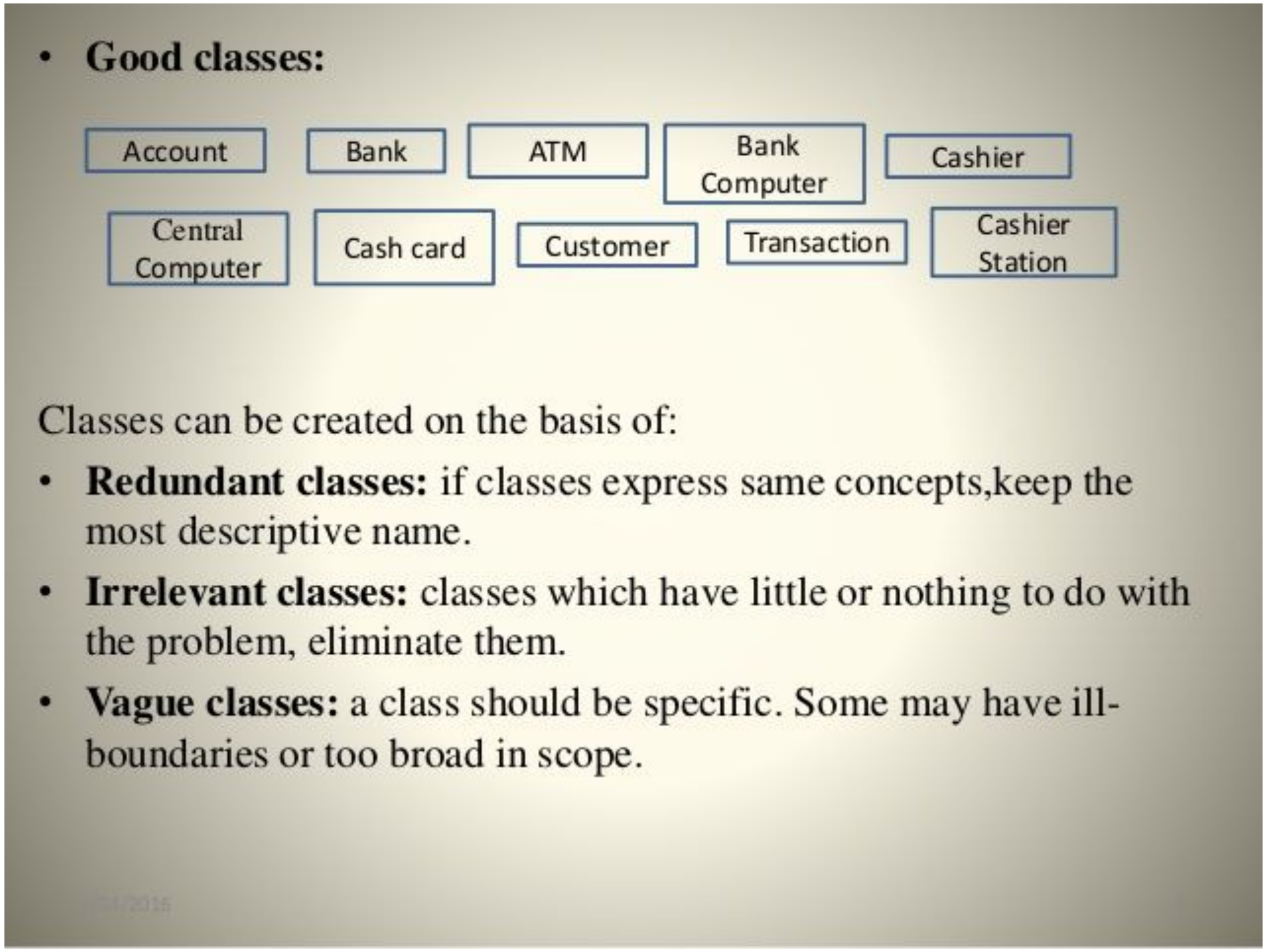
Multiple Associations



Scenario For Practice …

Automated Teller Machine (ATM) is a banking system. This banking system allows customers or users to have access to financial transactions. These transactions can be done in public space without any need for a clerk, cashier, or bank teller. The user is authenticated when enters the plastic ATM card in a Bank ATM, Then enters the user name and PIN (Personal Identification Number). For every ATM transaction, a Customer Authentication is required and essential. User checks the bank balance as well as also demands the mini statement about the bank balance if they want. Then the user withdraws the money as per their need. If they want to deposit some money, they can do it. After complete action, the user closes the session. If there is any error or repair needed in Bank ATM, it is done by an ATM technician. ATM technician is responsible for the maintenance of the Bank ATM, upgrades for hardware, firmware or software, and on-site diagnosis.

Something more on classes….



