ASSIGNMENT1

UML DIAGRAM THEORY

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously putting effort to make a truly industry standard.

* UML stands for **U**nified **M**odeling **L**anguage.
* UML is different from the other common programming languages like C++, Java, COBOL etc.
* UML is a pictorial language used to make software blue prints.

So UML can be described as a general purpose visual modeling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

UML diagrams are not only made for developers but also for business users, common people and anybody interested to understand the system. The system can be a software or non software. So it must be clear that UML is not a development method rather it accompanies with processes to make a successful system.

At the conclusion the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today.s complex environment.

A conceptual model of UML:

To understand conceptual model of UML first we need to clarify *What is a conceptual model?* and*Why a conceptual model is at all required?*

* A conceptual model can be defined as a model which is made of concepts and their relationships.
* A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

# BASIC NOTATIONS

# Structural Things:

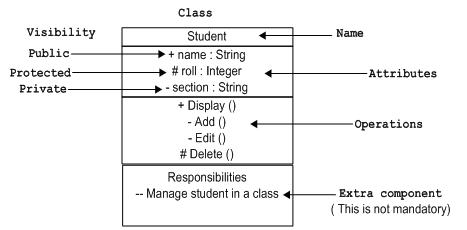
Graphical notations used in structural things are the most widely used in UML. These are considered as the nouns of UML models. Following are the list of structural things.

* Classes
* Interface
* Collaboration
* Use case
* Active classes
* Components
* Nodes

## Class Notation:

UML *class* is represented by the diagram shown below. The diagram is divided into four parts.

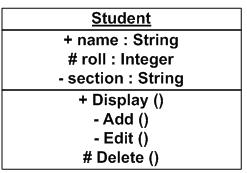
* The top section is used to name the class.
* The second one is used to show the attributes of the class.
* The third section is used to describe the operations performed by the class.
* The fourth section is optional to show any additional components.



Classes are used to represent objects. Objects can be anything having properties and responsibility.

## Object Notation:

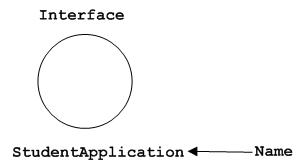
The *object* is represented in the same way as the class. The only difference is the *name* which is underlined as shown below.



As object is the actual implementation of a class which is known as the instance of a class. So it has the same usage as the class.

## Interface Notation:

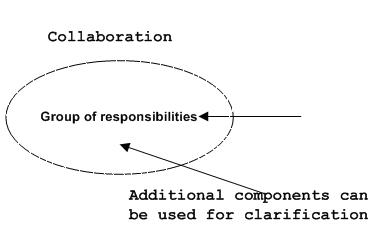
Interface is represented by a circle as shown below. It has a name which is generally written below the circle.



Interface is used to describe functionality without implementation. Interface is the just like a template where you define different functions not the implementation. When a class implements the interface it also implements the functionality as per the requirement.

## Collaboration Notation:

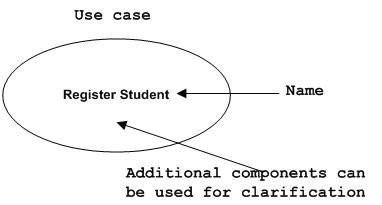
Collaboration is represented by a dotted eclipse as shown below. It has a name written inside the eclipse.



Collaboration represents responsibilities. Generally responsibilities are in a group.

## Use case Notation:

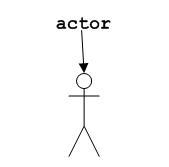
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.



Use case is used to capture high level functionalities of a system.

## Actor Notation:

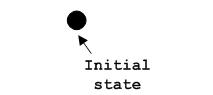
An actor can be defined as some internal or external entity that interacts with the system.



Actor is used in a use case diagram to describe the internal or external entities.

## Initial State Notation:

Initial state is defined to show the start of a process. This notation is used in almost all diagrams.



The usage of Initial State Notation is to show the starting point of a process.

