# UML Diagram:

A UML (Unified Modeling Language) diagram is a graphical representation of a system, process, or object-oriented program, using standardized symbols and notation. UML diagrams are used to visualize, design, and document the structure and behavior of software systems. UML diagrams come in several different types, including class diagrams, use case diagrams, activity diagrams, sequence diagrams, and more. Each type of UML diagram is used to represent different aspects of the system, and they are often used in combination to provide a comprehensive view of the system being modeled. UML diagrams are widely used in software engineering and other fields related to system design and development.

## Types:

UML (Unified Modeling Language) can be divided into two broad categories of diagrams: Structural UML diagrams and Behavioral UML diagrams.

### Structural UML diagrams:

Structural UML diagrams are used to represent the static structure of a system. They focus on the physical and conceptual components of a system and how they relate to each other. Examples of Structural UML diagrams include:

#### Class diagram -

Used to describe the structure of a system by showing classes, attributes, methods, and their relationships.

#### Object diagram -

Used to show a snapshot of the objects in a system at a particular point in time and their relationships.

#### Component diagram -

Used to represent the physical components of a system, including libraries, executables, files, and more.

#### Deployment diagram -

Used to show the hardware and software components of a system and their relationships.

### Behavioral UML diagrams:

Behavioral UML diagrams are used to represent the dynamic behavior of a system. They focus on the interactions between components and how the system behaves in response to different events. Examples of Behavioral UML diagrams include:

#### Use case diagram -

Used to describe the interactions between actors and a system to accomplish a goal or task.

#### Activity diagram -

Used to describe the flow of activities in a system, such as business processes or workflows.

#### State diagram -

Used to represent the behavior of an object or system in response to different events and conditions.

#### Sequence diagram -

Used to describe the interactions between objects in a system, showing the messages that are passed between them over time.

# Class Diagram:

A class diagram is a type of UML (Unified Modeling Language) diagram that represents the structure of a system by showing classes, their attributes, methods, and relationships. It is used to describe the static structure of a system, and it is one of the most widely used UML diagrams in software engineering.

In a class diagram, each class is represented as a rectangle, with the name of the class at the top. The attributes of the class are listed below the name, and the methods are listed below the attributes. The attributes represent the data or properties of the class, while the methods represent the operations that can be performed on the class.

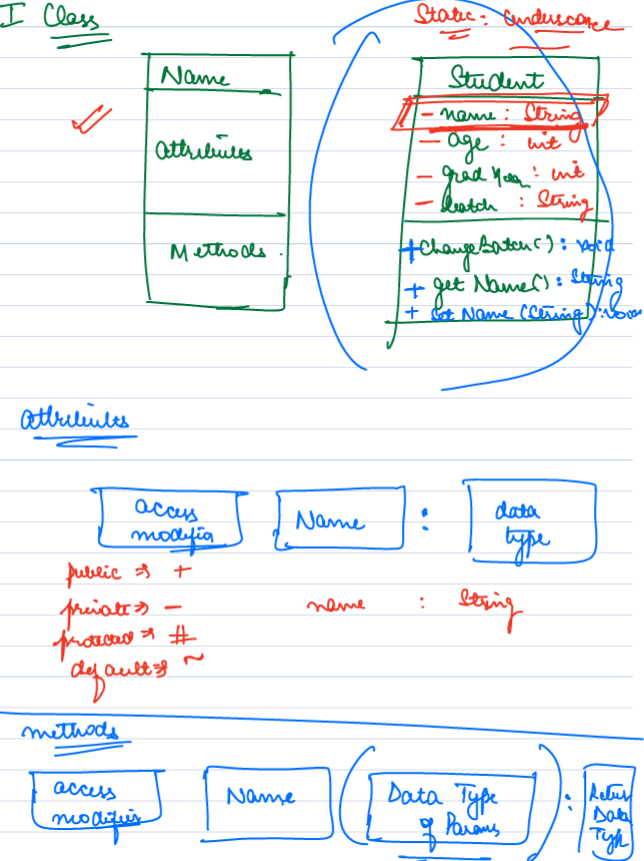
Classes in a class diagram are connected by various types of relationships, including association, aggregation, and inheritance. The relationships between classes are represented by lines between the rectangles.

An association relationship shows how classes are connected to each other, and it can be one-to-one, one-to-many, or many-to-many. An aggregation relationship represents a "has-a" relationship between classes, where one class is composed of other classes. Inheritance represents an "is-a" relationship between classes, where one class inherits the properties and methods of another class.

Class diagrams are essential for designing and communicating the structure of a software system. They are used in various stages of software development, including analysis, design, and implementation.

