

**Tutorials Point, Simply Easy Learning**

1 | Page

**UML Tutorial**

**Tutorialspoint.com**

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

**UML was created by Object Management Group and UML 1.0 specification draft was proposed to the OMG in January 1997. This tutorial gives an initial push to start you with UML. For more detail kindly check tutorialspoint.com/uml**

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 Unified Modelling Language.

• 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 modelling 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 is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization UML is become an OMG (Object Management Group) standard.

**Goals of UML:**

A picture is worth a thousand words, this absolutely fits while discussing about UML. Object oriented concepts were introduced much earlier than UML. So at that time there were no standard methodologies to organize and consolidate the object oriented development. At that point of time UML came into picture.

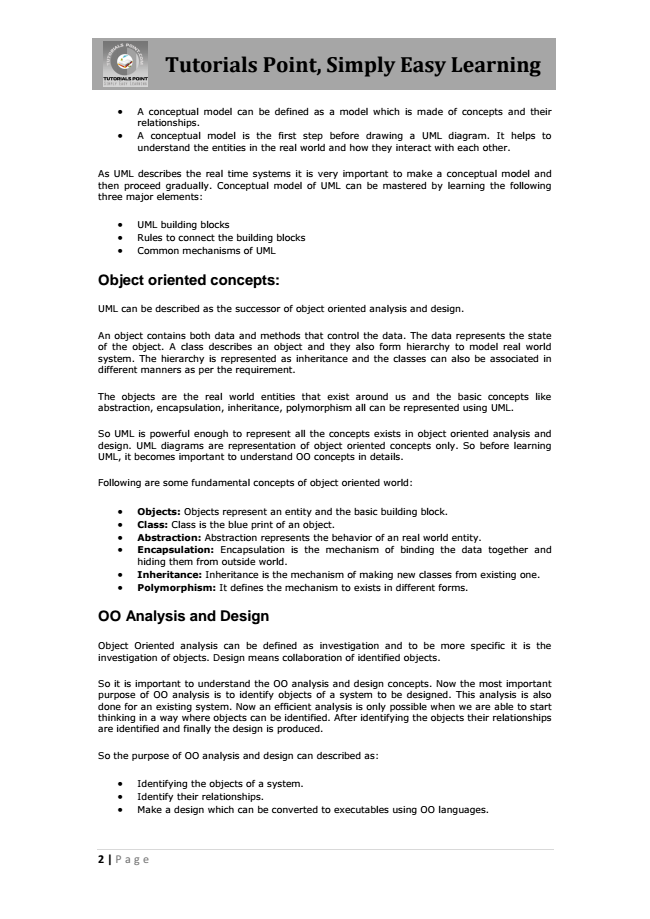
There are a number of goals for developing UML but the most important is to define some general purpose modelling language which all modelers can use and also it needs to be made simple to understand and use.

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. 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 modelling 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?*

**

**Tutorials Point, Simply Easy Learning**

• 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.

As UML describes the real time systems it is very important to make a conceptual model and then proceed gradually. Conceptual model of UML can be mastered by learning the following three major elements:

• UML building blocks

• Rules to connect the building blocks

• Common mechanisms of UML

**Object oriented concepts:**

UML can be described as the successor of object oriented analysis and design.

An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form hierarchy to model real world system. The hierarchy is represented as inheritance and the classes can also be associated in different manners as per the requirement.

The objects are the real world entities that exist around us and the basic concepts like abstraction, encapsulation, inheritance, polymorphism all can be represented using UML.

So UML is powerful enough to represent all the concepts exists in object oriented analysis and design. UML diagrams are representation of object oriented concepts only. So before learning UML, it becomes important to understand OO concepts in details.

Following are some fundamental concepts of object oriented world:

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

• Class: Class is the blue print of an object.

• Abstraction: Abstraction represents the behavior of an real world entity.

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

• Inheritance: Inheritance is the mechanism of making new classes from existing one.

• Polymorphism: It defines the mechanism to exists in different forms.

**OO Analysis and Design**

Object Oriented analysis can be defined as investigation and to be more specific it is the investigation of objects. Design means collaboration of identified objects.

So it is important to understand the OO analysis and design concepts. Now the most important purpose of OO analysis is to identify objects of a system to be designed. This analysis is also done for an existing system. Now an efficient analysis is only possible when we are able to start thinking in a way where objects can be identified. After identifying the objects their relationships are identified and finally the design is produced.

So the purpose of OO analysis and design can described as:

• Identifying the objects of a system.

• Identify their relationships.

