

# Object Oriented Analysis and Design Using the UML

---

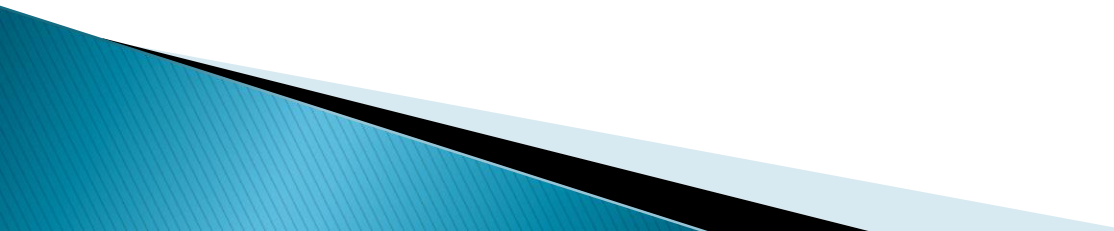
## Object Oriented Analysis



# Object-Oriented Analysis And Design (OOAD)

---

It's a structured method for analyzing, designing a system by applying the object-oriented concepts (abstraction, encapsulation, inheritance , polymorphism), and developing a set of graphical system models during the development life cycle of the software.



# Object Oriented Analysis

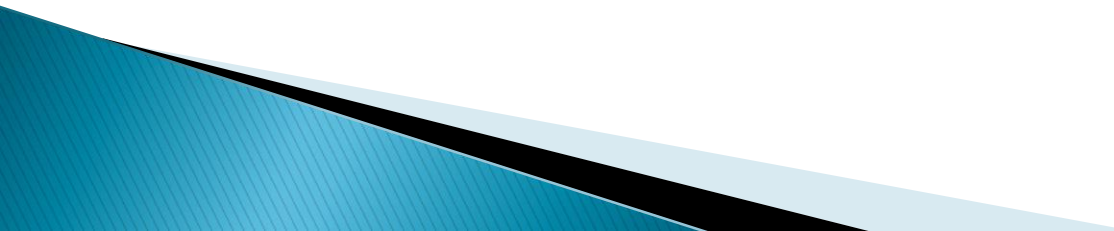
## Process of Object Modeling

---

**Elicit requirements:** Define what does the software need to do, and what's the problem the software trying to solve.


**Specify requirements:** Describe the requirements, usually, using use cases (and scenarios) or user stories.

**Conceptual model:** Identify the important objects, refine them, and define their relationships and behavior and draw them in a simple diagram.



# Object-oriented development

---

- ▶ Object-oriented analysis, design and programming are related but distinct
  - ▶ OOA is concerned with developing an object model of the application domain
  - ▶ OOD is concerned with developing an object-oriented system model to implement requirements
  - ▶ OOP is concerned with realising an OOD using an OO programming language such as Java or C++
- 

# Object Oriented Analysis

## Process of Object Modeling

---

- ◆ **Identifying objects:**
  - Using concepts.
- ◆ **Organising the objects:**
  - classifying the objects identified, so similar objects can later be defined in the same class.
- ◆ **Identifying relationships between objects:**
  - this helps to determine inputs and outputs of an object.
- ◆ **Defining operations of the objects:**
  - the way of processing data within an object.
- ◆ **Defining objects internally:**
  - information held within the objects.

# Goals of OO Analysis

---

- ▶ What are the two main goals of OO analysis?
  - 1) Understand the customer's requirements
  - 2) Describe problem domain as a set of classes and relationships
- ▶ What techniques have we studied for the 1st goal?
  - Develop a requirements specification
  - Describe scenarios of use in user's language as use cases
- ▶ What techniques have we studied for the 2nd goal?
  - Use strategies to discover classes
  - Use UML class diagrams to represent classes & relationships
  - Use Sequence diagrams to model dynamic behavior of a system

# System Modeling

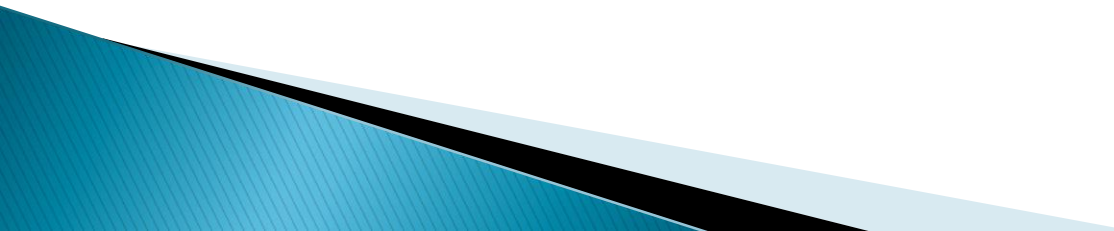
---

- ▶ System modeling is the process of developing models of the system, with each model representing a different perspectives of that system.
- ▶ The most important aspect about a system model is that it leaves out detail; It's an abstract representation of the system.
- ▶ The models are usually based on graphical notation, which is almost based on the notations in the Unified Modeling Language (UML).
- ▶ Models are used during the analysis process to help to elicit and specify the requirements, during the design process to describe how the system will be implemented, and the implementation to document the system structure and operation.

# Different Perspectives of System Modeling

---

The models represent the system from different perspectives.

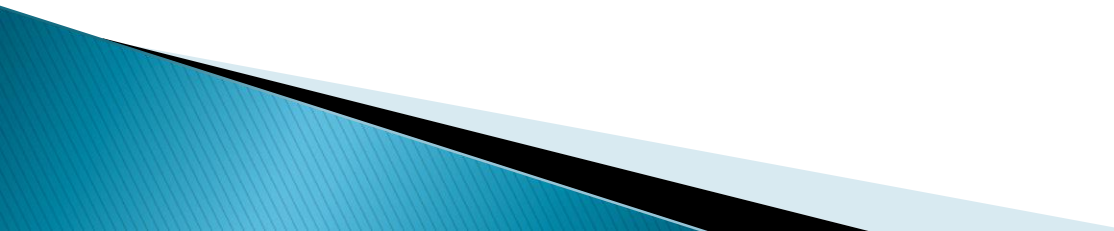
- ▶ **External**, where you model the context or the environment of the system.
  - ▶ **Interaction**, where you model the interaction between components of a system, or between a system and other systems.
  - ▶ **Structural**, where you model the organization of the system, or the structure of the data being processed by the system.
  - ▶ **Behavioral**, where you model the dynamic behavior of the system and how it respond to events.
- 



# Unified Modeling Language (UML)

---

The unified modeling language become the standard modeling language for object-oriented modeling. It has many diagrams, however, the most diagrams that are commonly used are:

- ▶ **Use case diagram:** It shows the interaction between a system and it's environment (users or systems) within a particular situation.
  - ▶ **Class diagram:** It shows the different objects, their relationship, their behaviors, and attributes.
  - ▶ **Sequence diagram:** It shows the interactions between the different objects in the system, and between actors and the objects in a system.
  - ▶ **Statechart diagram:** It shows how the system respond to external and internal events.
  - ▶ **Activity diagram:** It shows the flow of the data between the processes in the system.
- 

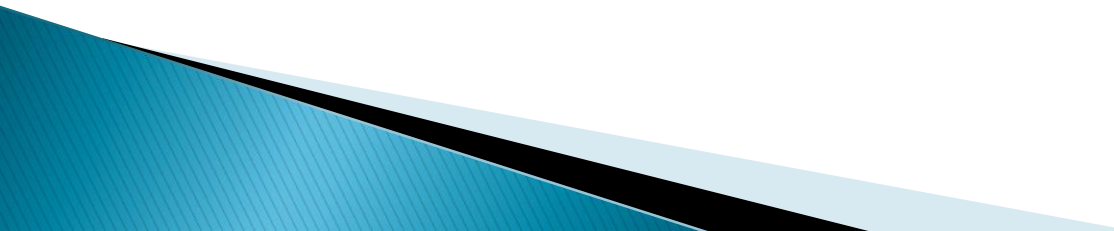
# Use case diagrams

---

Use Case Diagrams describe the functionality of a system and users of the system.

