# LAB : 1

## OBJECTIVE :

(a) Introduction to Unified Modelling Language (UML) and its type.

(b) Draw Use Case diagram for ATM application

## Requrements :

(a) Windows PC (Windows 7/8/10) / Mac

(b) Star UML Tool

## Procedure :

## (1.a) Introduction to Unified Modelling Language (UML) and its type.

### Unified Modeling Language (UML) :

UML, as the name implies, is a modeling language. It may be used to visualize, specify, construct, and document the artifacts of a software system. It provides a set of notations (e.g. rectangles, lines, ellipses, etc.) to create a visual model of the system. Like any other language, UML has its own syntax (symbols and sentence formation rules) and semantics (meanings of symbols and sentences). Also, we should clearly understand that UML is not a system design or development methodology, but can be used to document object-oriented and analysis results obtained using some methodology.

UML was developed to standardize the large number of object-oriented modeling notations that existed and were used extensively in the early 1990s. The principles ones in use were:

- Object Management Technology [Rumbaugh 1991]
- Booch's methodology [Booch 1991]
- Object-Oriented Software Engineering [Jacobson 1992]
- Odell's methodology [Odell 1992]

• Shaler and Mellor methodology [Shaler 1992]

It is needless to say that UML has borrowed many concepts from these modeling techniques. Especially, concepts from the first three methodologies have been heavily drawn upon. UML was adopted by Object Management Group (OMG) as a de facto standard in 1997.

## UML diagrams

UML can be used to construct nine different types of diagrams to capture five different views of a system. Just as a building can be modeled from several views (or perspectives) such as ventilation perspective, electrical perspective, lighting perspective, heating perspective, etc.; the different UML diagrams provide different perspectives of the software system to be developed and facilitate a comprehensive understanding of the system. Such models can be refined to get the actual implementation of the system.

The UML diagrams can capture the following five views of a system:

1. **User's view** : This view defines the functionalities (facilities) made available by the system to its users. The users' view captures the external users' view of the system in terms of the functionalities offered by the system. The users' view is a black-box view of the system where the internal structure, the dynamic behavior of different system components, the implementation etc. are not visible.
2. **Structural view** : The structural view defines the kinds of objects (classes) important to the understanding of the working of a system and to its implementation. It also captures the relationships among the classes (objects). The structural model is also called the static model, since the structure of a system does not change with time.
3. **Behavioral view** : The behavioral view captures how objects interact with each other to realize the system behavior. The system behavior captures the time-dependent (dynamic) behavior of the system.
4. **Implementation view** : This view captures the important components of the system and their dependencies.
5. **Environmental view** : Environmental view: This view models how the different components are implemented on different pieces of hardware.

Fig. 1.1 shows the UML diagrams responsible for providing the different views

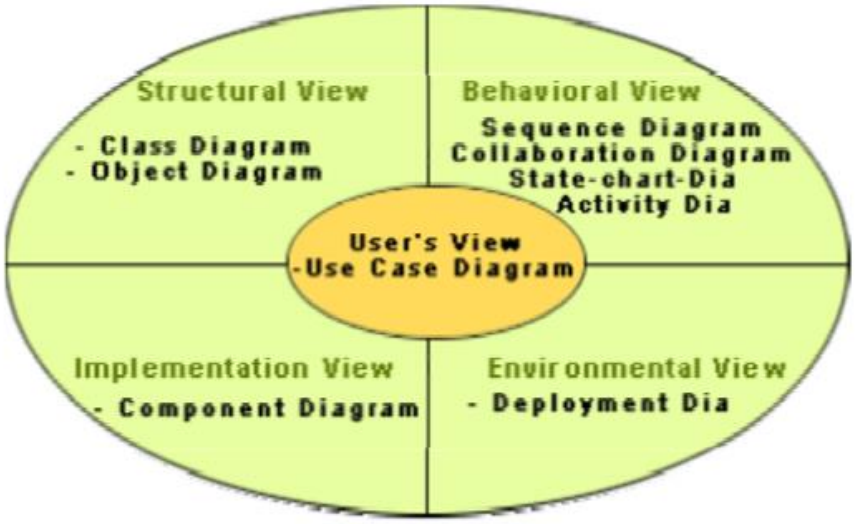



**Fig. 1.1** Different types of diagrams and views supported in UML

Different UML Diagrams are as following :

**Use Case Diagram :** As the most known diagram type of the behavioral UML types, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.

**Class Diagram :** Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.

**Object Diagram :** Object Diagrams, sometimes referred to as Instance diagrams are very similar to class diagrams. Like class diagrams, they also show the relationship between objects but they use real-world examples.

**Sequence Diagram :** Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows.

**Collaboration Diagram :**The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

**State Machine Diagram :** The state machine diagram is also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system. It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.

**Activity Diagram :** In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

**Component Diagram :** A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

**Deployment Diagram :** The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software

will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

Some other UML Diagrams are :

**Interaction Diagram :** As the name suggests, the interaction diagram portrays the interactions between distinct entities present in the model. It amalgamates both the activity and sequence diagrams. The communication is nothing but units of the behavior of a classifier that provides context for interactions.

A set of messages that are interchanged between the entities to achieve certain specified tasks in the system is termed as interaction. It may incorporate any feature of the classifier of which it has access. In the interaction diagram, the critical component is the messages and the lifeline.

**Timing Diagram :** In UML, the timing diagrams are a part of Interaction diagrams that do not incorporate similar notations as that of sequence and collaboration diagram. It consists of a graph or waveform that depicts the state of a lifeline at a specific point of time. It illustrates how conditions are altered both inside and between lifelines alongside linear time axis.

The timing diagram describes how an object underwent a change from one form to another. A waveform portrays the flow among the software programs at several instances of time.

## (1.b) Draw Use Case diagram for ATM application

- On Star UML Tool

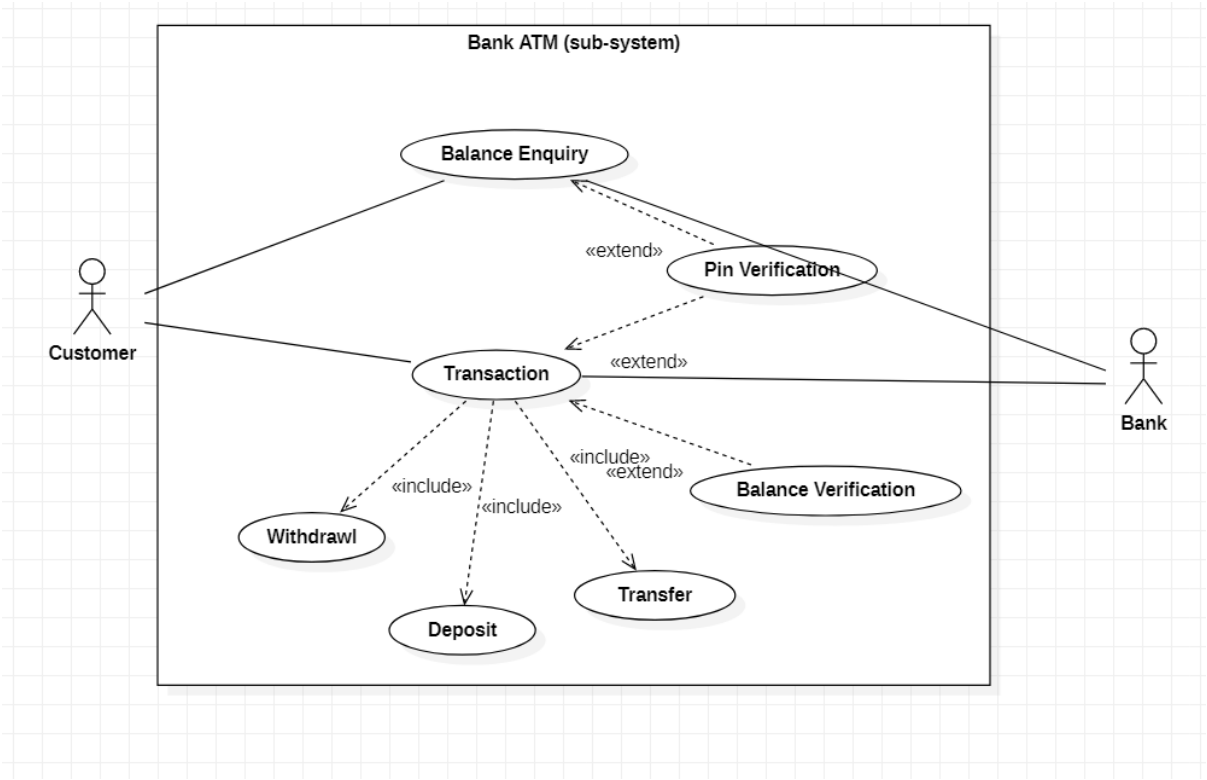



**Fig. 1.2** UML Diagram for a Bank Atm (sub-system)

# LAB : 2

## OBJECTIVE :

(a) Study Use case diagrams to answer given situaton.
(b) Draw Use Case diagram for Ticket Distributor for a Train System

## Requrements :

(a) Windows PC (Windows 7/8/10) / Mac

(b) Star UML Tool

## Procedure :

## (a) Study Use case diagrams to answer given situaton

**2-1 Consider an ATM system. Identify at least three different actors that interact with this system.**

An actor is any entity (user or system) that interacts with the system of interest. For an ATM, possible actors are :

- Bank Customer
- ATM Maintainer
- Central Bank Computer

**2–2 Can the system under consideration be represented as an actor? Justify your answer.**

The system under consideration is not external to the system and shouldn't be represented as an actor. There are a few cases, however, when representing the system as an actor may clarify the use case model. These include situations where

the system initiates uses cases, for example, as time passes (Check for Outdated Articles, Send Daily Newsletter).

**2–3 What is the difference between a scenario and a use case? When do you use each construct?**

A **scenario** is an actual sequence of interactions (i.e., an instance) describing one specific situation; a use case is a general sequence of interactions (i.e., a class) describing all possible scenarios associated with a situation. Scenarios are used as examples and for clarifying details with the client.

