Google 도착어: 한국어 원본 표시 옵션 ▼

사용 사례 다이어그램 - 통합 모델링 언어(UML)

최종 업데이트 : 2024년 10월 14일

통합 모델링 언어(UML) 의 사용 사례 다이어그램은 사용자(행위자)와 시스템 간의 상호 작용을 보여주는 시각적 표현입니다. 시스템의 기능적 요구 사항을 포착하여 다양한 사용자가 시스템 내의 다양한 사용 사례 또는 특정 기능을 사용하는 방식을 보여줍니다. 사용 사례 다이어그램은 시스템 동작에 대한 개략적인 개요를 제공하므로 이해 관계자, 개발자 및 분석가가 사용자 관점에서 시스템이 어떻게 작동하도록 의도되었는지, 그리고 다양한 프로세스가 서로 어떻게 관련되어 있는지 이해하는 데 유용합니다. 이는 시스템 범위와 요구 사항을 정의하는 데 필수적입니다.

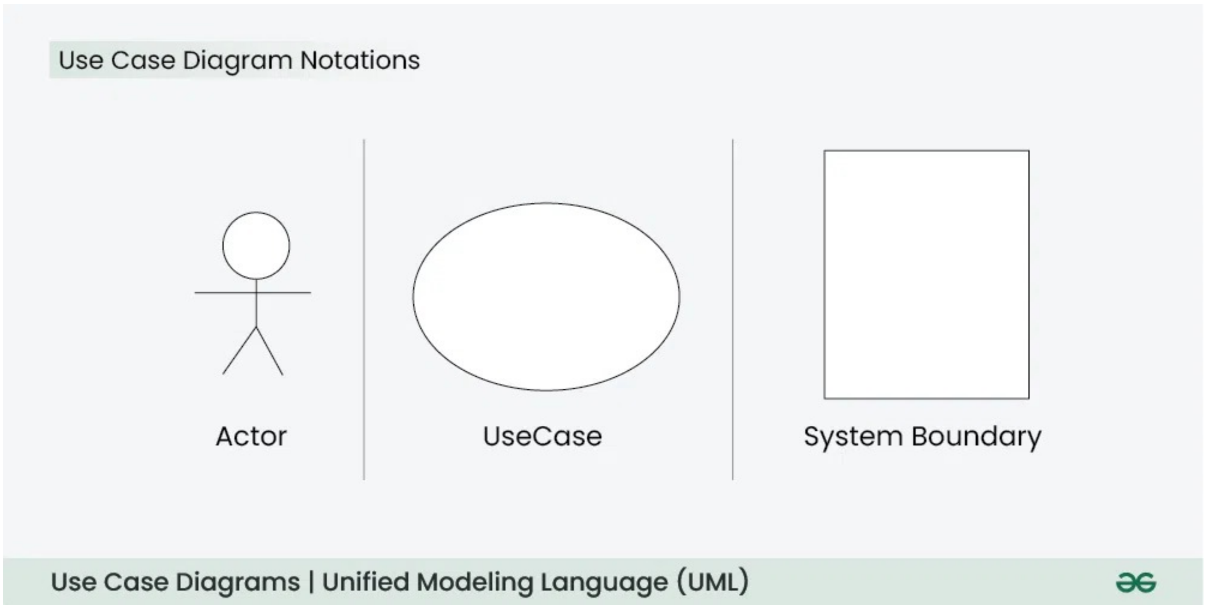
목차

당사는 귀하가 당사 웹사이트에서 최상의 브라우징 경험을 할 수 있도록 쿠키를 사용합니다. 당사 사이트를 사용함으로써 귀하는 당사의 쿠키 정책 및 개인정보 보호 정책을 읽고 이해했음을 인정합니다.

알았어요 !

UML에서 사용 사례 다이어그램이란 무엇입니까?

사용 사례 다이어그램은 특정 목표를 달성하기 위해 고려 중인 시스템(사용자 또는 외부 시스템)과 행위자 간의 상호 작용을 나타내는 통합 모델링 언어(UML) 다이어그램의 한 유형입니다. 사용자가 시스템과 상호 작용할 수 있는 다양한 방법을 설명하여 시스템 기능에 대한 높은 수준의 뷰를 제공합니다.