Describe the functional behavior of the system as seen by the user.

These diagrams contain the following elements:

- **Actors**, which represent users of a system, including human users and other systems.
  - **Use Cases**, which represent functionality or services provided by a system to users.
- 

# Use case diagrams

---

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

To draw an use case diagram we should have the following items identified. Functionalities to be represented as an use case  
Actors Relationships among the use cases and actors.

# Use case diagrams

---

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

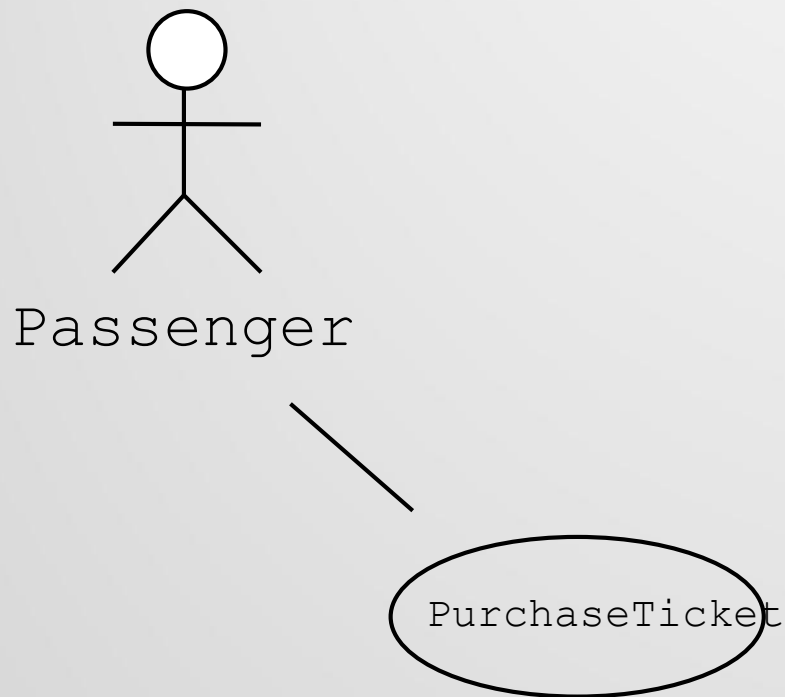
The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.

Use note when ever required to clarify some important points.



# Use Case Diagrams

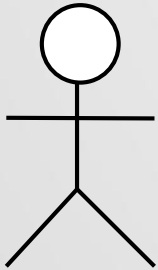
---



- ▶ Used during requirements elicitation to represent external behavior
- ▶ **Actors** represent roles, that is, a type of user of the system
- ▶ **Use cases** represent a sequence of interaction for a type of functionality
- ▶ The use case model is the set of all use cases. It is a complete description of the functionality of the system and its environment

# Actors

---



Passenger

- ▶ An actor models an external entity which communicates with the system:
  - User
  - External system
  - Physical environment
- ▶ An actor has a unique name and an optional description.
- ▶ Examples:
  - Passenger: A person in the train
  - GPS satellite: Provides the system with GPS coordinates

# Use Case

---

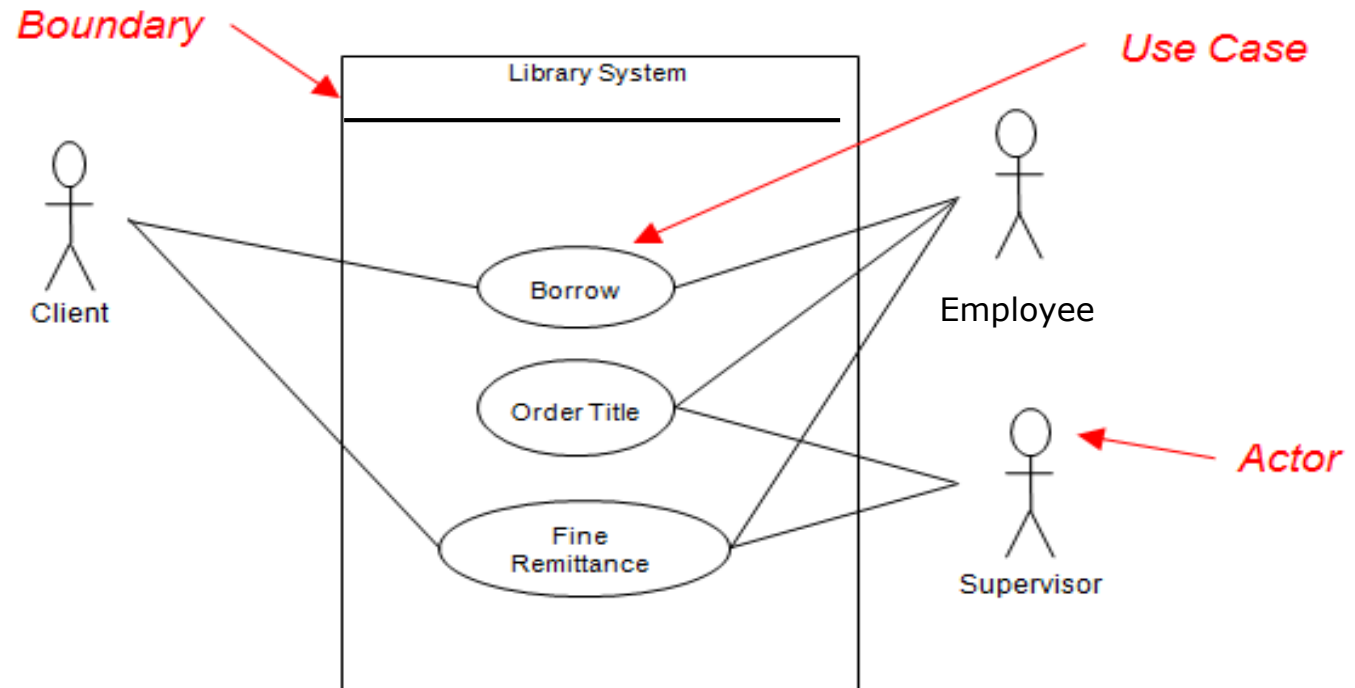


A use case represents a class of functionality provided by the system as an event flow.

A use case consists of:

- ▶ Unique name
- ▶ Participating actors
- ▶ Entry conditions
- ▶ Flow of events
- ▶ Exit conditions
- ▶ Special requirements