**Use cases** are used as complete descriptions to specify a user task or a set of related system features.

## (b) Draw Use Case diagram for Ticket Distributor for a Train System

**Problem Statement :**

Draw a use case diagram for a ticket distributor for a train system. The system includes two actors: a traveler,who purchases different types of tickets, and a central computer system, which maintains a reference database for the tariff.

Use cases should include: BuyOneWayTicket, BuyWeeklyCard, BuyMonthlyCard, UpdateTariff. Also include the following exceptional cases: Time-Out (i.e., traveler took too long to insert the right amount), TransactionAborted (i.e., traveler selected the cancel button without completing the transaction), DistributorOutOfChange, and DistributorOutOfPaper

**Use Case Diagram :**

1. The traveler (actor) decides which kind of ticket he wants to buy, one way, weekly or monthly card.

2. Further user can face the exceptions like time runs out or transaction gets aborted in between or distributor gets out of change.

3. The relationship between the exceptional use cases and normal use cases uses <> relationship, as extend is used to extend the use cases in order to add new functionality in the use cases.

4. On the other side, CentralComputerSystem will see the function of UpdateTariff.

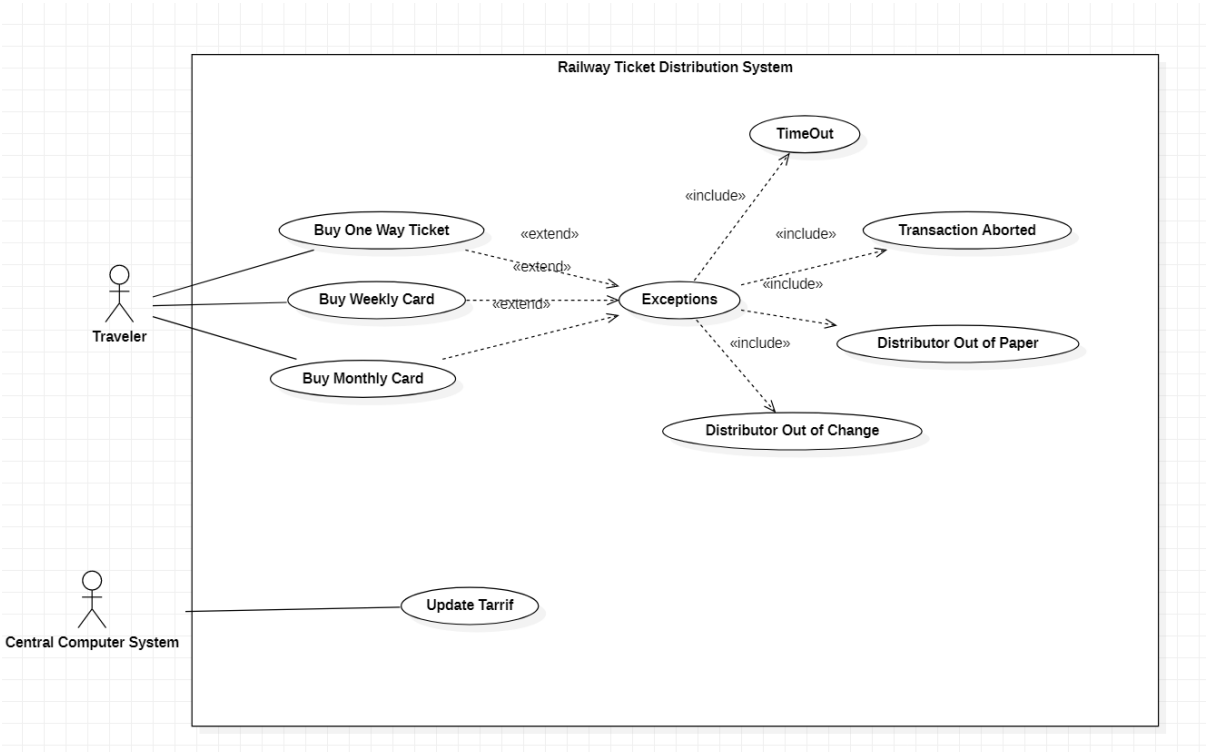



**Fig 2.1** Use case diagram for a ticket distributor

# LAB : 3

## OBJECTIVE :

(c) Write the flow of events and specify all fields for the use case UpdateTariff that you drew in Lab 2. Do not forget to specify any relationships.
(d) Draw a class diagram representing a book defined by the following statement: "A book is composed of a number of parts, which in turn are composed of a number of chapters. Chapters are composed of sections." Focus only on classes and relationships.
(e) Add multiplicity to the class diagram you produced in Exercise (b).
(f) Draw an object diagram representing the first part of this book (i.e., Part I, Getting Started). Make sure that the object diagram you draw is consistent with the class diagram of Exercise (b).

## Requrements :

(a) Windows PC (Windows 7/8/10) / Mac

(b) Star UML Tool

## Procedure :

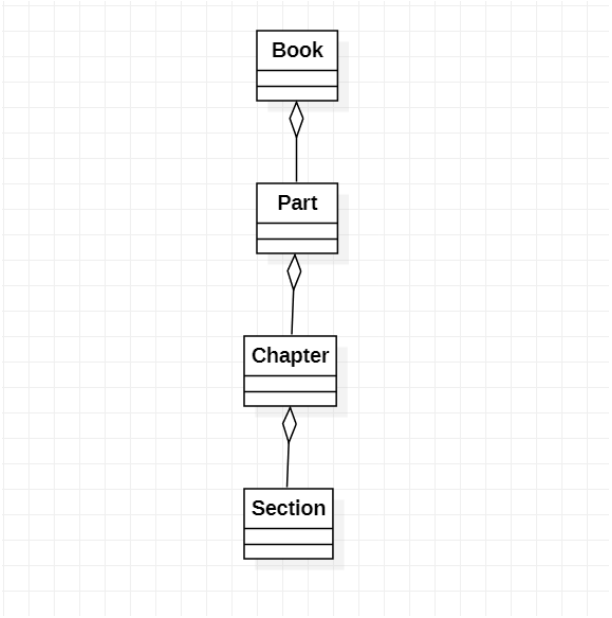
(a) Write the flow of events and specify all fields for the use case UpdateTariff that you drew in Lab 2. Do not forget to specify any relationships.

- **Use case name** UpdateTariff
- **Participating actor** Initiated by CentralComputerSystem
- **Flow Of Events**
  1. The CentralComputerSystem activates the "UpdateTariff" function of the ticket distributors available on the network.
     i. The ticket distributor disables the traveler interface and posts a sign indicating that the ticket distributor is under maintenance.
     ii. The ticket distributor waits for the new database from the CentralComputerSystem

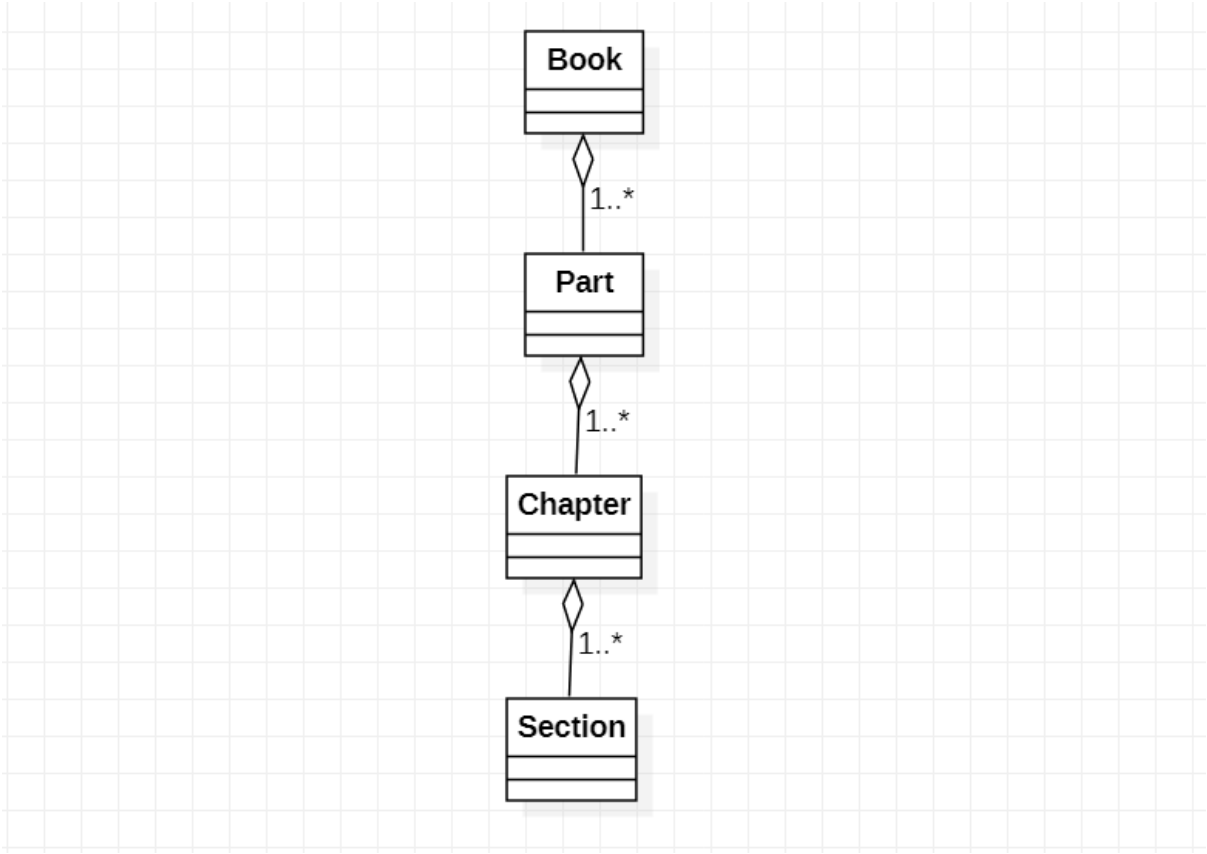
2. After waiting a minute for the ticket distributors to reach a waiting state, the CentralComputerSystem broadcasts the new database.
    i. 5. The ticket distributor system receives the new database of tariff. Upon complete, the ticket distributor sends an acknowledgement to the CentralComputerSystem
    ii. After acknowledgment, the ticket distributor enables the traveler interface and can issue tickets at the new tariff.
3. The CentralComputerSystem hecks if all ticket distributors have acknowledged the new database. If not, the CentralComputerSystem invokes the Check NonRespondingDistributors use case.