Use Case Diagram Notations
Actor
UseCase
System Boundary
Use Case Diagrams | Unified Modeling Language (UML)

효과적인 사용 사례 다이어그램을 만드는 방법에 대한 더 많은 통찰력을 얻으려면 시스템 설계 과정에서 시스템 설계의 기본을 다루고 시스템 설계의 실제 응용 프로그램을 제공합니다.

언제 Use Case Diagram을 적용해야 하나요?

사용 사례 다이어그램은 여러 상황에서 유용합니다. 다음은 사용 사례 다이어그램을 고려해야 하는 경우입니다.

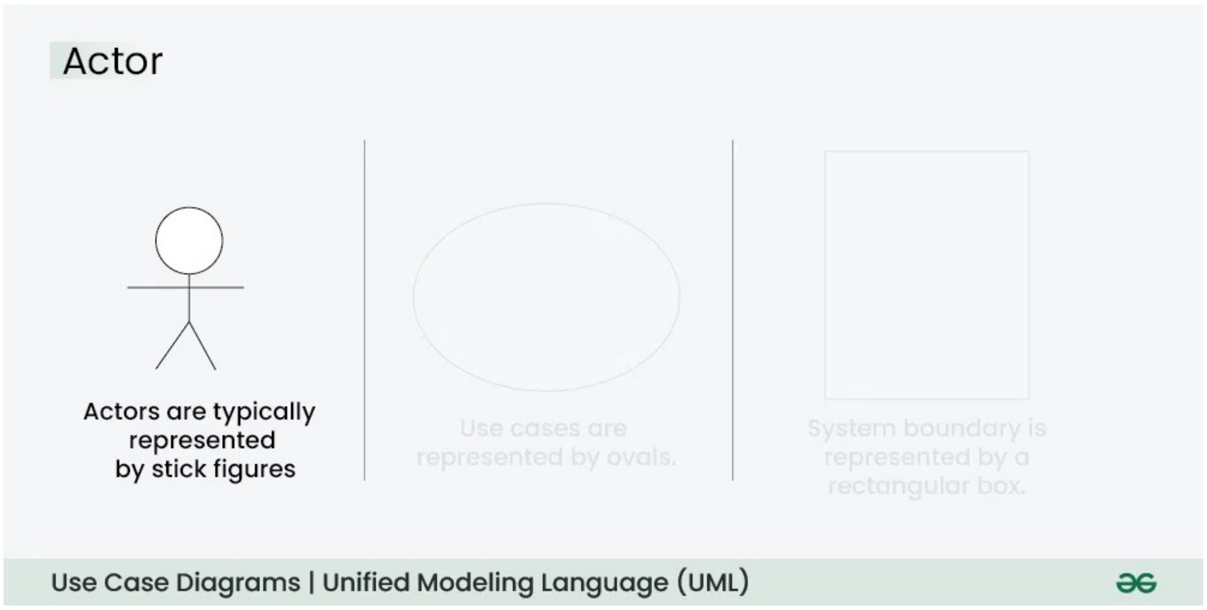
- 사용자 요구사항을 수집하고 명확히 해야 할 때, 사용 사례 다이어그램은 다양한 사용자가 시스템과 상호작용하는 방식을 시각화하는 데 도움이 됩니다.
- 비기술적 이해 관계자를 포함한 다양한 그룹과 협력하는 경우 이러한 다이어그램은 시스템 기능을 전달하는 명확하고 간단한 방법을 제공합니다.
- 시스템 설계 단계에서 사용 사례 다이어그램은 사용자 상호작용을 개략적으로 설명하고 기능을 계획하는 데 도움이 되며, 이를 통해 설계가 사용자 요구 사항과 일치하는지 확인할 수 있습니다.
- 시스템에 포함되는 것과 외부적인 것을 정의할 때, 사용 사례 다이어그램은 이런 경계를 명확히 하는 데 도움이 됩니다.

사용 사례 다이어그램 표기법

UML 표기법은 소프트웨어 개발자, 설계자 및 기타 이해 관계자가 일관되고 이해하기 쉬운 방식으로 시스템 설계, 아키텍처 및 동작을 전달하고 문서화할 수 있는 시각적 언어를 제공합니다.