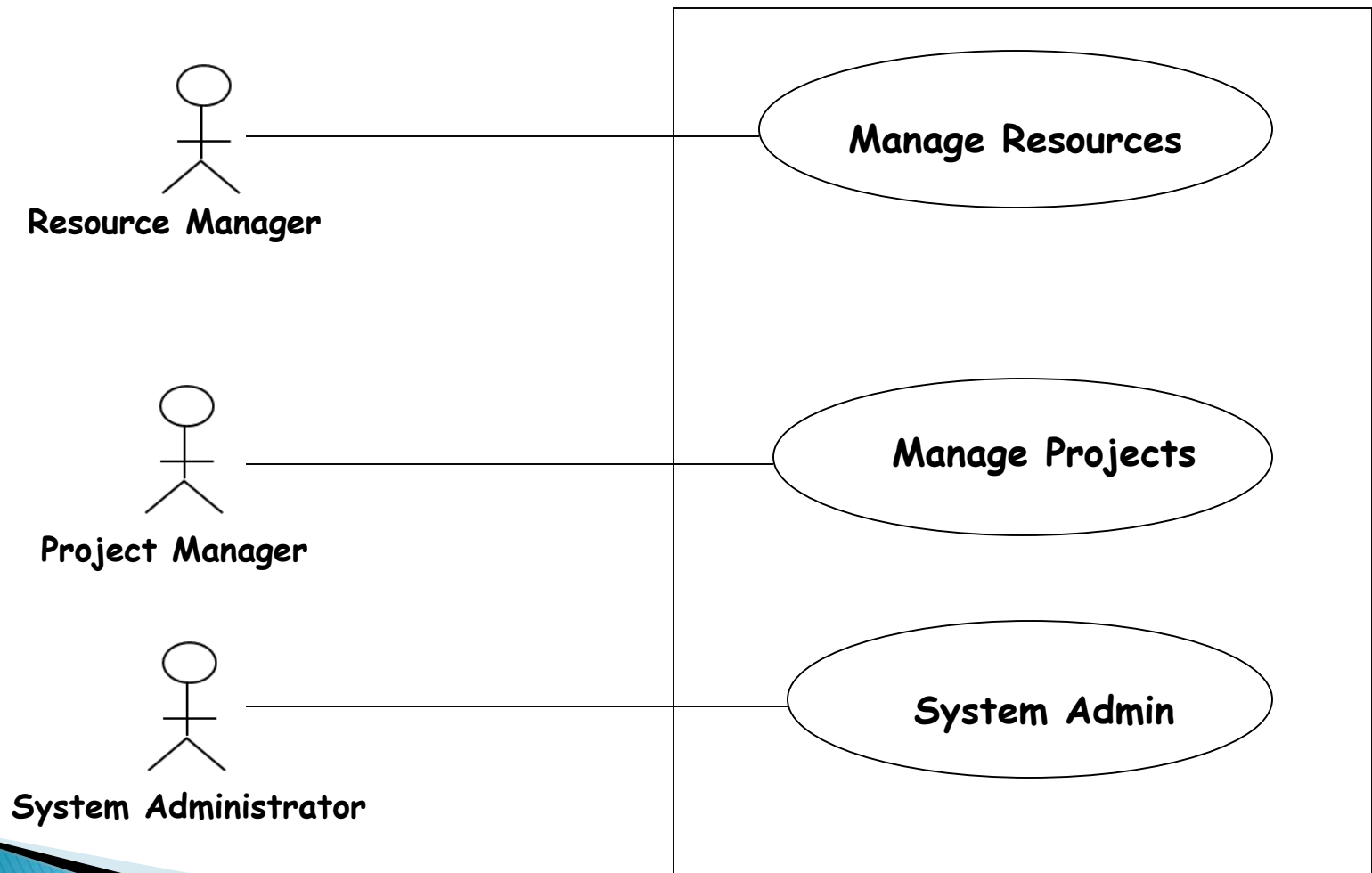
# Use case diagrams



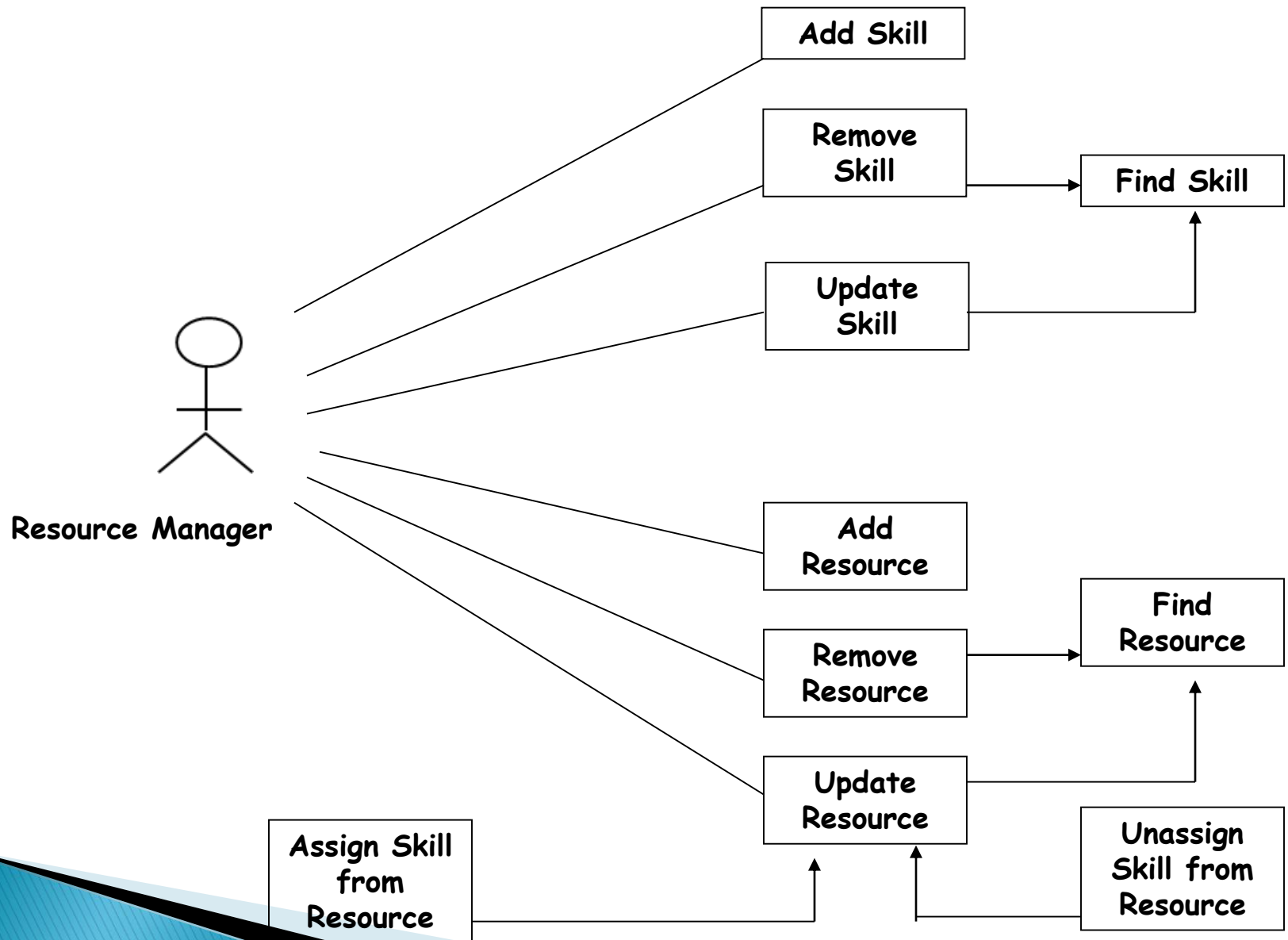


# Example: High Level Use Case Diagram

---



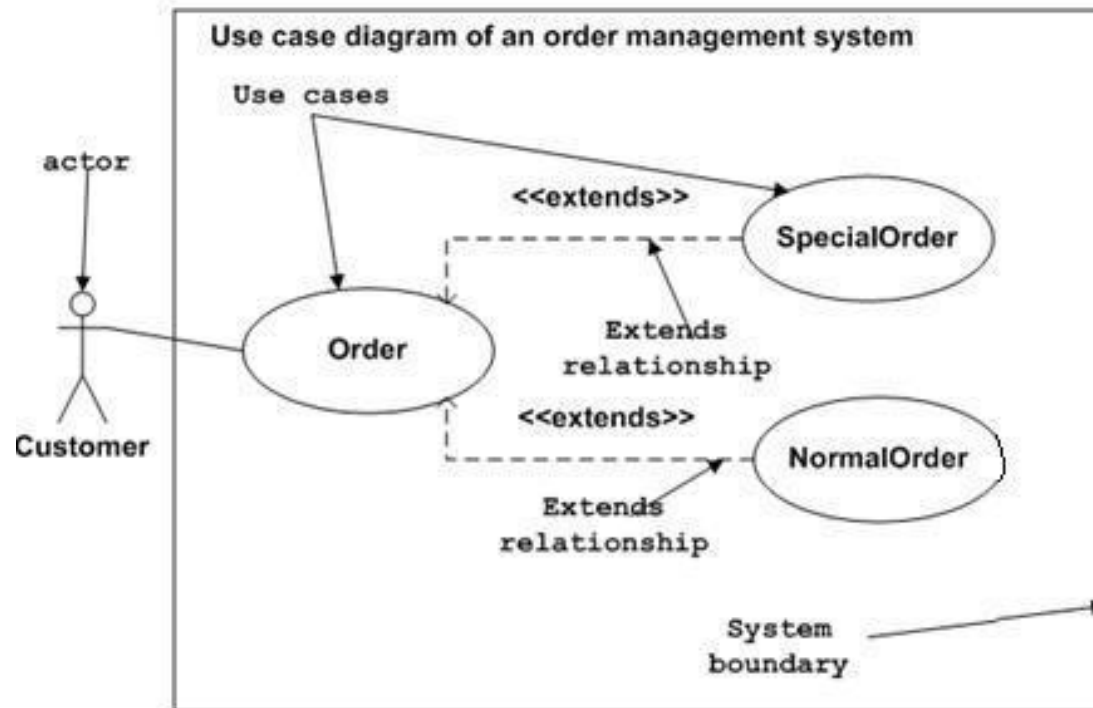
# Example: Managing Resources Use Case Diagram



# Example: Order management system

The following is a sample use case diagram representing the order management system. So if we look into the diagram then we will find three use cases (Order, SpecialOrder and NormalOrder) and one actor which is customer.

