Object Oriented Analysis and Design with Java

UE19CS353

Prof. Mahitha G, Prof. Phalachandra and Prof. Sindhu R Pai
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

UE19CS353: Object Oriented Analysis and Design with Java

Use case Modelling: Use case Diagrams

Prof. Phalachandra H L and Prof. Sindhu R Pai

Department of Computer Science and Engineering

Use case Modelling - Agenda

- Introduction to Use case Modelling
- Use case diagrams - Use case and Actor
- Example use case diagrams
- Relation between these use case and Actor
- Relation between the use cases
- Identify the relationship?
- Use case specification/description
- Practice: ATM
- Few inputs

Use case Modelling: Introduction

- Describes the **interaction of users and the system**
- Describes **what functionality does a system provides to its users**.
- Use case model has **two important elements** - **actors** and **use cases**.

Actor/s: One or set of objects who directly interacts with the system

Every actor has a defined purpose while interacting with the system.

An actor can be a person, device or another system.

Use case: A piece of functionality that a system offers to its users.

Set of all use cases defines the entire functionality of the system.

Also define the error conditions that may occur while interacting with the system

Use case Diagrams

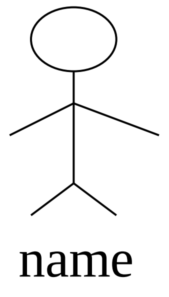
- Incorporates both **actor and use cases and also the relationship** between them in the **graphical representation**.
- Used to visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis.
- Provide a way for developers, domain experts and end-users to Communicate.
- Serve as basis for testing

Use case

- Use cases **specify the desired behavior**.
- A use case is
  - a description of a set of sequences of actions a system performs to yield an observable result of value to an actor.
  - includes variants
- Name starts with a verb.
- Each sequence represent an interaction of actors with the system.

Actor

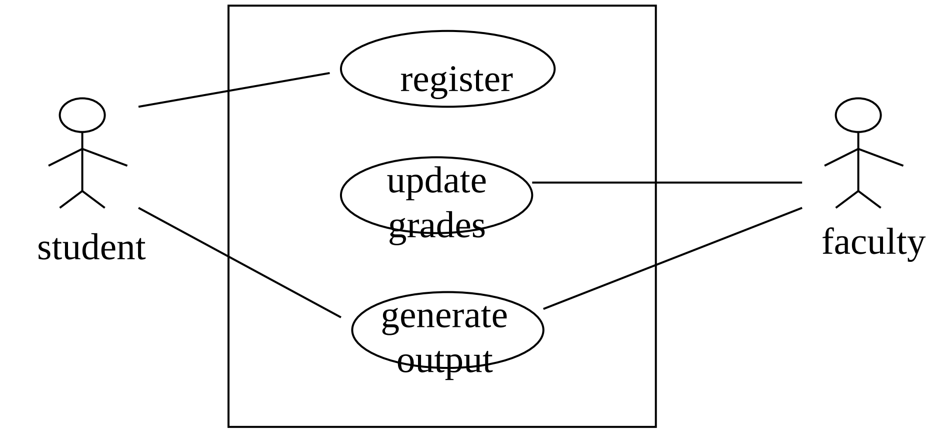
- Represents a set of roles that users of use case play when interacting with these use cases.
- Can **be human or automated systems.**
- Actor is someone interacting with use case (system function). **Named by noun.**
- Actors are entities which require help from the system to perform their task or are needed to execute the system's functions.
- Actors are not part of the system.
- An **Actor triggers use case**
- Actor has responsibility toward the system (inputs), and have expectations from the system (outputs).

