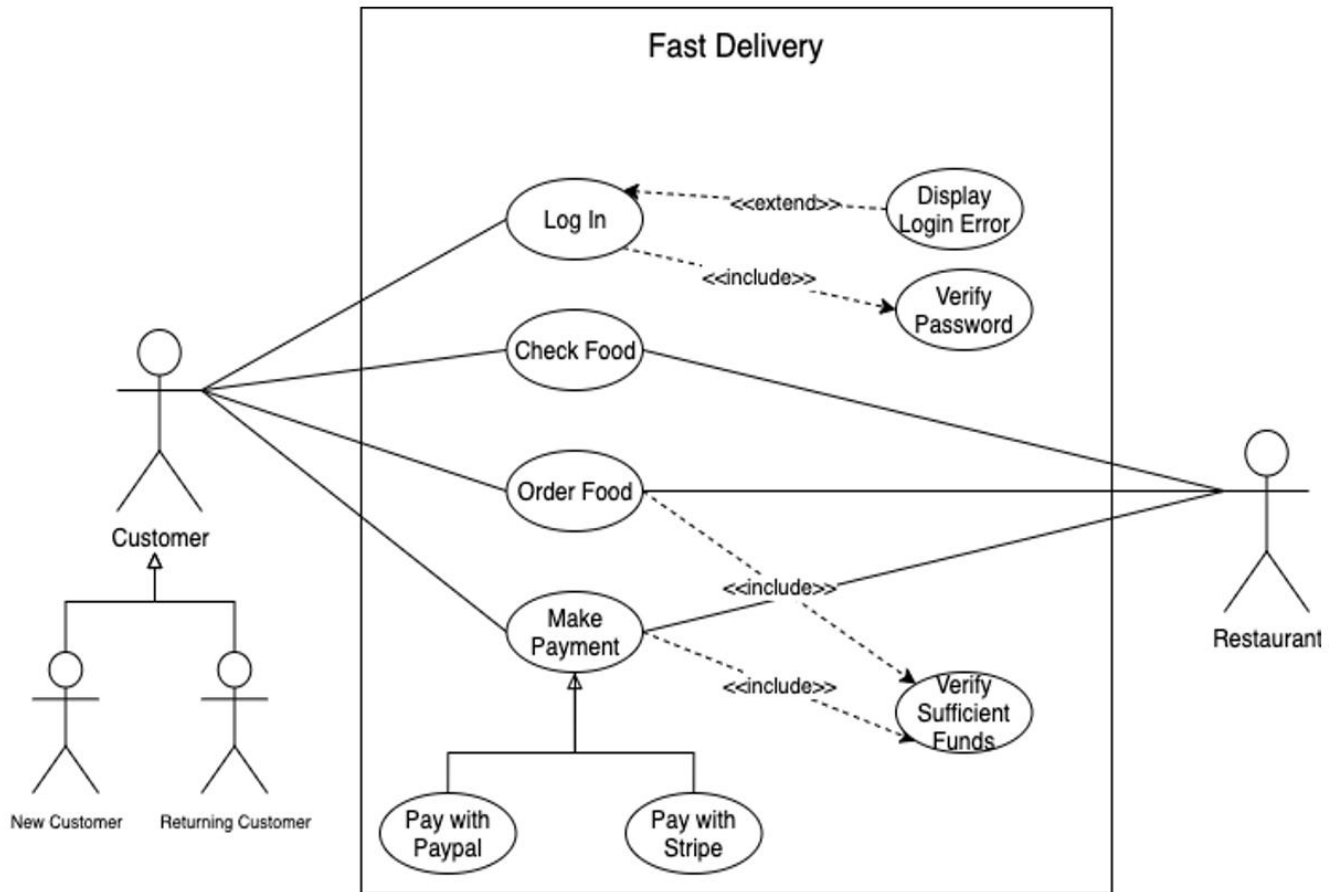


Diagrams

Use Case Diagrams

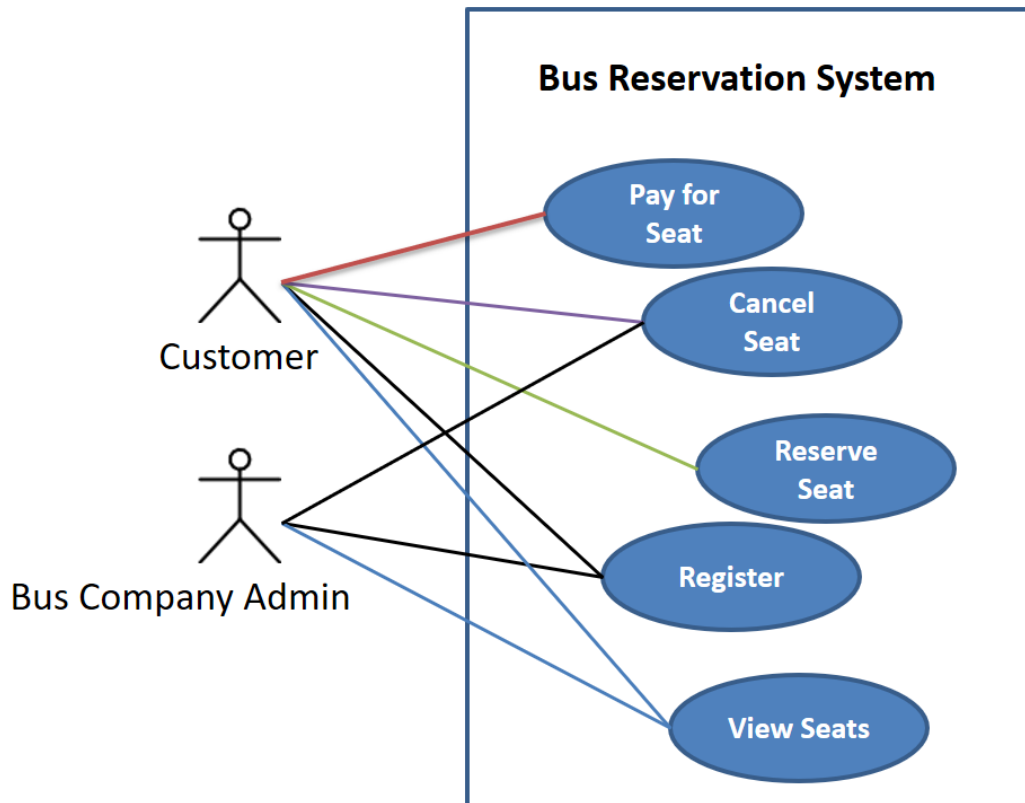


Actor Goals?

Customer Reserves the Seat
Customer Pays for the Seat
Customer & Admin Cancels the Seat
Customer & Admin must Register
Customer & Admin can View the Seat

Complete the Use Case Diagram

Consider an online **reservation system** for a **bus company**. The bus company includes several buses and realizes trips to different cities. Each bus is identified by its plate number and a separately assigned bus number. The trips are based on a predefined schedule and stop at predefined bus stations. Each bus can have only one trip per day. Each bus includes a driver and one hostess. For long trips, the bus will have breaks at service and rest areas. There are two types of trips, normal trips and express trips. Express trips do not stop at intermediate stations and get faster at the destination. Seats can be reserved by customers on the web site of the bus company. The customer has the option to directly pay for the seat through the website. In that case, the seat cannot be cancelled (neither by the customer nor by the bus company). If the customer has not paid for the seat, the bus company can cancel the seat if the customer does not show up one hour before the trip. When the reservation is cancelled, the seat will become free and can be sold to another customer. Both the customer and the company staff must authenticate themselves for performing operations with the system.



Use Case Description Template

Identifier	Give a unique Identifier for the Use Case		
Name	Write the name of use case		
Purpose	Brief Description of the process (what is happening?) in the use case		
Priority	High/Medium/Low		
Actors	List all actors (people, systems etc.) associated with this requirement		
Pre-conditions	What must occur before the use case begins		
Post-conditions	What has occurred as a result of the use case		
Dependencies	List identifiers of use cases on which this use case is dependent		
Typical Course of Action (Primary Flow)			
S#	Actor Action	S#	System Response
1	List the actor steps here	1.1	List the system steps here
2		2.1	
Alternative Course of Action			
S#	Actor Action	S#	System Response
1	List the actor steps here	1.1	List the system steps here
Exceptions: List the sequence of actions that prevents getting to the post condition			

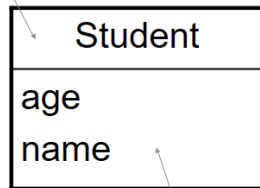
Identifier	UC-01		
Name	Search & Invite a contractor		
Purpose	This use case describes the procedure to invite a contractor while creating a contract.		
Priority	High		
Actors	User		
Pre-conditions	1 - Internet Connection should be good 2 - The User Account should exist. 3 - The User Account should be verified. 4 - The User has already Signed-up.		
Post-conditions	An invitation will be sent to the requested contractor.		
Dependencies	None		
Typical Course of Action (Primary Flow)			
S#	Actor Action	S#	System Response
1	Enter username/name/email of the contractor for invitation.	1.1	Display a list of matched contractors.
2	Select one of the contractors from the list.	2.1	Display selected contractor
3	Click "Send".	3.1	Send a "Contract Invitation" email to the selected contractor.
Alternative Course of Action			
S#	Actor Action	S#	System Response
1	Enter username/name/email of the contractor for invitation.	1.1	Display "No user found" error.
2	Click "Back" Button	2.1	Display Previous page
Exceptions: User presses "Cancel" button while records were being searched. "An error occurred while searching" message is displayed			

Class Diagrams

Syntax of Domain Model

Class Name

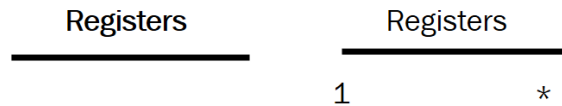
Conceptual Class:



Attributes

Association

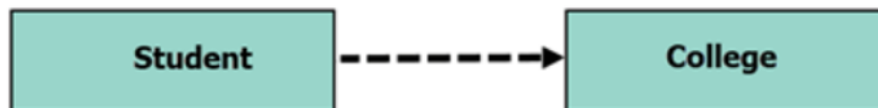
Multiplicity/Cardinality



■ There are mainly three kinds of relationships in UML:

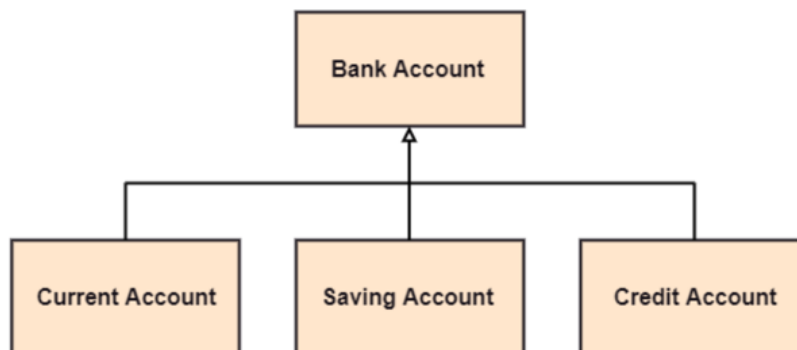
1. Dependency
2. Generalization
3. Association

- A dependency means the relation between two or more classes in which a change in one may force changes in the other. However, it will always create a weaker relationship. Dependency indicates that one class depends on another.
- In the following UML class diagram examples, Student has a dependency on College



Dependent Class

A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class. For example, The Current Account, Saving Account, and Credit Account are the generalized form of Bank Account.