*The SpecialOrder and NormalOrder use cases are extended from Order use case. So they have extends relationship.*

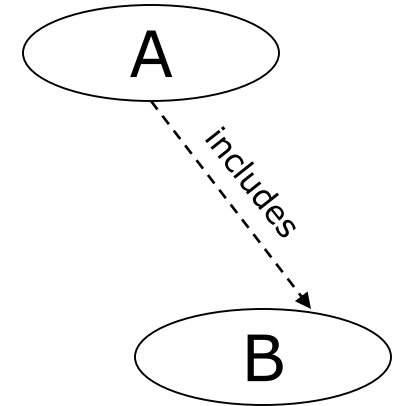


# Dependences

---

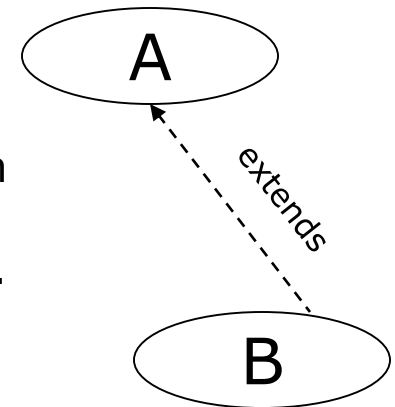
## Include Dependencies

An include dependency from one use case (called the base use case) to another use case (called the inclusion use case) indicates that the base use case will include or call the inclusion use case. A use case may include multiple use cases, and it may be included in multiple use cases.



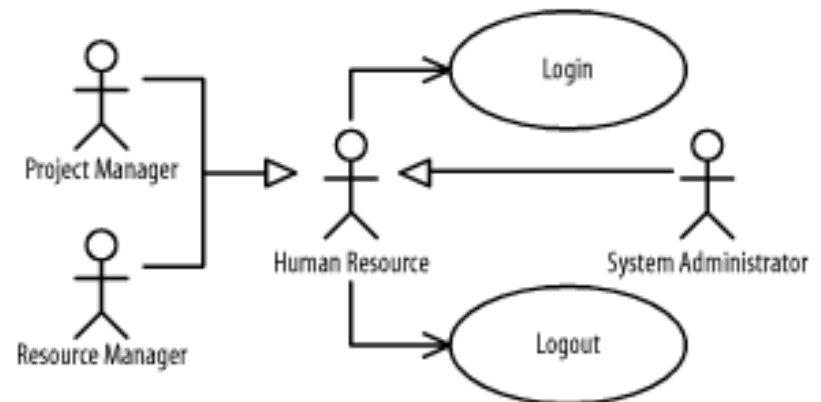
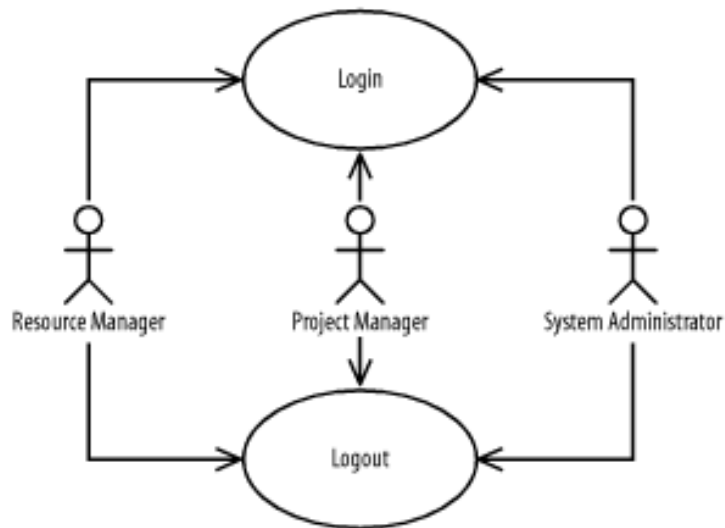
## Extend Dependencies

An extend dependency from one use case (called the extension use case) to another use case (called the base use case) indicates that the extension use case will extend (or be inserted into) and augment the base use case. A use case may extend multiple use cases, and a use case may be extended by multiple use cases.



# Generalizations

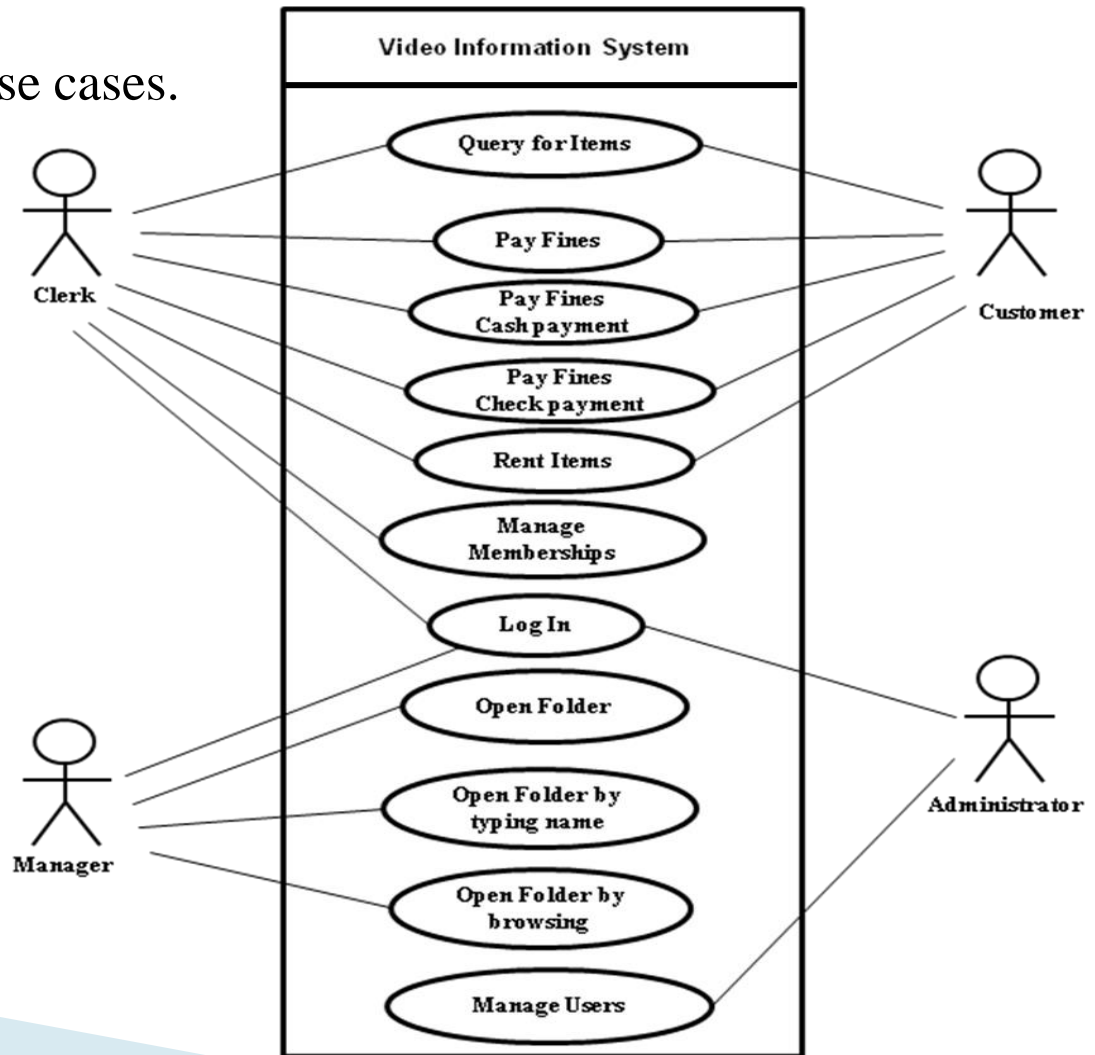
- Actors may be similar in how they use a system; for example, project managers, resource managers, and system administrators may log in and out of our project management system.
- Use cases may be similar in the functionality provided to users; for example, a project manager may publish a project's status in two ways: by generating a report to a printer or by generating a web site on a project web server.