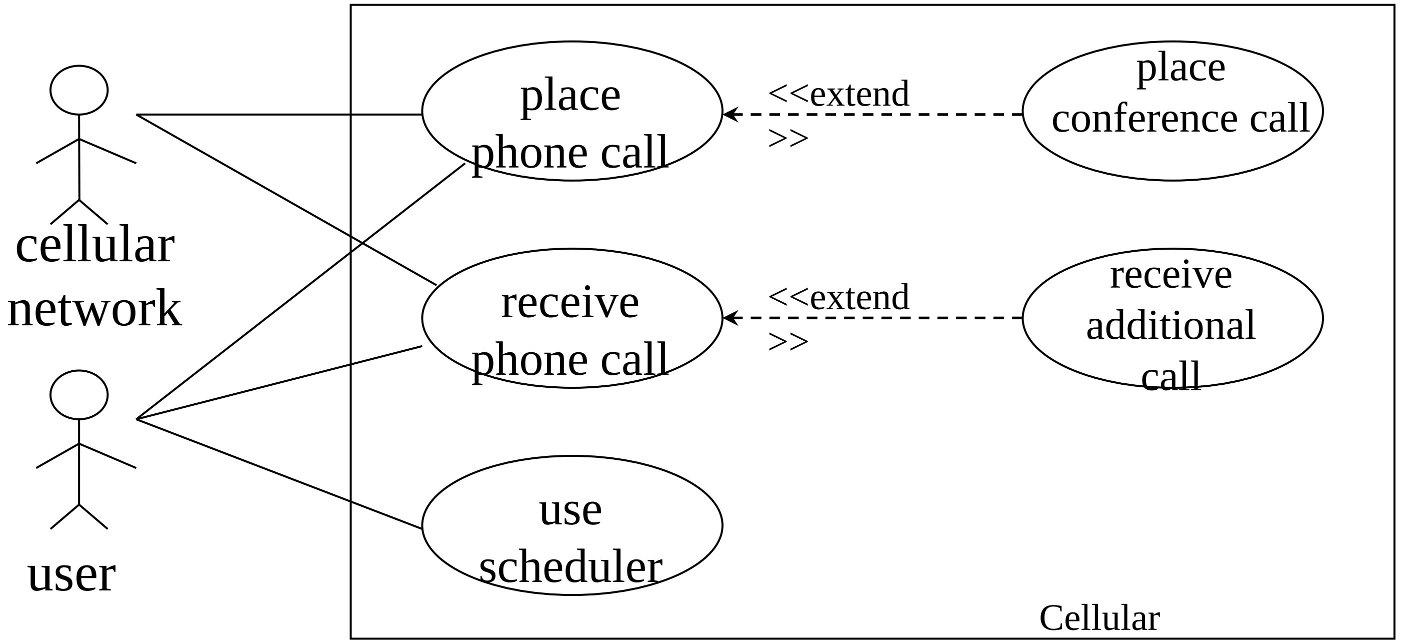


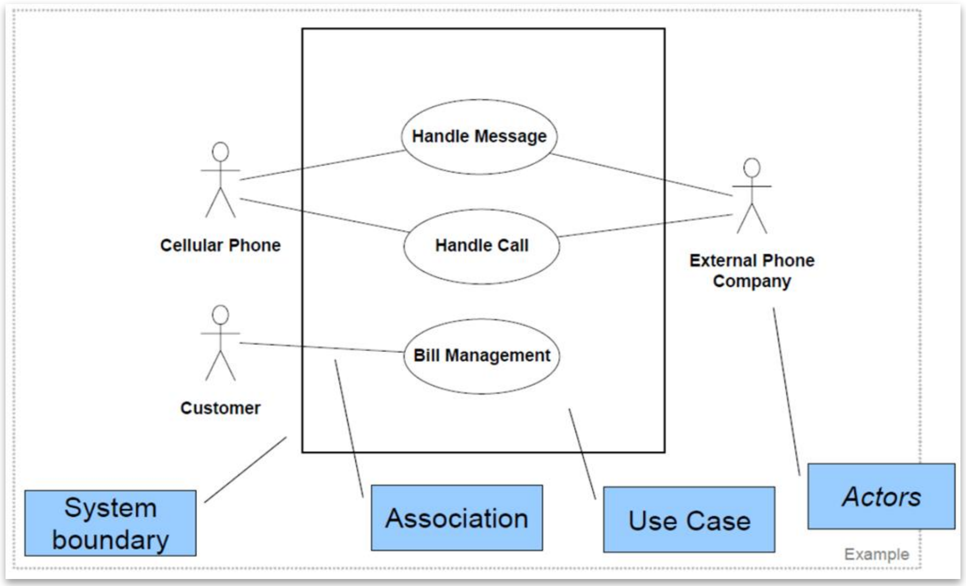
How to create Use case diagrams?

