Object Diagrams | Unified Modeling Language (UML)

Last Updated : 31 Jan, 2024

Object diagrams are a visual representation in UML (Unified Modeling Language) that illustrates the instances of classes and their relationships within a system at a specific point in time. They display objects, their attributes, and the links between them, providing a snapshot of the system's structure during execution. Since object diagrams depict behavior when objects have been instantiated, we can study the behavior of the system at a particular instant.

Important Topics for the Object Diagrams

1. What are Object Diagrams?

An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them.

- An object diagram in UML is useful because it provides a clear and visual representation of specific instances of classes and their relationships at a particular point in time, aiding in understanding and communicating the structure and interactions within a system.
- In other words, "An object diagram in the Unified Modeling Language (UML), is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

Object diagrams in UML are depicted using a simple and intuitive notations to show a snapshot of a system at a specific point in time, displaying instances of classes and their relationships.

What is an Object?

An object refers to a specific instance of a class within a system. A class is a blueprint or template that defines the common attributes and behaviors shared by a group of objects. An object, on the other hand, is a concrete and individual occurrence of that class, possessing unique values for its attributes.

What is a Classifier?

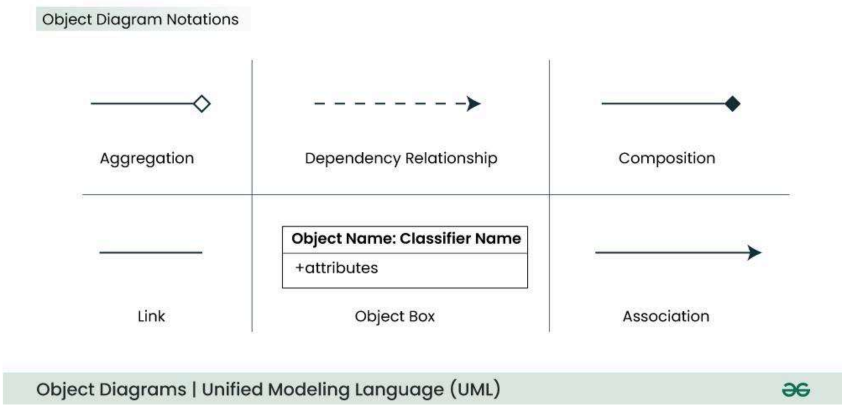
In UML a classifier refers to a group of elements that have some common features like methods, attributes and operations. A classifier can be thought of as an abstract metaclass which draws a boundary for a group of instances having common static and dynamic features.

For example:

we refer a class, an object, a component, or a deployment node as classifiers in UML since they define a common set of properties. We are able to design object diagrams by instantiating classifiers.

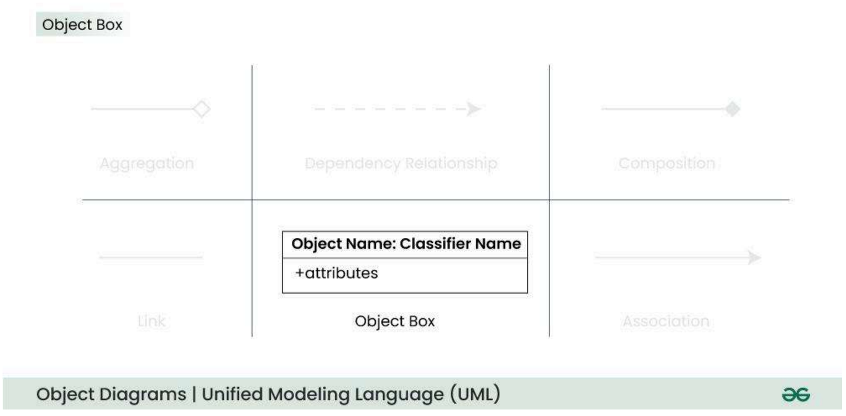
4. Object Diagram Notations

The object diagram in UML uses specific notations to represent instances of classes and their relationships at a particular moment in time.



1. Objects or Instance specifications

When we instantiate a classifier in a system, the object we create represents an entity which exists in the system. We can represent the changes in object over time by creating multiple instance specifications. We use a rectangle to represent an object in an object diagram.