- **Good classes:**

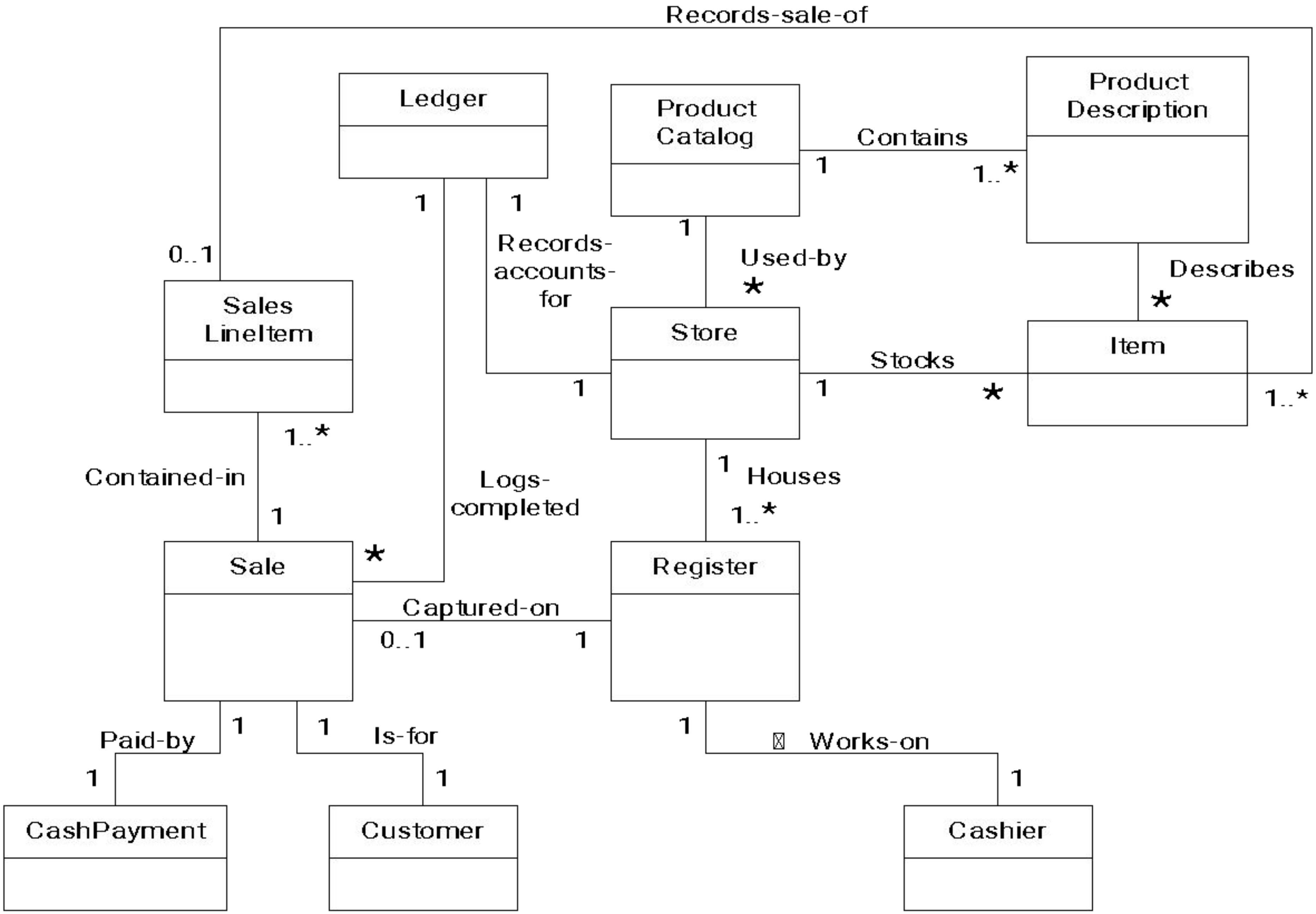
Account | Bank | ATM | Bank Computer | Cashier

Central Computer | Cash card | Customer | Transaction | Cashier Station

Classes can be created on the basis of:

- **Redundant classes:** if classes express same concepts,keep the most descriptive name.
- **Irrelevant classes:** classes which have little or nothing to do with the problem, eliminate them.
- **Vague classes:** a class should be specific. Some may have ill-boundaries or too broad in scope.

POS Domain Model Example:


Records-sale-of
Ledger
Product Catalog
Product Description
Contains
1
1..*
1
1
1
0..1
Records-accounts-for
Used-by
*
Describes
*
Sales LineItem
Store
Item
Stocks
1
1
*
1..*
1..*
Contained-in
1
Logs-completed
1
Houses
1..*
Sale
*
Register
Captured-on
0..1
1
1
1
Paid-by
Is-for
1
Works-on
1
1
1
CashPayment
Customer
Cashier