- List main system functions (use cases) in a column
- Draw ovals around the function labels
- Draw system boundary
- Draw actors and connect them with use cases
- Specify include and extend relationships between use cases

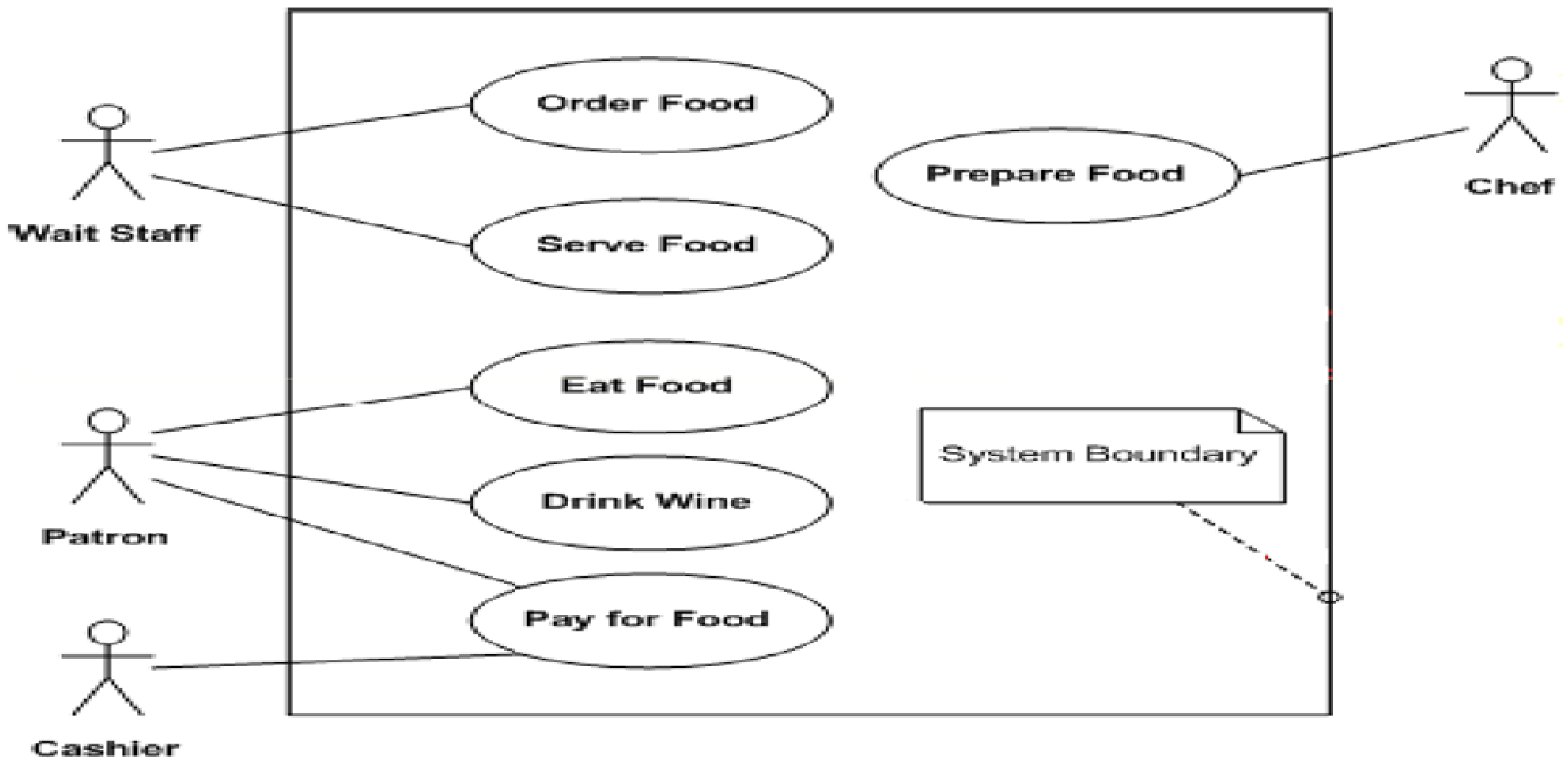
Example use case diagram





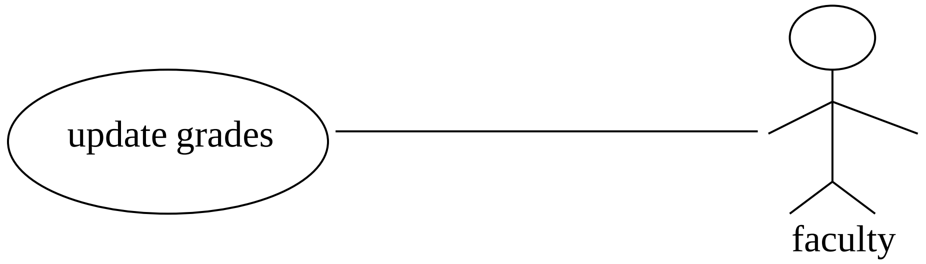


Example use case diagram for a restaurant



Relationships between Use Cases and Actors

- Actors may be connected to use cases by **associations**, indicating that the actor and the use case communicate with one another using messages.



Relationship between use cases

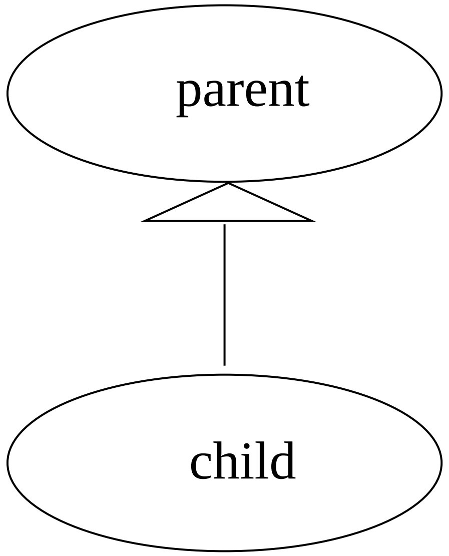
1. **Generalization** - Use cases that are specialized versions of other use cases.