- **Entry condition** The ticket distributor is connected to a network reachable by the CentralComputerSystem.
- **Exit condition**
    - The ticket distributor can issue tickets under the new tariff, OR •
    - The ticket distributor is disabled and displays a sign denoting that it is under maintenance.
- **Quality requirements**
    - The ticket distributor stays offline at most 2 minutes and is considered out-of-order otherwise.

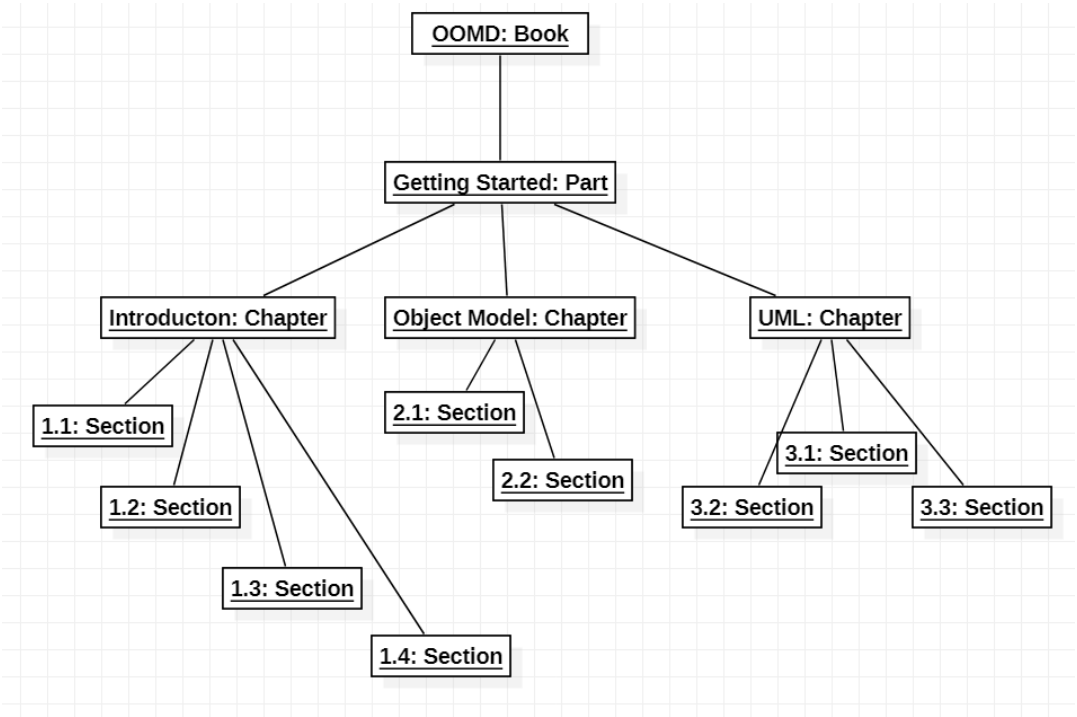
(b) Draw a class diagram representing a book defined by the following statement: "A book is composed of a number of parts, which in turn are composed of a number of chapters. Chapters are composed of sections." Focus only on classes and relationships.




(c) Add multiplicity to the class diagram you produced in Exercise (b).




(d) Draw an object diagram representing the first part of this book (i.e., Part I, Getting Started). Make sure that the object diagram you draw is consistent with the class diagram of Exercise (b).



# LAB : 4

## OBJECTIVE :

(a) Extend the class diagram of Book to include the following attributes:
- a book includes a publisher, publication date, and an ISBN
- a part includes a title and a number
- a chapter includes a title, a number, and an abstract
- a section includes a title and a number

(b) Consider the class diagram of above question. Note that the Part, Chapter, and Section classes all include a title and a number attribute. Add an abstract class and a generalization relationship to factor out these two attributes into the abstract class.

## Requrements :

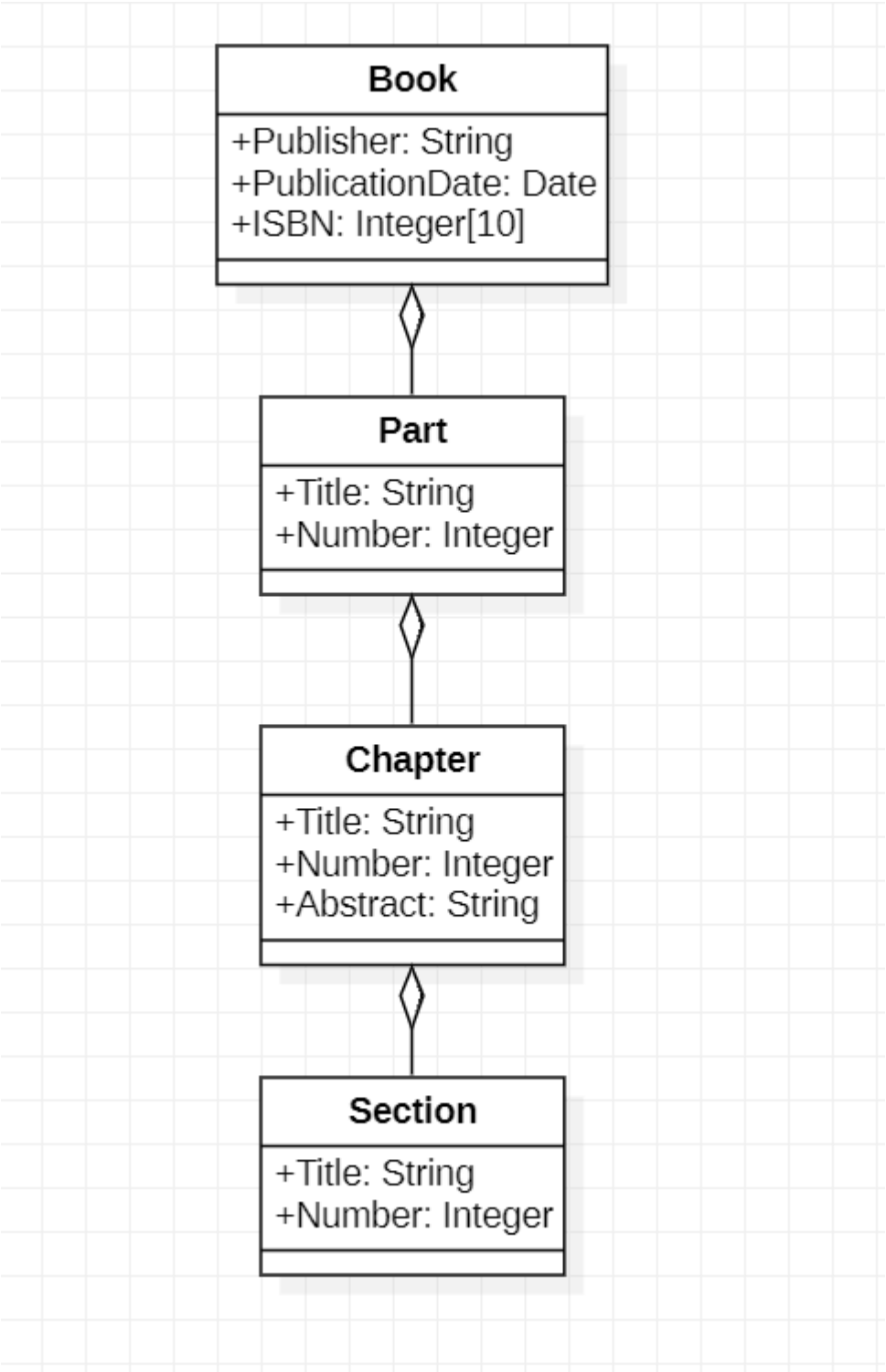
(a) Windows PC (Windows 7/8/10) / Mac

(b) Star UML Tool

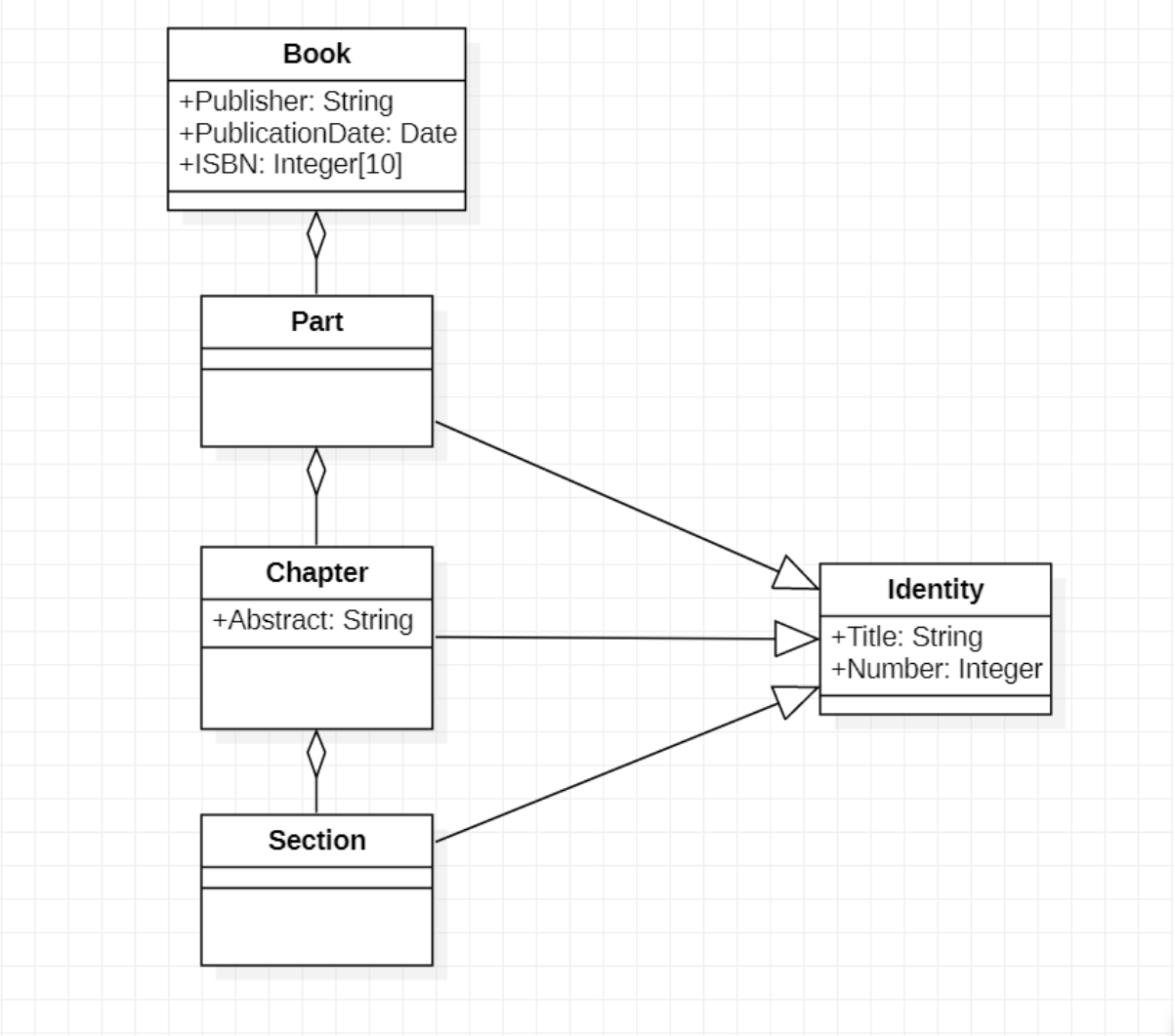
## Procedure :

