***UML Frequently Asked Questions***

1. **What are the Goals of UML?**

**Ans:** There are a number of goals for developing UML but the most important is to define some general purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use.

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 system. Thus it must be clear that UML is not a development method

rather it accompanies with processes to make it a successful system.

1. **Why uml for modeling?**

**Ans:**

* Use graphical notation to communicate more clearly than natural language (imprecise) and code(too detailed).
* Help acquire an overall view of a system.
* UML is *not* dependent on any one language or technology.
* The communication of the desired structure and behaviour of a system between analysts, architects, developers, stakeholders and users.

1. **What are the fundamental concepts of the object-oriented world ?**

**Ans:**

**Objects**: Objects represent an entity and the basic building block.

**Class:** Class is the blueprint of an object.

**Abstraction:** Abstraction represents the behavior of a real world entity.

**Encapsulation:** Encapsulation is the mechanism of binding the data together and hiding them from the outside world.

**Inheritance:** Inheritance is the mechanism of making new classes from existing ones.

**Polymorphism:** It defines the mechanism to exist in different forms.

1. **What is the Role of UML in OO Design?**

**Ans:** UML is a modeling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on modeling OO software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects.

If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects. Hence, the relation between OO design and UML is very important to understand.

The OO design is transformed into UML diagrams according to the requirement. Before understanding the UML in detail, the OO concept should be learned properly. Once the OO analysis and design is done, the next step is very easy. The input from OO analysis and design is the input to UML diagrams.

1. **Mention how many daigrams in UML ?**

**Ans:** UML includes the following nine diagrams.

**1) Structural Diagram**

-Class Diagram

-Object Diagram

-Component Diagram

-Deployment Diagram

**2) Behavioral Diagram**

-Use case Diagram

-Sequence Diagram

-Interaction Diagram

-StatechartDiagram

-Activity Diagram

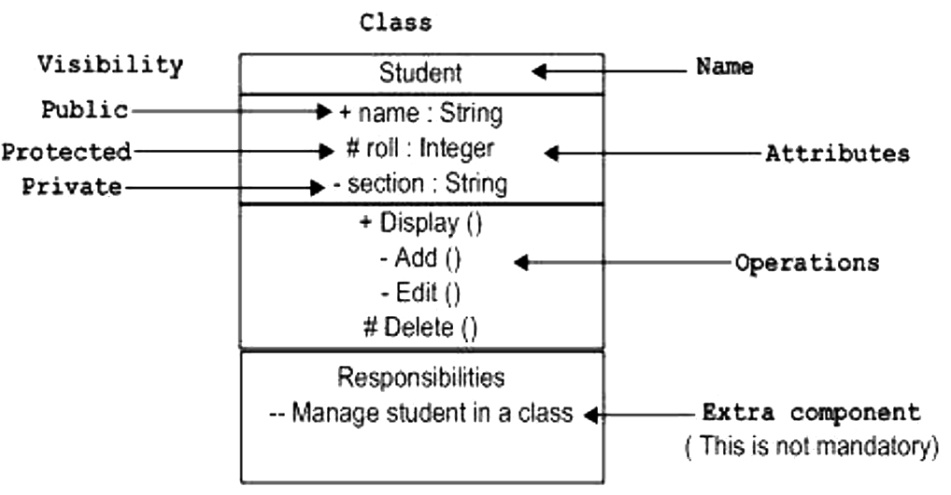
1. **Explain UML Notations ?**

**Ans:**

**Class Notation**

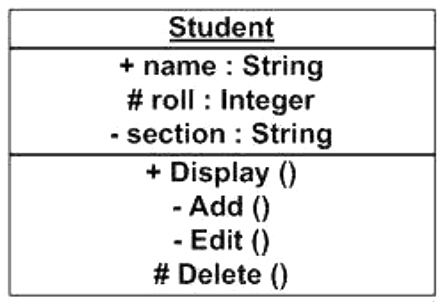
UML class is represented by the following figure. The diagram is divided into four parts.

* + 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.



**Object Notation**

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

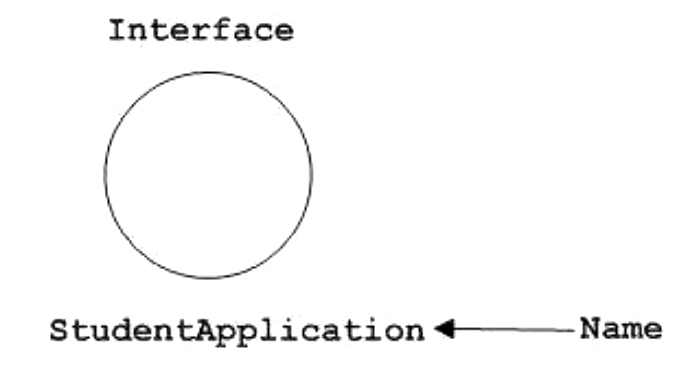


**Interface Notation**

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

Interface is used to describe the functionality without implementation.