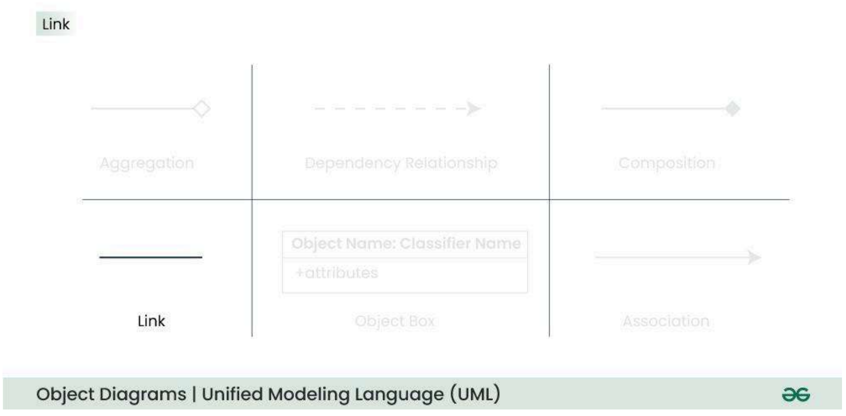


2. Attributes and Values

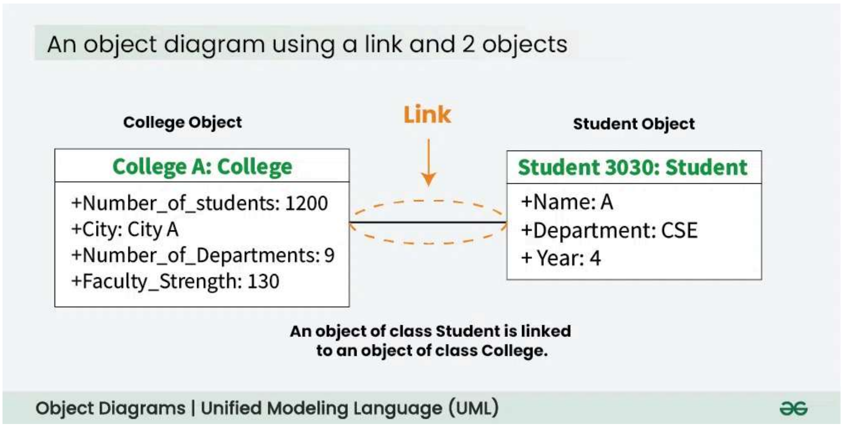
Inside the object box, attributes of the object are listed along with their specific values.

3. Link

We use a link to represent a relationship between two objects. We represent the number of participants on the link for each, at the end of the link. The term link is used to specify a relationship between two instance specifications or objects. We use a solid line to represent a link between two objects.



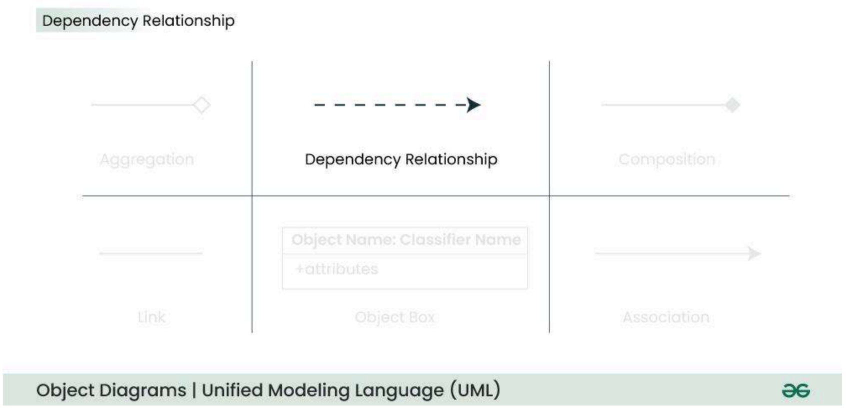
For Example *– In the figure below, an object of class Student is linked to an object of class College.*