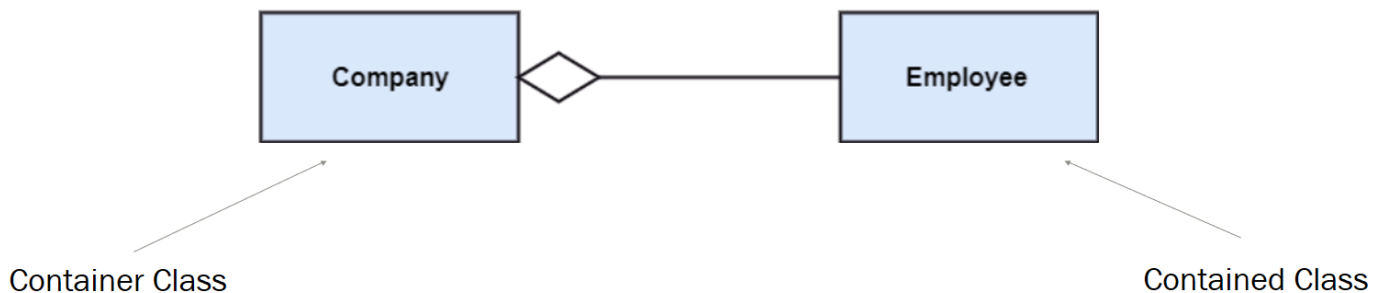
Multiplicity

*	T	zero or more; "many"
1..*	T	one or more
1..40	T	one to 40
5	T	exactly 5
3, 5, 8	T	exactly 3, 5, or 8

A multiplicity is a factor associated with an attribute. It specifies how many instances of attributes are created when a class is initialized. If a multiplicity is not specified, by default one is considered as a default multiplicity.

Aggregation

- An aggregation is a **subset of association**, which represents has a relationship. It is more specific than association. It defines a **part-whole** or **part-of relationship**. In this kind of relationship, the child class can exist independently of its parent class.



Composition

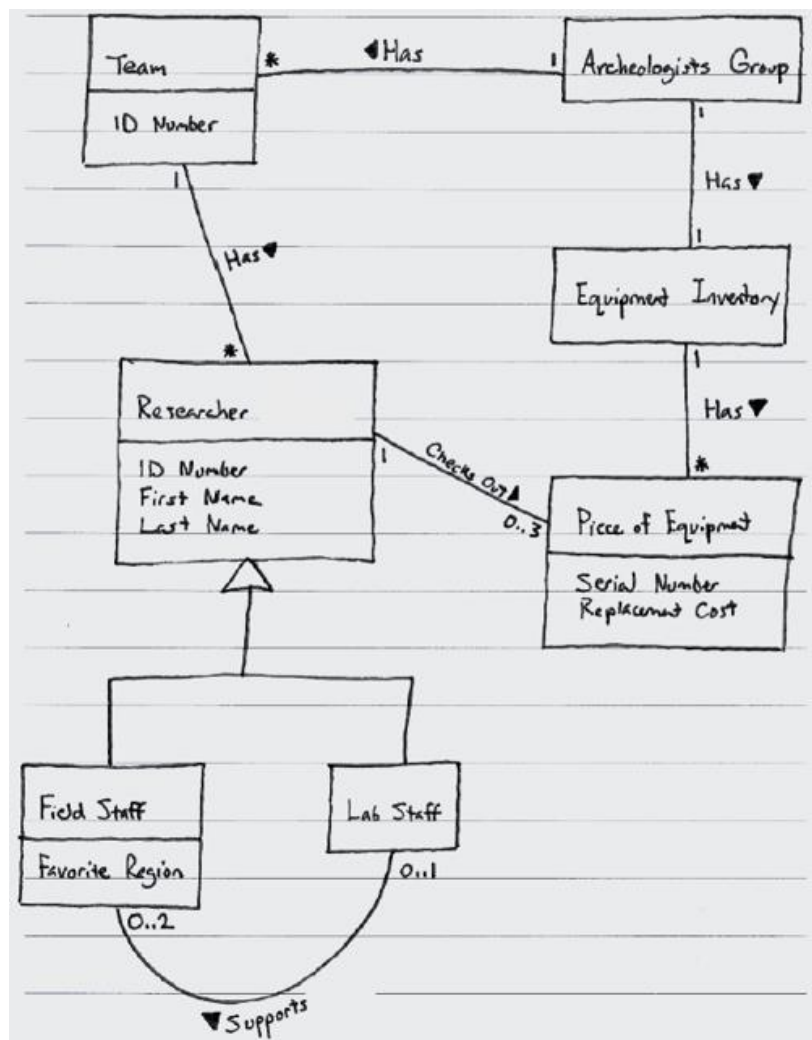
- The composition is a **subset of aggregation**. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.
- A contact book consists of multiple contacts, and if you delete the contact book, all the contacts will be lost.



Archaeologist Management System

Identify Associations and Multiplicity

You have been asked to build a management system for a group of archeologists. The group is comprised of multiple teams of researchers. Each team has a letter ID (e.g., team A, team B). Each researcher belongs to one of the teams, and has an ID number, a first name, and a last name. There are two types of researchers: field staff and lab staff. Each field staff member has a favorite region (string). Each lab researcher supports up to 2 field researchers. Some researchers may not be supported by a lab researcher. The archaeologist group also manages an inventory of equipment. Researchers of any type may check out up to 3 pieces of equipment. Each piece of equipment has a serial number and replacement cost.

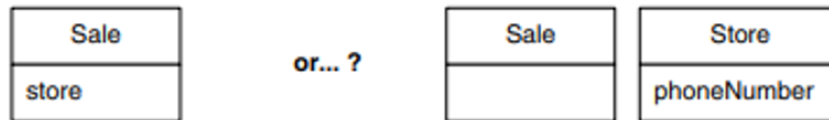


A Common Mistake in Identifying Concepts:

Attribute OR Concept?

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

As an example, should *store* be an attribute of *Sale*, or a separate conceptual class *Store*?



If in doubt, make it a separate concept.

During a semester, a lecturer reads one or more lectures

Sometimes the lecturer is on leave to focus on doing research, in this case (s)he does not give a lecture

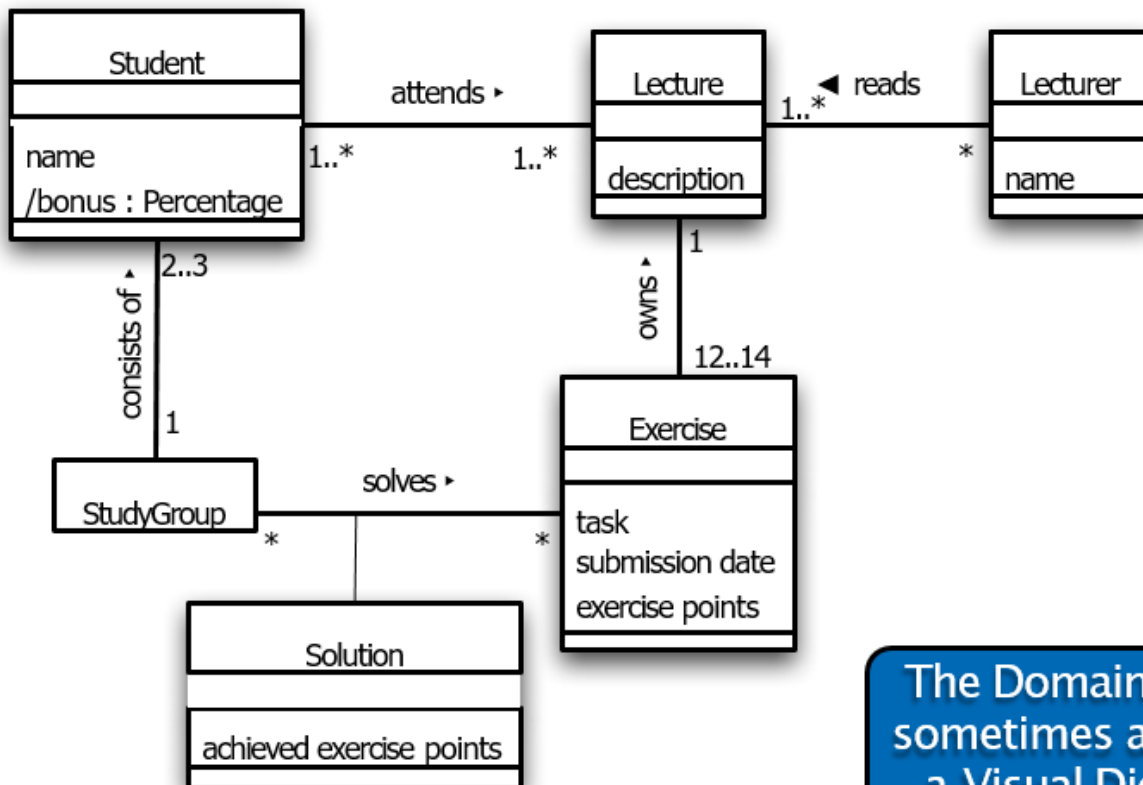
A student usually attends one or more lectures, unless (s)he has something better to do

During the semester there will be several exercises which are meant to be solved by small study groups

Each student is assigned to one particular study group for the whole semester

A study group consists of two to three students

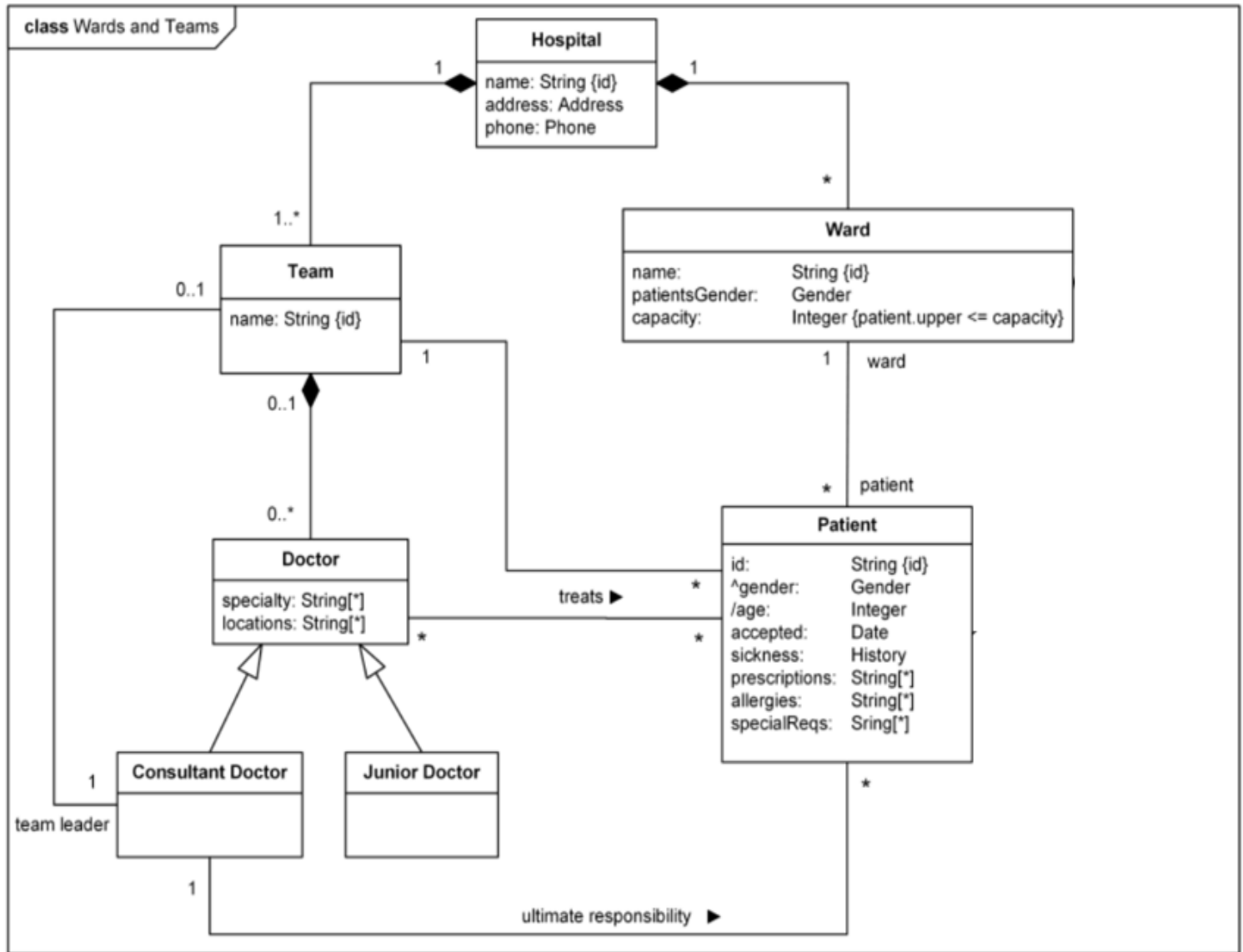
After submission of a solution by a study group it is graded by a tutor



The Domain Model is sometimes also called a Visual Dictionary.