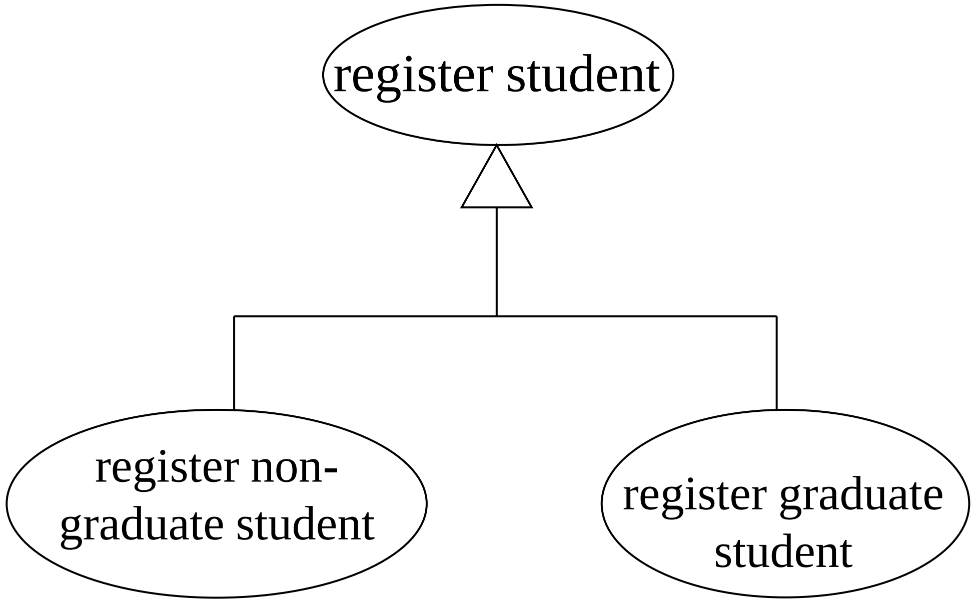
2. **Include** - Use cases that are included as parts of other use cases. Enable to factor common behavior.

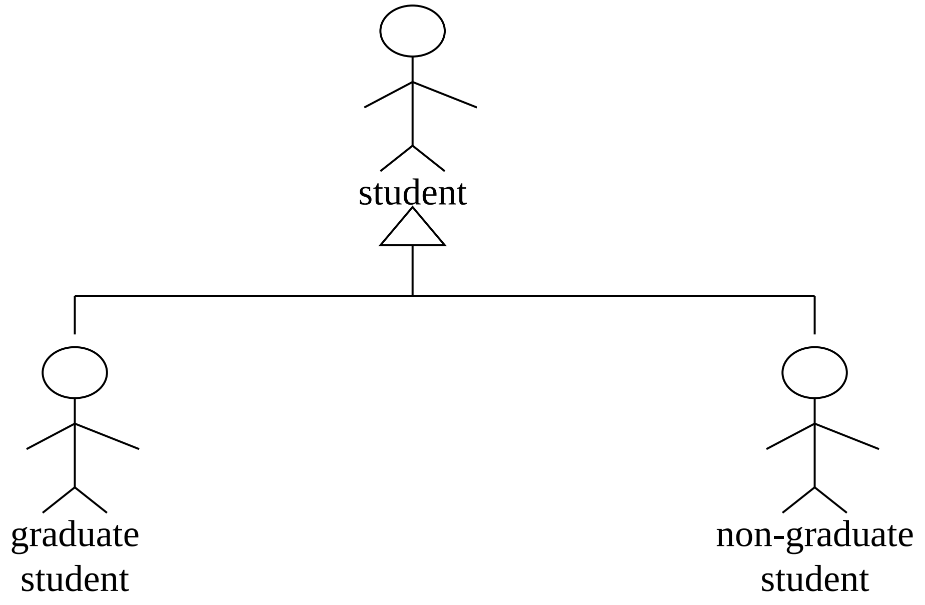
3. **Extend** - Use cases that extend the behavior of other core use cases. Enable to factor variants.

Generalization

- The child use case inherits the behavior and meaning of the parent use case.
- The child may add to or override the behavior of its parent.
- Share the same relationship to the actor