An object diagram using a link and 2 objects
College Object
Link
Student Object
College A: College
+Number_of_students: 1200
+City: City A
+Number_of_Departments: 9
+Faculty_Strength: 130
Student 3030: Student
+Name: A
+Department: CSE
+ Year: 4
An object of class Student is linked to an object of class College.
Object Diagrams | Unified Modeling Language (UML)

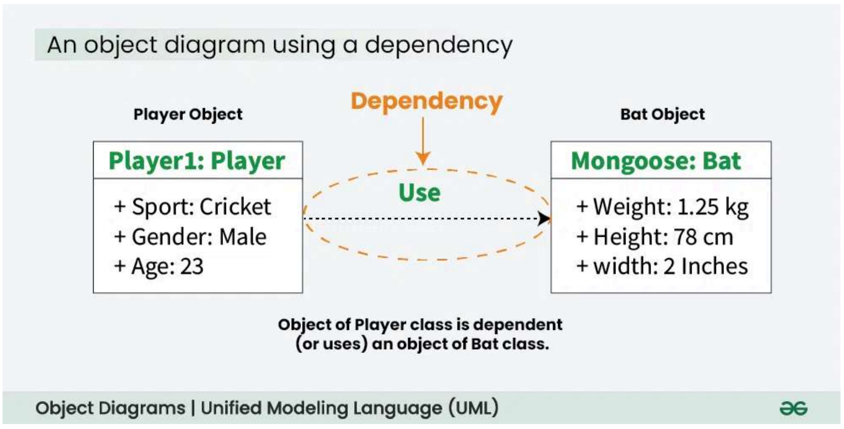
4. Dependency Relationships

We use a dependency relationship to show when one element depends on another element. A dependency is used to depict the relationship between dependent and independent entities in the system.

- Any change in the definition or structure of one element may cause changes to the other.
- This is a unidirectional kind of relationship between two objects.
- Dependency relationships are of various types specified with keywords like Abstraction, Binding, Realization, Substitution and Usage are the types of dependency relationships used in UML.

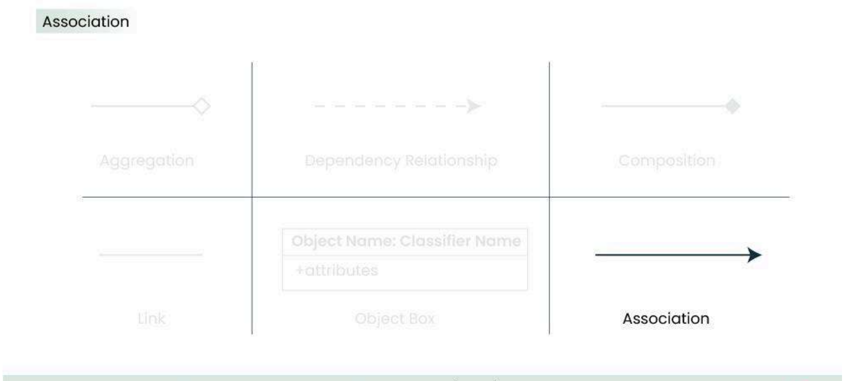


For example *– In the figure below, an object of Player class is dependent (or uses) an object of Bat class.*


An object diagram using a dependency
Player Object
Dependency
Bat Object
Player1: Player
+ Sport: Cricket
+ Gender: Male
+ Age: 23
Use
Mongoose: Bat
+ Weight: 1.25 kg
+ Height: 78 cm
+ width: 2 Inches
Object of Player class is dependent (or uses) an object of Bat class.
Object Diagrams | Unified Modeling Language (UML)

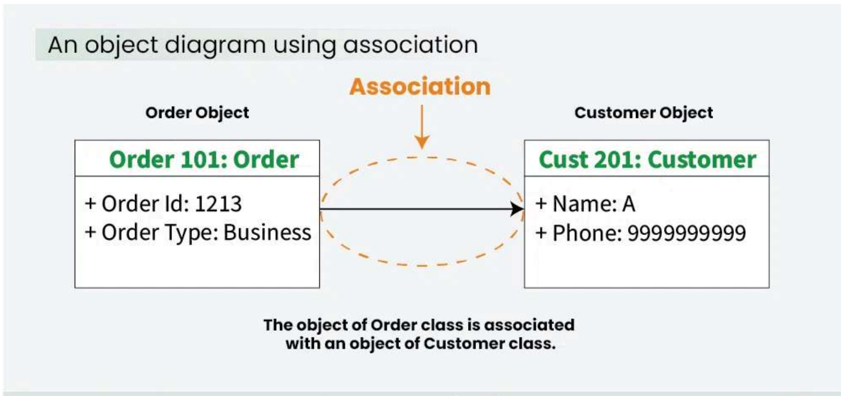
5. Association

Association is a reference relationship between two objects (or classes). An association line connects two object boxes, representing a relationship between instances of two classes. We use association when one object references members of the other object. Association can be uni-directional or bi-directional. We use an arrow to represent association.



Object Diagrams | Unified Modeling Language (UML)

For example *– The object of Order class is associated with an object of Customer class.*

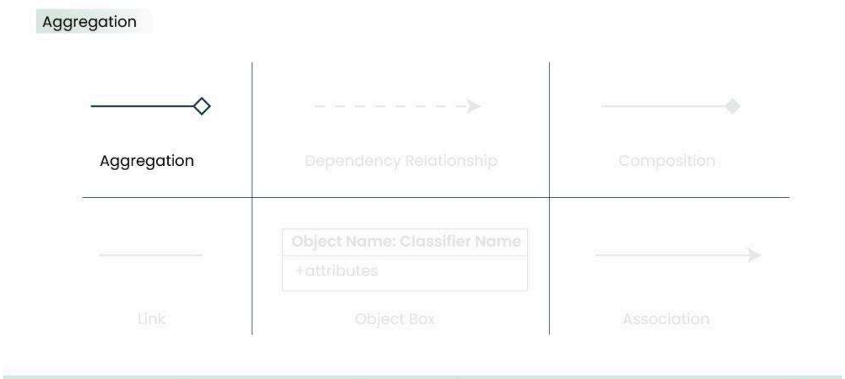


Object Diagrams | Unified Modeling Language (UML)

6. Aggregation

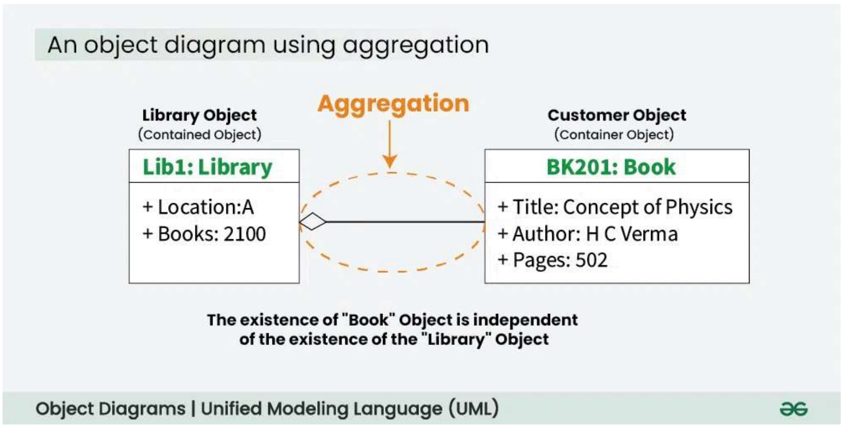
Aggregation represents a "has a" relationship. We use a hollow diamond on the containing object with a line which joins it to the contained object.

- Aggregation is a specific form of association.
- It is a kind of parent-child relationship however it isn't inheritance.
- Aggregation occurs when the lifecycle of the contained objects does not strongly depend on the lifecycle of container objects.



