15CSE202 Object Oriented Programming
Lecture 5

OO Analysis with Domain / Object Model

Nalinadevi Kadiresan
CSE Dept.
Amrita School of Engg.

Object Oriented Analysis - recap

- Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain
- Steps in examining requirements are:
  1. Use Case Document preparation
  2. Object Model preparation

What is a domain model?

- Domain modeling is a technique used to understand the project problem description and to translate the requirements of that project into software components of a solution.
- The software components are commonly implemented in an object oriented programming language.
- A domain model contains conceptual classes, associations between conceptual classes, and attributes of a conceptual class.
- "Informally, a conceptual class is an idea, thing, or object".

OOA through domain modeling

- The primary tasks in object-oriented analysis (OOA) are:
  - Identifying objects
  - Organizing the objects by creating object model diagram
  - Defining the internals of the objects, or object attributes
  - Defining the behavior of the objects, i.e., object actions
  - Describing how the objects interact

Domain Model in OO

- Is an Object and it has
  - identity,
  - state, and
  - behaviour.
- An object can be
  - related to other objects and
  - complex (with sub-objects).

Identifying Objects

1. Modify or reuse an existing model.
2. Use a conceptual class category list.
3. Identify noun phrases.

Example categories:

- **Physical or tangible objects**: quiz, game piece, die
- **Specification, designs, or description of things**: game piece image, marking scheme
- **Transaction:** game move, order item, work item grade
- **Roles of people:** marker, instructor, player
- **Container of things**: game board, card deck
- **Events**: start turn, finish game
- **Rules and policy**: games ruler, submit assignment policy
- **Records**: work item grade

Using Noun Phrases

- Conceptual class can be identified by studying the use case looking for relevant noun phrases
- Some heuristics for identification are:
  - terms that developers or users need to clarify in order to understand the use case,
  - recurring nouns (higher frequency),
  - real world entities that the system needs to track,
  - real world activities
  - data sources or sinks

Categories of Association

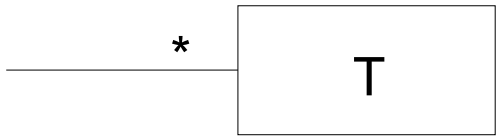
Association Example	Example
A is a physical part of B	Memory is part of a CPU
A is a logical part of B	The p tag is grouped by a div tag
A is contained in B	A card is in a deck
A is a description of B	A house plan describes a house
A is a line item of B	A part in a part list
A is reported in B	An error event is reported in a log
A is a member of B	A faculty member is a member of a department
A used B	Tax rules are used to calculate the total in a point of sales terminal
A communicates with B	A computer communicates with a printer

Heuristics for Identifying Associations

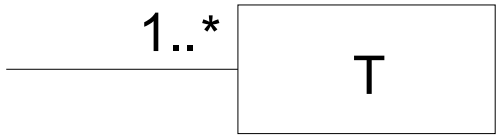
- Some heuristics for identifying associations are:
  - examine verb phrases,
  - ensure that the roles and association names are clear,
  - only add an association if it improves the understanding of the domain,
  - wait until the list of associations are stable before considering the multiplicity

Multiplicity in Association

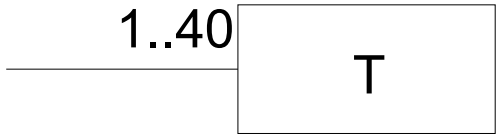
- Multiplicity defines how many instances of a class *A* can be associated with one instance of a class *B*.



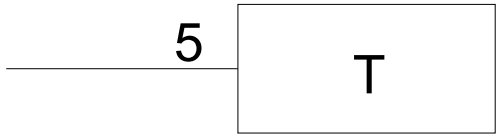
zero or more;
"many"



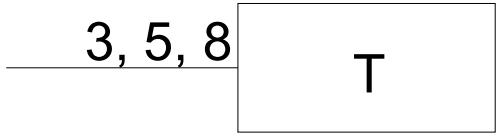
one or more



one to 40



exactly 5



exactly 3, 5, or 8

Identifying Attributes

- An attribute is a logical data (property) of an object (e.g., my eyes are hazel).
- Most attributes can be represented by simple data types
- Some heuristics for identifying attributes:
  - an attribute is part of the state of an object (a car's speed is 100 km/h, weight of a work item)
  - attributes are required by the use case (i.e., ignore irrelevant attributes).