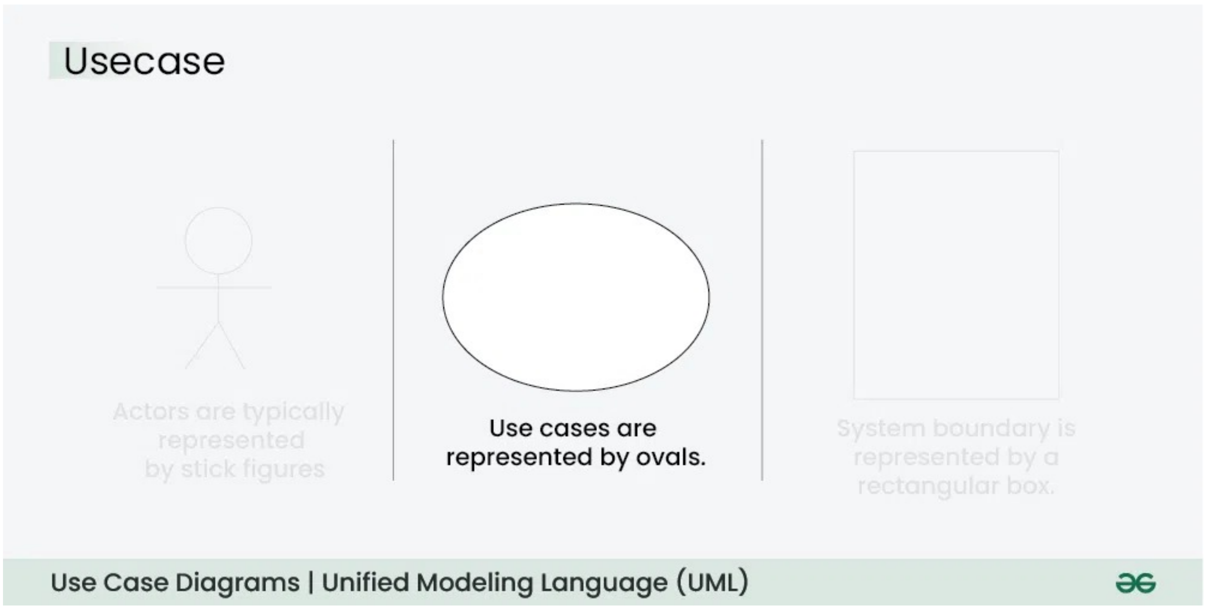
1. 배우들

Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In the context of a Use Case Diagram, actors initiate use cases and receive the outcomes. Proper identification and understanding of actors are crucial for accurately modeling system behavior.


Actor
Actors are typically represented by stick figures
Use cases are represented by ovals.
System boundary is represented by a rectangular box.
Use Case Diagrams | Unified Modeling Language (UML)

2. Use Cases

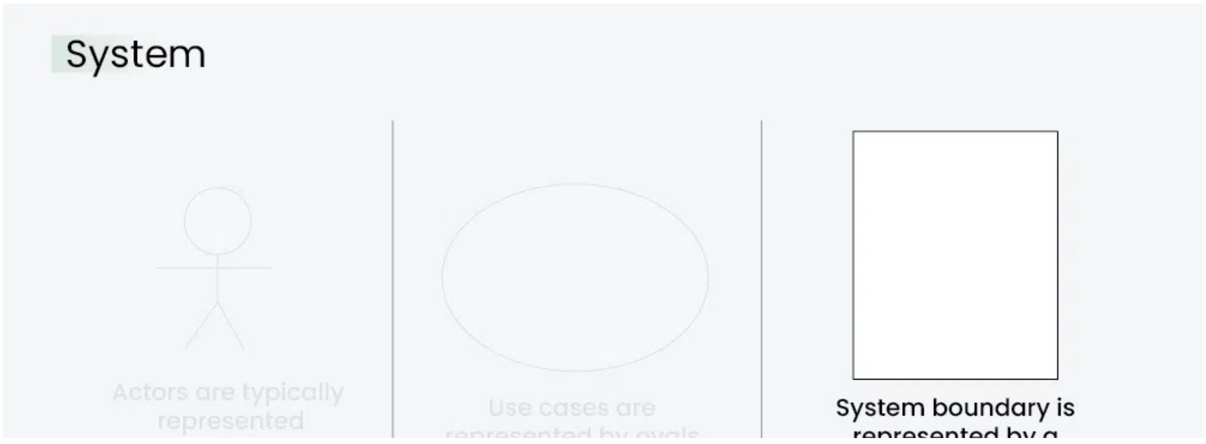
Use cases are like scenes in the play. They represent specific things your system can do. In the online shopping system, examples of use cases could be "Place Order," "Track Delivery," or "Update Product Information". Use cases are represented by ovals.

