Palash Bajpai

UML Class

Diagrams

Table of Contents

[**1. What is UML?** 2](#_Toc191888509)

[1. Types of UML 2](#_Toc191888510)

[**2. Understanding Class Diagrams** 4](#_Toc191888511)

# **1. What is UML?**

Unified Modeling Language (UML) is a standardized visual modeling language used to design and document software systems. It helps developers, architects, and stakeholders understand and communicate the structure and behavior of a system

✔️ provides a standard way to visualize system architecture.

✔️ Supports object-oriented design (OOD).

✔️ Used for designing, analyzing, and documenting software systems.

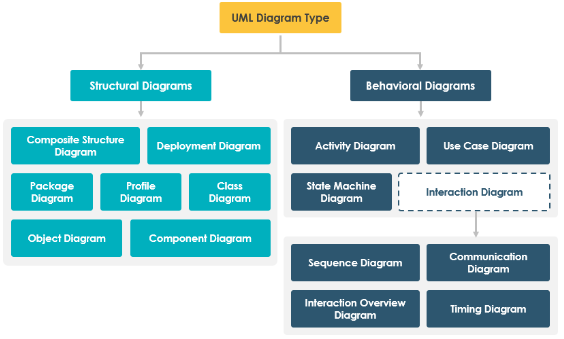
✔️ Independent of programming languages and development methodologies.

**Real World Example:**

Imagine you are developing an E-commerce Application (like Amazon). UML helps visualize:

* How users interact with the system.
* How products, orders, and payments are related.
* How different components communicate.

## 1. Types of UML



**Structural Diagrams**

These diagrams represent the static aspects of a system—how components are structured.

| **UML Diagram** | **Purpose** | **Example** |
| --- | --- | --- |
| **Class Diagram** | Defines object structure & relationships. | Customer has Orders in an e-commerce system. |
| **Object Diagram** | Represents instances of classes at a specific time. | A real-time snapshot of Order and its associated Product. |
| **Component Diagram** | Shows how components interact. | Payment module in a banking app. |
| **Deployment Diagram** | Represents hardware/software deployment. | AWS cloud infrastructure for a web app. |
| **Package Diagram** | Organizes related classes. | Organizing Customer, Order, and Product in different packages. |

**Behavioral Diagrams**

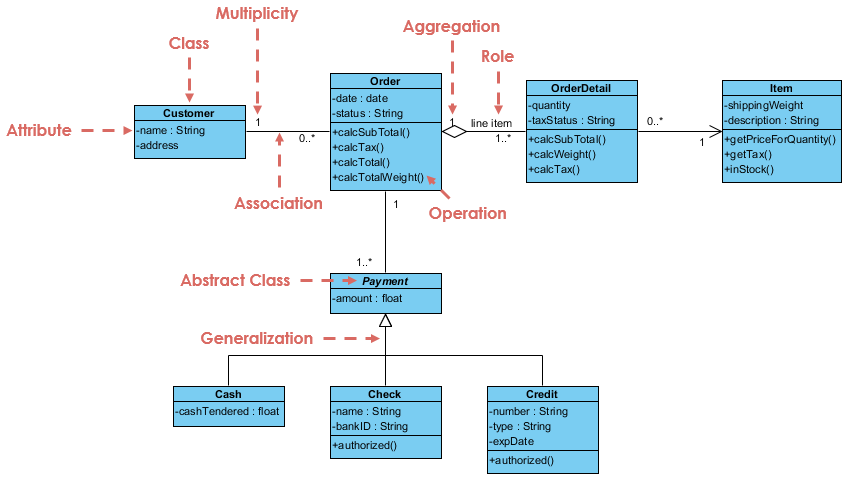
These diagrams represent the dynamic aspects of a system—how components behave over time.

| **UML Diagram** | **Purpose** | **Example** |
| --- | --- | --- |
| **Use Case Diagram** | Defines user interactions with the system. | A Customer placing an Order. |
| **Sequence Diagram** | Shows message flow between objects. | User authentication in a login system. |
| **Activity Diagram** | Represents workflows & processes. | Steps in an online payment transaction. |
| **State Diagram** | Represents object states & transitions. | Order transitioning from Placed → Shipped → Delivered. |

# **2. Understanding Class Diagrams**

A Class Diagram is a UML diagram that represents the static structure of a system. It shows:

* Classes (objects) and their properties/methods.
* Relationships between classes (Association, Inheritance, etc.).
* Visibility (public, private, protected).✔️ provides a standard way to visualize system architecture.



**Role of Class Diagrams**

* Blueprint of the System: Helps design object-oriented software before coding.
* Easy Communication: Developers, architects, and stakeholders understand relationships better.
* System Maintenance: Helps in debugging and extending functionalities.
* Reduces Complexity: Breaks large software into smaller, manageable parts..

**Similarity with OOPs**

Mapping of UML concepts to OOP

| **UML Concept** | **OOP Concept** | **Example** |
| --- | --- | --- |
| **Class** | Blueprint of objects | BankAccount, Customer, Order |
| **Attributes** | Class variables | balance in BankAccount |
| **Methods** | Functions in a class | withdraw(), deposit() |
| **Association** | Object relationship | Customer has multiple Orders |
| **Generalization** | Inheritance | AdminUser extends User |
| **Composition** | "Has-a" relationship | Car has an Engine |

# **3. Basics of UML Class Diagrams**

## Class Representation

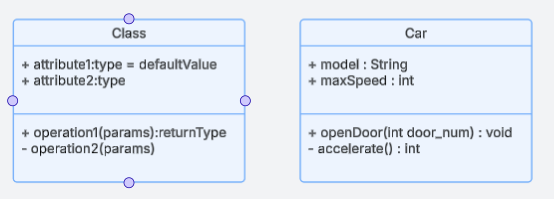
A Class in UML represents a blueprint for objects. It defines attributes (data) and methods (behavior).

A class in UML is depicted as a rectangle divided into three sections:

1️. Class Name (First Section)

2️. Attributes (Properties/Fields) (Second Section)

3️. Methods (Functions/Operations) (Third Section)



## Attributes and methods

**Attributes:** Attributes define the state of a class (fields/variables).

**Methods:** Methods define behavior (actions).

**Visibility:** Visibility notations indicate the access level of attributes and methods.

| **Visibility** | **Symbol** | **Meaning** |
| --- | --- | --- |
| Public | + | Accessible by any class |
| Private | - | Only accessible within the class |
| Protected | # | Accessible by the class and subclasses |