• Make a design which can be converted to executables using OO languages.

2 | Page



**Tutorials Point, Simply Easy Learning**

There are three basic steps where the OO concepts are applied and implemented. The steps can be defined as

OO Analysis --> OO Design --> OO implementation using OO languages

Now the above three points can be described in details:

• During object oriented analysis the most important purpose is to identify objects and describing them in a proper way. If these objects are identified efficiently then the next job of design is easy. The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed. When these responsibilities are collaborated the purpose of the system is fulfilled.

• The second phase is object oriented design. During this phase emphasis is given upon the requirements and their fulfilment. In this stage the objects are collaborated according to their intended association. After the association is complete the design is also complete.

• The third phase is object oriented implementation. In this phase the design is implemented using object oriented languages like Java, C++ etc.

**Role of UML in OO design:**

UML is a modelling language used to model software and non software systems. Although UML is used for non software systems the emphasis is on modelling object oriented software applications. Most of the UML diagrams discussed so far are used to model different aspects like static, dynamic etc. Now what ever 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.

So 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 details the OO concepts should be learned properly. Once the OO analysis and design is done the next step is very easy. The input from the OO analysis and design is the input to the UML diagrams.

**UML Building Blocks:**

As UML describes the real time systems it is very important to make a conceptual model and then proceed gradually. Conceptual model of UML can be mastered by learning the following three major elements:

• UML building blocks

• Rules to connect the building blocks

• Common mechanisms of UML

This chapter describes all the UML building blocks. The building blocks of UML can be defined as:

• Things

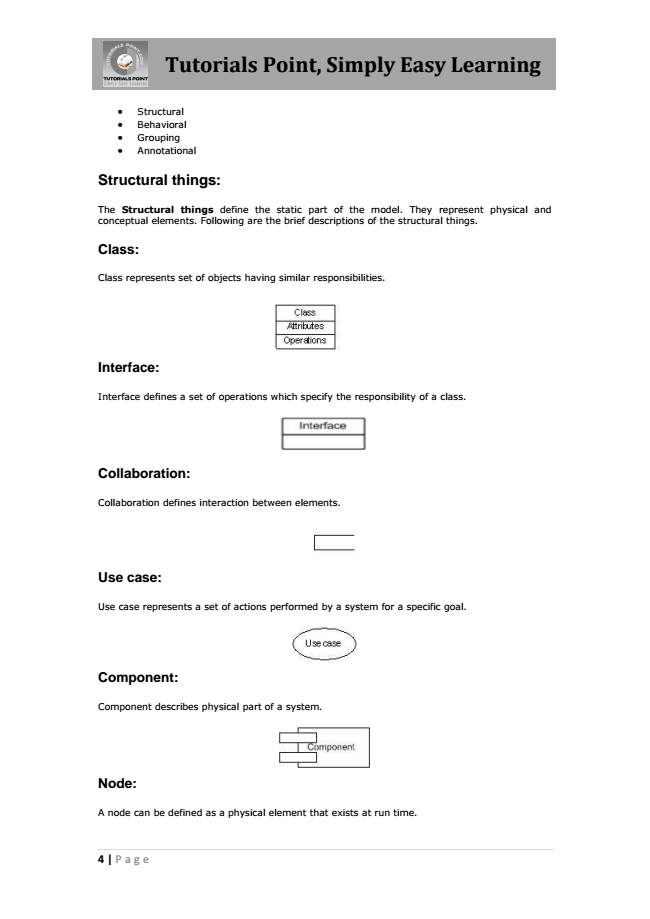
• Relationships

• Diagrams

**(1) Things:**

Things are the most important building blocks of UML. Things can be:

3 | Page



**Tutorials Point, Simply Easy Learning**

• Structural

• Behavioral

• Grouping

• Annotational

**Structural things:**

The Structural things define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

**Class:**

Class represents set of objects having similar responsibilities.

**Interface:**

Interface defines a set of operations which specify the responsibility of a class.

**Collaboration:**

Collaboration defines interaction between elements.

**Use case:**

Use case represents a set of actions performed by a system for a specific goal.