Hospital Management System

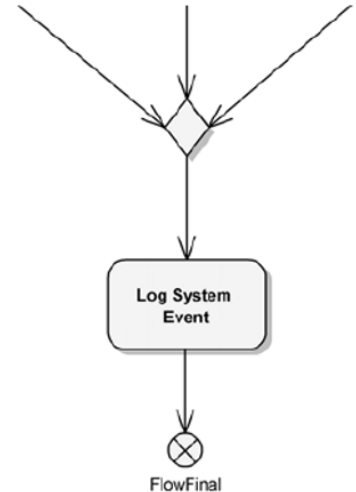
- **Ward** is a division of a **hospital** or a suite of **rooms** shared by **patients** who need a similar kind of care. In a hospital, there are a number of wards, each of which may be empty or have on it one or more patients. Each ward has a **unique name**. Wards are differentiated by **gender** of its patients, i.e. male wards and female wards. A ward can only have patients of the gender admitted to it. Every ward has a fixed **capacity**, which is the maximum number of patients that can be on it at one time (i.e. the capacity is the number of beds in the ward).
- Different wards may have different capacities. The **doctors** in the hospital are organized into **teams** (also called firms). Each team has a unique **name or code** (e.g. Orthopedics or Pediatrics) and is headed by a consultant doctor (in the UK, Republic of Ireland, and parts of the Commonwealth) or attending physician (also known as staff physician) (in the United States).
- **Consultant doctor** or attending physician is the senior doctor who has completed all of his or her specialist training, residency and practices medicine in a clinic or hospital, in the specialty learned during residency. She or he can supervise fellows, residents, and medical students.
- The rest of the team are all **junior doctors**. Each doctor could be a member of no more than one team. Each patient is on a single ward and is under the care of a single team of doctors. A patient may be treated by any number of doctors but they must all be in the team that cares for the patient.
- A doctor can treat any number of patients. The team leader accepts ultimate responsibility, legally and otherwise, for the care of all the patients referred to him/her, even with many of the minute-to-minute decisions being made by subordinates.



Activity Diagrams

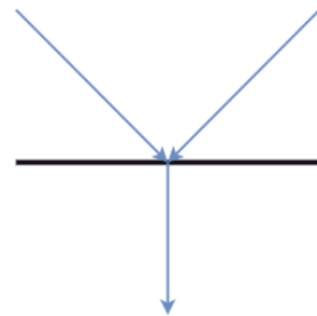
Merge Nodes (Control Nodes)

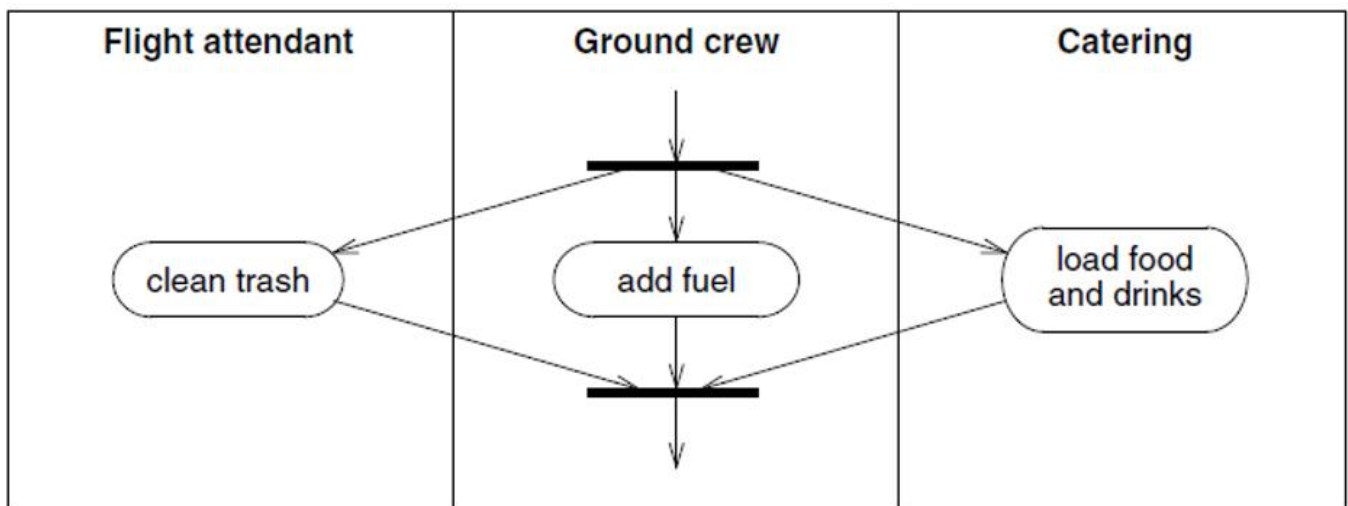
- Merge nodes **take multiple input flows and direct any and all of them to one outgoing flow**.
- There is **no waiting or synchronization** at a merge node.
- Whenever any of the three incoming flows reach the merge point (shown as a diamond), each will be routed through it to the `Log System Event` action. Thus, multiple events will be logged.



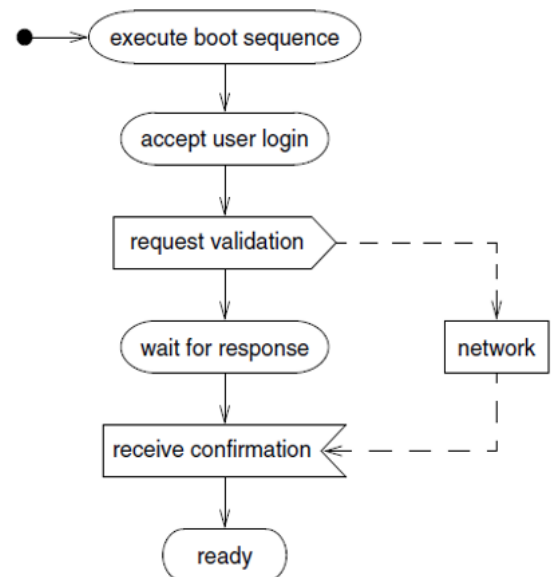
Joins (Control Nodes)

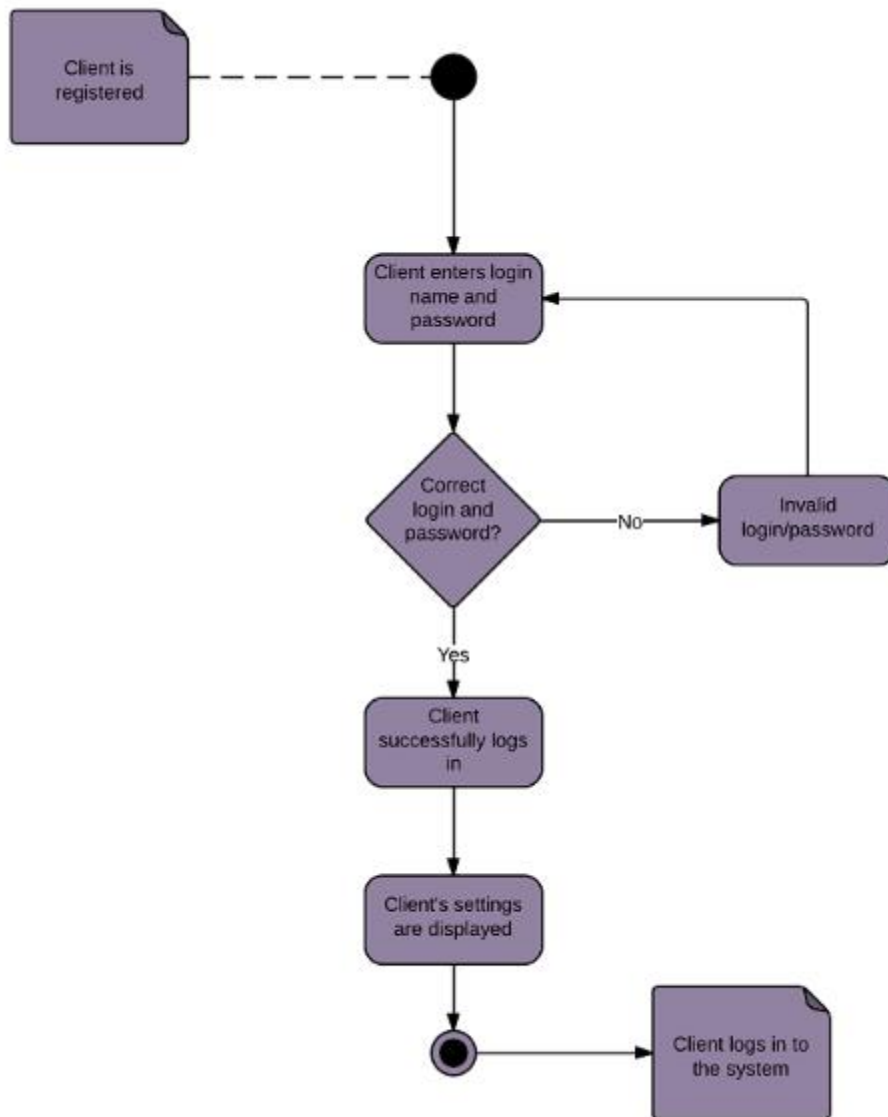
- A join has **multiple incoming flows and a single outbound flow**, like merge nodes.
- With a join, all the **incoming flows must be completed before the outbound flow commences**.



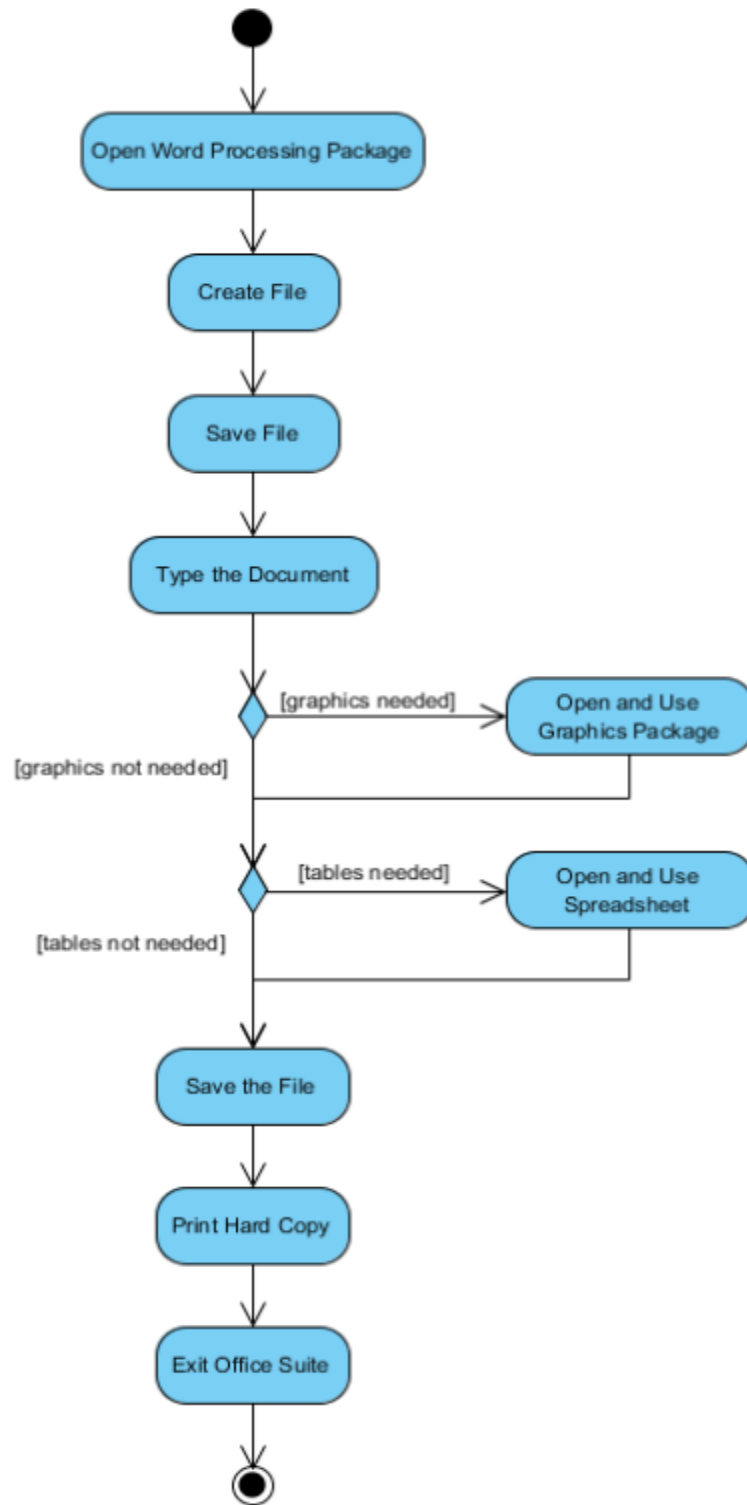


- Consider a workstation that is turned on.
- It goes through a boot sequence and then requests that the user log in.
- After entry of a name and password, the workstation queries the network to validate the user.
- Upon validation, the workstation then finishes its startup process.