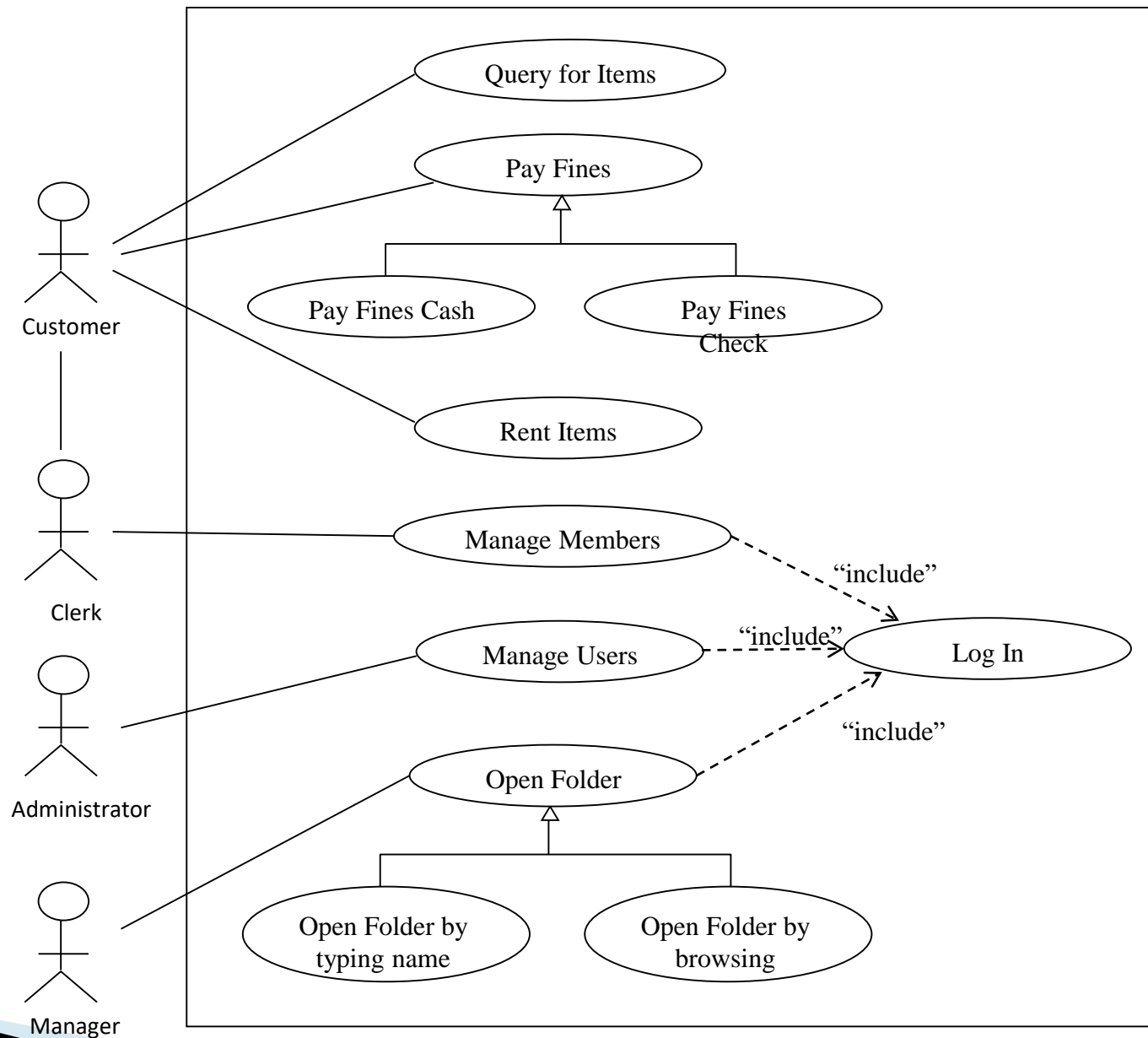


# Example

Redraw the given use case diagram after applying:

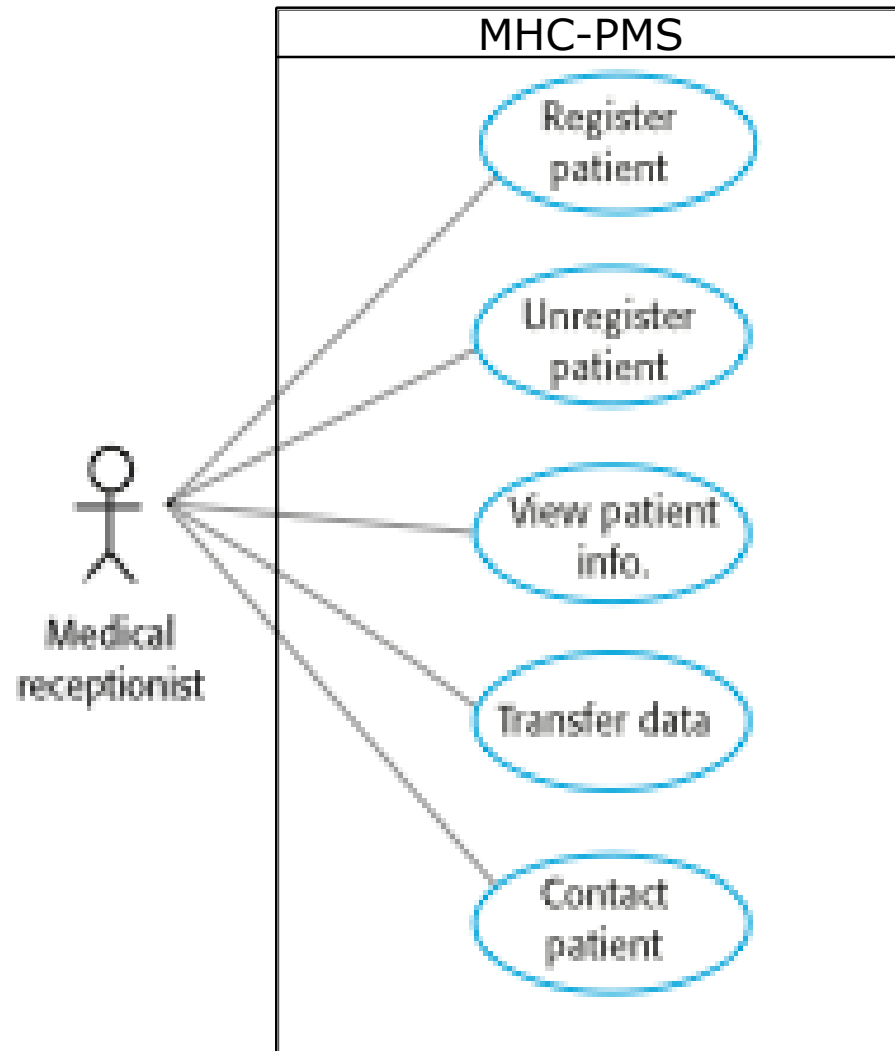
- Inheritance between actors
- Extend between use cases;
- Include relationship between use cases.