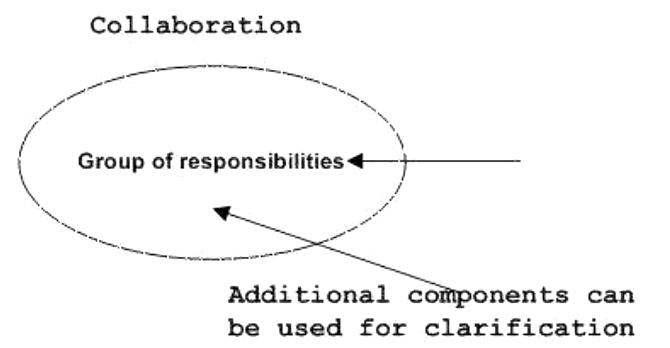
Interface is just like a template where you define different functions, not the implementation.



**Collaboration Notation**

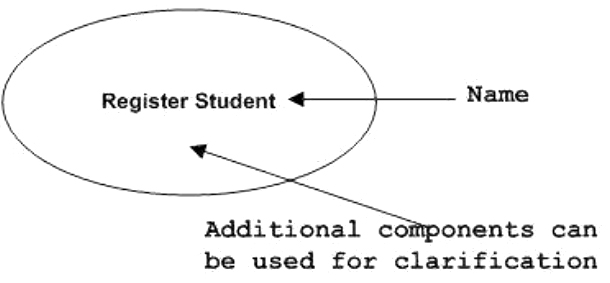
Collaboration is represented by a dotted eclipse as shown in the following figure. It has a name written inside the eclipse.

Collaboration represents responsibilities. Generally, responsibilities are in a group.



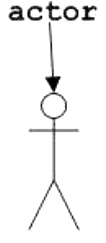
**Use Case Notation**

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.

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



**Final State Notation**

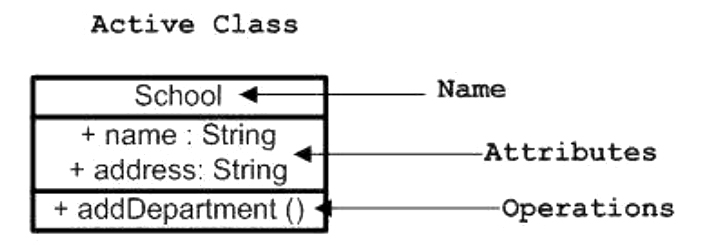
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.

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



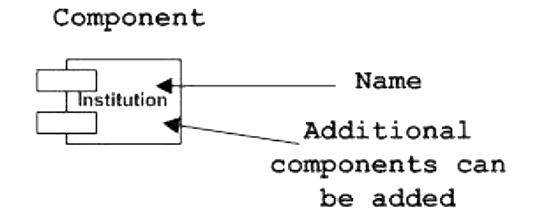
**Active Class Notation**

Active class looks similar to a class with a solid border. Active class is generally used to describe the concurrent behavior of a system. Active class is used to represent the concurrency in a system.



**Component Notation**

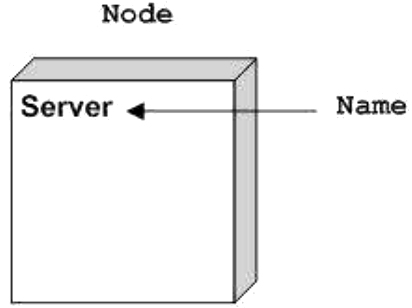
A component in UML is shown in the following figure with a name inside. Additional elements can be added wherever required. Component is used to represent any part of a system for which UML diagrams are made.



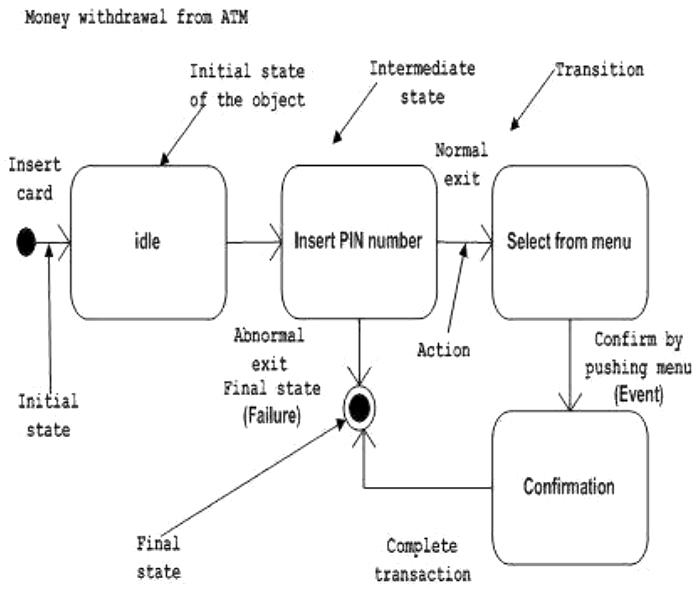
**Node Notation**

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

Node is used to represent the physical part of a system such as the server, network, etc.