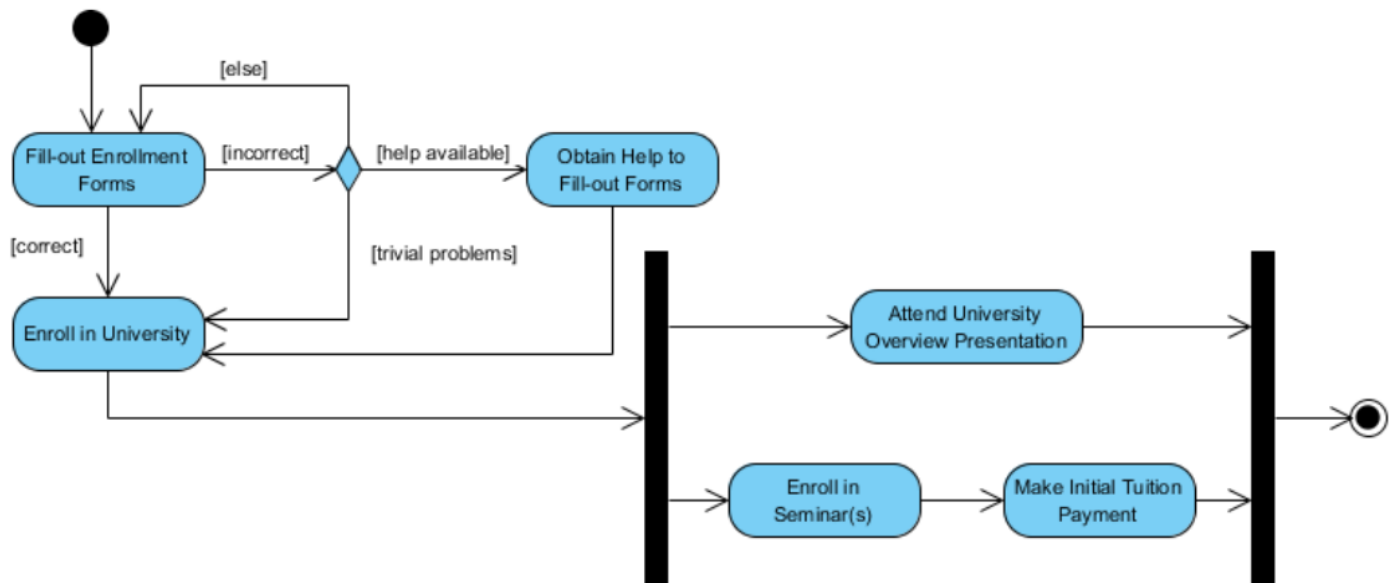


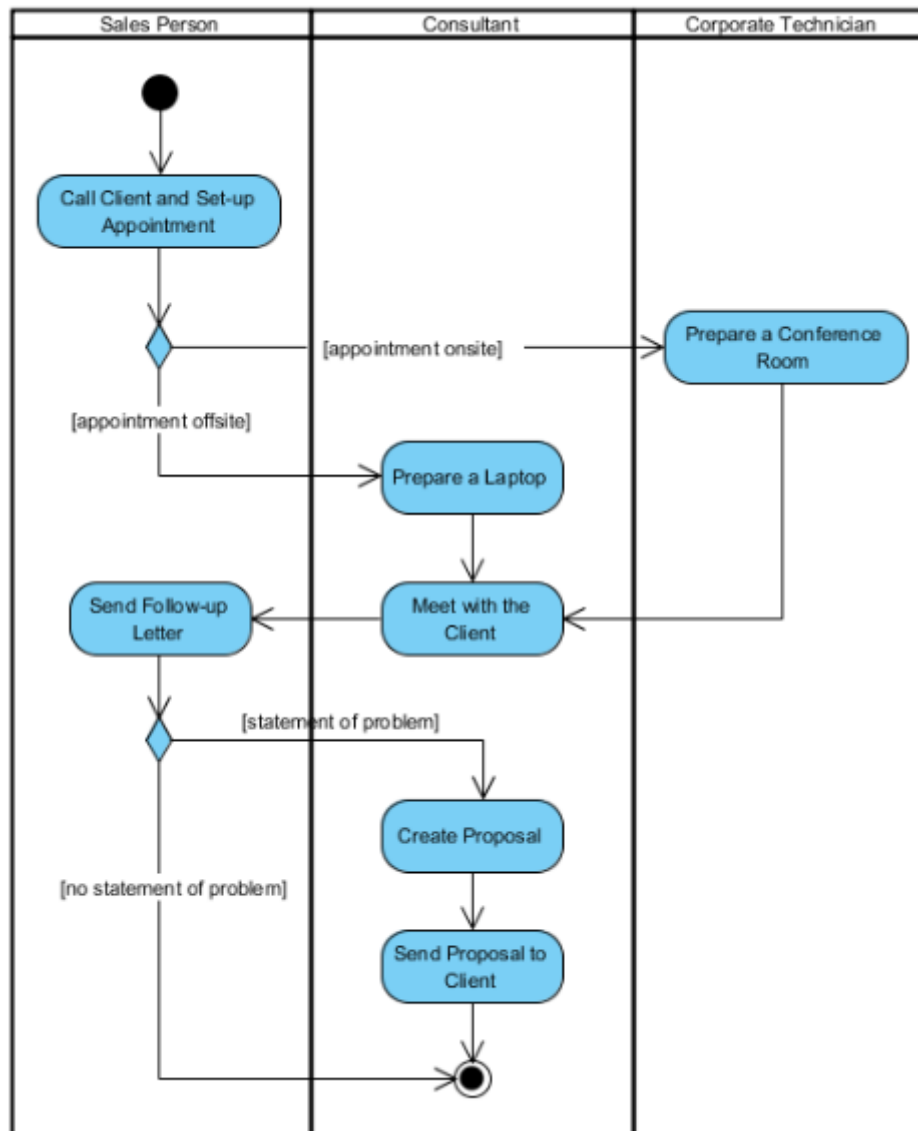


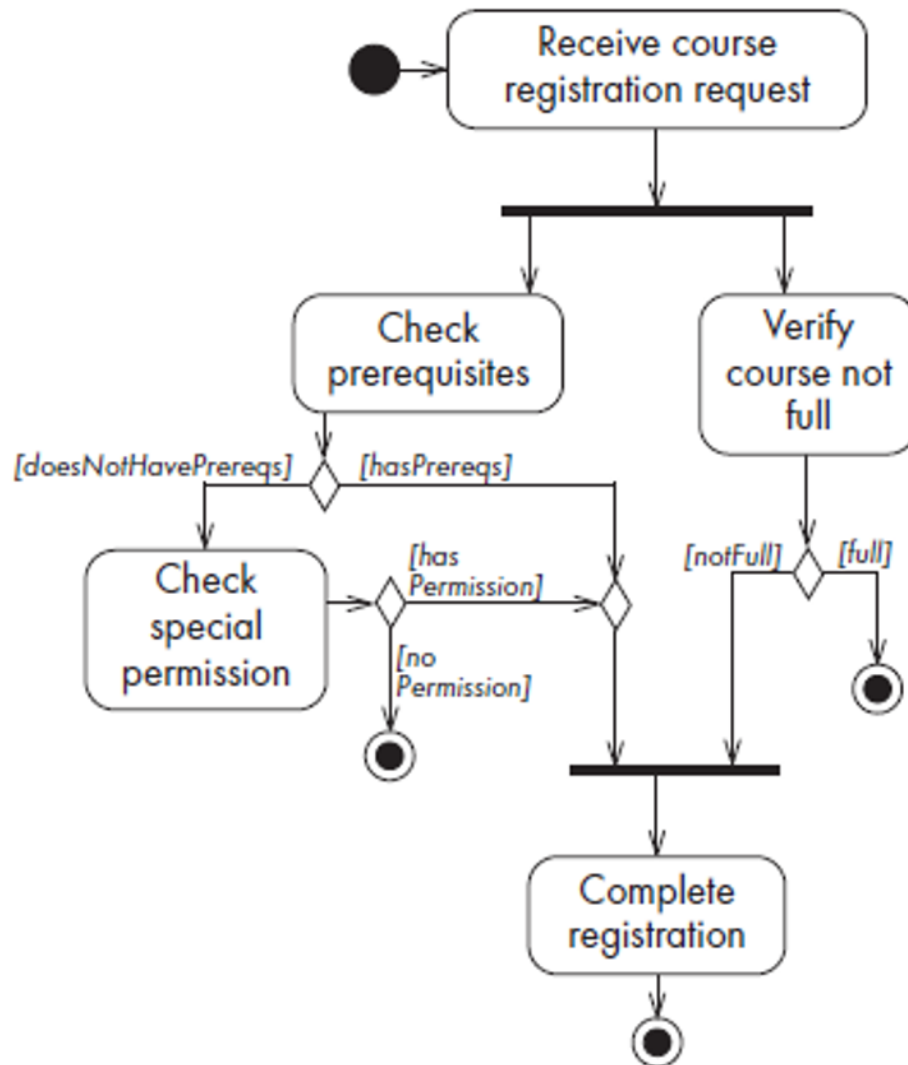
- Open the word processing package.
- Create a file.
- Save the file under a unique name within its directory.
- Type the document.
- If graphics are necessary, open the graphics package, create the graphics, and paste the graphics into the document.
- If a spreadsheet is necessary, open the spreadsheet package, create the spreadsheet, and paste the spreadsheet into the document.
- Save the file.
- Print a hard copy of the document.
- Exit the word processing package.



- An applicant wants to enroll in the university.
- The applicant hands a filled out copy of Enrollment Form.
- The registrar inspects the forms.
- The registrar determines that the forms have been filled out properly.
- The registrar informs student to attend in university overview presentation.
- The registrar helps the student to enroll in seminars
- The registrar asks the student to pay for the initial tuition.