## Final State Notation:

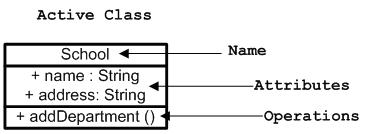
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.



The usage of Final State Notation is to show the termination point of a process.

## Active class Notation:

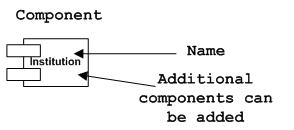
Active class looks similar to a class with a solid border. Active class is generally used to describe concurrent behaviour of a system.



Active class is used to represent concurrency in a system.

## Component Notation:

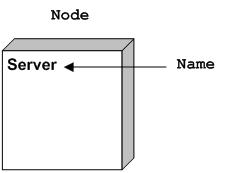
A component in UML is shown as below with a name inside. Additional elements can be added wherever required.



Component is used to represent any part of a system for which UML diagrams are made.

## Node Notation:

A node in UML is represented by a square box as shown below with a name. A node represents a physical component of the system.



Node is used to represent physical part of a system like server, network etc.

# Behavioural Things:

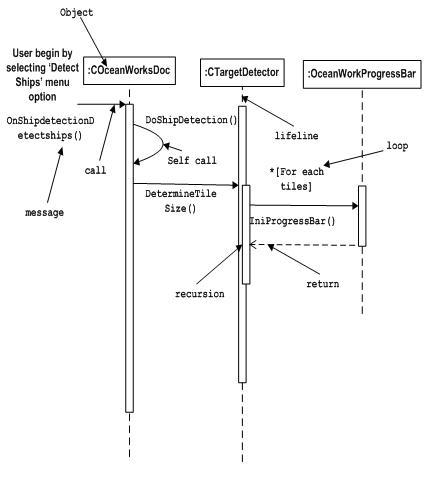
Dynamic parts are one of the most important elements in UML. UML has a set of powerful features to represent the dynamic part of software and non software systems. These features include*interactions* and *state machines*.

Interactions can be of two types:

* Sequential (Represented by sequence diagram)
* Collaborative (Represented by collaboration diagram)

## Interaction Notation:

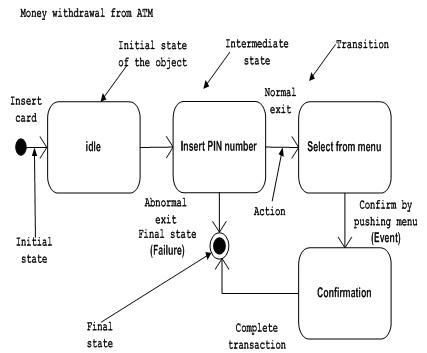
Interaction is basically message exchange between two UML components. The following diagram represents different notations used in an interaction.



Interaction is used to represent communication among the components of a system.

## State machine Notation:

State machine describes the different states of a component in its life cycle. The notations are described in the following diagram.



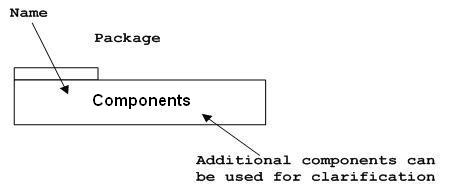
State machine is used to describe different states of a system component. The state can be active, idle or any other depending upon the situation.

# Grouping Things:

Organizing the UML models are one of the most important aspects of the design. In UML there is only one element available for grouping and that is package.

## Package Notation:

Package notation is shown below and this is used to wrap the components of a system.

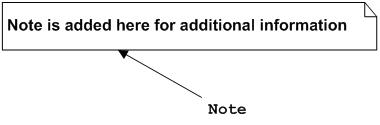


# Annotational Things:

In any diagram explanation of different elements and their functionalities are very important. So UML has *notes* notation to support this requirement.

## Note Notation:

This notation is shown below and they are used to provide necessary information of a system.