Making a domain model

1. Identify conceptual classes
2. Draw the class diagram
3. Add any associations between classes
4. Add attributes (properties) to the classes

Example-1 Car Rental Application Use case

- **Use Case Name**: Release a Vehicle (to a Customer)
- **Actors**:
  1. Front-Desk Clerk
  2. Customer
- **Preconditions**: Vehicle has been assigned to the customer

Basic Flow ("Sunny Day Scenario"):

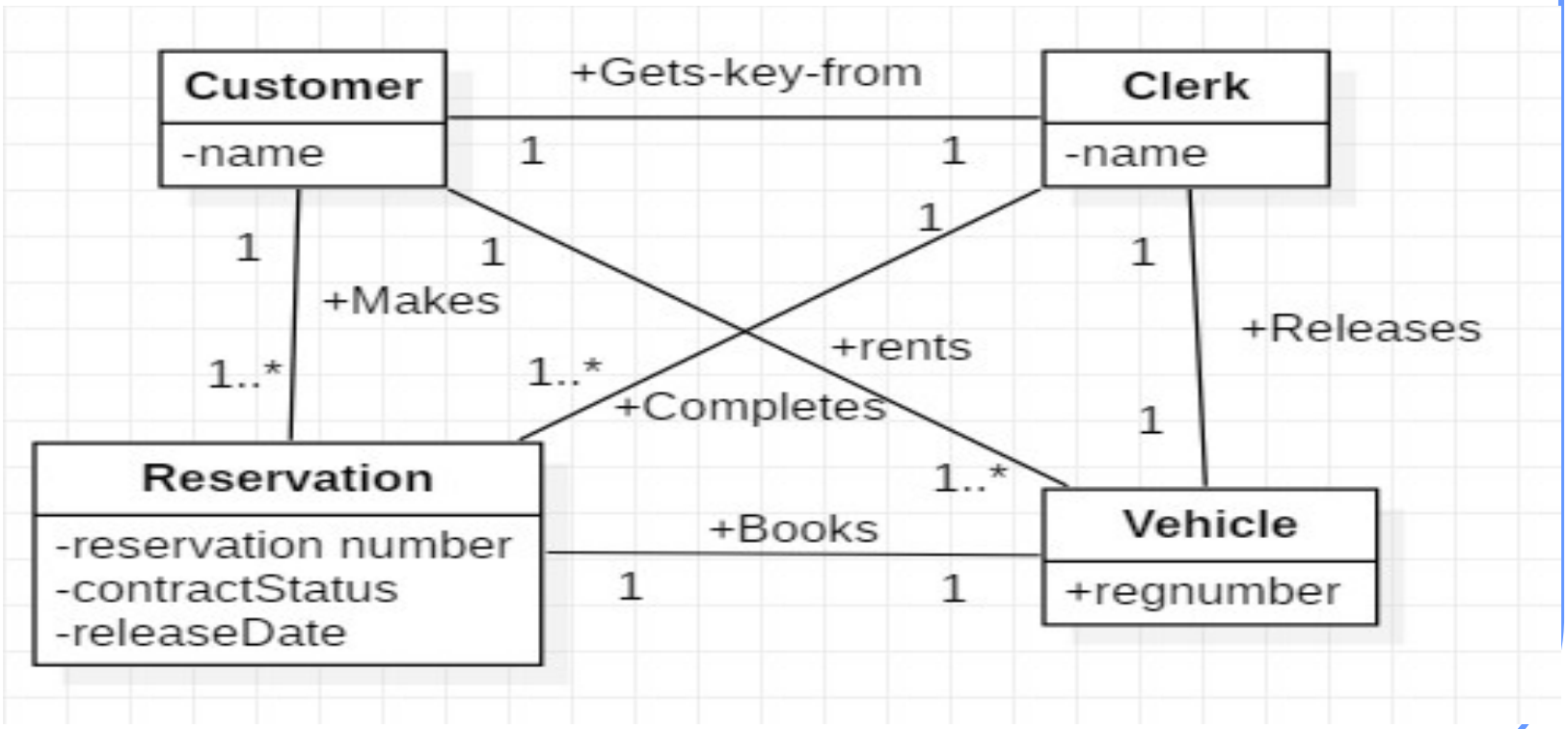
1. A **customer** comes to the **office** to acquire a **vehicle**.
2. The **clerk** locates the **vehicle reservation contract** by means of the **reservation number** and/or **customer name**. [Exception: Required vehicle type is not available due to late arrivals.]
3. The **customer** signs the **contract** and the **clerk** gives the **keys** to the **vehicle**.
4. The **clerk** then marks the **contract** active by entering the **vehicle release date** (today's date) onto **the vehicle reservation contract**.
5. The use case terminates at this point.