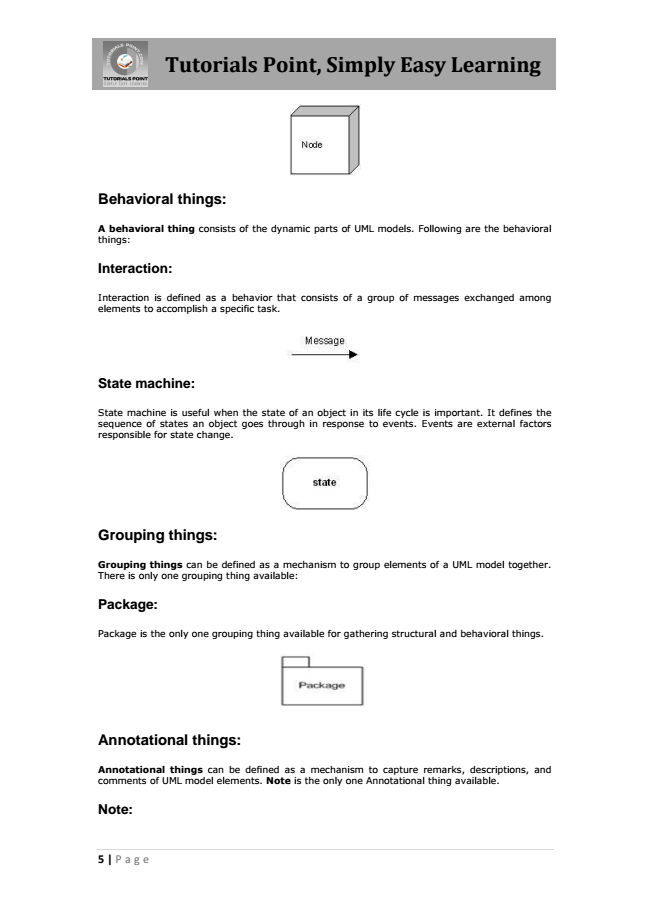
**Component:**

Component describes physical part of a system.

**Node:**

A node can be defined as a physical element that exists at run time.

4 | Page



**Tutorials Point, Simply Easy Learning**

**Behavioral things:**

A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:

**Interaction:**

Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

**State machine:**

State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.

**Grouping things:**

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

**Package:**

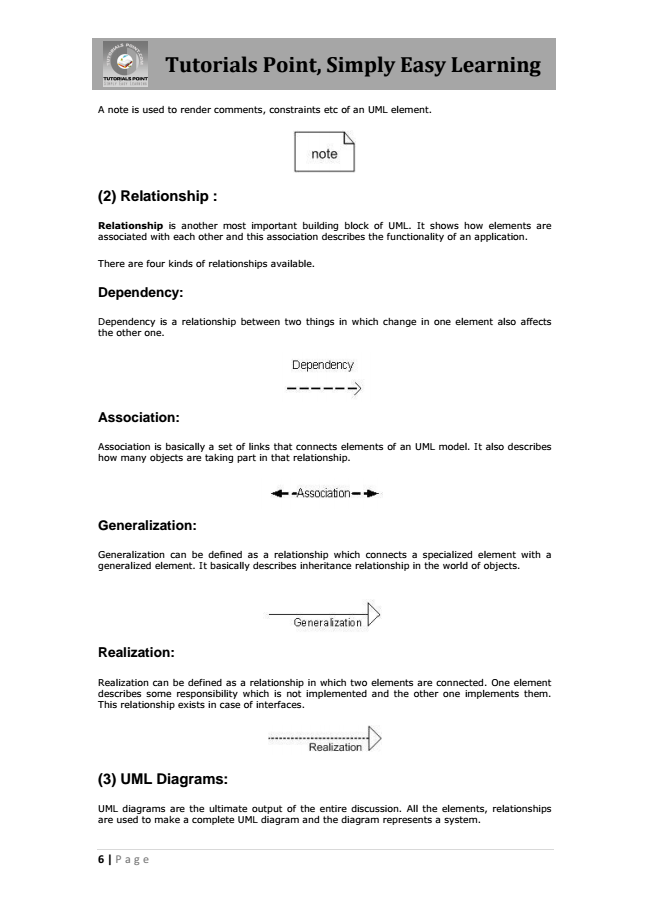
Package is the only one grouping thing available for gathering structural and behavioral things.

**Annotational things:**

Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note is the only one Annotational thing available.

**Note:**

5 | Page



**Tutorials Point, Simply Easy Learning**

A note is used to render comments, constraints etc of an UML element.

**(2) Relationship :**

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

**Dependency:**

Dependency is a relationship between two things in which change in one element also affects the other one.