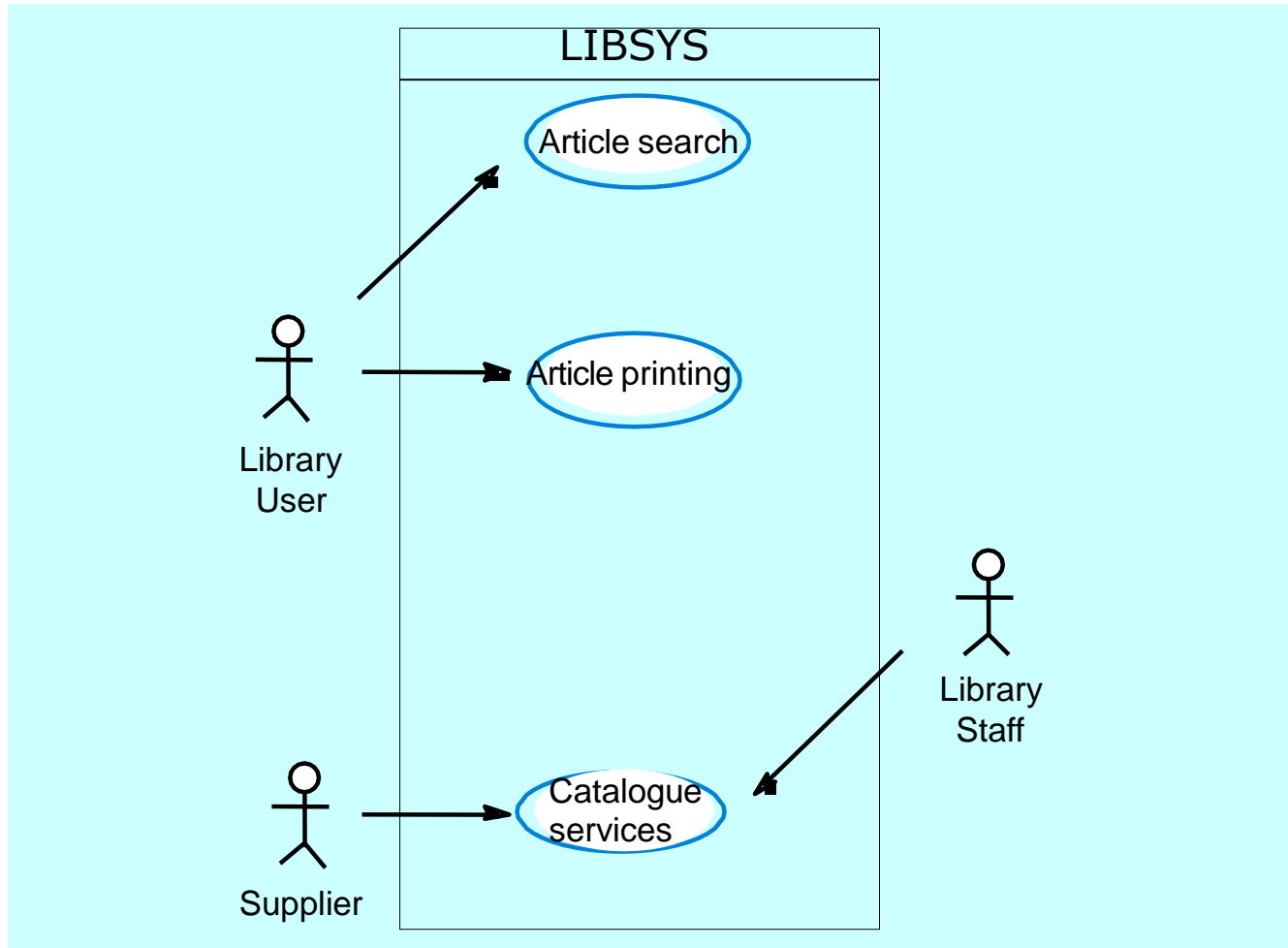
# Use cases in the MHC-PMS involving the role 'Medical Receptionist'

---





# Example: LIBSYS use cases



# Example: Trading House System

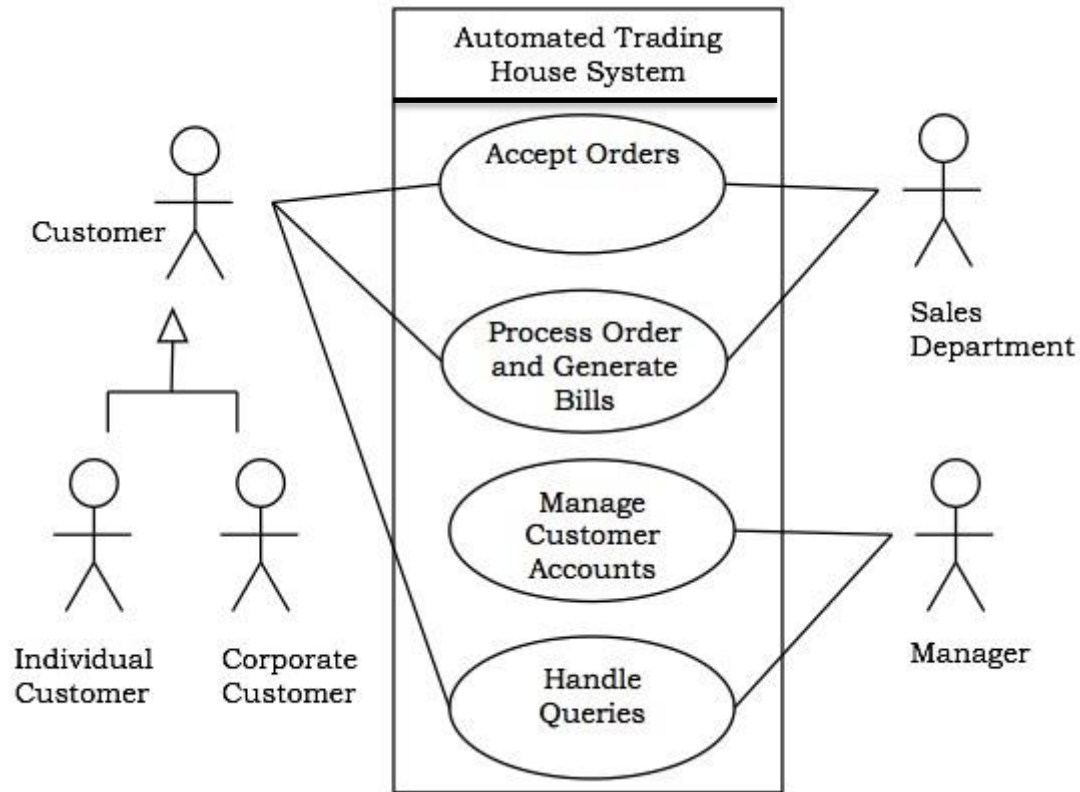
---

Let us consider an Automated Trading House System. We assume the following features of the system:

- The trading house has transactions with two types of customers, individual customers and corporate customers.
- Once the customer places an order, it is processed by the sales department and the customer is given the bill.
- The system allows the manager to manage customer accounts and answer any queries posted by the customer.

*Consider the following incomplete use case diagram and complete it. Use appropriate relation labels and direction arrows when needed.*

# UCD: Trading House System



# Use Case Description

---

**Use case *text* provides the detailed description of a particular use case**

Use Case ID:	Give each use case a unique integer sequence number identifier.		
Use Case Name:	Start with a verb.		
Created By:		Last Updated By:	
Date Created:		Date Last Updated:	

Actors:	Calls on the system to deliver its services.
Description:	"user-goal" or "sub-function"
Stakeholders and Interests:	Who cares about this use case, and what do they want?
Trigger:	Identify the event that initiates the use case.
Pre-conditions:	What must be true on start, and worth telling the reader?
Post-conditions:	Describe the state of the system at the conclusion of the use case execution.
Normal Flow:	A typical, unconditional happy path scenario of success.
Alternative Flows (Extensions):	Alternative scenarios of success or failure.
Priority:	Indicate the relative priority of implementing the functionality required to allow this use case to be executed.
Technology and Data Variations List	Varying I/O methods and data formats.
Special Requirements:	Related non-functional requirements.
Notes and Issues:	Such as open issues.