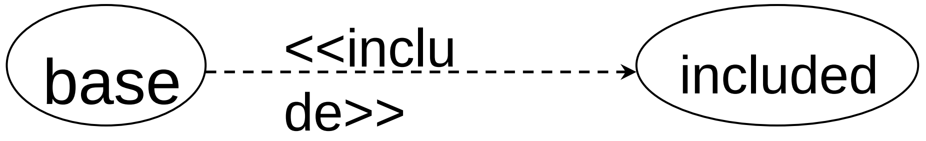




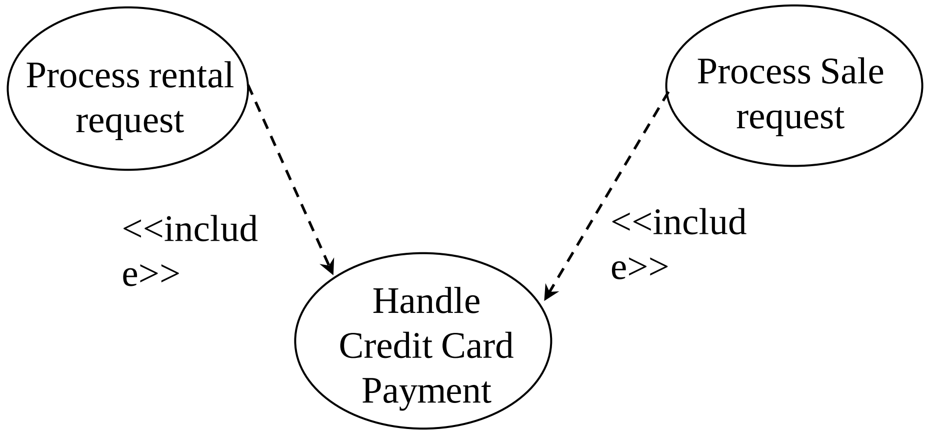
Generalization as a Relationships between Actors

Include

- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.

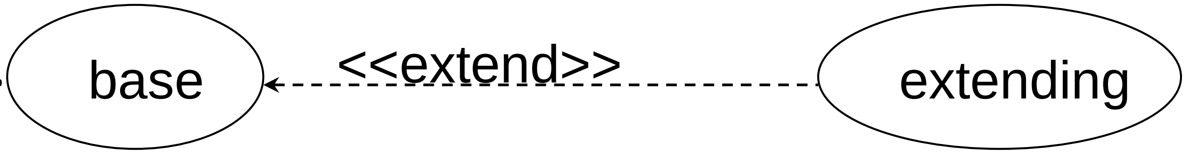


- Enables to avoid describing the same flow of events several times by putting the common behavior in a use case of its own

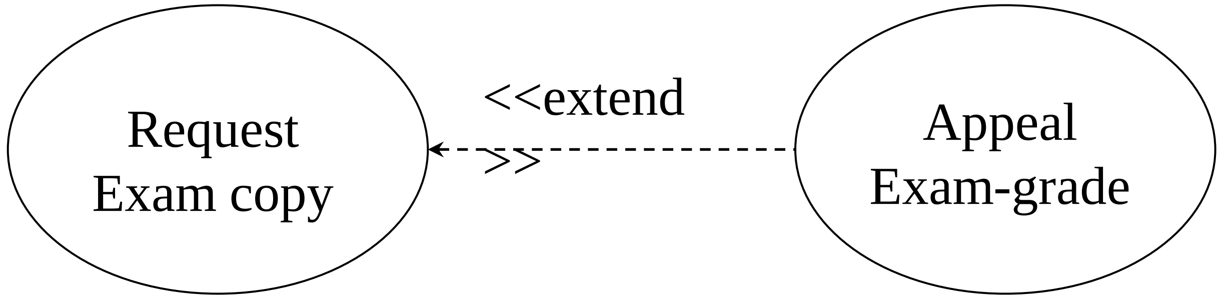


Extend

- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- **Adds incremental behavior to use case**
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.



- Enables to model optional behavior or branching under conditions.



Identify the relationship?????