Example-1 Car Rental Applicat
Use case

Each color represents a conceptual class

- **Noun-Phrases Identified**: Customer, Customer name, Clerk, Office, Vehicle, keys, Vehicle reservation contract, Reservation number, Vehicle release date
- **Noun as conceptual class:**
  - Customer, Clerk, Vehicle, Reservation

Example-1 Car Rental Application Object Model



Example-2 Coin Flip

Use Case Name: Coin Flip

Actor:

1. A **player** who makes a prediction
2. A **player** who gets the other option
3. **Coin**
4. **Coin game**

Pre-condition:

1. **Two players are available**
2. **A coin is available**

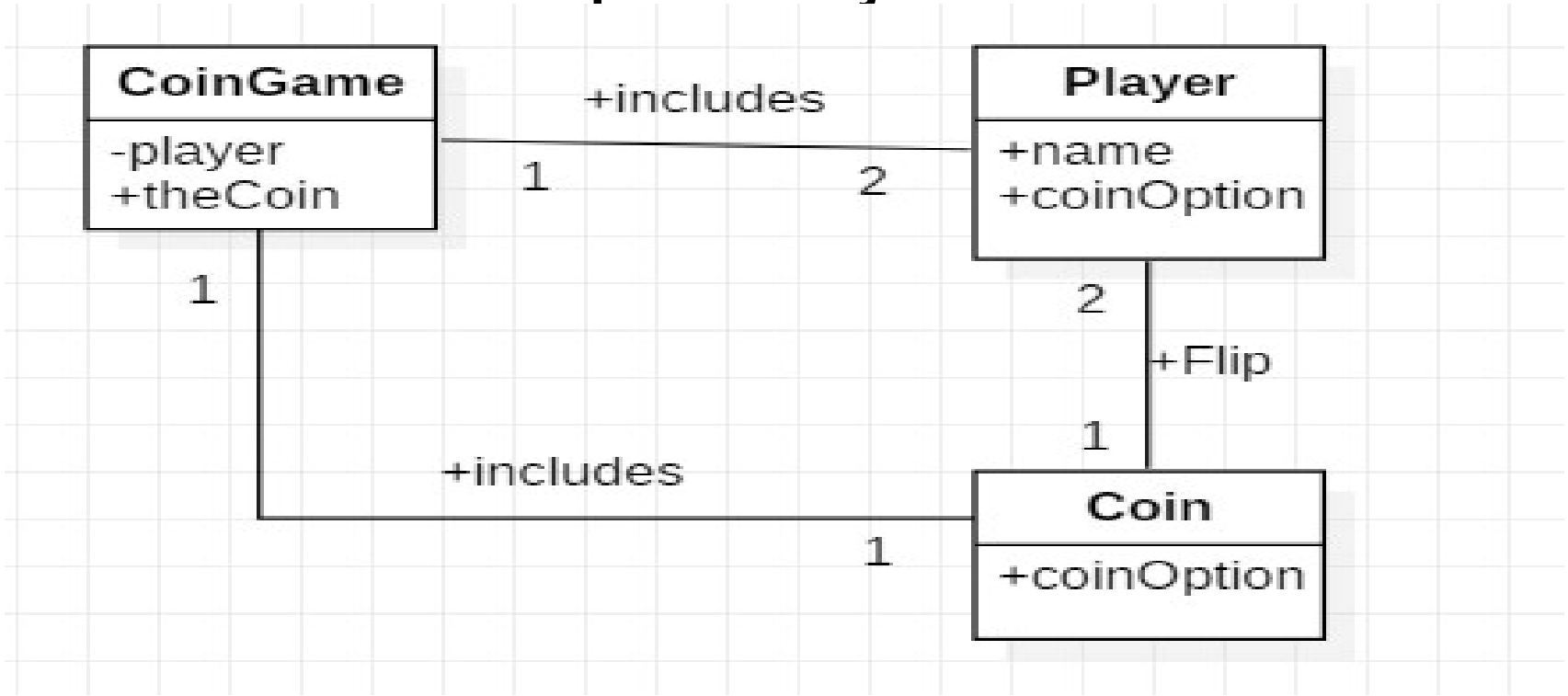
Basic Flow:

1. A **player** at random is picked to predict the **coin** flip
2. The **player** picked offers a prediction of **coin** flip
3. The other **player** gets the other **coin** flip option
4. The **coin** is flipped and **result** is provided
5. A **winner** and **loser** is picked
6. Offer to try again

Noun Phrases: Player, Coin, Coin game, winner, loser

Same as Player

Example-2
Coin Flip -- Object Model



Example-3: Online Support Request Use Case

Use Case Name: Support Request

Actor:

1. Person who generates support request by clicking button
2. Customer Support Representative

Pre-condition:

1. The person who generates the request needs to have an active internet connection
2. The website needs to be accessible by the customer
3. The support agent needs to have an active internet connection

Example-3: Online Support Request Use Case

- **Basic Flow:**
  1. The customer visits the support website
  2. The customer clicks the "generate support ticket button"
  3. The customer is taken to a page where they are told that support will be present shortly
  4. The ticket is sent to the support department
  5. A customer support representative takes hold of the ticket
  6. The customer support representative is sent to the same page as the user where there is a chat box
  7. The customer support representative provides the help and support to the customer
  8. The customer closes the chat window
  9. The customer support representative closes the support ticket and enters information about the session.
  10. The use case ends

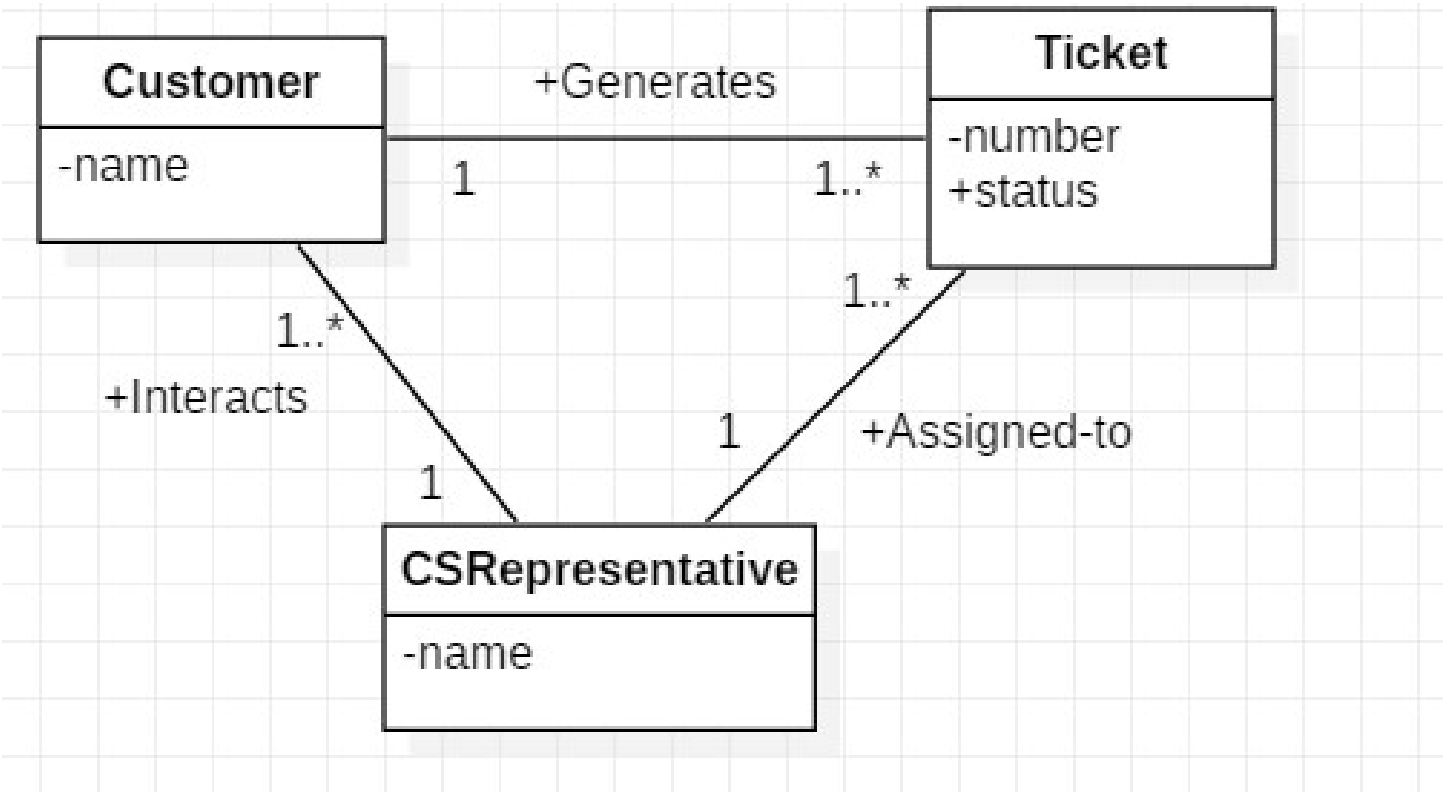
Noun Phrases: customer, support website, support ticket button, ticket, support department, customer support representative, chat window

Example-3: Online Support Request Noun Identification

- **Noun Phrases: customer, support website, support ticket button, ticket, support department, customer support representative, chat window, page**
- **Classes Identified**
  - **Customer, Website, Chat window, Ticket, Support Department, CS Representative**
- **Conceptual Classes**
  - **Customer, Ticket, CS Representative**

Example-3: Online Support Request

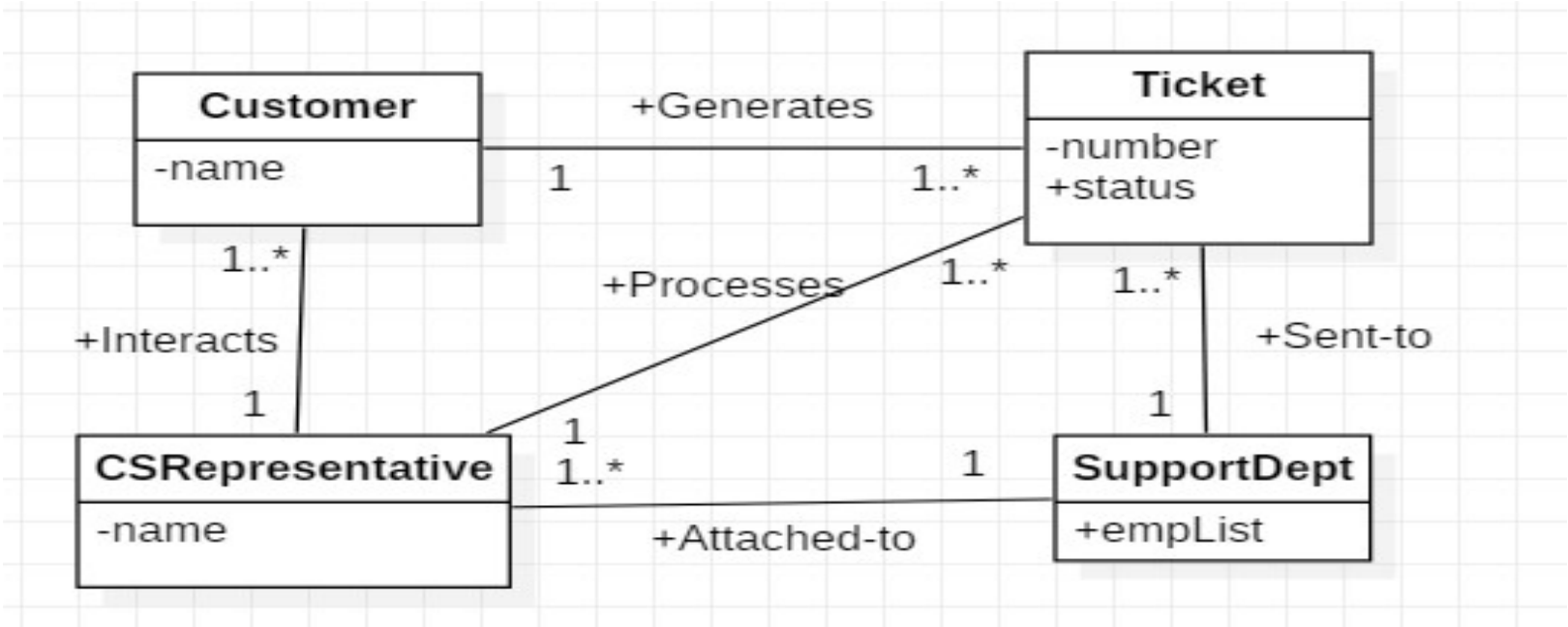
Simple Object Model



Example-3: Online Support Request Fine –grained Object Model

- **Noun Phrases: customer, support website, support ticket button, ticket, support department, customer support representative, chat window, page**
- **Classes Identified**
  - **Customer, Website, Chat window, Ticket, Support Department, CS Representative**
- **Conceptual Classes**
  - **Customer, Ticket, Support Department, CS Representative**

Example-3: Online Support Request Fine –grained Object Model



Need Not Add Website

Example-4 Process Sale

- **Brief:** A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items

Example-4 Process Sale Use Case

- Primary Actor: Cashier
- Main Success Scenario (or Basic Flow):

1. Customer arrives at 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 3-4 until indicates done.

Example-4 Process Sale Use Case

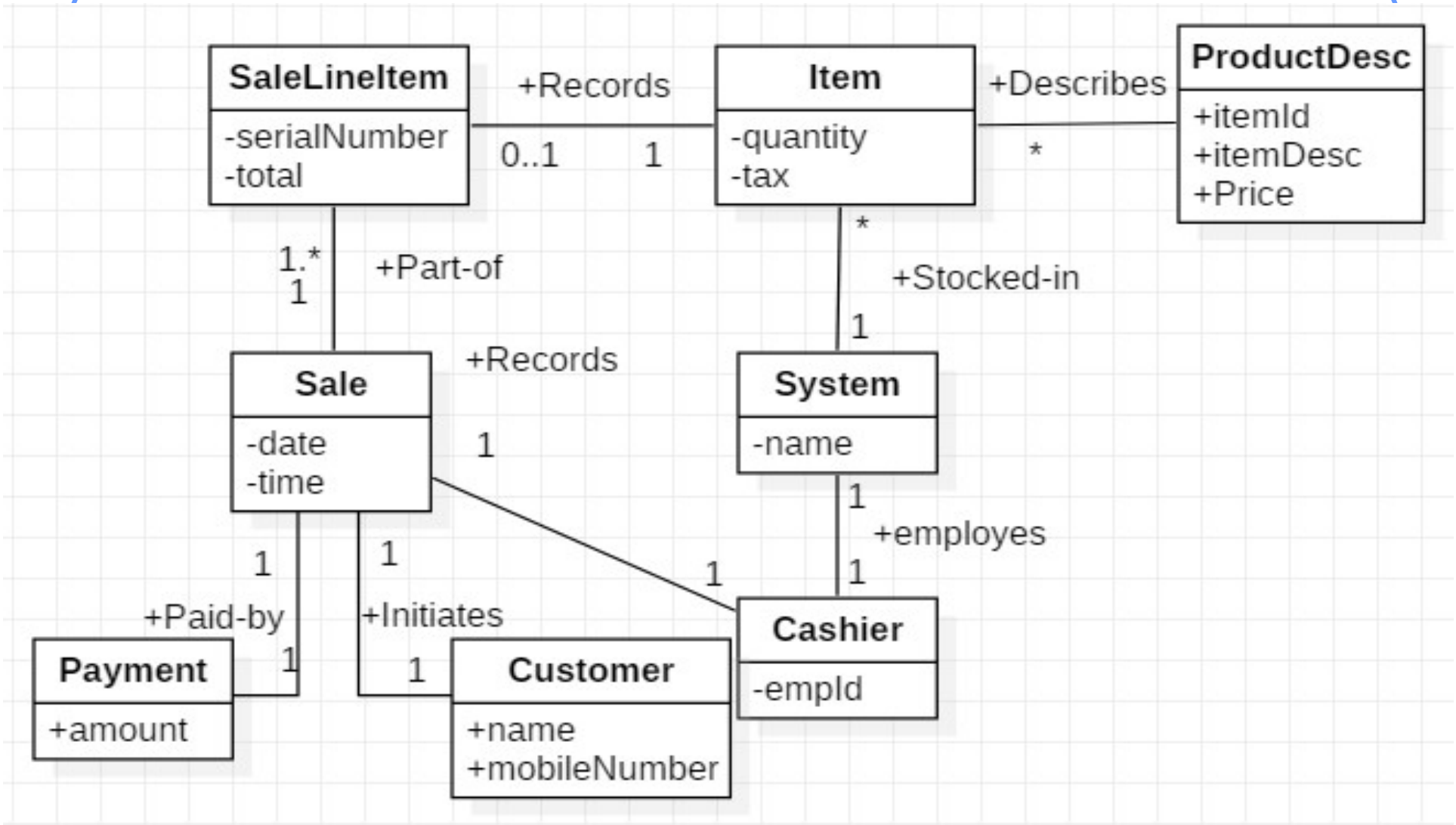
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory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

श्रद्धावान् लभते ज्ञानम्

Example-4 Process Sale
Conceptual Classes

Noun Phrases	Conceptual Class
Cashier	**Cashier**
Customer	**Customer**
POS Checkout	**Sale**
Sale, Receipt	
Sale Line item, Goods/Services, taxes, running total	**Sale Line Item, Item**
Item identifier, item description, price	**ProductSpec**
System	**Store**
Payment	**Payment**

Example-4 Process Sale --Object Model



Try!!!!!!!!!!
Use case and Object Model

Use Case: Enroll in Course (Casual Version)

Student must be registered in the system before starting this use case.

The student logs onto the system and may search for courses to enroll in. The Student searches by course name, course code, department, level and availability. The Student then selects the course to enroll in and the system will respond by displaying the course details. The student then confirms the enrollment and logs out of the system or enrolls for more courses.

If the system goes down at any time, the system will display an appropriate error message and the use case will end. If the student tries to enroll for a course that is full, the system will inform the Student. The Student may then continue to enroll in other courses.

The system must respond to any search request within 3 seconds. The system must be able to perform up to 1000 concurrent searches. The System should be able to support 500 concurrent users enrolling in classes. Class enrollment should take no longer than 10 seconds to confirm.

Next Session will be

Class and Objects in Java