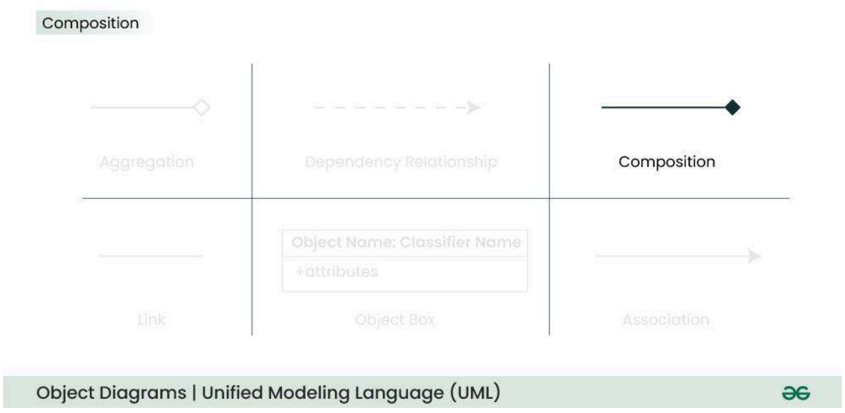
Object Diagrams | Unified Modeling Language (UML)

For example *– A library has an aggregation relationship with books. Library has books or books are a* *part of library. The existence of books is independent of the existence of the library.*

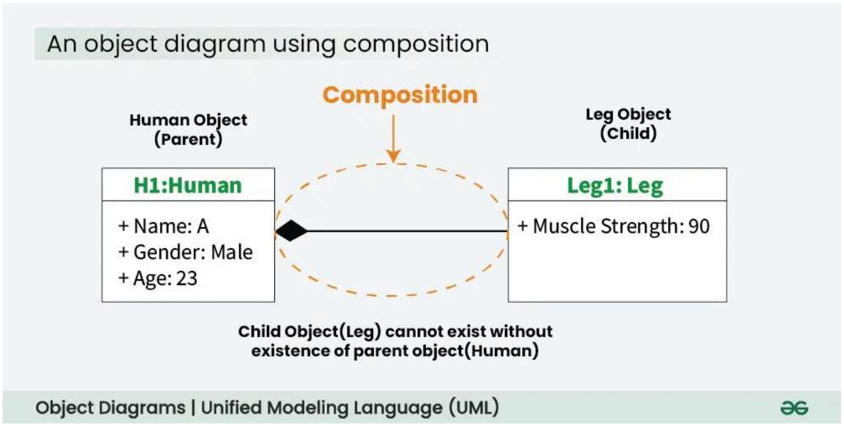

An object diagram using aggregation
Aggregation
Library Object
(Contained Object)
Lib1: Library
+ Location:A
+ Books: 2100
Customer Object
(Container Object)
BK201: Book
+ Title: Concept of Physics
+ Author: H C Verma
+ Pages: 502
The existence of "Book" Object is independent of the existence of the "Library" Object
Object Diagrams | Unified Modeling Language (UML)

7. Composition

Composition is a type of association where the child cannot exist independent of the other. We use a filled diamond on the containing object with a line which joins it to the contained object. Composition is also a special type of association. It is also a kind of parent child relationship but it is not inheritance. So whenever independent existence of the child is not possible we use a composition relationship.



Consider the example of a boy Gurkaran: Gurkaran is composed of legs and arms. Here Gurkaran has a composition relationship with his legs and arms. Here legs and arms can't exist without the existence of their parent object.


An object diagram using composition
Composition
Human Object
(Parent)
H1:Human
+ Name: A
+ Gender: Male
+ Age: 23
Leg Object
(Child)
Leg1: Leg
+ Muscle Strength: 90
Child Object(Leg) cannot exist without existence of parent object(Human)
Object Diagrams | Unified Modeling Language (UML)

5. Purpose of Object Diagrams

The main purpose of using object diagrams is:

- They offer a detailed view of how objects interact with each other in specific scenarios.
- Proper design and analysis of applications can be faster and efficient.
- Object diagrams are beneficial during the implementation phase of software development.
- Promoting a shared understanding of specific instances and their relationships, facilitating collaboration among team members.