# Use Case Description

---

## ► Use Case Identification

- **Use Case ID**

Give each use case a unique integer sequence number identifier.

- **Use Case Name**

State a concise, results-oriented name for the use case. These reflect the tasks the user needs to be able to accomplish using the system. Include an action verb and a noun.

- **Use Case History**

- Created By
- Date Created
- Last Updated By
- Date Last Updated

- **Actors**

An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Name the actor that will be initiating this use case and any other actors who will participate in completing the use case.

- **Description**

Provide a brief description of the reason for and outcome of this use case.

- **Stakeholders and Interests**

Who cares about this use case, and what do they want?

- **Trigger**

Identify the event that initiates the use case. This could be an external business event or system event that causes the use case to begin, or it could be the first step in the normal flow.

# Use Case Description

---

## Use Case Definition

- **Pre-conditions**

List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each precondition. Examples:

- User's identity has been authenticated.
- User's computer has sufficient free memory available to launch task.

- **Post-conditions**

Describe the state of the system at the conclusion of the use case execution. Number each post-condition. Examples:

- Price of item in database has been updated with new value.

- **Normal (basic) Flow of events – Happy path – Successful path – Main Success Scenario**

Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description.

- **Alternative Flows (Extensions): Alternate scenarios of success or failure**

Document other, legitimate usage scenarios that can take place within this use case separately in this section. State the alternative flow, and describe any differences in the sequence of steps that take place. Number each alternative flow in the form "X.Y", where "X" is the Use Case ID and Y is a sequence number for the alternative flow. For example, "5.3" would indicate the third alternative flow for use case number 5.

# Use Case Description

---

- **Priority**

Indicate the relative priority of implementing the functionality required to allow this use case to be executed. The priority scheme used must be the same as that used in the software requirements specification.

- **Technology and Data Variations List**

Varying I/O methods and data formats.

- **Special Requirements**

Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.

- **Notes and Issues**

List any additional comments about this use case or any remaining open issues or TBDs (To Be Determined) that must be resolved. Identify who will resolve each issue, the due date, and what the resolution ultimately is.

# Example of Use Case Description

Use Case Name: Borrow Books		ID: 2	Importance Level: High
Primary Actor: Borrower		Use Case Type: Detail, Essential	
Stakeholders and Interests: Borrower - wants to check out books Librarian - wants to ensure borrower only gets books deserved			
Brief Description: This use case describes how books are checked out of the library.			
Trigger: Borrower brings books to check out desk.			
Type: External			
Relationships: Association: Borrower, Personnel Office, Registrar's Office Include: Extend: Generalization:			
Normal Flow of Events: 1. The Borrower brings books to the Librarian at the check out desk. 2. The Borrower provides Librarian their ID card. 3. The Librarian checks the validity of the ID Card. If the Borrower is a Student Borrower, Validate ID Card against Registrar's Database. If the Borrower is a Faculty/Staff Borrower, Validate ID Card against Personnel Database. If the Borrower is a Guest Borrower, Validate ID Card against Library's Guest Database. 4. The Librarian checks whether the Borrower has any overdue books and/or fines. 5. The Borrower checks out the books.			
SubFlows:			
Alternate/Exceptional Flows: 4a. The ID Card is invalid, the book request is rejected. 5a. The Borrower either has overdue books, fines, or both, the book request is rejected.			



# Text and Diagrams

---

- ▶ Use case *text* provides the detailed description of a particular use case
- ▶ Use case *diagram* provides an overview of interactions between actors and use cases

## Example: Point of Sale – Problem Description

---

- ❑ The Point-of-Sale terminal is a computerized system used to record sales and handle payments; it is typically used in a retail store. It includes hardware components such as a computer and bar code scanner, and software to run the system.
- ❑ It interfaces to various service applications, such as a third-party tax calculator and inventory control. These systems must be relatively fault-tolerant; that is, even if remote services are temporarily unavailable (such as the inventory system), they must still be capable of capturing sales and handling at least cash payments (so that the business is not crippled).

## Example: Point of Sale – Problem Description

---

- ❑ A POS system increasingly must support multiple and varied client-side terminals and interfaces. These include a thin-client Web browser terminal, a regular personal computer with something like a Java Swing graphical user interface, touch screen input, wireless PDAs, and so forth.
- ❑ Furthermore, we are creating a commercial POS system that we will sell to different clients with disparate needs in terms of business rule processing. Each client will desire a unique set of logic to execute at certain predictable points in scenarios of using the system, such as when a new sale is initiated or when a new line item is added.

# Example: Point of Sale – Problem Description

---

- Therefore, we will need a mechanism to provide this flexibility and customization. Using an iterative development strategy, we are going to proceed through requirements, object-oriented analysis, design, and implementation.

# Example: Point of Sale - Actors

---

- Actors:
  - Cashier
  - Customer
  - Supervisor
- Choosing actors:
  - Identify system boundary
  - Identify entities, human or otherwise, that will interact with the system, from outside the boundary.

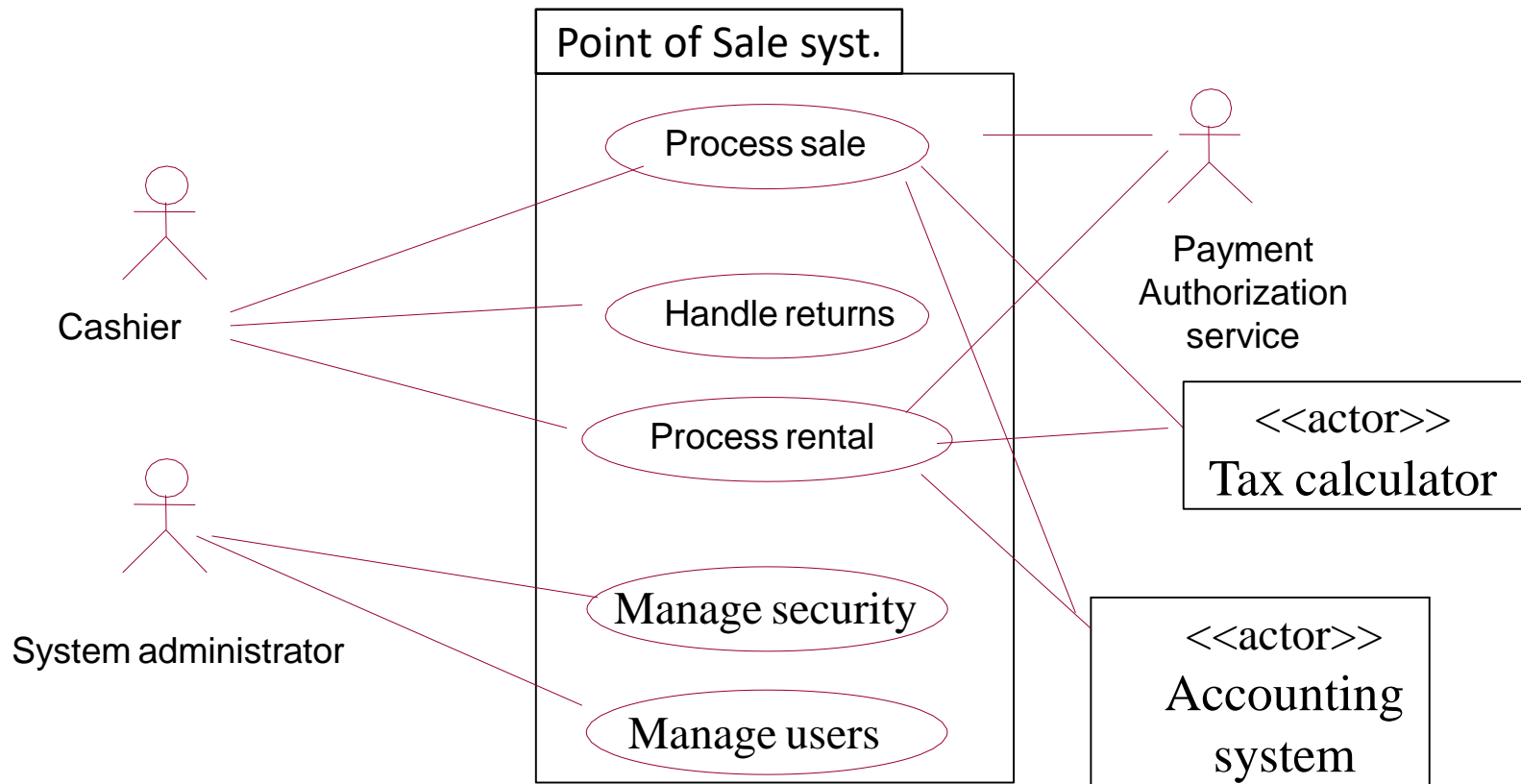
# Example: Point of Sale – Actor-Goal List

