Class Diagram

A class diagram is one of the key UML (Unified Modelling Language) diagrams used in software engineering. It provides a static structure of a system by showing its classes, attributes, operations (methods), and relationships between objects. Class diagrams help in visualizing the blueprint of a system and are used extensively during the design phase of software development.

Key Components of a Class Diagram:

Class:

Represents a blueprint or template for objects.

Depicted as a rectangle divided into three parts:

Class Name: (at the top).

Attributes: (in the middle).

Methods or operations (at the bottom).

Example:

diff

Copy code

```
+--------------------+
|    Student         |
+--------------------+
| -studentID         |
| -name              |
+--------------------+
| +enroll()          |
| +attendClass()     |
+--------------------+
```

Attributes:

Properties or data members of the class (variables).

Example: In the class "Student", attributes could be studentID, name, etc.

Methods (Operations):

Functions that define the behavior of a class.

Example: In the "Student" class, methods might include enroll() or attendClass().

Relationships in Class Diagrams:

Association:

Represents a relationship between two classes.

Example: A "Student" can be associated with a "Course" class.

Inheritance:

A class can inherit properties and behaviours from another class (parent-child relationship).

Example: "Undergraduate Student" inherits from "Student".

Aggregation:

Shows a "whole-part" relationship where the part can exist independently.

Example: A "Library" class might contain a "Book" class.

Composition:

A stronger form of aggregation where the part cannot exist independently of the whole.

Example: A "Car" class contains a "Engine" class, and an engine cannot exist without the car.

Benefits of Class Diagrams:

Helps visualize the structure of a system before implementation.

Defines relationships between different components (classes).

Simplifies system design by representing all the necessary attributes and behaviours of a system.

Clarifies the roles of various components in object-oriented design.