**Association:**

Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

**Generalization:**

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

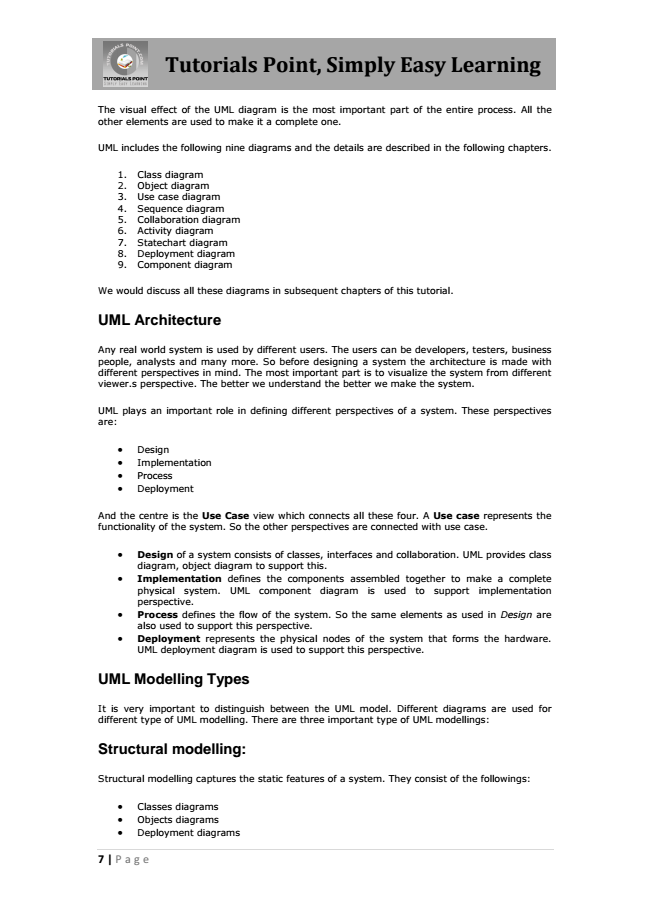
**Realization:**

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.

**(3) UML Diagrams:**

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

6 | Page



**Tutorials Point, Simply Easy Learning**

The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.

UML includes the following nine diagrams and the details are described in the following chapters.

1. Class diagram 2. Object diagram 3. Use case diagram 4. Sequence diagram 5. Collaboration diagram 6. Activity diagram 7. Statechart diagram 8. Deployment diagram 9. Component diagram

We would discuss all these diagrams in subsequent chapters of this tutorial.

**UML Architecture**

Any real world system is used by different users. The users can be developers, testers, business people, analysts and many more. So before designing a system the architecture is made with different perspectives in mind. The most important part is to visualize the system from different viewer.s perspective. The better we understand the better we make the system.

UML plays an important role in defining different perspectives of a system. These perspectives are:

• Design

• Implementation

• Process

• Deployment

And the centre is the Use Case view which connects all these four. A Use case represents the functionality of the system. So the other perspectives are connected with use case.

• Design of a system consists of classes, interfaces and collaboration. UML provides class diagram, object diagram to support this.

• Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support implementation perspective.

• Process defines the flow of the system. So the same elements as used in Design are also used to support this perspective.

• Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.

**UML Modelling Types**

It is very important to distinguish between the UML model. Different diagrams are used for different type of UML modelling. There are three important type of UML modellings:

**Structural modelling:**

Structural modelling captures the static features of a system. They consist of the followings:

• Classes diagrams

• Objects diagrams

• Deployment diagrams

7 | Page



**Tutorials Point, Simply Easy Learning**

• Package diagrams

• Composite structure diagram

• Component diagram

Structural model represents the framework for the system and this framework is the place where all other components exist. So the class diagram, component diagram and deployment diagrams are the part of structural modelling. They all represent the elements and the mechanism to assemble them.

But the structural model never describes the dynamic behavior of the system. Class diagram is the most widely used structural diagram.

**Behavioral Modelling:**

Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modelling shows the dynamic nature of the system. They consist of the following:

• Activity diagrams

• Interaction diagrams

• Use case diagrams

All the above show the dynamic sequence of flow in a system.

**Architectural Modelling:**

Architectural model represents the overall framework of the system. It contains both structural and behavioral elements of the system. Architectural model can be defined as the blue print of the entire system. Package diagram comes under architectural modelling.

**UML Basic Notations**

UML is popular for its diagrammatic notations. We all know that UML is for visualizing, specifying, constructing and documenting the components of software and non software systems. Here the Visualization is the most important part which needs to be understood and remembered by heart.

UML notations are the most important elements in modelling. Efficient and appropriate use of notations is very important for making a complete and meaningful model. The model is useless unless its purpose is depicted properly.