# Relationships

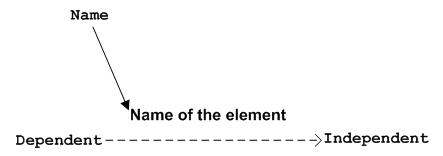
A model is not complete unless the relationships between elements are described properly. The*Relationship* gives a proper meaning to an UML model. Following are the different types of relationships available in UML.

* Dependency
* Association
* Generalization
* Extensibility

## Dependency Notation:

Dependency is an important aspect in UML elements. It describes the dependent elements and the direction of dependency.

Dependency is represented by a dotted arrow as shown below. The arrow head represents the independent element and the other end the dependent element.

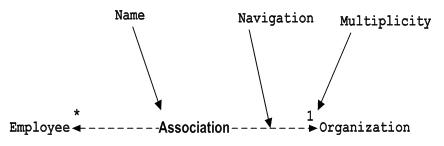


Dependency is used to represent dependency between two elements of a system.

## Association Notation:

Association describes how the elements in an UML diagram are associated. In simple word it describes how many elements are taking part in an interaction.

Association is represented by a dotted line with (without) arrows on both sides. The two ends represent two associated elements as shown below. The multiplicity is also mentioned at the ends (1, \* etc) to show how many objects are associated.



Association is used to represent the relationship between two elements of a system.

## Generalization Notation:

Generalization describes the inheritance relationship of the object oriented world. It is parent and child relationship.

Generalization is represented by an arrow with hollow arrow head as shown below. One end represents the parent element and the other end child element.

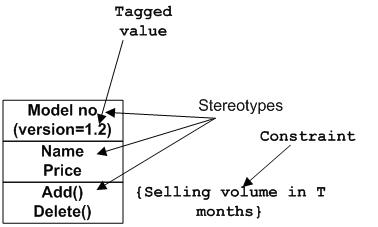
Generalization Notation

Generalization is used to describe parent-child relationship of two elements of a system.

## Extensibility Notation:

All the languages (programming or modeling) have some mechanism to extend its capabilities like syntax, semantics etc. UML is also having the following mechanisms to provide extensibility features.

* Stereotypes (Represents new elements)
* Tagged values (Represents new attributes)
* Constraints (Represents the boundaries)



Extensibility notations are used to enhance the power of the language. It is basically additional elements used to represent some extra behaviour of the system. These extra behaviours are not covered by the standard available notations.

STANDARD DIAGRAMS:

There are two broad caetgories of diagrams and then are again divided into sub-categories:

* Structural Diagrams
* Behavioral Diagrams

# Structural Diagrams:

The *structural diagrams* represent the static aspect of the system. These static aspects represent those parts of a diagram which forms the main structure and therefore stable.

These static parts are represents by classes, interfaces, objects, components and nodes. The four structural diagrams are:

* Class diagram
* Object diagram
* Component diagram
* Deployment diagram

## Class Diagram:

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations and collaboration.

Class diagrams basically represent the object oriented view of a system which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system.

Class diagram represents the object orientation of a system. So it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

## Object Diagram:

Object diagrams can be described as an instance of class diagram. So these diagrams are more close to real life scenarios where we implement a system.

Object diagrams are a set of objects and their relationships just like class diagrams and also represent the static view of the system.

The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from practical perspective.

## Component Diagram:

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces or collaborations.

So Component diagrams represent the implementation view of a system.

During design phase software artifacts (classes, interfaces etc) of a system are arranged in different groups depending upon their relationship. Now these groups are known as components.

Finally, component diagrams are used to visualize the implementation.

## Deployment Diagram:

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed.

Deployment diagrams are used for visualizing deployment view of a system. This is generally used by the deployment team.

***Note:****If the above descriptions and usages are observed carefully then it is very clear that all the diagrams are having some relationship with one another. Component diagrams are dependent upon the classes, interfaces etc which are part of class/object diagram. Again the deployment diagram is dependent upon the components which are used to make a component diagrams.*

# Behavioral Diagrams:

Any system can have two aspects, static and dynamic. So a model is considered as complete when both the aspects are covered fully.