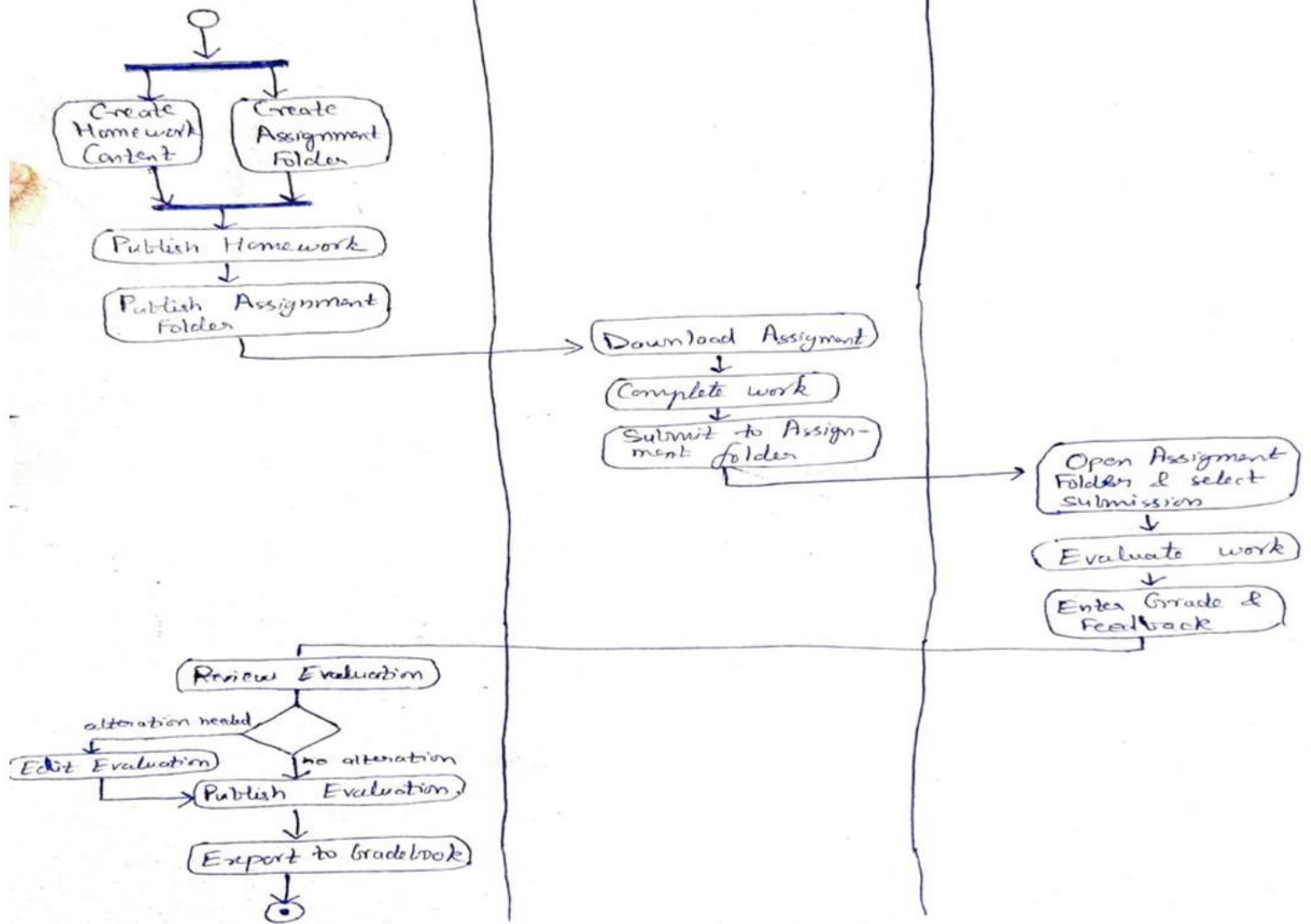


- Consider the following process for assigning, submitting, and grading Homework. The process begins when the Instructor creates the Homework instruction content on the course site. Then the Instructor creates an Assignment Folder for the Homework. These two tasks, creating the Homework content and creating the Assignment Folder have no particular order in which they must be completed, but they must both be completed concurrently before the process can continue. The process continues when the Instructor publishes the Homework and then the Assignment Folder. The student downloads the assignment and begins work. When the student has completed the work, they submit it to the Assignment Folder. The TA opens the Assignment Folder and selects the student's submission. The TA evaluates the work, then enters the grade and feedback to the submission. The Instructor reviews the evaluation. If the evaluation needs no alterations, then the Instructor publishes the evaluation. Otherwise, the Instructor edits the evaluation and then publishes it. After the evaluation is complete, the Instructor exports the grade to the Gradebook and the process ends.

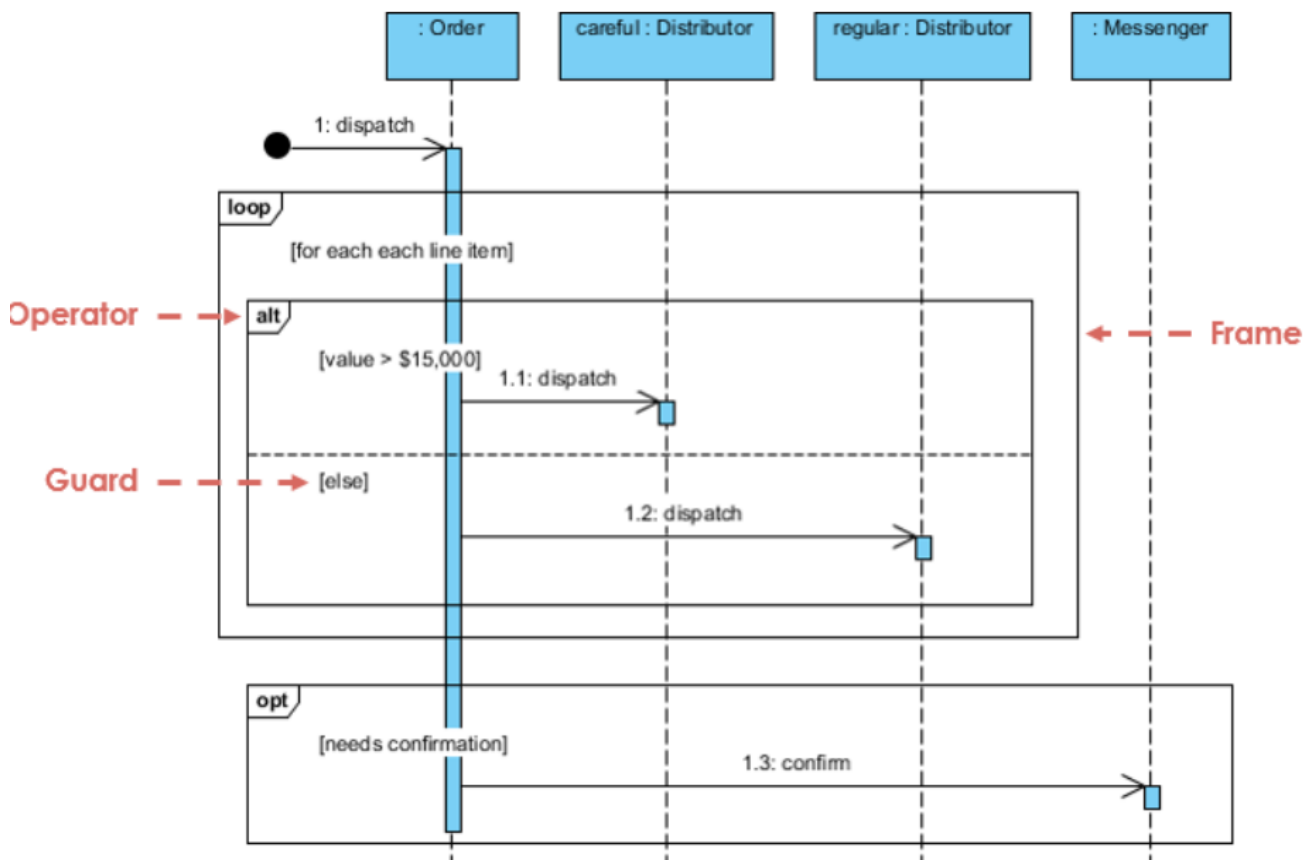
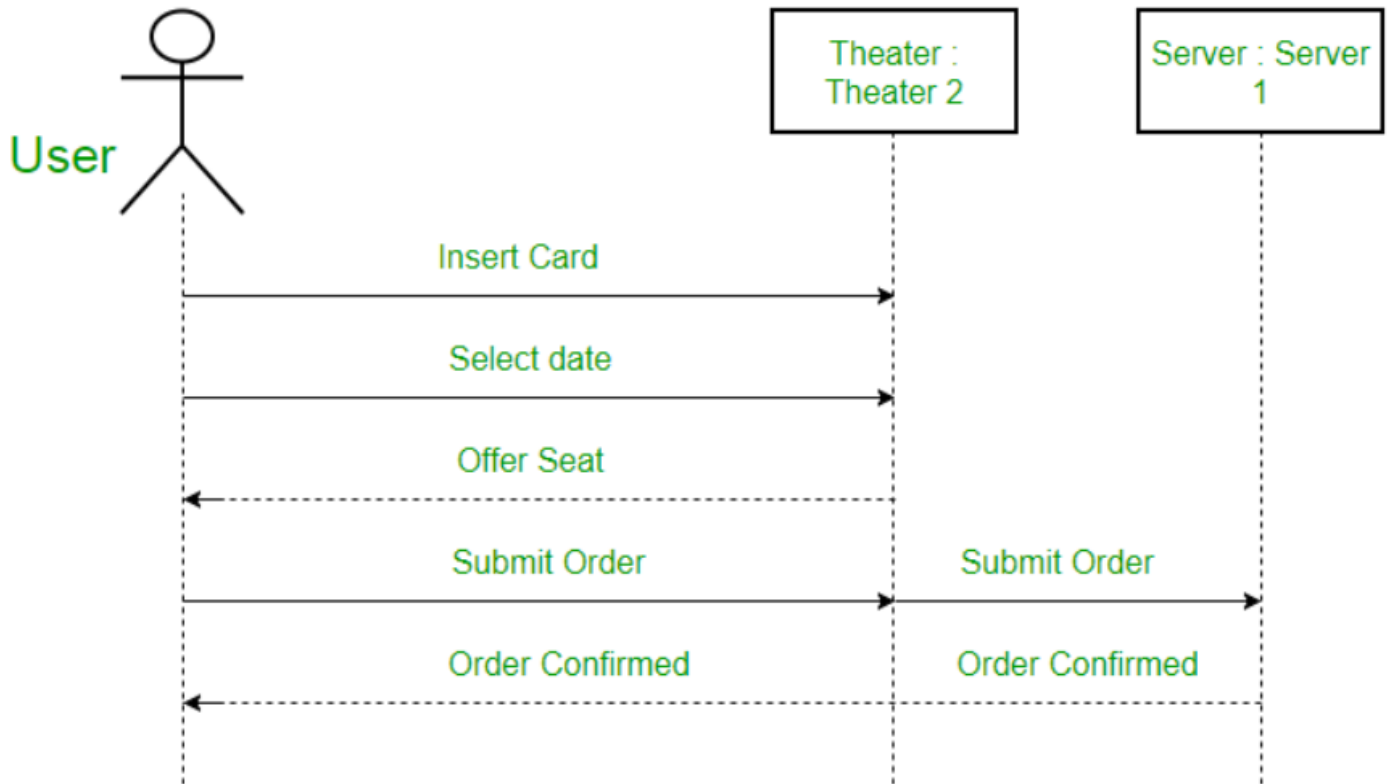
Instructor

Student

TA

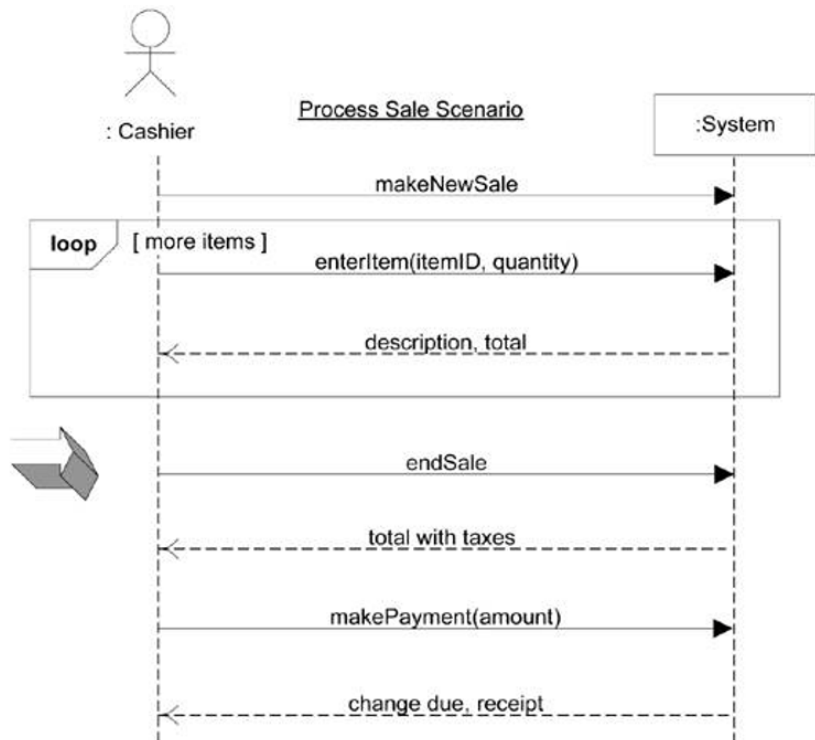


Sequence Diagrams



Simple cash-only Process Sale scenario:

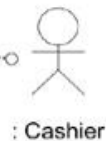
1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.
- Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
- ...



the name could be "NextGenPOS" but "System" keeps it simple

the ":" and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML

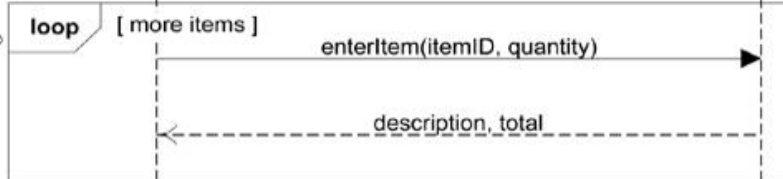
external actor to system



Process Sale Scenario

:System

a UML loop interaction frame, with a boolean guard expression



return value(s) associated with the previous message

an abstraction that ignores presentation and medium


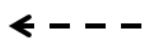
the return line is optional if nothing is returned

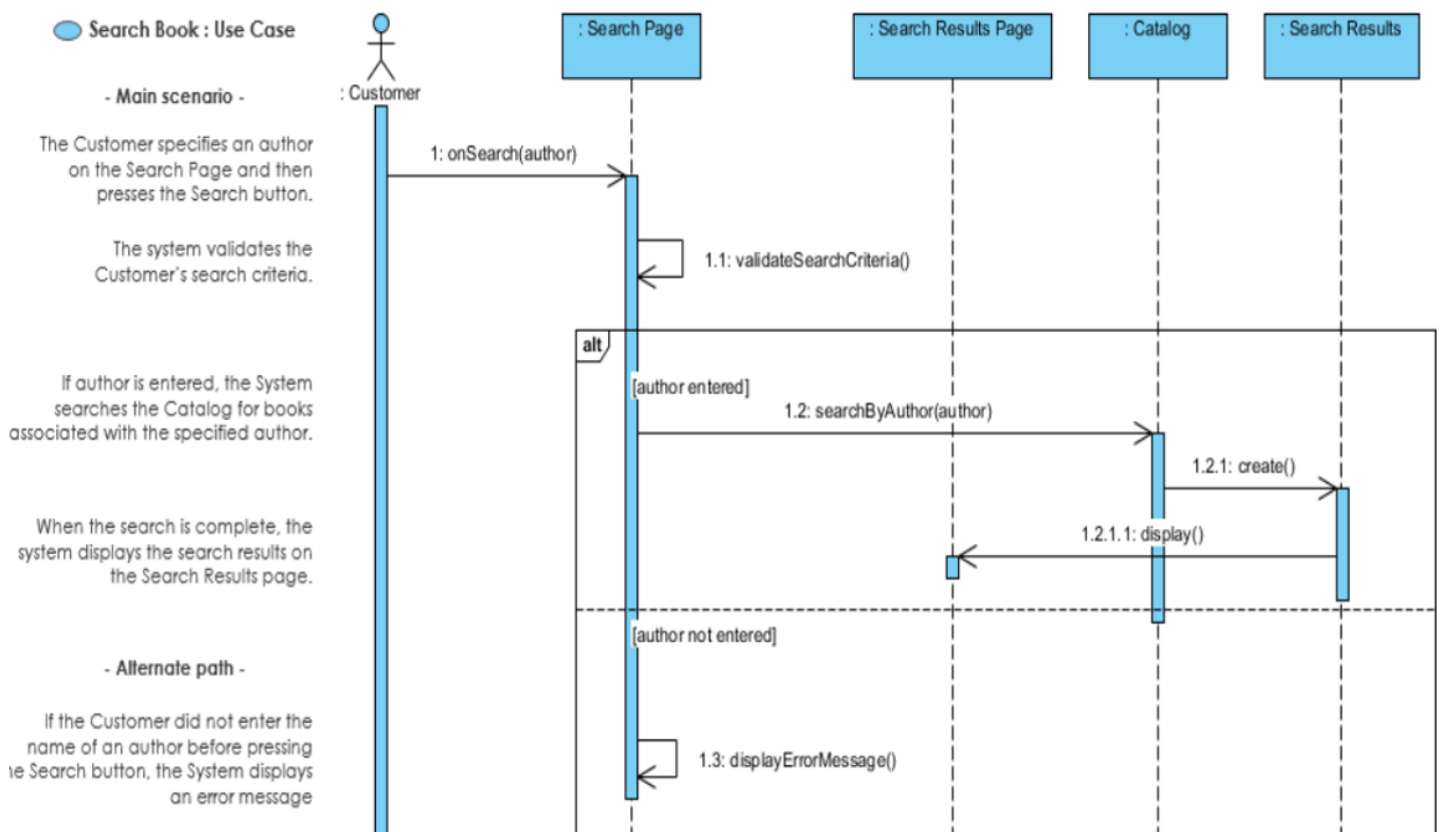
a message with parameters

it is an abstraction representing the system event of entering the payment data by some mechanism

Sequence Diagrams

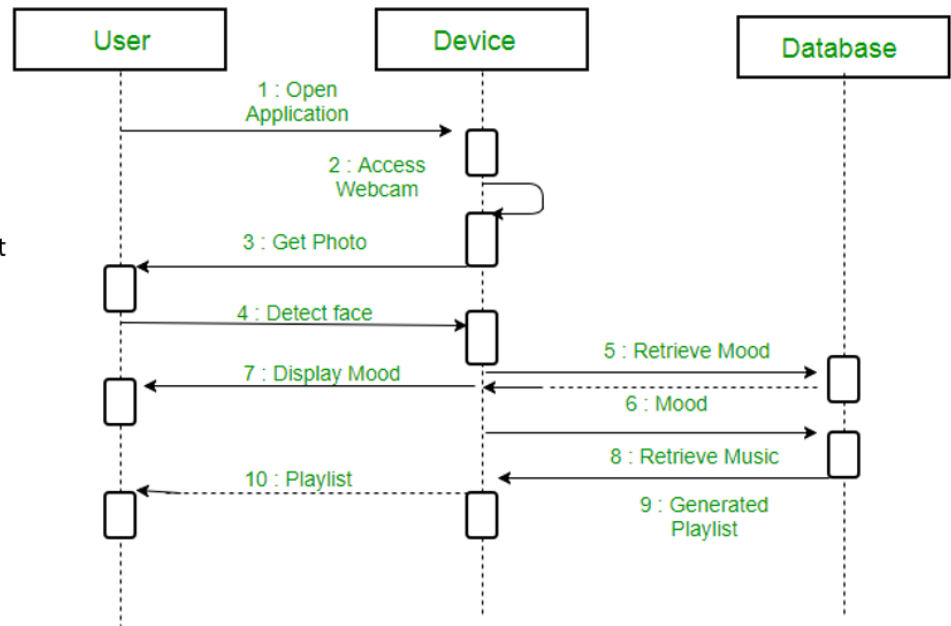
- Messages are indicated by a solid line with an arrow.

Arrow	Meaning	Arrowhead style UML 2.0
Synchronous	Transfer control and wait for answer. (sequential operations)	
Return	Returns a value to the caller (optional)	



Sequence diagram for an emotion-based music player

- Firstly, the application is opened by the user.
- The device then gets access to the web cam.
- The webcam captures the image of the user.
- The device uses algorithms to detect the face and predict the mood.
- It then requests database for dictionary of possible moods.
- The mood is retrieved from the database.
- The mood is displayed to the user.
- The music is requested from the database.
- The playlist is generated and finally shown to the user.



Use case description for the “Transfer Funds”

Use Case Name: Transfer Funds

Actors: User

Summary: This use case allows the user to transfer funds between their own accounts or to other accounts.

Preconditions: The user must be logged in to the mobile banking application.

The user must have at least one account set up in the application.

Basic Flow of Events:

1. The user selects the “Transfer Funds” option from the main menu.
2. The application presents the user with a list of their accounts and prompts them to select the account they want to transfer funds from.
3. The user selects the account they want to transfer funds from.
4. The application presents the user with a form to fill out with the recipient’s account information, including the account number and the name of the recipient.
5. The user fills out the form and enters the amount they want to transfer.
6. The application validates the recipient’s account information and the available balance in the user’s account.
7. If the validation is successful, the application deducts the transfer amount from the user’s account and adds it to the recipient’s account.
8. The application displays a confirmation message to the user with the details of the transfer.

Guards

- When modeling object interactions, there will be times when a **condition** must be met for a message to be sent to the object.
- Guards are used throughout UML diagrams to **control flow**.

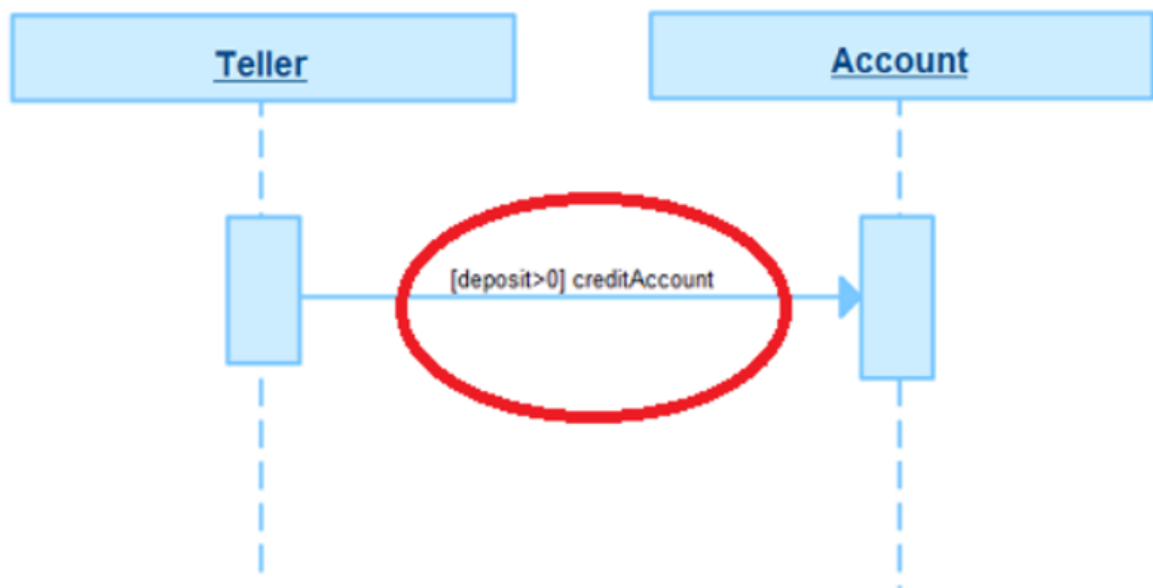
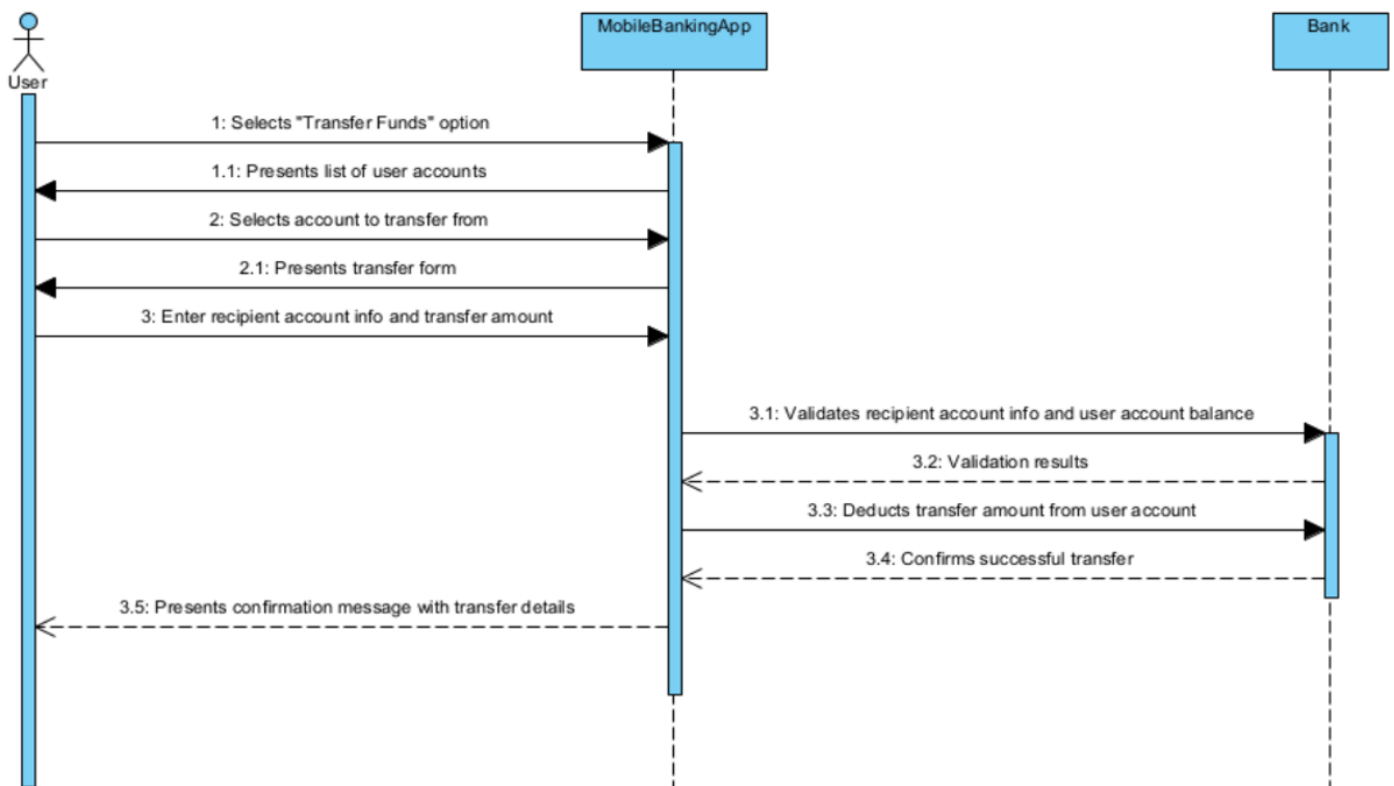