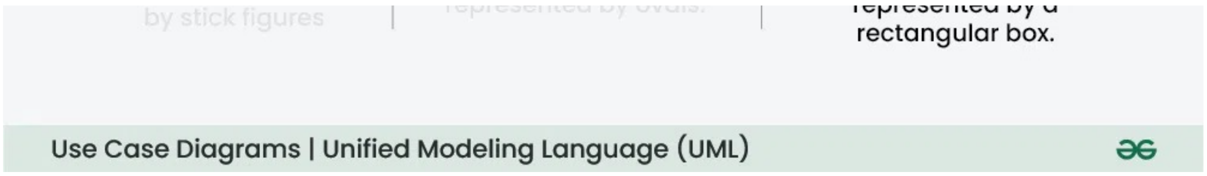

Usecase
Actors are typically represented by stick figures
Use cases are represented by ovals.
System boundary is represented by a rectangular box.
Use Case Diagrams | Unified Modeling Language (UML)

3. System Boundary

The system boundary is a visual representation of the scope or limits of the system you are modeling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it. The system boundary is typically represented by a rectangular box that surrounds all the use cases of the system.

> *The purpose of system boundary is to clearly outlines the boundaries of the system, indicating which components are internal to the system and which are external actors or entities interacting with the system.*




by stick figures
represented by ovals.
represented by a rectangular box.
Use Case Diagrams | Unified Modeling Language (UML)

Use Case Diagram Relationships

In a Use Case Diagram, relationships play a crucial role in depicting the interactions between actors and use cases. These relationships provide a comprehensive view of the system's functionality and its various scenarios. Let's delve into the key types of relationships and explore examples to illustrate their usage.

1. Association Relationship

The Association Relationship represents a communication or interaction between an actor and a use case. It is depicted by a line connecting the actor to the use case. This relationship signifies that the actor is involved in the functionality described by the use case.

Example: Online Banking System

- **Actor:** Customer
- **Use Case:** Transfer Funds
- **Association:** A line connecting the "Customer" actor to the "Transfer Funds" use case, indicating the customer's involvement in the funds transfer process.

2. Include Relationship

The Include Relationship indicates that a use case includes the functionality of another use case. It is denoted by a dashed arrow pointing from the including use case to the included use case. This relationship promotes modular and reusable design.

Example: Social Media Posting

- **Use Cases:** Compose Post, Add Image
- **Include Relationship:** The "Compose Post" use case includes the functionality of "Add Image." Therefore, composing a post includes the action of adding an image.

3. Extend Relationship

The Extend Relationship illustrates that a use case can be extended by another use case under specific conditions. It is represented by a dashed arrow with the keyword "extend." This relationship is useful for handling optional or exceptional behavior.

> ***Example: Flight Booking System***

- **Use Cases:** Book Flight, Select Seat
- **Extend Relationship:** The "Select Seat" use case may extend the "Book Flight" use case when the user wants to choose a specific seat, but it is an optional step.

4. Generalization Relationship

The Generalization Relationship establishes an "is-a" connection between two use cases, indicating that one use case is a specialized version of another. It is represented by an arrow pointing from the specialized use case to the general use case.

> ***Example: Vehicle Rental System***

- **Use Cases:** Rent Car, Rent Bike
- **Generalization Relationship:** Both "Rent Car" and "Rent Bike" are specialized versions of the general use case "Rent Vehicle."

Generalization Relationship

How to draw a Use Case diagram in UML?

Below are the main steps to draw use case diagram in UML:

- **Step 1: Identify Actors**: Determine who or what interacts with the system. These are your actors. They can be users, other systems, or external entities.
- **Step 2: Identify Use Cases**: Identify the main functionalities or actions the system must perform. These are your use cases. Each use case should represent a specific piece of functionality.
- **Step 3: Connect Actors and Use Cases**: Draw lines (associations) between actors and the use cases they are involved in. This represents the interactions between actors and the system.
- **Step 4: Add System Boundary**: Draw a box around the actors and use cases to represent the system boundary. This defines the scope of your system.
- **Step 5: Define Relationships**: If certain use cases are related or if one use case is an extension of another, you can indicate these relationships with appropriate notations.
- **Step 6: Review and Refine**: Step back and review your diagram. Ensure that it accurately represents the interactions and relationships in your system. Refine as needed.
- **Step 7: Validate**: Share your use case diagram with stakeholders and gather feedback. Ensure that it aligns with their understanding of the system's functionality.