6. Benefits of Object Diagrams

- **Detailed Insight into Relationships:**
    - They offer a detailed view of relationships and collaborations between instances of classes. This helps in understanding the specific interactions and dependencies among objects.
- **Implementation Guidance:**
    - During the system implementation phase, object diagrams assist developers in building and testing the actual instances of classes. They provide guidance on how to represent objects in code.
- **Integration Testing Assistance:**
    - They are valuable for integration testing, allowing testers to evaluate how different objects collaborate and exchange information. This ensures that integrated components of the system work seamlessly.
- **Validation of Code Implementation:**
    - Developers can use object diagrams to validate that the actual code aligns with the intended relationships and interactions specified in the design. This helps maintain consistency between the design and the implementation.
- **Scenario Illustration:**
    - Object diagrams are useful for illustrating and documenting specific scenarios or use cases, providing a clear visual representation of how objects behave in different situations.

7. How to draw an Object Diagram?

1. **Identify Classes:** Determine the classes relevant to the scenario you want to depict. Classes are the blueprints that define the attributes and behaviors shared by their instances.
2. **Identify Objects:** Identify specific instances or objects of each class that you want to include in the diagram. These represent the actual things in your system.
3. **Create Object Boxes:** Draw rectangles to represent the specific instances or objects of each class. Write the name of each object inside the box.
4. **Add Attributes and Values:** Inside each object box, list the attributes of that object along with their specific values.
5. **Draw Relationships:** Connect the object boxes with lines to represent relationships or associations between instances. Use arrows to indicate the direction of the association if necessary.
6. **Label Relationships:** Label the relationships with multiplicity and role names if needed. Label the association lines with a verb or phrase to describe the nature of the relationship.
7. **Review and Refine:** Review your Object diagram to ensure it accurately represents the system's structure and relationships. Refine the diagram as needed based on feedback and requirements.
8. **Use Tools for Digital Drawing:** While you can draw class diagrams on paper, using digital tools can provide more flexibility and ease of modification. UML modeling tools, drawing software, or even specialized diagramming tools can be helpful.

8. Use Cases of Object Diagrams

Object diagrams in UML also play a crucial role in various phases of software development. Here are some use cases for object diagrams:

- **System Implementation:**
  - Object diagrams provide a practical representation of specific instances of classes, aiding in the implementation phase. Object diagrams help developers when they are building the actual software. These diagrams show real examples of things (objects) in the system and how they work together.
- **Communication and Collaboration:**
  - Similar to class diagrams, object diagrams serve as a communication tool among stakeholders. They facilitate discussions about the relationships and interactions between specific objects at a particular moment, promoting a shared understanding among team members.
- **Test Case Design:**
  - Testers use object diagrams to design test cases based on the relationships and behaviors of specific instances. This helps ensure thorough testing of object interactions in the system.
- **Debugging and Troubleshooting:**
  - During the debugging process, object diagrams can be valuable for understanding the state of specific objects at a particular point in time. This aids developers in identifying and resolving issues related to object interactions.
- **Training and Documentation:**
  - Object diagrams can be used for training purposes, helping new team members understand the structure and behavior of specific instances in the system. They also contribute to documentation by providing visual representations of object relationships.

Previous Article
Class Diagram | Unified Modeling Language (UML)

Next Article
Deployment Diagram in Unified Modeling Language(UML)

Similar Reads

Class Diagrams vs Object Diagrams | Unified Modeling Language(UML)
UML class diagrams and object diagrams are the key tools for understanding the structure of a system, yet they serve distinct purposes. The differences between UML class diagrams and object diagrams provide insights...
3 min read

Behavioral Diagrams | Unified Modeling Language(UML)
Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them. So UML becomes essential to communicate with non-...
7 min read

State Machine Diagrams | Unified Modeling Language (UML)
A State Machine Diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions....

7 min read

Activity Diagrams | Unified Modeling Language (UML)

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. It is a type of behavioral diagram and we can depict both sequential processing and...

9 min read

Sequence Diagrams | Unified Modeling Language (UML)

Unified Modelling Language (UML) is a modeling language in the field of software engineering that aims to set standard ways to visualize the design of a system. UML guides the creation of multiple types of diagrams suc...

10 min read

View More Articles

Article Tags : Design Pattern System Design UML

DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Visualisation Tutorial
Pandas Tutorial
NumPy Tutorial
NLP Tutorial
Deep Learning Tutorial

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial
Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps
Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design
High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation
Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects
Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos
DSA
Python
Java
C++
Web Development
Data Science
CS Subjects