## Actor-Goal List

Sales activity system is a remote application that will frequently request sales data from each POS node on the network

Actor	Goal
Cashier	<ul style="list-style-type: none"><li>•Process sales</li><li>•Process returns</li><li>•Cash in</li><li>•Cash out</li></ul>
Manager	<ul style="list-style-type: none"><li>•Start up</li><li>•Shut down</li></ul>
System administrator	<ul style="list-style-type: none"><li>•Add users</li><li>•Modify users</li><li>•Delete users</li><li>•Manage security</li></ul>
Sales activity system (external computer sys)	<ul style="list-style-type: none"><li>•Analyse sales &amp; performance data</li></ul>

# Example: Point of Sale – Use Case Diagram



**Use Case Diagram:** illustrates a set of use cases for a system.

# Exercise: movie ticket machine

- Problem Description

Implement a simple movie ticket vending machine. The movie theater that will use the machine has only one movie and one show time each day. Every morning, the theater manager will turn on the ticket machine, and it will ask him for the name of the movie and the ticket price that day. It will also ask how many seats are in the theater (so it won't sell too many tickets).

When a customer walks up to the ticket machine, he will see the name of the movie, the time, and the ticket price displayed. There is a slot to insert money, a keypad of buttons to enter a number into the "Number of Tickets" field, and a "Buy" button.

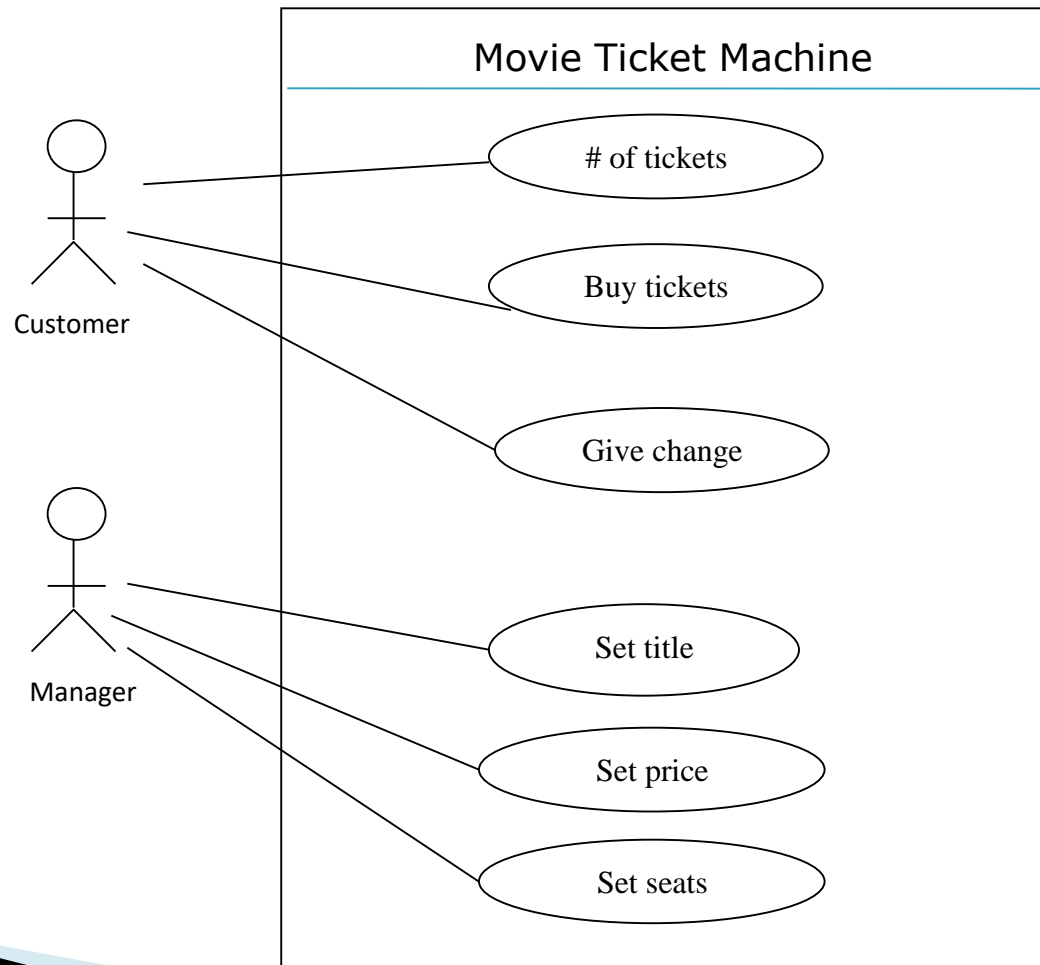
Printed tickets come out of a slot at the bottom of the machine. Above the ticket slot is a message display (for error messages like "Please enter more money or request fewer tickets" or "SOLD OUT!"). An additional display shows the customer's balance inside the machine.

Finally, there is a "Return Change" button so the customer can get his unspent money back.

- Who or what are the actors?
- What are the use cases (goals of actors)?



# Use case diagram for Movie Ticket Machine



# Identification of Use Cases

## Use cases for Manager

- Use case: Set title

- Actors: Manager, Machine
- 1. Manager requests a change of movie title
- 2. Machine asks manager for new movie title
- 3. Manager enters movie title

- Use case: Set price

- Actors: Manager, Machine
- 1. Manager requests a change of ticket price
- 2. Machine asks manager for new price for movie title
- 3. Manager enters ticket price
- Alternatives: Invalid price
- If manager enters price below SR5 or greater than SR50
- 3a. Machine asks manager to reenter price

- Use case: Set seats

- Actors: Manager, Machine
- 1. Manager requests a change in number of seats
- 2. Machine asks manager for number of seats in theatre
- 3. Manager enters number of seats
- Alternatives: Invalid number of seats
- If manager enters number less than 20 or greater than 999
- 3a. Machine asks manager to reenter number of seats

# Identification of Use Cases

## Use cases for Customer

- Use case: # of tickets

- Actors: Customer, Machine
- 1. Customer enters number of tickets
- 2. Machine displays total balance due
- Alternative: Customer wants zero tickets
- At step 1, customer enters zero tickets
- 1a. Display thank you message
- 1b. Set balance to \$0.0

- Use case: Return change to customer

- Actors: Customer, Machine
- 1. Customer requests change
- 2. Machine dispenses money
- 3. Machine updates customer balance

- Use case: Buy tickets

- Actors: Customer, Machine
- 1. Customer requests tickets
- 2. Machine tells customer to put balance due in money slot
- 3. Customer enters money in money slot
- 4. Machine updates customer balance
- 5. Customer requests tickets
- 6. Machine prints tickets
- 7. Machine updates number of seats
- Alternative: Insufficient seats
- At step 1, if number of tickets requested is less than available seats,
- 1a. Display message and end use case
- Alternative: Insufficient funds
- At step 5, if money entered < total cost,
- 5a. Display insufficient amount entered
- 5b. Go to step 3