Use Case Diagram example(Online Shopping System)

Let's understand how to draw a Use Case diagram with the help of an Online Shopping System:

- **Actors:**
  - Customer
  - Admin
- **Use Cases:**
  - Browse Products
  - Add to Cart

    - Checkout
    - Manage Inventory (Admin)
- **Relations:**
    - The Customer can browse products, add to the cart, and complete the checkout.
    - The Admin can manage the inventory.

Below is the use case diagram of an Online Shopping System:

What are common Use Case Diagram Tools and Platforms?

Several tools and platforms are available to create and design Use Case Diagrams. These tools offer features that simplify the diagram creation process, facilitate collaboration among team members, and enhance overall efficiency. Here are some popular Use Case Diagram tools and platforms:

- **Lucidchart:**
    - Cloud-based collaborative platform.
-
    - Real-time collaboration and commenting.
    - Templates for various diagram types.
- **draw.io:**
    - Free, open-source diagramming tool.
    - Works offline and can be integrated with Google Drive, Dropbox, and others.
    - Offers a wide range of diagram types, including Use Case Diagrams.
- **Microsoft Visio:**
    - Part of the Microsoft Office suite.
    - Supports various diagram types, including Use Case Diagrams.
-
    - Extensive shape libraries and templates.

- **SmartDraw:**
    - User-friendly diagramming tool.
    - Templates for different types of diagrams, including Use Case Diagrams.
    - Integration with Microsoft Office and Google Workspace.
- **PlantUML:**
    - Open-source tool for creating UML diagrams.
    - Text-based syntax for diagram specification.
    - Supports collaborative work using version control systems.

What are Common Mistakes while making Use Case Diagram?

Avoiding common mistakes ensures the accuracy and effectiveness of the Use Case Diagram. Here are key points for each mistake:

- Adding too much detail can confuse people.
- Unclear connections lead to misunderstandings about system interactions.
- Different names for the same elements create confusion.
- Incorrectly using generalization can misrepresent relationships.
- Failing to define the system’s limits makes its scope unclear.
- Treating the diagram as static can make it outdated and inaccurate.

Best Practices for Use Case Diagram

Crafting clear and effective Use Case Diagrams is essential for conveying system functionality and interactions. Here are some best practices to consider:

- Use Case Diagram focus on capturing the core functions of the system, avoiding extraneous details.
- They uses a uniform naming scheme for use cases and actors throughout the diagram to enhance clarity and prevent misunderstandings.
- They ensure uniformity in the appearance of elements such as ovals

(for use cases), stick figures (for actors), and connecting lines to create a polished presentation.
- They help in organizing use cases into coherent groups that represent distinct modules or subsystems within the overall system.
- Use Case Diagrams adopt an iterative method, updating the diagram as the system changes or as new information emerges.

What is the Purpose and Benefits of Use Case Diagrams?

The Use Case Diagram offers numerous benefits throughout the system development process. Here are some key advantages of using Use Case Diagrams:

- Use Case Diagrams offer a clear visual representation of a system’s functions and its interactions with external users. This representation helps stakeholders, including those without technical expertise, in grasping the system’s overall behavior.
- They establish a shared language for articulating system requirements, ensuring that all team members have a common understanding.
- Use Case Diagram illustrate the different ways users engage with the system, contributing to a thorough comprehension of its functionalities.
- In the design phase, Use Case Diagrams help outline how users (actors) will interact with the system. They support the planning of user interfaces and aid in structuring system functionalities.

Conclusion

In conclusion, a Use Case Diagram in UML serves as a powerful tool for capturing and visualizing the functional requirements and interactions within a system. By representing actors, use cases, and their relationships in a clear and concise manner, this diagram provides a high-level overview of the system's behavior.

FAQs on Use Case Diagram in UML

Below are the main faqs on Use Case Diagram in UML:

Q1: How do use case diagrams support agile methodologies?

> *They provide a clear overview of user requirements and prioritize features based on user needs.*

Q2: How does include and exclude relationships differ from each other in Use Case Diagram?

> *"Include" indicates mandatory functionality that is reused across use cases, while "extend" represents optional behavior that enhances a use case under certain conditions.*

Q3: How communication is improved among stakeholders using Use Case Diagram?