Diagram Frames in UML Sequence Diagrams

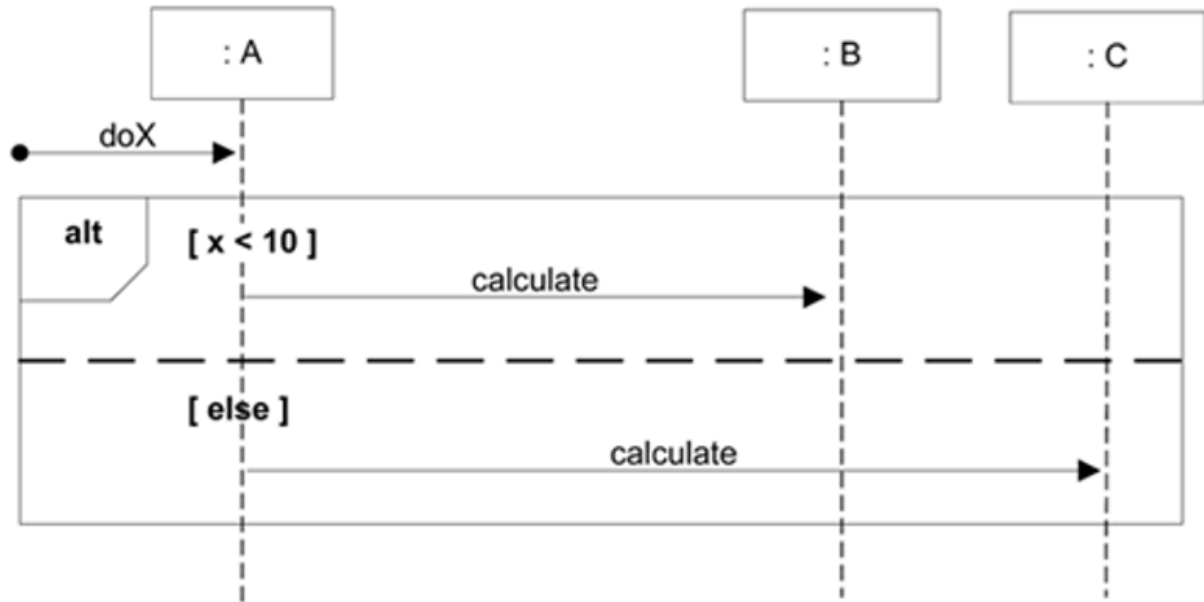
- To support **conditional** and **looping** constructs (among many other things), the UML uses **frames**.
- Frames are regions or fragments of the diagrams; they have an **operator** or **label** (such as loop) and a **guard** (conditional clause).



Normal Flow Sequence Diagram:

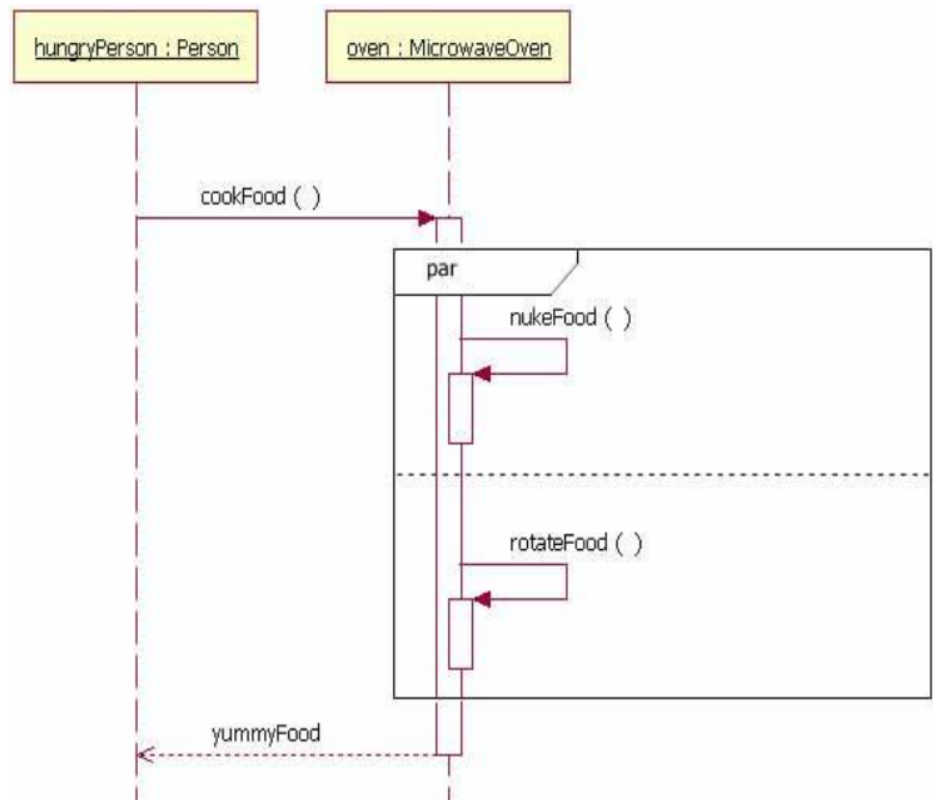


An ALT frame is placed around the mutually exclusive alternatives

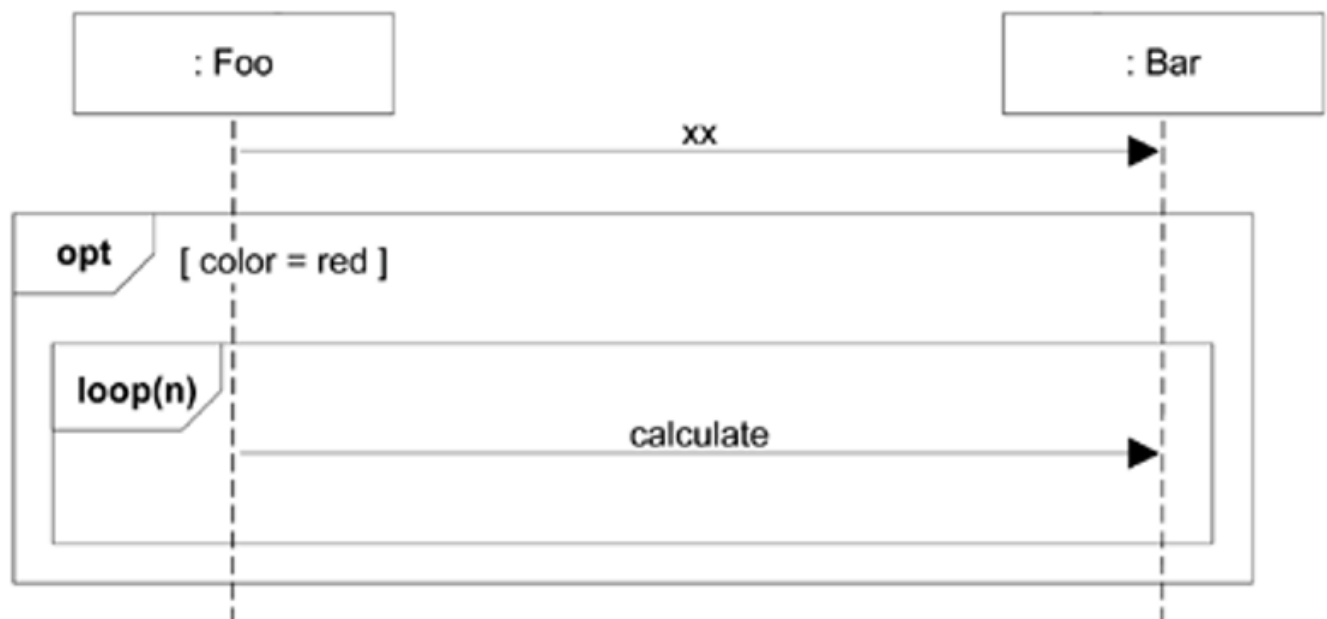


Parallel Frame

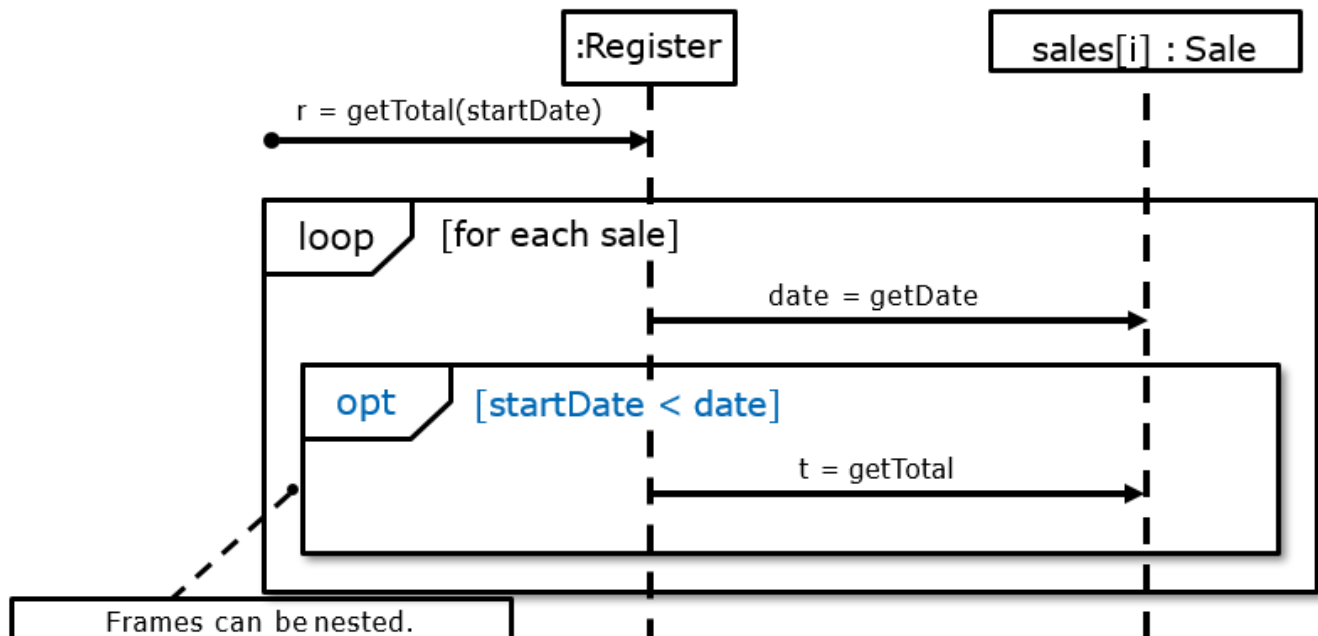
- When the processing time required to complete portions of a **complex task** is longer than desired, some systems handle parts of the processing in parallel.
- The parallel combination fragment element needs to be used when creating a sequence diagram that shows parallel processing activities.