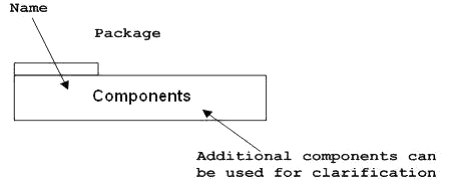


**State Machine Notation**

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

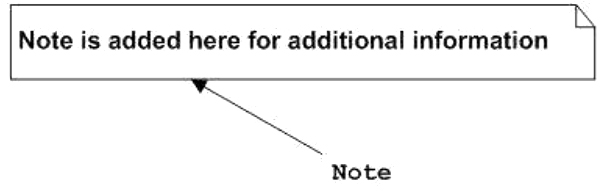
**Package Notation**

Package notation is shown in the following figure and is used to wrap the components of a system.



**Note Notation**

This notation is shown in the following figure. These notations are used to provide necessary information of a system.



**Dependency Notation**

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

Dependent ----------> Independent

**Association Notation**

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

Employee<----------- Association ----------> Organization

**Generalization Notation**

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

C:\Users\umr\Desktop\notatoons\ganeralization.jpg

1. **What is the Purpose of Class Diagrams?**

**Ans:** The purpose of the class diagram can be summarized as:

* + Analysis and design of the static view of an application.
  + Describe responsibilities of a system.
  + Base for component and deployment diagrams.
  + Forward and reverse engineering.

1. **How to Draw a Class Diagram? With an Example**

**Ans:** The following points should be remembered while drawing a class diagram:

* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified inadvance.
* Responsibility (attributes and methods) of each class should be clearly identified.
* For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
* Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

1. **Where to Use Class Diagrams?**

**Ans:** Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.

In a nutshell it can be said, class diagrams are used for:

* Describing the static view of the system.
* Showing the collaboration among the elements of the static view.
* Describing the functionalities performed by the system.
* Constructing software applications using object-oriented languages.

1. **What is the Purpose of Object Diagrams?**

**Ans:** The purpose of the object diagram can be summarized as:

* + Forward and reverse engineering
  + Object relationships of a system
  + Static view of an interaction
  + Understand object behavior and their relationship from a practical perspective

1. **How to Draw an Object Diagram? With an Example**

**Ans:**

* First, analyze the system and decide which instances have important data and association.
* Second, consider only those instances, which will cover the functionality.
* Third, make some optimization as the number of instances are unlimited.

Before drawing an object diagram, the following things should be remembered and

understood clearly:

* Object diagrams consist of objects.
* The link in object diagram is used to connect objects.
* Objects and links are the two elements used to construct an object diagram.

After this, the following things are to be decided before starting the construction of the

diagram:

* + The object diagram should have a meaningful name to indicate its purpose.
  + The most important elements are to be identified.
  + The association among objects should be clarified.
  + Values of different elements need to be captured to include in the object diagram.
  + Add proper notes at points where more clarity is required.

1. **Where to Use Object Diagrams?**

**Ans:** object diagrams are used for:

* + Making the prototype of a system.
  + Reverse engineering.
  + Modeling complex data structures.
  + Understanding the system from a practical perspective.

1. **How to Draw a Component Diagram? With an Example**

**Ans:** Before drawing a component diagram, the following artifacts are to be identified clearly:

 Files used in the system.

 Libraries and other artifacts relevant to the application.

 Relationships among the artifacts.

After identifying the artifacts, the following points need to be kept in mind.

 Use a meaningful name to identify the component for which the diagram is to be drawn.

 Prepare a mental layout before producing the using tools.

 Use notes for clarifying important points.

1. **Where to Use Component Diagrams?**

**Ans:**

* + Model the components of a system.
  + Model the database schema.
  + Model the executables of an application.
  + Model the system's source code

1. **How to Draw a Deployment Diagram? Explain with an Example.**

**Ans:** Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important as it controls the following parameters:

* + Performance
  + Scalability
  + Maintainability
  + Portability

Before drawing a deployment diagram, the following artifacts should be identified:

* Nodes
* Relationships among nodes

1. **Where to Use Deployment Diagrams?**

**Ans:** Deployment diagrams are mainly used by system engineers. These diagrams are used to describe the physical components (hardware), their distribution, and association. Deployment diagrams can be visualized as the hardware components/nodes on which the software components reside.

Software applications are developed to model complex business processes. Efficient software applications are not sufficient to meet the business requirements. Business requirements can be described as the need to support the increasing number of users,

quick response time, etc.

1. **Explain Sequence Diagram with an example?**

**Ans:** Sequence diagram is an Interaction diagram that shows how objects operate with one another & in what order

1. **Draw an Activity Diagram?**

**Ans:** Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

**Purpose of Activity Diagrams**

activity diagram is used to show message flow from one activity to another.

Before drawing an activity diagram, we should identify the following elements:

* Activities
* Association
* Conditions
* Constraints

