UNIT-III

Domain Model Example

MADHESWARI.K
AP/CSE
SSNCE

Domain Model

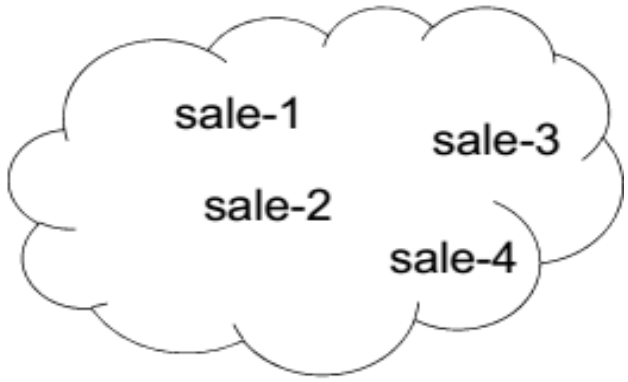
- A visual representation of conceptual classes or real-world objects in a domain of interest
- ***IS NOT*** a visualization of software components
- **IS**
  - *Class diagrams* (UML notation) show:
    - conceptual classes
    - attributes of conceptual classes
    - associations between conceptual classes

Conceptual classes

- Three ways of viewing a conceptual class:
  - Symbol
  - Intension
  - Extension
- In a domain model, we are interested in both the symbol and the intension of a conceptual class.

Sale
date time

"A sale represents the event of a purchase transaction. It has a date and a time."



How to identify conceptual classes

1. Reuse and modify existing models
2. Use a category list
3. Identify noun phrases

Category list

Conceptual class category	Examples
Business transaction	**Sale, Payment, Reservation**
Transaction line items	**SalesLineItem**
Product or service related to a transaction or transaction line item	**Item** **Flight, Seat, Meal**
Where is the transaction recorded	**Register, Ledger**
Place of transaction	**Store, Airplane, Seat**

NextGen POS system

Use Case	1.Process_Sale
Descirption	A user arrives at a POS sales point and tries to purchase an item.
Actors	Cashier (Primary)
Assumptions	Cashier is identified and authenticated
Steps	1. Customer arrives at POS checkout with goods to purchase. 2. Cashier starts a new sale. 3. WHILE more items DO 1. Cashier enters item identifier. 2. System records sale line item and presents item description, price and running total. END WHILE 4. System presents total with taxes calculated. 5. Cashier tells Customer the total, and asks for payment. 6. Customer pays and System handles payment. 7. System presents receipt. 8. Customer leaves with receipt and goods (if any).

How to identify conceptual classes

Linguistic analysis: Identify noun phrases in the use cases.

Process Sale ...

Main Success Scenario:

1. **Customer** arrives at a **POS checkout** with **goods** 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 **total**. ...

Use a conceptual class category list.

- physical objects
- specifications, descriptions
- places
- transactions
- transaction line items
- roles of people
- containers or contained item
- event
- catalog etc.

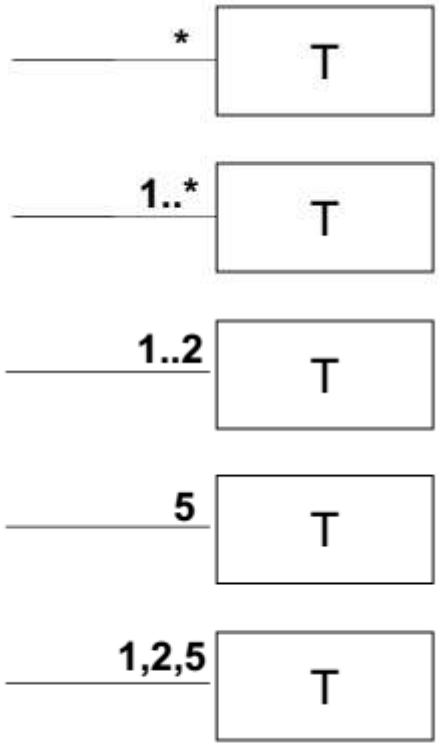
How to identify associations

- An association is a relationship between object instances that indicates some meaningful and interesting connection
  - worth *remembering*
  - derived from the Common Associations List
    - *A is a physical (or logical) part of B*
    - *A is physically (or logically) contained in*
    - *A is a description of B*
    - *A is known/captured/logged/recorded in B*
    - *A uses or manages B*
    - *A is related to a transaction B*
    - *Etc.*

Candidate associations

- Relationships that need to be remembered:
  - Register **Records** Sale (why?)
  - Sale **Paid-by** Payment (why?)
  - ProductCatalog **Records** ProductSpecification (why?)
- Relationships derived from the Common Associations List
  - SalesLineItem **Is-contained-in** Sale
  - Store **Contains** Item
  - ProductSpecification **Describes** Item
  - Cashier **Is-member-of** Store
  - Cashier **Uses/Manages** Register
  - Customer/Cashier Makes/Receives (Is-related-to) Payment
  - Etc.

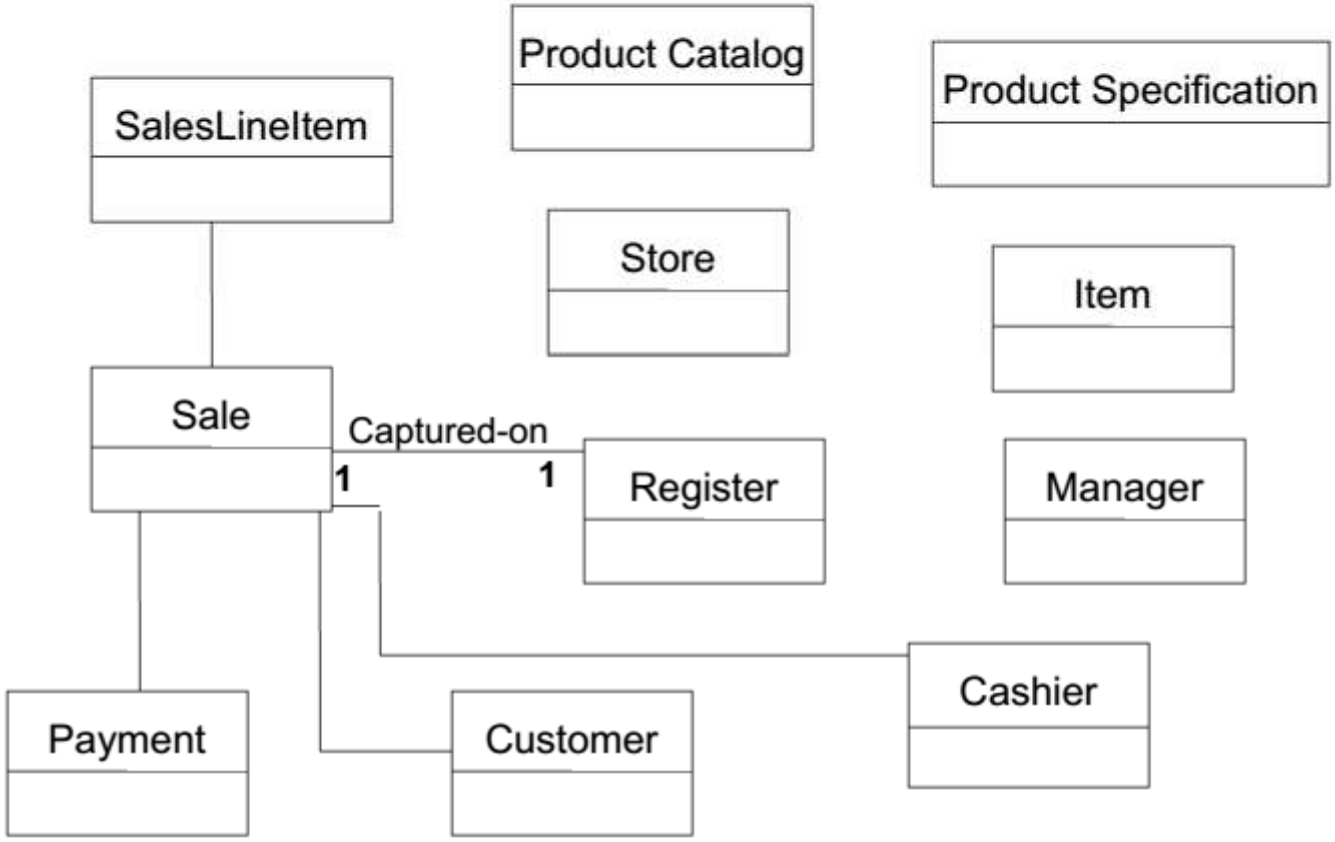
Multiplicity of associations



Attention!

The multiplicity value of a relationship indicates how many object instances can be validly associated with another, **at a particular moment**, rather than over a span of time.

Adding associations (names/multipl.)



How to identify attributes

- Attributes in a domain model should preferably be **simple attributes** or **data types**
- Common types: boolean, date, number, string, text, time
- Other types: address, color, geometrics, phone number, national insurance number, universal product code, postal codes
- Do not relate conceptual classes with an attribute!
- Represent a data type as a non-primitive class if:
  - *It is composed of separate sections*
  - *Operations are associated with it (e.g. parsing)*
  - *It has other attributes*
  - *It is an abstraction of one or more types*

Adding attributes