1. **What is Object-Oriented Programming (OOP)?**

OOP is a programming paradigm that models real-world objects and systems using classes and objects.

**Key Concepts:**

* **Classes**: Blueprints for objects, defining properties and behavior.
* **Objects**: Instances of classes, with their own attributes and methods.
* **Inheritance**: A child class inherits properties and behavior from a parent class.
* **Polymorphism**: Objects can take on multiple forms through method overriding or overloading.
* **Abstraction**: Showing only necessary information, hiding internal details.
* **Encapsulation**: Bundling data and methods, restricting direct access.

1. **What is the difference between an interface and an abstract class?Interface vs Abstract Class:**

Interfaces and abstract classes are used to define a blueprint for other classes, but they serve different purposes.

**Interface:**

* A contract specifying methods that must be implemented by any class that implements it.
* Cannot be instantiated on its own.
* All methods are abstract.
* A class can implement multiple interfaces.

**Abstract Class:**

* A class that cannot be instantiated on its own and is intended to be inherited.
* Can have both abstract and concrete methods.
* A class can only inherit from one abstract class.

**Key differences:**

* Multiple inheritance: Interfaces allow multiple inheritance, while abstract classes do not.
* Method implementation: Interfaces have no implementation, while abstract classes can have both abstract and concrete methods.

1. What is dependency Injection?
   * Dependency Injection (DI) is a software design pattern that allows components to be loosely coupled, making it easier to test, maintain, and extend systems. In DI, components are given their dependencies rather than creating them themselves. This is achieved through a process called **inversion of control**, where the control of creating dependencies is inverted from the component to an external entity, such as a container or a factory.

Imagine you're making a sandwich. You need bread, meat, cheese, and veggies. Instead of making each component yourself, you ask a friend to bring the bread, another friend to bring the meat, and so on. This way, you can focus on assembling the sandwich without worrying about how each component is made.

1. What is CI/CD?
   * Continuous Integration is a development practice where developers frequently merge their code changes into a shared repository, and automated build and test processes are triggered for each code commit. The main goals of CI are to detect integration issues early and to ensure that the codebase is always in a working state.
   * Continuous Deployment/Delivery (CD):

Continuous Deployment and Continuous Delivery are practices that extend the principles of CI to the deployment stage. They automate the process of releasing code changes to production environments, enabling frequent and reliable deployments.

Continuous Deployment: In this approach, every successful code change that passes the automated tests is automatically deployed to production. The process is fully automated, and there is little to no manual intervention required.

Continuous Delivery: Continuous Delivery is similar to Continuous Deployment, but the actual deployment to production is done manually or triggered by a human decision. The automated pipeline ensures that the code is always in a deployable state, and the team can choose when to release the changes to

1. What is MVC?
   * MVC stands for Model-View-Controller, and it is a software architectural pattern commonly used in the development of web applications. The MVC pattern separates the application logic into three interconnected components, each with its own responsibilities. This separation allows for better organization, maintainability, and scalability of the codebase.

1. Model:

The Model represents the application's data and business logic. It is responsible for handling data manipulation, database interactions, and business rules. The Model layer encapsulates the application's data and exposes interfaces for other components to interact with it.

2. View:

The View represents the user interface and is responsible for presenting the data to the end-users. It displays the information provided by the Model and allows users to interact with the application. The View is typically responsible for HTML rendering in web applications.

3. Controller:

The Controller acts as an intermediary between the Model and the View. It receives user input from the View, processes it, and updates the Model accordingly. The Controller also retrieves data from the Model and passes it to the appropriate View for display. In web applications, the Controller handles HTTP requests and determines which Model and View to invoke.

1. What is ORM?
   * ORM stands for Object-Relational Mapping. It is a programming technique and a software design pattern that allows developers to interact with relational databases using object-oriented programming principles.
   * In traditional database interactions, data is stored in relational databases, and developers use SQL (Structured Query Language) to query and manipulate the data. However, when working with object-oriented programming languages like Java, C#, Python, etc., data is represented as objects with properties and methods. The process of converting data between these two different paradigms (objects and relational databases) is where ORM comes into play.
2. What is n-tier arch ?
   * N-Tier architecture is a software architecture pattern that separates an application into multiple layers, each with a specific responsibility. The number of layers can vary, but typically includes:
     1. Presentation Tier: Client-side user interface (UI) and user experience (UX)
     2. Application Tier: Server-side business logic and application processing
     3. Business Logic Tier: Server-side business rules and logic
     4. Data Tier: Server-side database and data storage
     5. Integration Tier: Server-side integration with external systems and services
   * Each tier communicates with adjacent tiers through well-defined interfaces, allowing for loose coupling and scalability. N-Tier architecture provides several benefits, including:
     1. Improved maintainability and scalability
     2. Enhanced security through separation of concerns
     3. Easier integration with external systems
     4. Better support for distributed computing and cloud deployments
   * N-Tier architecture is commonly used in large-scale enterprise applications, where complexity and scalability are major concerns.
3. Explain AD?
   * Active Directory is a directory service developed by Microsoft that provides a centralized repository for storing information about objects on a network. It authenticates users and computers, controls access to network resources, and enables single sign-on (SSO) and group policy management.
   * Key Features:
     1. Directory service
     2. Authentication and authorization
     3. Single sign-on (SSO)
     4. Group policy management
   * Benefits:
     1. Simplified user management
     2. Improved security
     3. Scalability
   * Common Use Cases:
     1. User authentication
     2. Computer management
     3. Resource access
4. What is AD? And
   * Active Directory (AD) authenticates users by verifying their credentials against the directory database and issuing Kerberos tickets. Here's a step-by-step overview:
     1. A user attempts to log on to a computer or access a network resource using their credentials.
     2. The user's computer sends an authentication request to a Domain Controller (DC).
     3. The DC verifies the user's credentials and generates a Kerberos ticket.
     4. The DC issues a Ticket-Granting Ticket (TGT) to the user.
     5. The user's computer sends the TGT to the Ticket-Granting Service (TGS).
     6. The TGS verifies the ticket and issues a Service Ticket.
     7. The user's computer presents the Service Ticket to the network resource.
     8. The network resource verifies the ticket and grants access to the user.
   * **Key Components:**
     1. Domain Controller (DC)
     2. Kerberos
     3. Ticket-Granting Ticket (TGT)
     4. Ticket-Granting Service (TGS)
     5. Service Ticket
   * AD ensures that only authorized users can access sensitive data and systems by verifying credentials and issuing tickets.