(e) Extend the class diagram of Book to include the given attributes

- a book includes a publisher, publication date, and an ISBN
- a part includes a title and a number
- a chapter includes a title, a number, and an abstract
- a section includes a title and a number


Book
+Publisher: String
+PublicationDate: Date
+ISBN: Integer[10]
Part
+Title: String
+Number: Integer
Chapter
+Title: String
+Number: Integer
+Abstract: String
Section
+Title: String
+Number: Integer

(f) Consider the class diagram of above question. Note that the Part, Chapter, and Section classes all include a title and a number attribute. Add an abstract class and a generalization relationship to factor out these two attributes into the abstract class.



# LAB : 5

## OBJECTIVE :

(a) Draw a class diagram representing the relationship between parents and children. Take into account that a person can have both a parent and a child. Annotate associations with roles and multiplicities.

(b) Draw a class diagram for bibliographic references. Use the references in Appendix C, Bibliography, to test your class diagram. Your class diagram should be as detailed as possible.

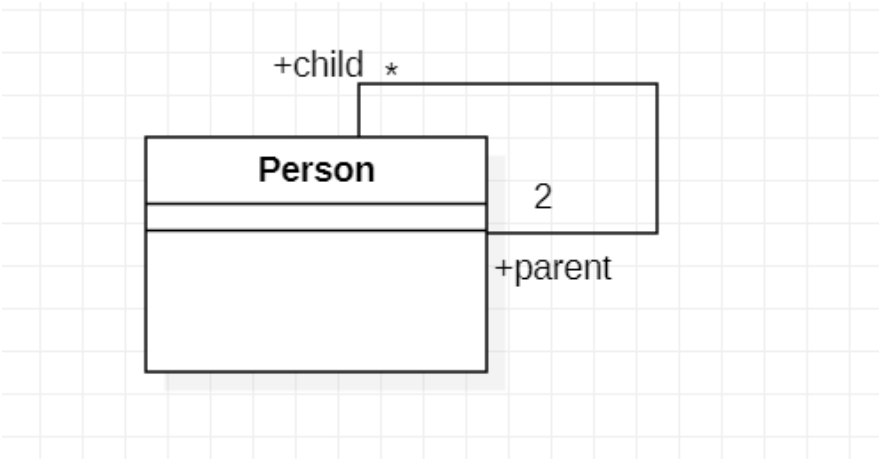
## Requrements :

(a) Windows PC (Windows 7/8/10) / Mac

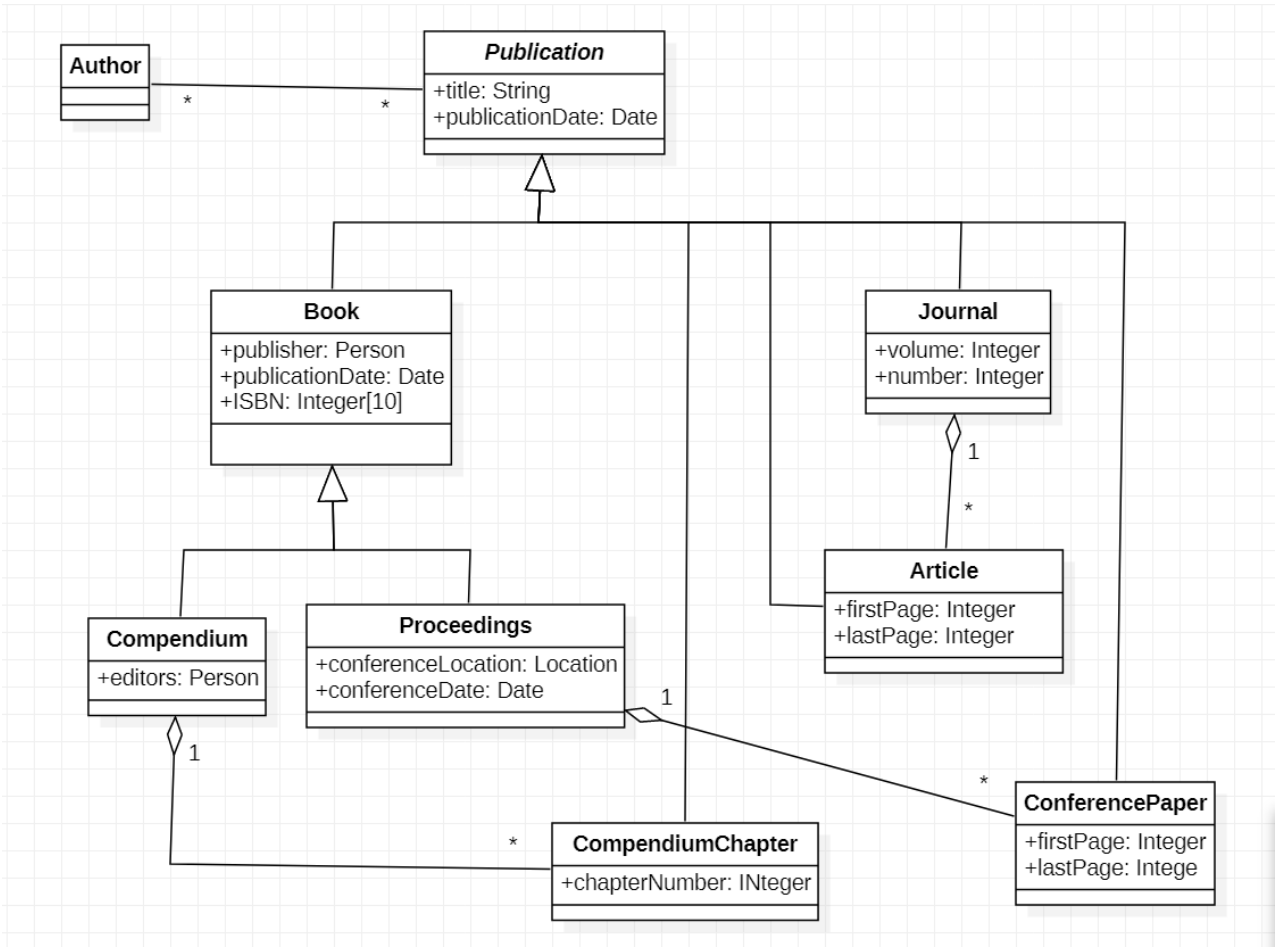
(b) Star UML Tool

## Procedure :

(a) Draw a class diagram representing the relationship between parents and children. Take into account that a person can have both a parent and a child. Annotate associations with roles and multiplicities.



(b) Draw a class diagram for bibliographic references. Use the references in Appendix C, Bibliography, to test your class diagram. Your class diagram should be as detailed as possible.



# LAB : 6

## OBJECTIVE :

1. Draw a sequence diagram for the warehouseOnFire scenario. Include the objects bob ,alice ,john , FRIEND , and instances of other classes you may need. Draw only the first five message sends. In addition to the UML rules on sequence diagrams, all correct sequence diagrams for this exercise should include one or more actors on the left of the diagram who initiate the scenario, one or more objects in the center of the diagram which represent the system, and a dispatcher actor on the right of the diagram who is notified of the emergency. All actors and objects should be instances
2. Draw a sequence diagram for the ReportIncident use case of Figure 6-1. Draw only the first five message sends. A correct solution should include a FieldOfficer actor on the left, a Dispatcher actor on the right, and one or more classes in the middle representing the FRIEND system. The answer to this exercise should be consistent with the answer to the previous exercise, in the sense that all class and operation names should be the same

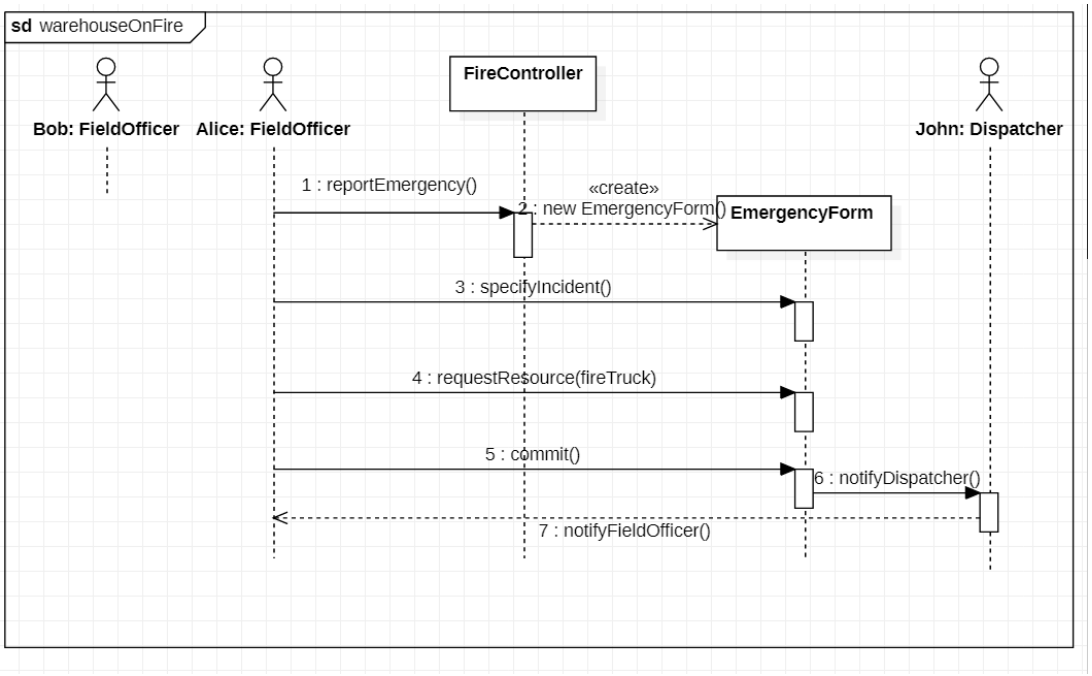
## Requrements :

(a) Windows PC (Windows 7/8/10) / Mac

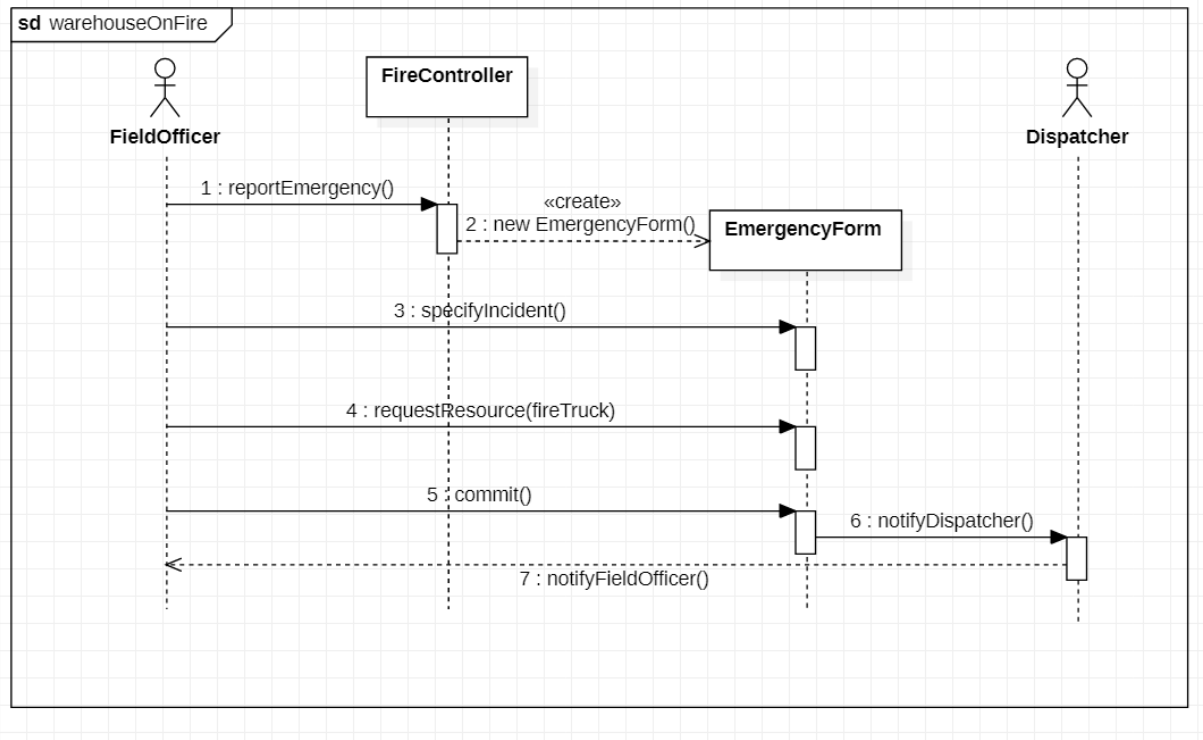
(b) Star UML Tool

## Sequence Diagram :

### 1. 6.1




### 2. 6.2



# LAB : 7

**TOPIC :** Activity Diagram

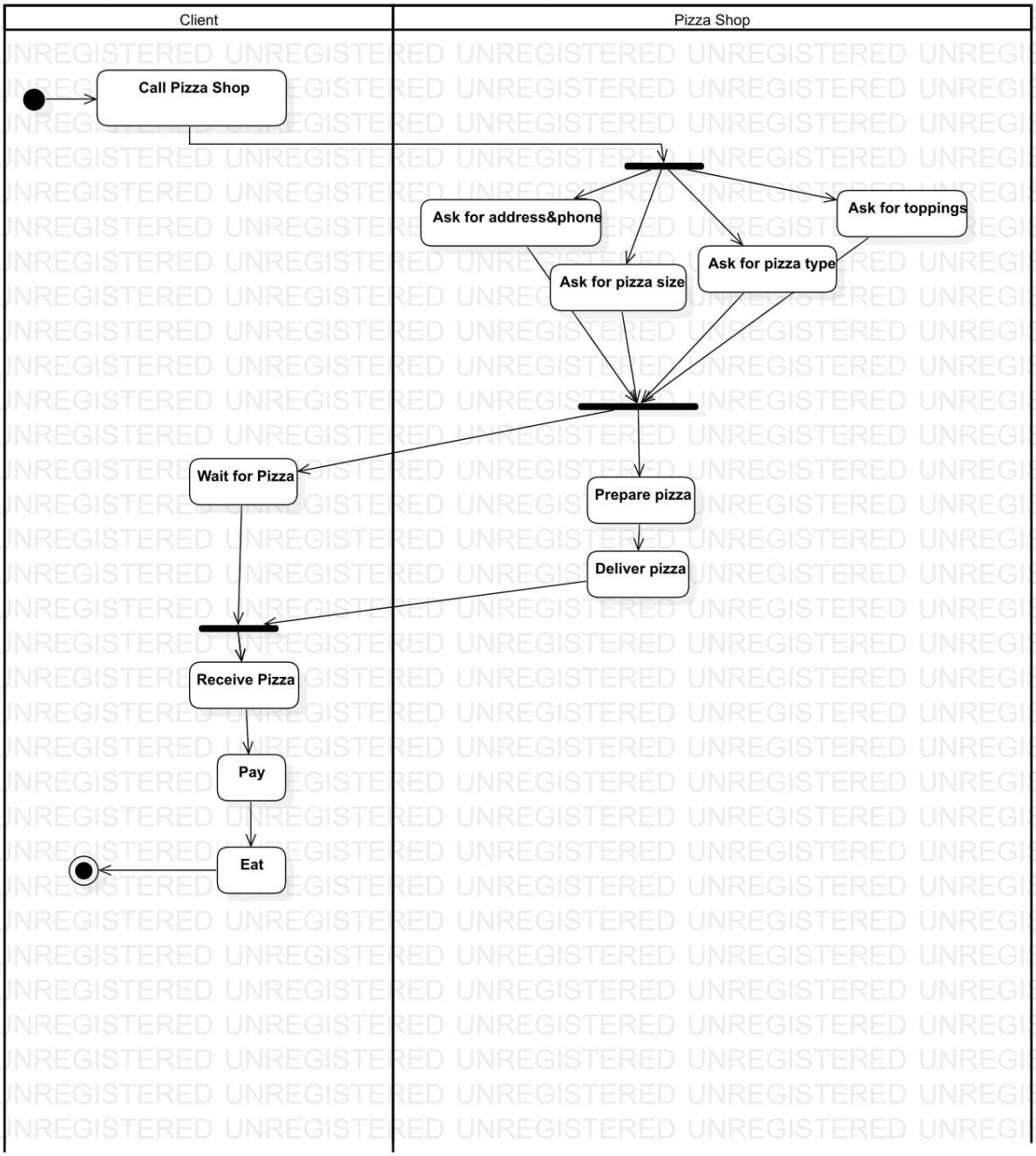
## Requrements :

(a) Windows PC (Windows 7/8/10) / Mac

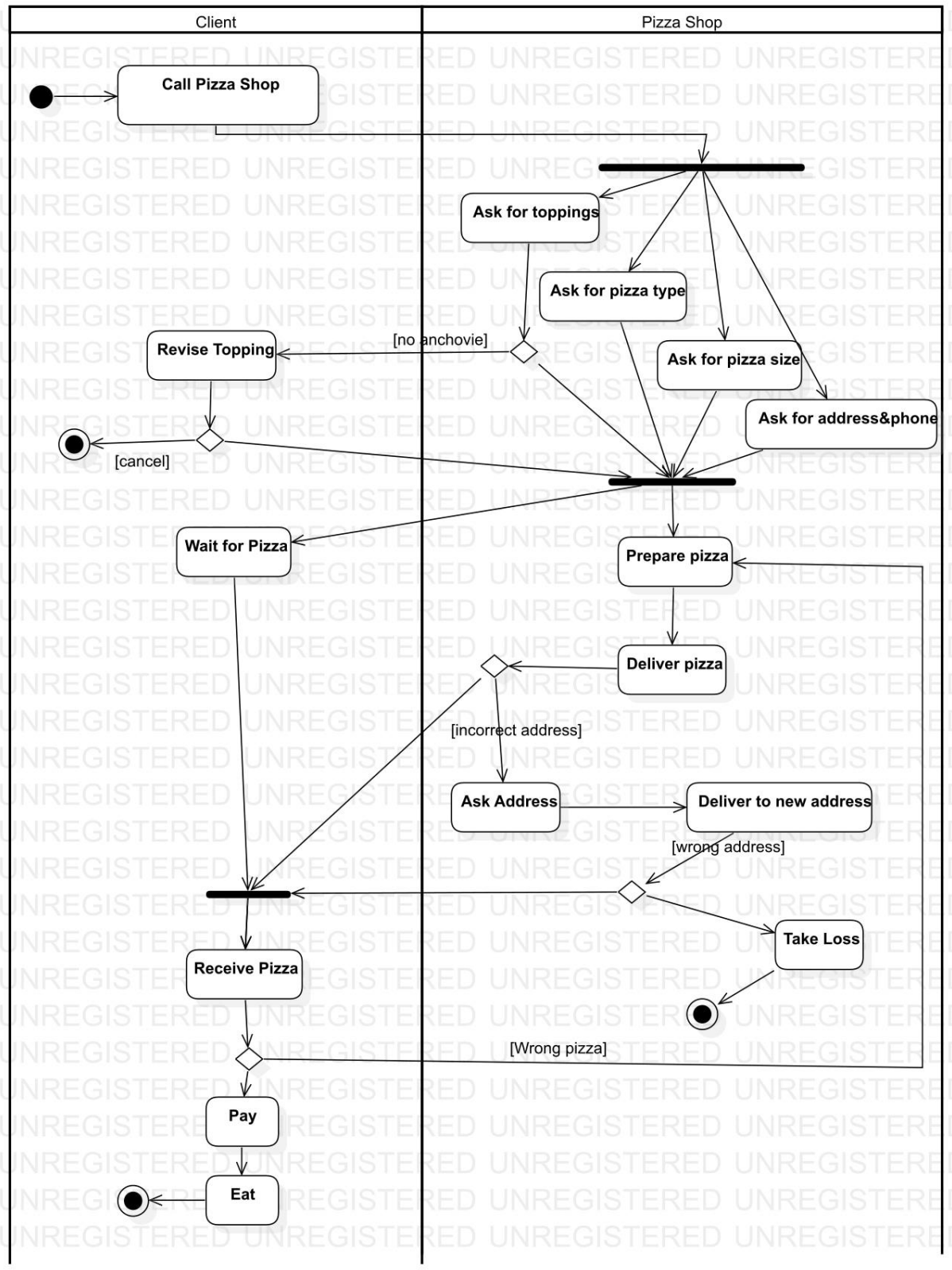
(b) Star UML Tool

## Activity Diagram :

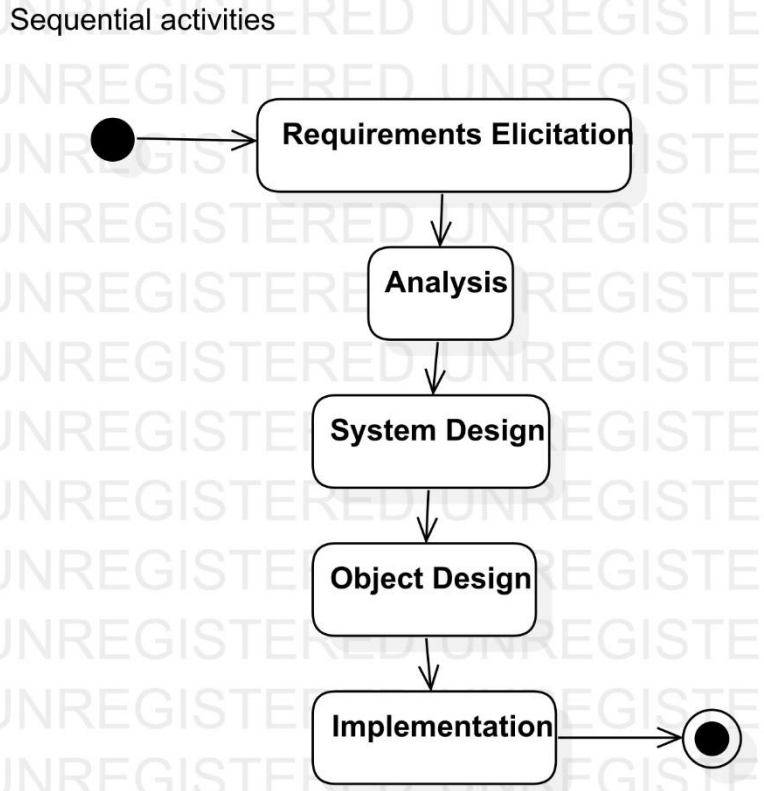
1. Consider the process of ordering a pizza over the phone. Draw an activity diagram representing each step of the process, from the moment you pick up the phone to the point where you start eating the pizza. Do not represent any exceptions. Include activities that others need to perform.

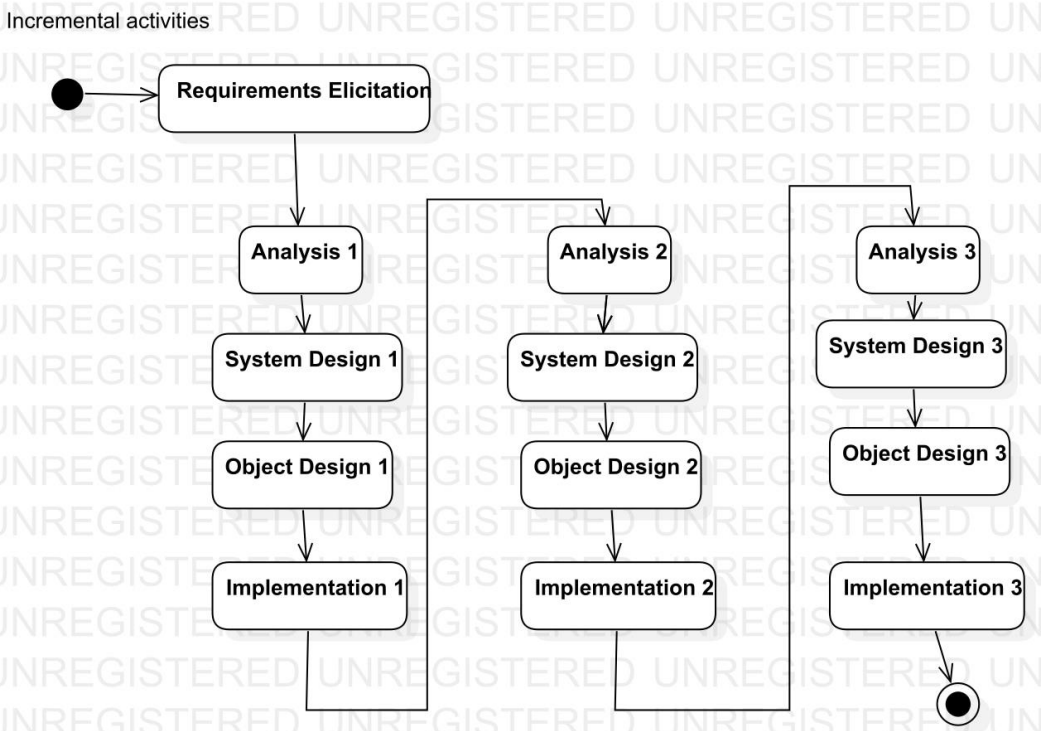

Client
Pizza Shop
Call Pizza Shop
Ask for address&phone
Ask for toppings
Ask for pizza size
Ask for pizza type
Wait for Pizza
Prepare pizza
Deliver pizza
Receive Pizza
Pay
Eat

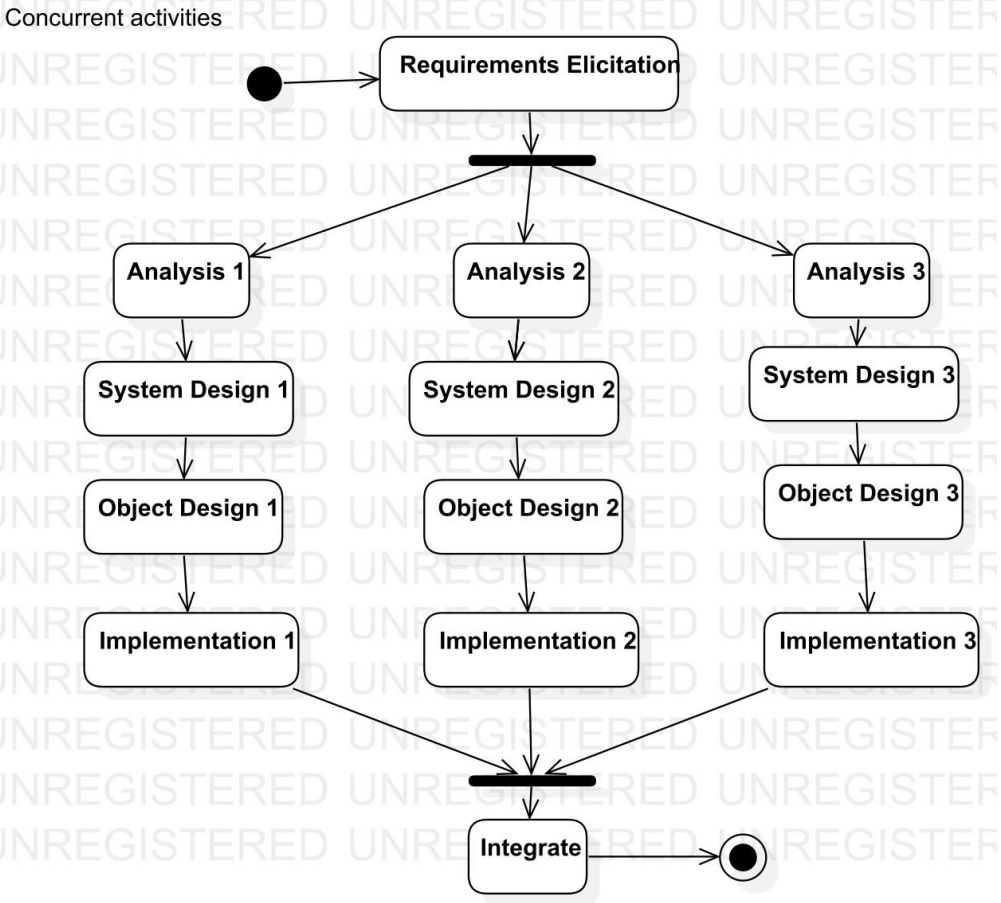
2. Add exception handling to the activity diagram you developed in Exercise 7-1. Consider at least three exceptions (e.g., delivery person wrote down wrong address, deliver person brings wrong pizza, store out of anchovies).



3. Consider the software development activities which we described in 7-1. Draw an activity diagram depicting these activities, assuming they are executed strictly sequentially. Draw a second activity diagram depicting the same activities occurring incrementally (i.e., one part of the system is analyzed, designed, implemented, and tested completely before the next part of the system is developed). Draw a third activity diagram depicting the same activities occurring concurrently.