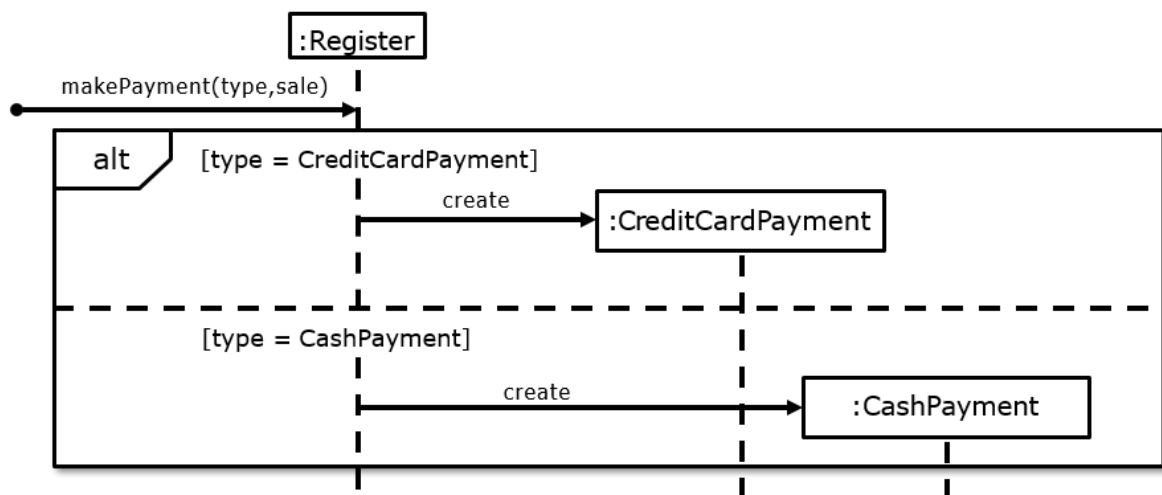
- Frames can be nested



Modeling task: Get the sum of all sales that happened today after 18:00 o'clock.

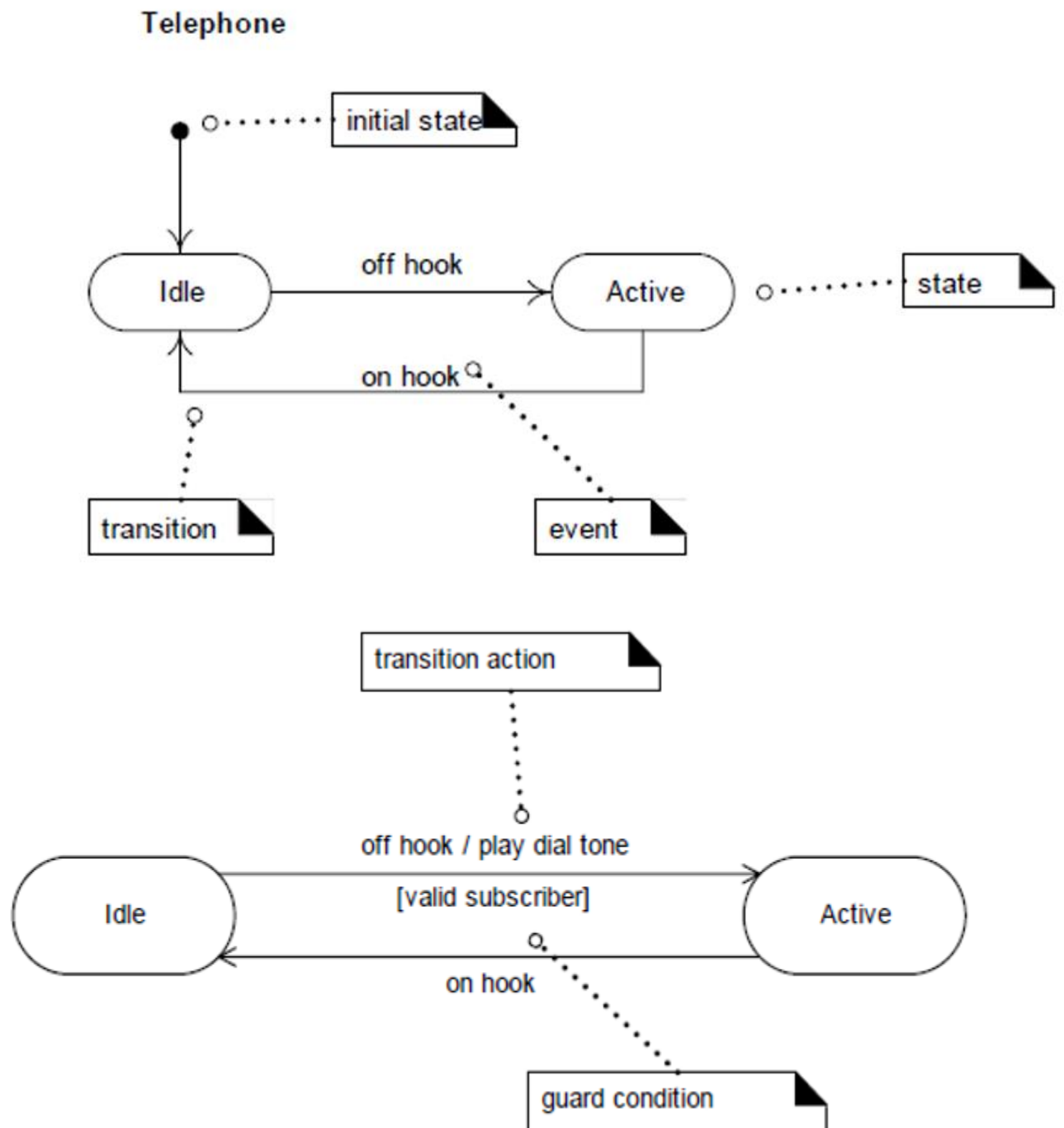


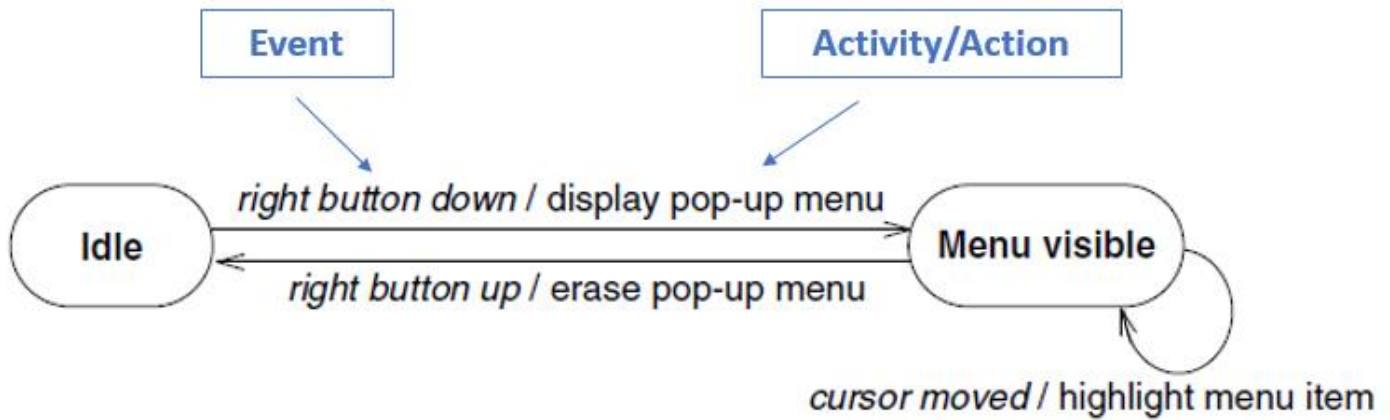
Use the UML alt frame to model between two and n mutually exclusive alternatives.



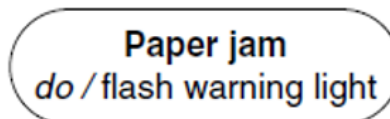
State Diagrams

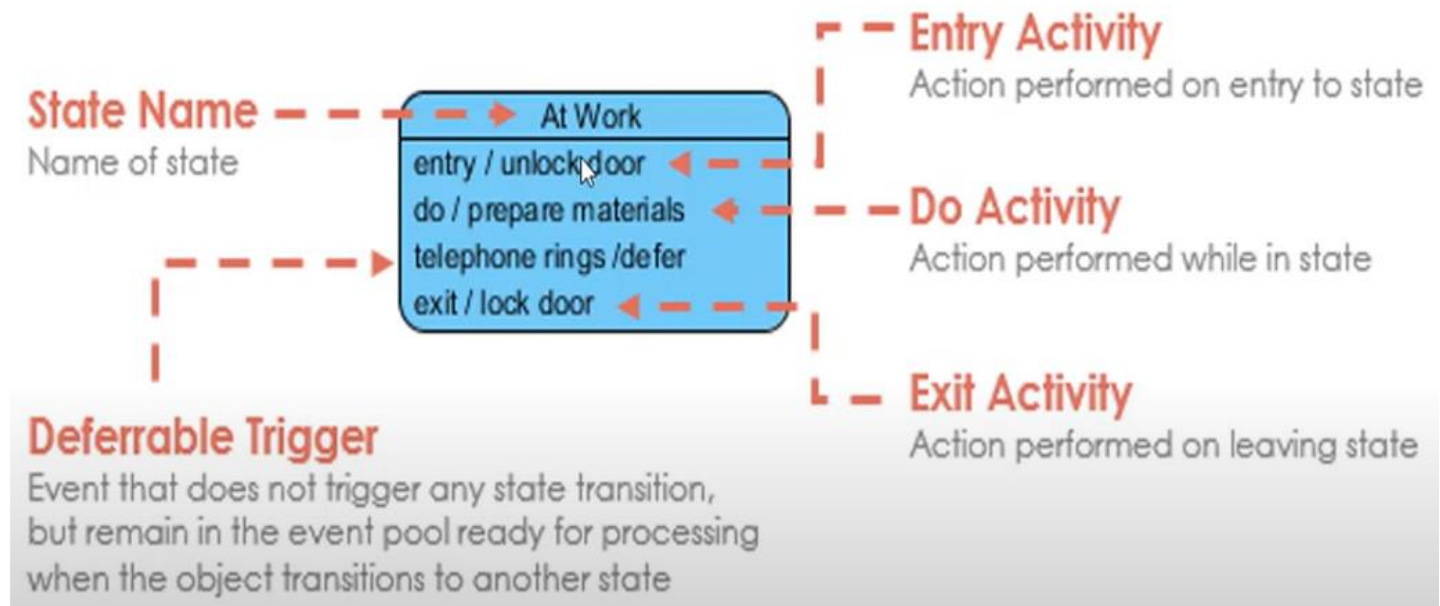
State Diagram for a Telephone





- A ***do-activity*** is an activity that continues for an **extended time**.
- The keyword *do* is reserved for indicating an ongoing activity
- By definition, a do-activity **can only occur within a state** and cannot be attached to a transition.
- For example, the warning light may flash during the *Paper jam* state for a copy machine





- A state allows nesting to contain substates; a substate inherits the transitions of its superstate (the enclosing state)

