Name of student: Abhay OmPrakash Prajapati		
Roll no: 41		**Tutorial No: 6**
Title of LAB Assignment: UML Diagrams (Class Diagram)		
DOP: 25-09-2023		**DOS:03-10-2023**
CO Mapped: CO1,CO2,CO3	**PO Mapped:** PO1 ,PO4 ,PO6	**Signature:**

Aim: UML Diagrams (Class Diagram)

Description:

Introduction to UML Activity Diagrams

What is UML?

The Unified Modeling Language (UML) is a standardized graphical language for visualizing, specifying, constructing, and documenting the artifacts of software systems. It provides a common notation for software development teams to create and understand models of

their systems.

Purpose of Class Diagrams

Class Diagrams are one of the most fundamental types of UML diagrams. They are used to model the static structure of a system, including the classes, their attributes, operations, and relationships. Class Diagrams are essential for understanding the design of a system and for communicating the design to others.

Benefits of Class Diagrams

Class Diagrams offer a number of benefits, including:

- **Clarity**: They provide a clear and concise visual representation of the system's structure and relationships.
- **Communication**: They facilitate communication between developers, designers, and stakeholders by offering a common visual language.
- **Design**: They help developers to understand the relationships and attributes of classes, which can lead to better system design.
- **Documentation**: They document the static structure of the system, which can aid in maintenance and future development.
- **Code Generation**: Some modeling tools can generate code from Class Diagrams, which can save developers time and effort.

Elements of a Class Diagram

A Class Diagram consists of the following key elements:

- **Classes**: Classes represent the building blocks of a system. They encapsulate data and behavior, and they are the blueprints for objects.
- **Attributes**: Attributes represent the properties or characteristics of a class.
- **Operations**: Operations represent the functions or methods associated

with a class.

- **Relationships**: Relationships represent the associations and connections between classes. There are several types of relationships in Class Diagrams, including association, aggregation, composition, inheritance, and dependency.

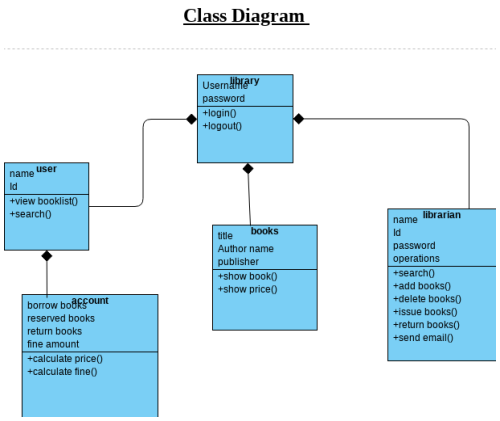
Class Diagram Notations

The following are some of the key notations used in Class Diagrams:

- **Class Notation**: Classes are typically represented as rectangles with three compartments: the top compartment contains the class name, the middle compartment lists attributes, and the bottom compartment lists operations.
- **Association Notation**: Associations between classes are depicted as lines connecting them. Multiplicity notations may be added to indicate the number of instances associated.
- **Inheritance Notation**: Inheritance relationships are represented with a line that has an arrowhead pointing to the superclass.
- **Multiplicity Notation**: Multiplicity notations are shown near the end of an association line and indicate how many instances of one class are associated with instances of another class.

Class Diagram Example

The following diagram shows a simple Class Diagram for a library system:



The diagram shows four classes:

Book, Author, Publisher, and Library. The Book class has attributes such as title, author, publisher, and publication_date. It also has operations such as checkOut() and checkIn().

The Author class has attributes such as name and email. It also has an operation called getBooks(), which returns a list of all the books written by the author.

The Publisher class has attributes such as name and address. It also has an operation called getBooks(), which returns a list of all the books published by the publisher.

The Library class has attributes such as name and address. It also has operations such as addBook(), removeBook(), checkOutBook(), and checkInBook().

The relationships between the classes are shown as follows:

- Association: A book can have one or more authors, and an author can write one or more books.
- Association: A book can have one publisher, and a publisher can publish one or more books.
- Aggregation: A library can have one or more books, but a book can only belong to one library.

Conclusion

UML Class Diagrams are a powerful tool for visualizing, designing, and documenting the static structure of a system. They offer a number of benefits, including clarity, communication, design, documentation, and code generation. Class Diagrams are essential for anyone involved in the software development process