Incremental activities
Requirements Elicitation
Analysis 1
System Design 1
Object Design 1
Implementation 1
Analysis 2
System Design 2
Object Design 2
Implementation 2
Analysis 3
System Design 3
Object Design 3
Implementation 3


Concurrent activities
Requirements Elicitation
Analysis 1
System Design 1
Object Design 1
Implementation 1
Analysis 2
System Design 2
Object Design 2
Implementation 2
Analysis 3
System Design 3
Object Design 3
Implementation 3
Integrate

# LAB : 8

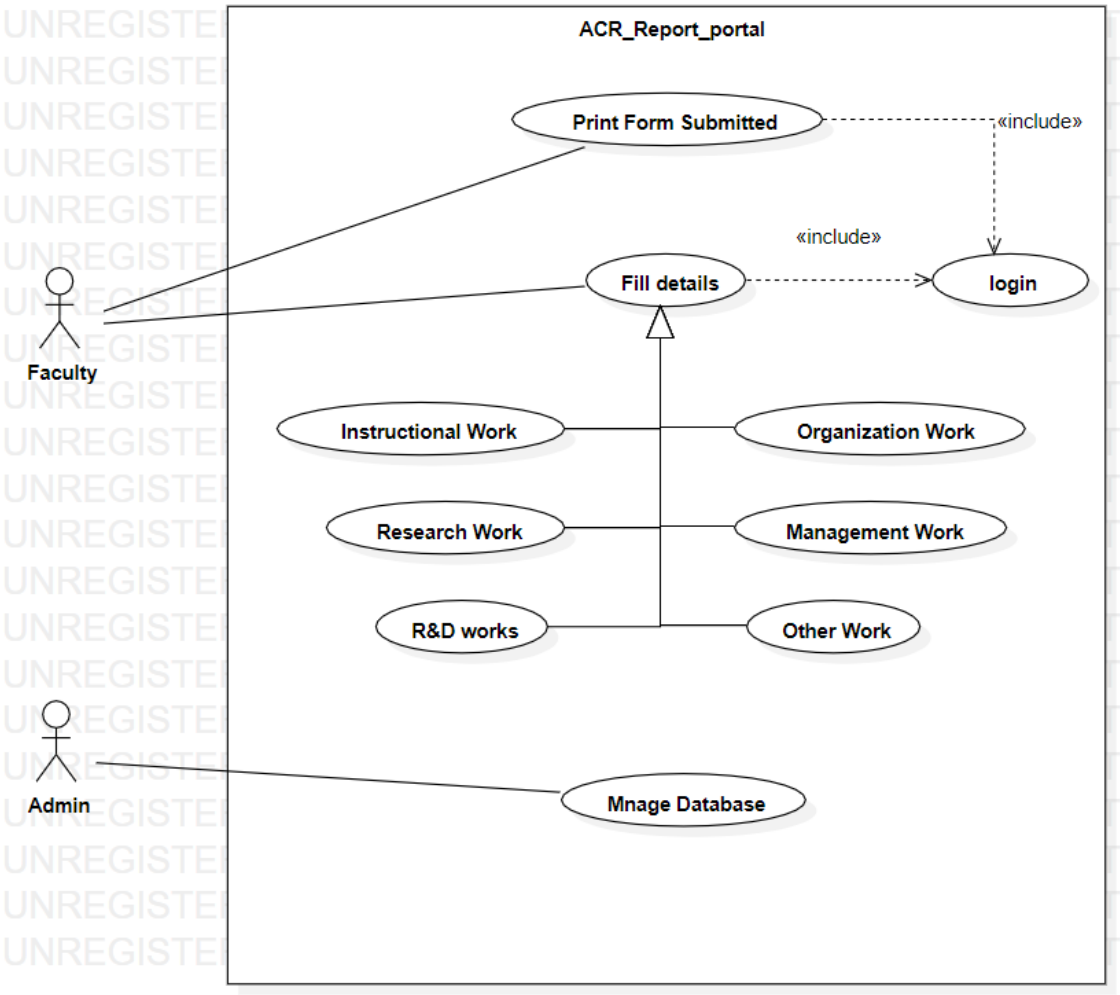
**TOPIC :** UML Diagrams of Minor Project – Online ACR Portal

## Requrements :

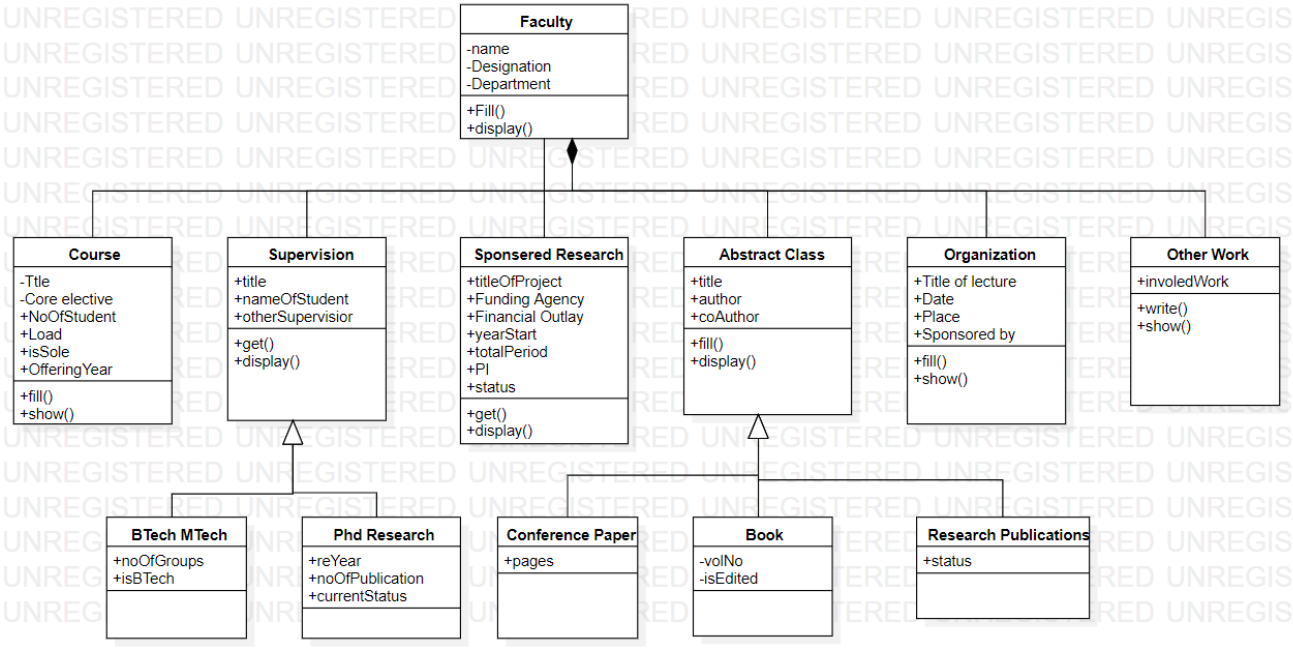
(a) Windows PC (Windows 7/8/10) / Mac

(b) Star UML Tool

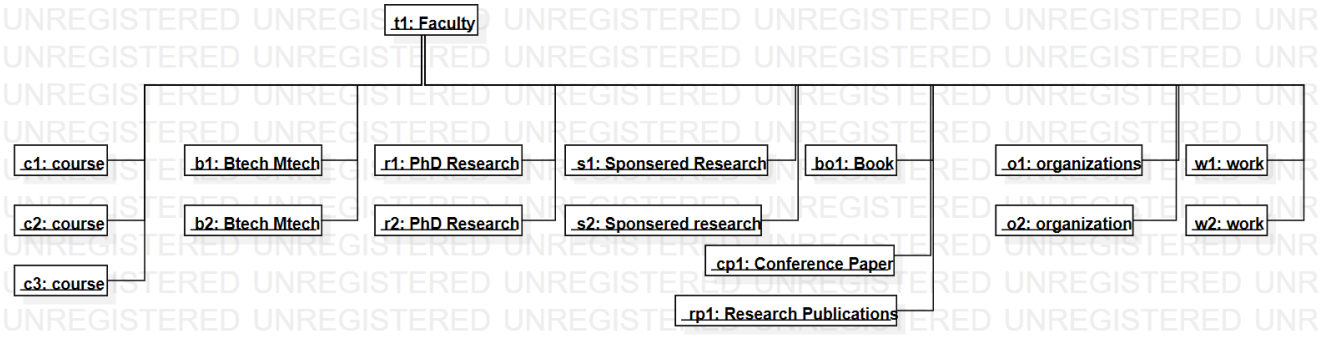
## Use Case Diagram :