- Create new order -> validate customer account
- Update order -> validate customer account
- Place order-> login
- Login account -> change password
- Withdraw funds -> update balance
- Choose folder -> upload document
- Book hotel package with flight -> book flight only
- Purchase item, purchase phone, purchase accessories
- Order food -> order wine
- ATM Transaction -> pin validation
- Make purchase -> view items
- Book ticket-> make payment
- Make payment-> update DB
- Process sale->handle gift voucher
- Process sale -> handle credit payment

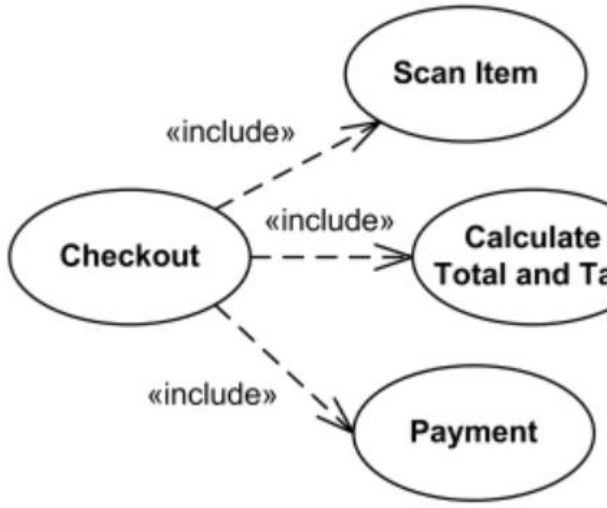
Identify the relationship?????

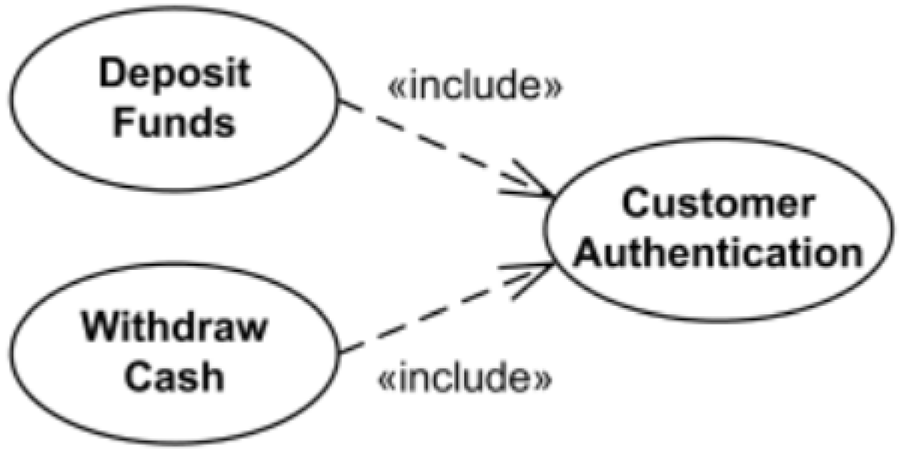
- Create new order -> validate customer account
- Update order -> validate customer account
- Place order-> login
- Login account -> change password
- Withdraw funds -> update balance
- Choose folder -> upload document
- Book hotel package with flight -> book flight
- Purchase item, purchase phone, purchase accessories
- Order food -> order wine
- ATM Transaction -> pin validation
- Make purchase -> view items
- Book ticket-> make payment thru credit card
- Make payment-> update DB
- Process sale->handle gift voucher
- Process sale -> handle credit payment

Red->include
Blue->extend
Brown – generalization

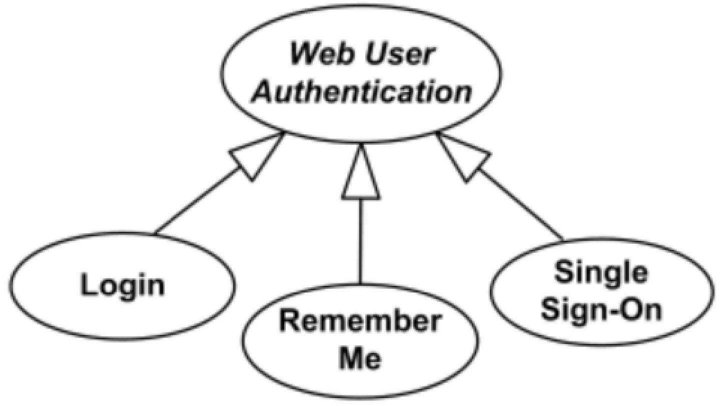
Identify the relationship?????