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams:

* Use case diagram
* Sequence diagram
* Collaboration diagram
* Statechart diagram
* Activity diagram

## Use case Diagram:

Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system.

So use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

## Sequence Diagram:

A sequence diagram is an interaction diagram. From the name it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Interaction among the components of a system is very important from implementation and execution perspective.

So Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

## Collaboration Diagram:

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.

The purpose of collaboration diagram is similar to sequence diagram. But the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.

## Statechart Diagram:

Any real time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system.

Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc.

State chart diagram is used to visualize the reaction of a system by internal/external factors.

## Activity Diagram:

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent or branched.

Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

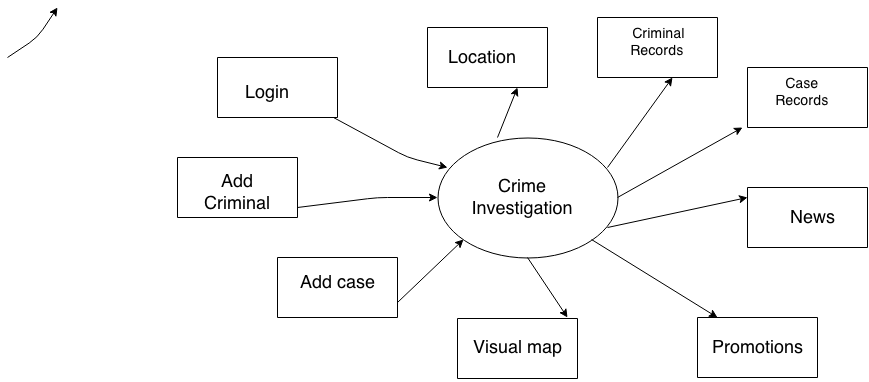
Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

***Note:****Dynamic nature of a system is very difficult to capture. So UML has provided features to capture the dynamics of a system from different angles. Sequence diagrams and collaboration diagrams are isomorphic so they can be converted from one another without losing any information. This is also true for statechart and activity diagram.*