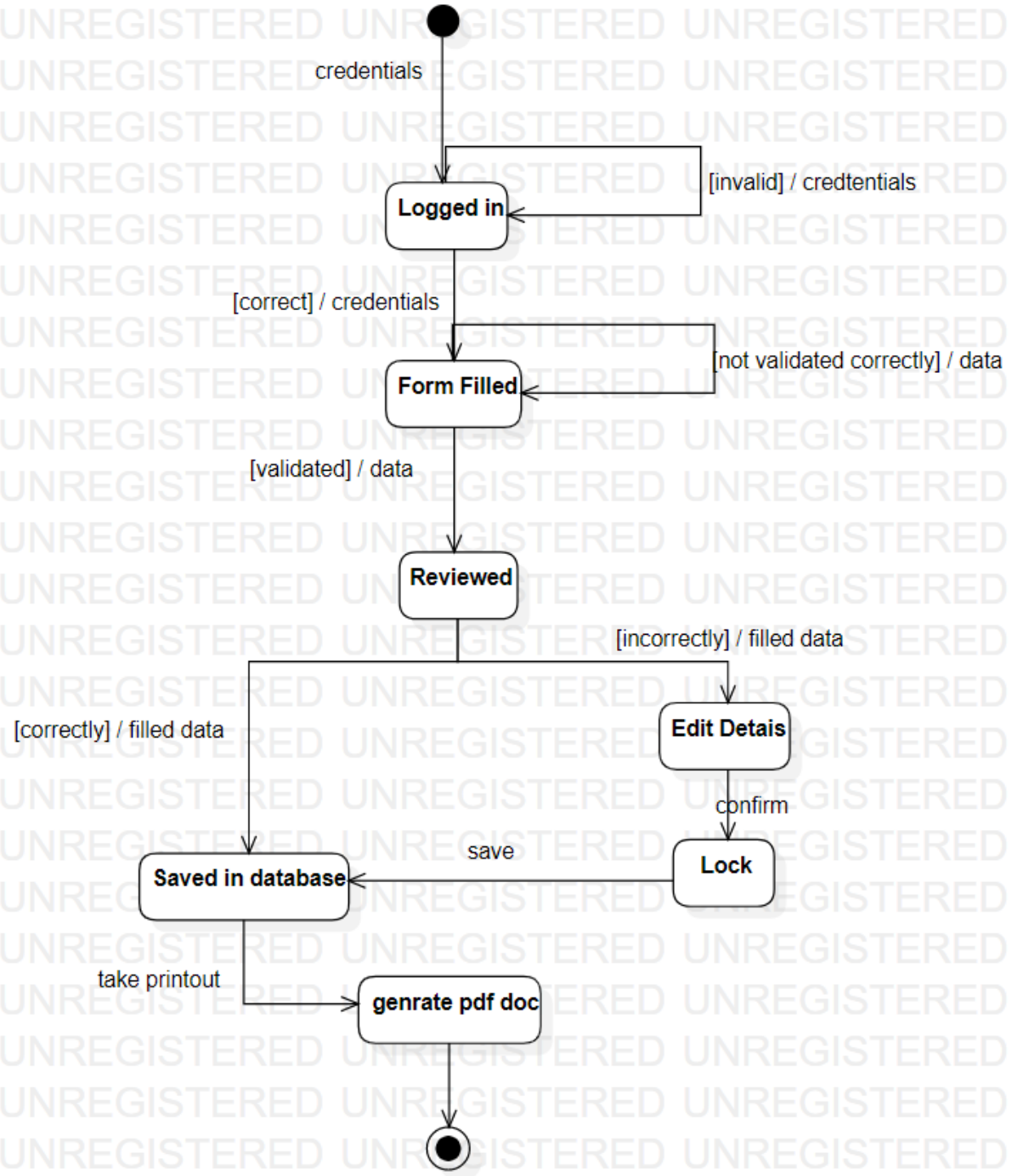
## Class Diagram :



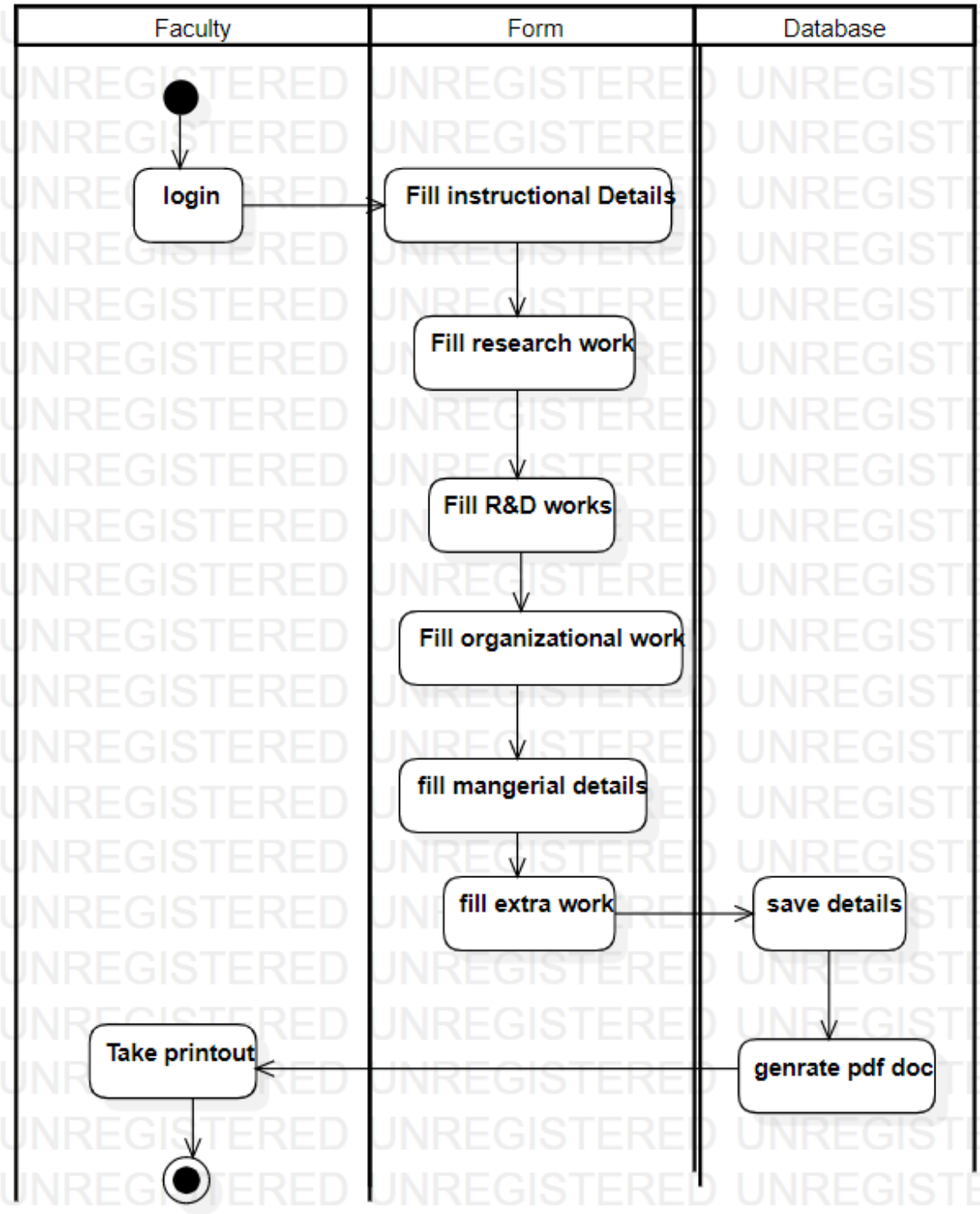

## Object Diagram :



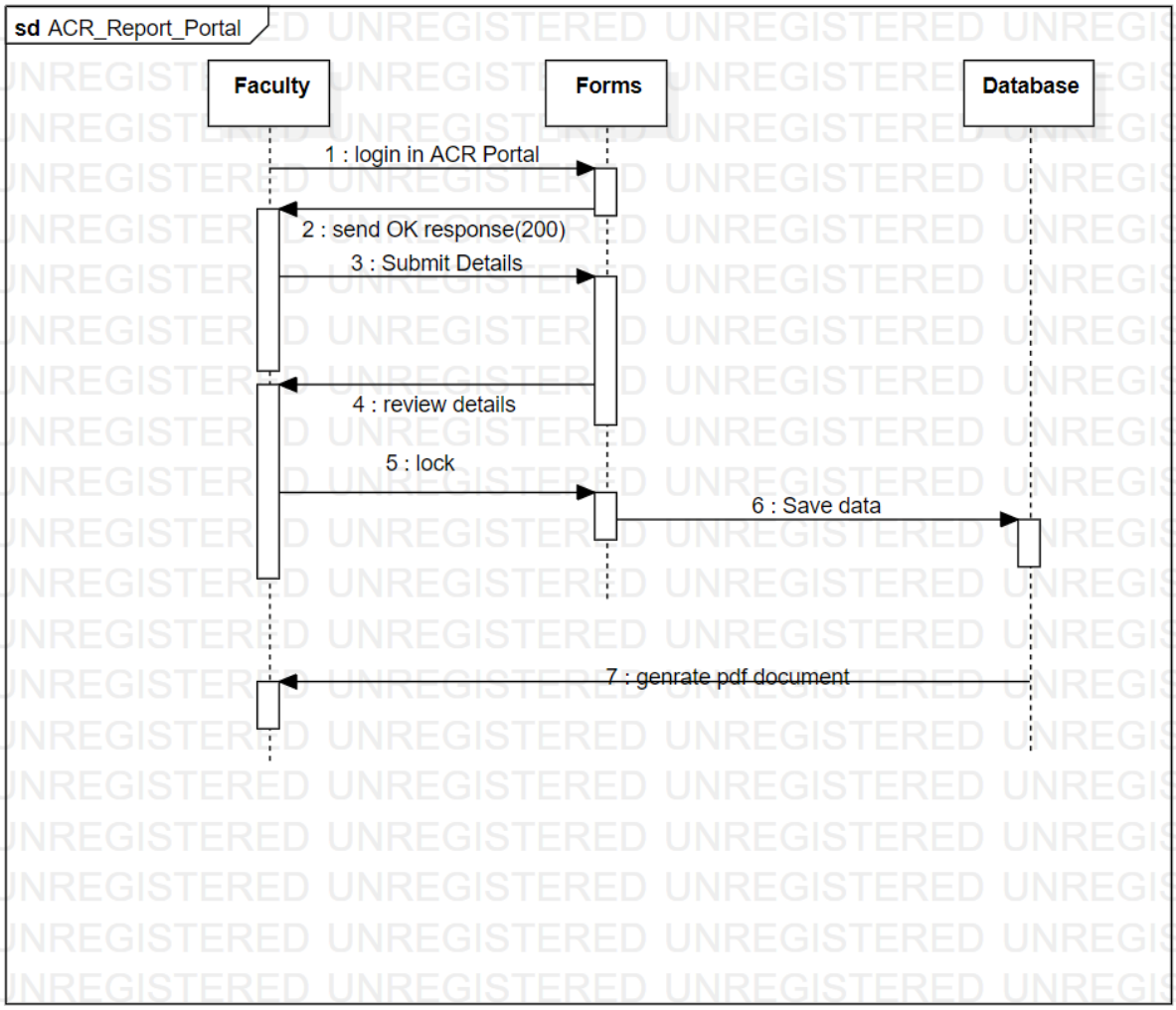
## Statechart Diagram :



## Activity Diagram :



## Sequence Diagram :



## Colaboration Diagram :