So learning notations should be emphasized from the very beginning. Different notations are available for things and relationships. And the UML diagrams are made using the notations of things and relationships. Extensibility is another important feature which makes UML more powerful and flexible.

The chapter describes the UML Basic Notations in more details. This is just an extension to the UML buildling block section I have discussed in previous chapter.

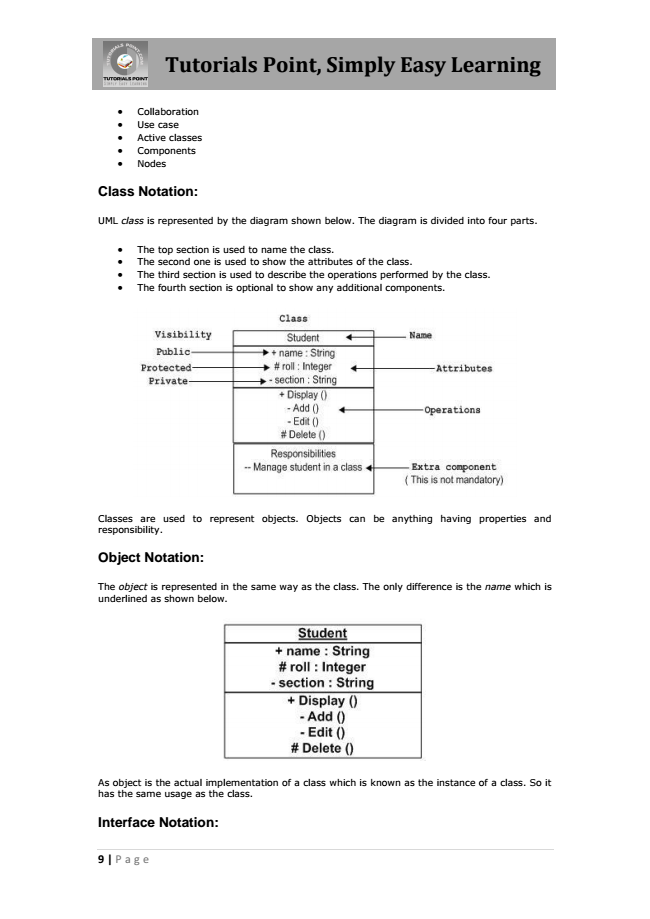
**Structural Things:**

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

8 | Page



**Tutorials Point, Simply Easy Learning**

• 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.

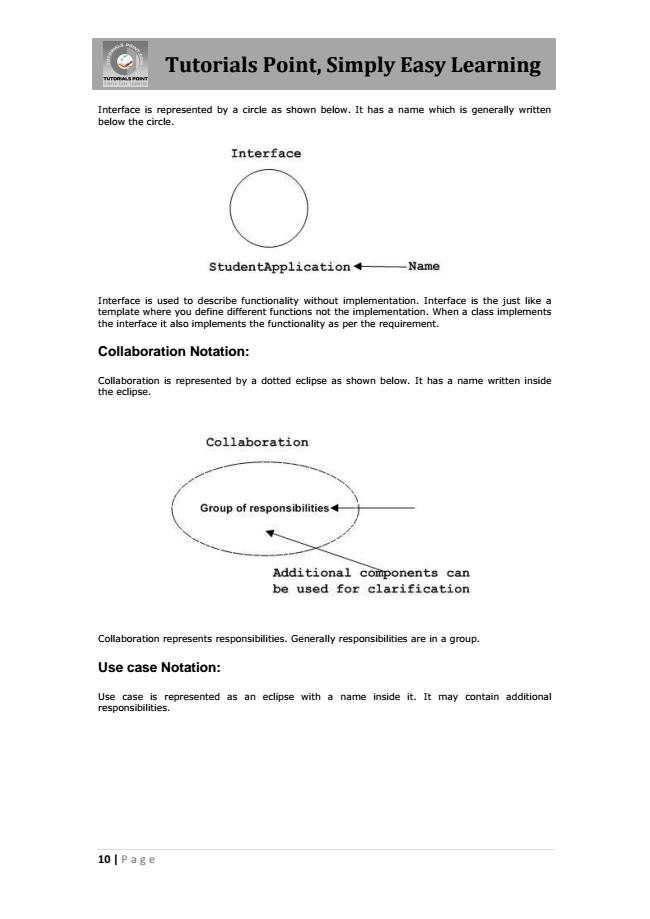
**Object Notation:**

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:**

9 | Page



**Tutorials Point, Simply Easy Learning**

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.

10 | Page