- Registration -> get help on registration
- Check out -> scan item -> calc total and tax -> payment
- Bank ATM transaction-> cust authentication





Deposit Funds *and* ***Withdraw Cash*** *use cases include* ***Customer Authentication*** *use case.*



Web User Authentication use case is ***abstract use case*** *specialized by Login, Remember Me and Single Sign-On use cases.*

Use case specification/Description

- Name (Must start with a verb)
- Summary
- Actors
- Pre-conditions
  - Conditions that must exist *before* the use case is executed
- Description
  - Textual description (may include steps to execute) and typically is the primary functionality
- Exceptions
  - These are paths which will need to handle exceptions which could be all to provide handling of things which are not provide you with a primary functionality including things like power failure
- Alternate Flows
  - Handles the other functionality paths for the summary these could be some in the exceptions too
- Post-conditions
  - Conditions that must exist *after* the use case is executed

Sample use case specification

- Name: Transfer Funds
- Summary/Overview : Transfer funds from one account to another
- Actor: Customer
- Pre-conditions: Source account must have sufficient funds
- Description:
  a. Customer identifies the accounts from which and to which funds have to be transferred
  b. Enters the amount to be transferred
  c. Confirm the transaction
- Exceptions
  - Cancel, Insufficient funds, Cannot identify destination account
  - Needs to handle ..say power failure, slow network
- Alternate Flows
  - Handles all the alternate paths (Accounts belonging to the same customer)
- Post-conditions: Funds transferred and account balances updated

Practice: ATM

- Actors: Customer
- Pre Condition:
    - The ATM must be in a state ready to accept transactions
    - The ATM must have at least some cash on hand that it can dispense
    - The ATM must have enough paper to print a receipt for at least one transaction
- Post Condition:
    - The current amount of cash in the user account is the amount before the withdraw minus the withdraw amount
    - A receipt was printed on the withdraw amount
    - The withdraw transaction was audit in the System log file

ATM : System and user Actions

Actor Actions	System Actions
1. Begins when a Customer arrives at ATM	
2. Customer inserts a Credit card into ATM	3. System verifies the customer ID and status
5. Customer chooses "Withdraw" operation	4. System asks for an operation type
7. Customer enters the cash amount	6. System asks for the withdraw amount
	8. System checks if withdraw amount is legal
	9. System dispenses the cash
	10. System deduces the withdraw amount from account
	11. System prints a receipt
13. Customer takes the cash and the receipt	12. System ejects the cash card

Alternative flow of events:

Step 3: Customer authorization failed. Display an error message, cancel the transaction and eject the card.

Step 8: Customer has insufficient funds in its account. Display an error message, and go to step 6.

Step 8: Customer exceeds its legal amount. Display an error message, and go to step 6.

Exceptional flow of events:

Power failure in the process of the transaction before step 9, cancel the transaction and eject the card

Few Inputs

- **One method to identify use cases is actor-based:**
  - Identify the actors related to a system or organization.
  - For each actor, identify the processes they initiate or participate in.
- **A second method to identify use cases is event-based:**
  - Identify the external events that a system must respond to.
  - Relate the events to actors and use cases.
- **The following questions may be used to help identify the use cases for a system:**
  - What are tasks of each actor ?
  - Will any actor create, store, change, remove, or read information in the system ?
  - What use cases will create, store, change, remove, or read this information ?
  - Will any actor need to inform the system about sudden, external changes ?
  - Does any actor need to be informed about certain occurrences in the system ?
  - Can all functional requirements be performed by the use cases ?

THANK YOU

Prof. Phalachandra H L and Prof. Sindhu R Pai

Department of Computer Science and Engineering

phalachandra@pes.edu

sindhurpai@pes.edu

+91 8277606459