### Class:

In a class diagram, a class is represented pictorially as a rectangle. The rectangle is divided into three parts, with the top part containing the name of the class, the middle part containing the attributes of the class, and the bottom part containing the methods of the class.



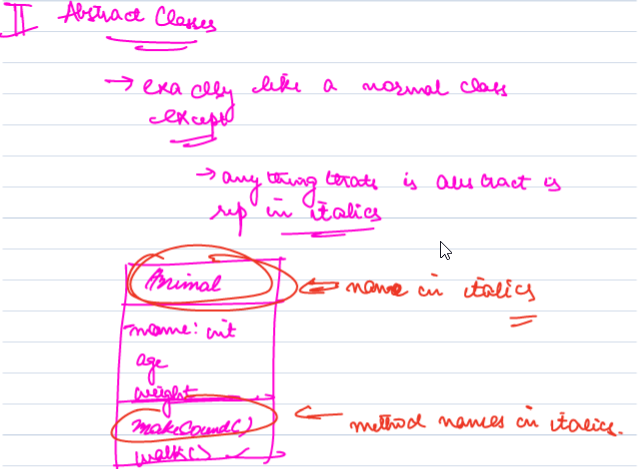
In this example, the class name is displayed at the top of the rectangle, and the attributes and methods are listed in the middle and bottom parts of the rectangle, respectively. Each attribute is represented by its name and data type, and each method is represented by its name and return type. The plus (+) sign before the name of each attribute and method indicates that they are public members of the class, while a minus (-) sign would indicate that they are private. The protected access modifier is denoted by the hash (#) symbol before the name of the attribute or method. It indicates that the member is accessible within the class and any subclasses, but not outside of them.

The default access modifier (also known as package-private) is denoted by leaving out any access modifier symbol before the name of the attribute or method. It indicates that the member is accessible within the same package as the class, but not outside of the package.

### Abstract Class:

In a class diagram, an abstract class is represented by italicizing the class name or by placing the {abstract} keyword before the class name. An abstract class is a class that cannot be instantiated and is intended to be used as a base class for other classes to inherit from.

Note that abstract classes are typically shown with italicized font instead of the {abstract} keyword. The use of italicized font is a convention that is widely used in UML diagrams to represent abstract classes and interfaces.

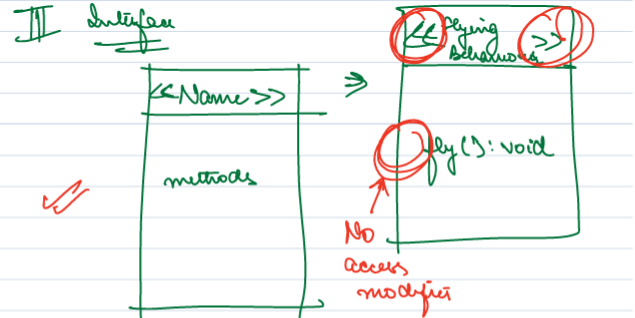


### Interface:

An interface is represented by a rectangle with the name of the interface inside, and the keyword "<<interface>>" or "{interface}" above or below the name of the interface.

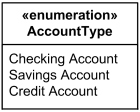
Note that interfaces in UML are similar to abstract classes in that they cannot be instantiated, but they are different in that they can only define method signatures and constants. Classes can implement multiple interfaces, but they can only inherit from a single class.

Also note that the use of the "<<interface>>" or "{interface}" keyword is optional in UML. It is used to indicate that the class is an interface, but the absence of the keyword does not change the fact that the class is an interface.



### Enums:

An enumeration (or enum) is represented by a rectangle with the name of the enum inside, and the keyword "<<enumeration>>" or "{enumeration}" above or below the name of the enum.



Note that an enumeration in UML is a data type that consists of a set of named values, which are often used to represent a fixed set of possible options. In programming languages, enums can be used in place of magic numbers or strings to make code more readable and less error-prone.

Also note that the use of the "<<enumeration>>" or "{enumeration}" keyword is optional in UML. It is used to indicate that the class is an enumeration, but the absence of the keyword does not change the fact that the class is an enum.

[Refer class notes](https://scaler-production-new.s3.ap-southeast-1.amazonaws.com/attachments/attachments/000/036/966/original/Note_11_Apr_2023_at_9_01_39_PM.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIDNNIRGHAQUQRWYA%2F20230428%2Fap-southeast-1%2Fs3%2Faws4_request&X-Amz-Date=20230428T063705Z&X-Amz-Expires=561600&X-Amz-SignedHeaders=host&X-Amz-Signature=421a94cf7e7cfc25b48db65f56d09d22902335c314be72ddc0fd38a89a54f65a)