> *They serve as a visual tool that bridges technical and non-technical stakeholders, strengthening a shared understanding of system requirements and functionalities.*

Q4: What is the significance of system boundaries in use case diagrams?

> *System boundaries define the scope of the system and help in clarifying what is included in the system versus what is external, helping to manage project scope.*

Q5: How do you prioritize use cases in a diagram?

> *Use cases can be prioritized based on factors like user importance, business value, frequency of use, or technical feasibility, guiding development focus.*

Want to be a Software Architect or grow as a working professional? Knowing both Low and High-Level System Design is highly necessary. As such, our course fits you perfectly: Mastering System Design: From Low-Level to High-Level Solutions. Get in-depth into **System Design** with hands-on projects and **Real-World Examples**. Learn how to come up with systems that are scalable, efficient, and robust—ones that impress. Ready to elevate your tech skills? Enrol now and build the future!

Comment

More info

Next Article

Sequence Diagrams - Unified Modeling Language (UML)

Similar Reads

What are UML Diagrams

Unified Modeling Language (UML) Diagrams

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to...

15 min read

UML Full Form

The full form of UML is "Unified Modeling Language". It is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize how a system has been designed. It...

3 min read

Structural Diagrams

Class Diagram | Unified Modeling Language (UML)

A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them. It helps everyone involved in a...

12 min read

Object Diagrams | Unified Modeling Language (UML)

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...

8 min read

Deployment Diagram in Unified Modeling Language(UML)

A Deployment Diagram is a type of Structural UML Diagram that shows the physical deployment of software components on hardware nodes. It illustrates the mapping of software components...

9 min read

Package Diagram – Unified Modeling Language (UML)

A package diagram is a type of structural diagram in UML (Unified Modeling Language) that organizes and groups related classes and components into packages. It visually represents the...

7 min read

Behavioral Diagrams

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 communicat...

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...

7 min read

Activity Diagrams - Unified Modeling Language (UML)

Activity diagrams are an essential part of the Unified Modeling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are...

10 min read

Use Case Diagram - Unified Modeling Language (UML)

A Use Case Diagram in Unified Modeling Language (UML) is a visual representation that illustrates the interactions between users (actors) and a system. It captures the functional requirements of a...

10 min read

Sequence Diagrams - Unified Modeling Language (UML)

A Sequence Diagram is a key component of Unified Modeling Language (UML) used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate wit...

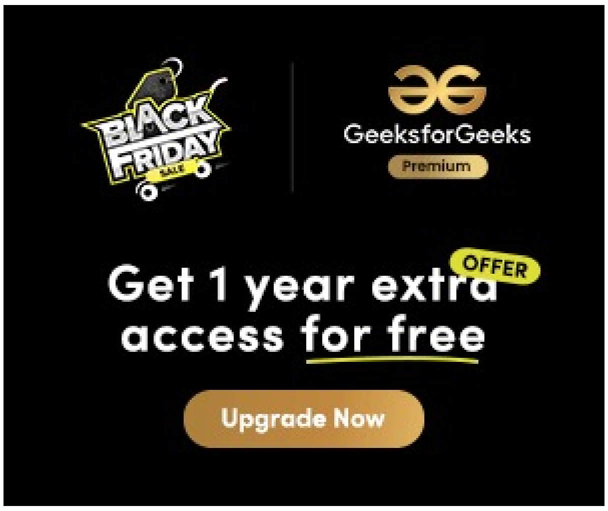
12 min read

Interaction Overview Diagrams | Unified Modeling Language (UML)

Interaction Overview Diagrams (IODs) in UML (Unified Modeling Language) provide a high-level view of the interactions between various components or objects in a system. They are used to...

8 min read

Article Tags : System Design Geeks Premier League Geeks Premier League 2023



GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:- A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

Company
About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Languages
Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA
Data Structures
Algorithms

Data Science & ML
Data Science With Python
Data Science For Beginner

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

Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

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
회사별 채용 프로세스
회사별 준비
적성 준비
퍼즐

학교 과목

수학
물리학
화학
생물학
사회 과학
영어 문법
상업
월드 GK

GeeksforGeeks 비디오

디에스에이
파이썬
자바
씨++
웹 개발
데이터 과학
CS 과목

@GeeksforGeeks, Sanchhaya Education Private Limited , 모든 권리 보유