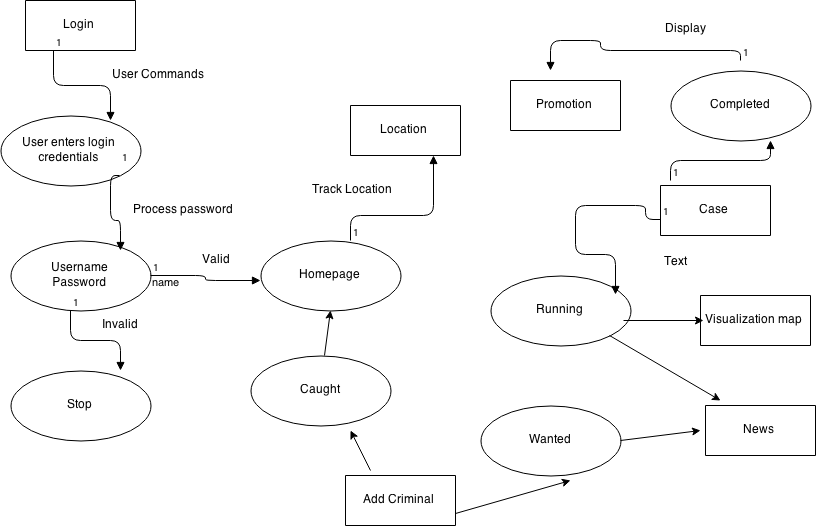
DATA DESIGN

DATA FLOW DIAGRAM

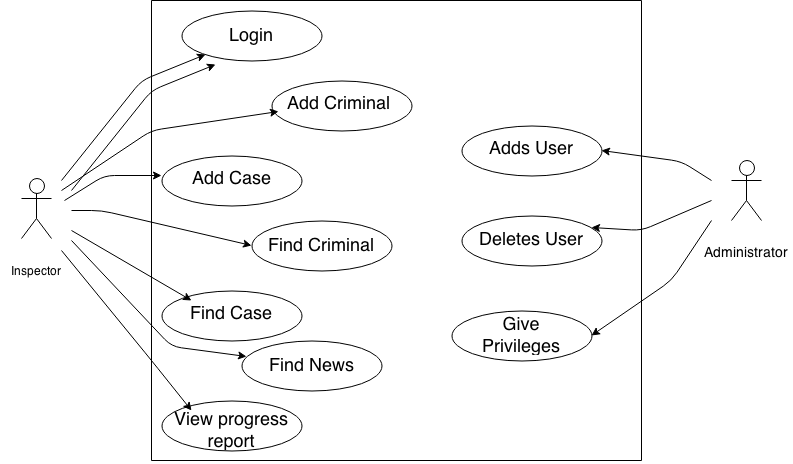
Level 0:



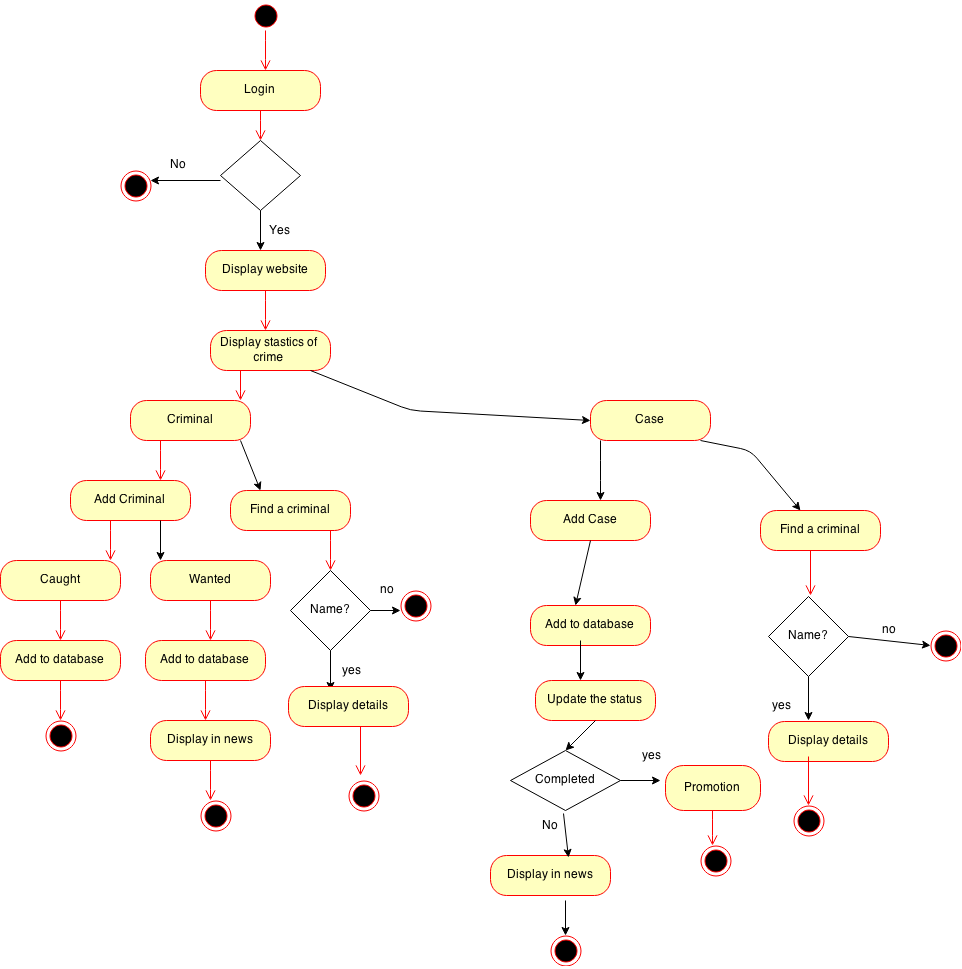
Level1:



Usecase diagram:



Activity Diagram:



Sequence Diagram:

