**Client Server**

1. **Tell us about the features of client/server**.

**Client-server architecture**, architecture of a computer network in which many [clients](https://www.britannica.com/technology/client) (remote processors) request and receive service from a centralized [server](https://www.britannica.com/technology/server) (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the hardware and software) that is providing the service. Clients are often situated at workstations or on personal computers, while servers are located elsewhere on the network, usually on more powerful machines. This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform.

1. **What is a Web server in a client server environment?**

"Web server" can refer to hardware or software, or both of them working together.

* On the hardware side, a web server is a computer that stores web server software and a website's component files (e.g. HTML documents, images, CSS stylesheets, and JavaScript files). It is connected to the Internet and supports physical data interchange with other devices connected to the web.
* On the software side, a web server includes several parts that control how web users access hosted files, at minimum an *HTTP server*. An HTTP server is a piece of software that understands [URLs](https://developer.mozilla.org/en-US/docs/Glossary/URL) (web addresses) and [HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP) (the protocol your browser uses to view webpages). It can be accessed through the domain names (like mozilla.org) of websites it stores and delivers their content to the end-user's device.

1. **What is the role of the presentation layer**

This tier, which is built with HTML5, cascading style sheets (CSS) and Javascript, is deployed to a computing device through a web browser or a web-based application. The presentation tier communicates with the other tiers through application program interface (API) calls

1. **They say this architecture is secure, how is it done in your opinion?**

To enhance the system security of the client computers, Virtual Machine Monitor (VMM) and the Security Manager (SM) were designed and used to form the trusted platform and check the Operating System's integrity outside the Virtual Machine (VM). Plus, a distinct hardware channel and I/O device, the secure I/O, was used to prevent low level attacks, like keyboard sniffers, screen scrapers, and so on.

To improve the reliability of the critical servers, an infrastructure was designed to dynamically construct and configure server farms using a peer-to-peer architecture. The decentralized, self-organization and scalability characteristics of peer-to-peer architecture ensure that the system will have significantly low management cost, while providing highly reliable services in spite of attacks.

1. **What is a Database Server in a client server environment?**

A database server is responsible for database storage, access and processing in a client server environment.

1. **What are Super servers in client server environments?**

A super-server starts other servers when needed, normally with access to them checked by a TCP wrapper. It uses very few resources when in idle state. This can be ideal for workstations used for local web development, client/server development or low-traffic daemons with occasional usage (such as ident and SSH).

1. **Explain 2-Tier and 3-Tier architecture**

**2-Tier Architecture:**

This is where you have direct communication between a **client** and a **server**with no intermediary. It is divided into two parts: client application and a database.

The client in a Two-tier**architecture** application has the code written for saving data in the database. The client sends a request to the server, where it then processes the request and sends back the data. Meaning the client handles both the Presentation layer (application interface) and Application layer (logical operations), while the server system handles the database layer.

**3-Tier Architecture:**

The **Three-tier Architecture** is divided into 3 parts

1. Presentation Layer (Client tier)
2. Application Layer (Business tier)
3. Database Layer (Data tier)

The Presentation Layer is the topmost layer of an application. This is the layer seen when using the software(Interface, web pages). By using the software we access web pages. Its main function is to communicate with the application layer. This layer passes information which is given by the user in terms of keyboard actions and mouse clicks to the application layer.

The Application Layer is also known as the Business logic layer. Here’s where we find logic controls and functionality that processes data received from the presentation layer and database layer. It acts as an intermediary between the presentation and database layer.

The Database Layer is the layer that stores data with the retrieval storage and execution methods made by the application layer. It contains methods that connect to the database and performs the required actions needed. These are Insert, update or delete.

1. **What is a File server?**

A file server is a server that provides access to files. It acts as a central file storage location that can be accessed by multiple systems. File servers are commonly found in enterprise settings, such as company networks, but they are also used in schools, small organizations, and even home networks.

**SOA & MicroServices**

1. **What are the main benefits of SOA?**

* Services are reusable
* Services are easily maintained : One of the advantages of SOA is the elimination of the complexities associated with interacting services within an environment. When it comes to updating, upgrading, and maintaining the services in the SOA environment there are no complications resulting from interactions with other connected and interacting services. This is because the program is complete and self-contained. In environments with complex components, this makes things easy.
* Reliability: SOA services are complete and self-contained programs. This makes it easy for testing, debugging or any form of maintenance.
* Availability: Normally, services in a SOA are available to any requester that needs them. For this reason, SOA services are made available via public directories where they can be searched for and discovered. Consumers, therefore, are always able to access the services.
* Scalability: Different organizations have different levels of complexity in their architecture. So, the ability of services to be successfully run on different servers within an environment increases the scalability and availability of the service.
* High Quality Services: Servers, as we have seen, may have many applications to which they may be applied. As such, there will be no functional redundancy. Data is always consistent, and the service codes are constantly maintained resulting in continuous high-quality applications.
* Platform Independence: One of the main advantages of SOA services is their public availability to any system that needs them, independent of the platform on which the system runs. This platform independence allows the integration of different requested services from different vendors to run harmoniously, regardless of the platform.
* Increased Productivity: The world is constantly changing, and technology is changing and evolving, trying to keep up. Services that have been versatile in their applications need to be constantly improved. Developers using the concepts of previous services need not create every application from scratch but can adapt and reuse old concepts and techniques when evolving into new services and technology.

1. **How can you achieve loose coupling in SOA**

There are a couple of approaches you can take. Our SOA architecture involves XML messages sent to and from the services. One way we achieve what you describe is by avoiding the use of a data binding library to our XML schema and use a generic XML parser to get just the data nodes you want ignoring those you aren't interested in. This way the service can add additional new nodes to the message without breaking anyone currently using it. We typically only do this when we need just one or two pieces of information from a larger schema structure.

Alternatively, the other (preferred) solution we use is versioning. A version of a service adheres to a particular schema/interface. When the schema changes (e.g. the interface is extended or modified), we create a new version of the service. At any time, we may have 2 or 3 versions on the go at any one time. In time, we deprecate and then remove older versions, while eventually migrating dependent code onto newer versions. This way those dependent on the service can continue using the existing version of the service while some particular dependency can 'upgrade' to the new version. Which versions of a service are called are defined in a configuration file for the dependent code. Note that it is not only the schema which gets versioned, but all of the underlying implementation code as well.

1. **Are web services and SOA the same?**

SOA is an architectural concept while web services are used to complete them. Web services are the preferred standards that are satisfied to achieve the architectural specifications of SOA. When one uses SOA all services need to be loosely coupled. Also, SOA services should be able to describe themselves and WSDL services will be describing how we can access the services. Also, these services can be accessed through WSDL. SOA services are located in a directory and here UDDI describes where these web services can be found.

1. **What is a reusable service?**

Reusable service is a stateless functionality that has the required granularity. It can be a part of a composite application or composite server. A reusable service should be identified with any activity prescribed by business and which has its specifications. A service constraint may be security, QoS, SLA or any usage policies. It may be defined by different runtime contracts, multiple interfaces, and different implementations. A reusable service is looked over by enterprise-level throughout its lifecycle, starting from design time through its runtime. Its reuse should also be promoted through a pre-defined process and its reuse can be measured.

1. **What are the disadvantages of SOA?**

* Extra Overload: In SOA, all inputs are validated before it is sent to the service. If you are using multiple services, then it will overload your system with extra computation.
* High Cost: SOA is costly in terms of human resource, development, and technology.
* High Bandwidth Server: As some web service sends and receives messages and information frequently so it easily reaches a million requests per day. So, it involves a high-speed server with a lot of data bandwidth to run a web service.

1. **What is ESB and where does it fit in?**

ESB stands for Enterprise Service Bus. Unlike other relationships, it provided any to any connectivity between different companies. Also, you may need to consider deployment services, IT services, etc. The SOA architecture enables SOA to meet all life’s priorities. The ESB is part of this reference architecture and provides the backbone of an SOA but it should not be considered an SOA by itself.

1. **In SOA do we need to build a system from scratch?**

No, if we need to integrate any existing system you just can loosely couple wrappers which help in wrapping all customer services and expose all functionalities in a generic manner.

1. **What is the most important skill needed to adopt SOA? technical or cultural?**

Surely cultural. SOA does require people to think of business and technology differently. Instead of thinking of technology first (e.g., If we implement this system, what kinds of things can we do with it?), practitioners must first think in terms of business functions, or services (e.g., My company does these business functions, so how can I set up my IT system to do those things for me most efficiently?).

It is expected that adoption of SOA will change business IT departments, creating service-oriented (instead of technology-oriented) IT organizations.

1. **List down the advantages of Microservices Architecture.**

* Better fault isolation; if one microservice fails, the others will continue to work.
* Code for different services can be written in different languages.
* The microservice architecture enables continuous delivery.
* Easy to understand since they represent a small piece of functionality, and easy to modify for developers, thus they can help a new team member become productive quickly.
* The code is organized around business capabilities.
* Scalability and reusability, as well as efficiency. Easy to scale and integrate with third-party services.
* Components can be spread across multiple servers or even multiple data centers.
* Complement cloud activities.
* Microservices simplify security monitoring because the various parts of an app are isolated. A security problem could happen in one section without affecting other areas of the project.
* Increase the autonomy of individual development teams within an organization, as ideas can be implemented and deployed without having to coordinate with a wider IT delivery function.

1. **What are the best practices to design Microservices?**

* Create a Separate Data Store for Each Microservice
* Keep Code at a Similar Level of Maturity
* Do a Separate Build for Each Microservice
* Deploy in Containers
* Treat Servers as Stateless

1. **How does Microservice Architecture work?**
2. **What are the pros and cons of Microservice Architecture?**

**Pros:**

* Greater agility
* Better scalability
* Faster development cycles (easier deployment and debugging)
* Easier to create a CI/CD pipeline for single-responsibility services
* Isolated services have better fault tolerance
* Platform- and language agnostic services
* Cloud-readiness
* Faster time to market

**Cons:**

* Needs more collaboration (each team has to cover the whole microservice lifecycle)
* Harder to test and monitor because of the complexity of the architecture
* Poorer performance, as microservices need to communicate (network latency, message processing, etc.)
* Harder to maintain the network (has less fault tolerance, needs more load balancing, etc.)
* Doesn’t work without the proper corporate culture (DevOps culture, automation practices, etc.)
* Security issues (harder to maintain transaction safety, distributed communication goes wrong more likely, etc.)

1. **What is the difference between Monolithic, SOA and Microservices Architecture?**
2. **What are the challenges you face while working Microservice Architectures?**

* Data synchronization (Consistency)
* Security
* Services Communication
* Data Staleness
* Testing: Testing is much more complex in a microservices environment due to the different services, their integration, and interdependencies. The team members responsible for quality assurance need to be knowledgeable on the order and channels of communications between services to have full coverage in their test cases. The asynchronous aspect of microservices also makes it harder to test in lower environments. Indistinct behaviors from microservices are harder to predict and validate.
* Monitoring and Performance : The traditional forms of monitoring and diagnostics will not align well with microservices since you have multiple services making up the same functionality previously supported by a single application. When a problem arises in the application, finding the root cause can be challenging if you do not have a means of monitoring and tracking the path a specific request took, like how many and which microservices were traversed for a specific request coming from a user interface.
* Embracing DevOps culture: Separate teams need agility, autonomy, and continuous delivery to be able to deliver initial releases and subsequent iterative changes. A lack of DevOps culture can bottle up releases and impact the overall time to market and the response to business requests and issues.
* Fault Tolerance: It is important that individual services do not bring down the overall system. Fault tolerance at the service level, and more importantly, at the overall solution level, is critical. Given the complexity of a microservices environment and the complex dependency chains, failure is inevitable. Microservices need to be able to withstand both internal and external failures. Robust resiliency testing is key to successful issue preparedness.
* Design with failure in mind: While this is counter-intuitive to many, expecting failure scenarios and building a robust set of microservices is imperative to a successful implementation. When more failure situations are predicted during design, the more exception handling mechanisms will be built, and seamless resolution of issues will be handled better. This is easier said than done.
* Cyclic dependencies: Dependency management across different services and their functionality is very important and cyclic dependencies can be a headache if not identified and resolved promptly. In microservice architecture, you’re even more vulnerable to errors coming from dependency issues. Decisions made around upgrades on related services with these dependencies are critical.

1. **What are the characteristics of Microservices?**

* Microservice architectural style is an approach to developing a single application as a suite of small services.
* Services are built around business capabilities, independently deployable and packaged, each running in its own process.
* Each Service should have separate database layer.
* Each Service can have independent codebase, CI/CD tooling sets.
* Each Service can be tested in isolation without dependent on other services
* Each service can pick the best technology stack for its use cases (no need to stick into one framework throughout the entire application).
* Each Service should have monitoring and troubleshooting capabilities for operation team
* Services should implement Retry functionality in case of network failure or system failure
* Each Service can implement independent security mechanism
* The main difference between SOA and Microservices is that a microservice employs a practice that attempts to eliminate any dependencies on other microservices. SOA does not make this  
  practice explicit as a requirement.
* Services can use HTTP(Rest) or messaging for communication or any other lightweight communication protocol.
* Well understood Distribution Transaction management
* Presenting API
* Clean and Clear Separation of Stateless and Stateful Services
* Do Not Share Libraries or SDKs
* Each Service can be run without waiting for other service to go online
* Implement a Self-Registration and Discovery Mechanism
* Service can use different language, framework and technologies
* Maintain Independent Revisions and Build Environments to maintains compatibility with other services.
* An architect can explicitly check for rules and constraints.

**HTTP Fundamentals:**

1. **What are the basic Features of HTTP?**

The basic features of HTTP are as follows:

* **HTTP is media independent:** It specifies that any type of media content can be sent by HTTP as long as both the server and the client can handle the data content.
* **HTTP is connectionless:** It is a connectionless approach in which HTTP client i.e., a browser initiates the HTTP request and after the request is sent the client disconnects from server and waits for the response.
* **HTTP is stateless:** The client and server are aware of each other during a current request only. Afterwards, both of them forget each other. Due to the stateless nature of protocol, neither the client nor the server can retain the information about different request across the web pages.

1. **What are request methods in HTTP?**

The following are the request methods in HTTP:

* **GET:** The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
* **HEAD:** The HEAD method asks for a response identical to that of a GET request, but without the response body.
* **POST:** The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
* **PUT:** The PUT method replaces all current representations of the target resource with the request payload.
* **DELETE:** The DELETE method deletes the specified resource.
* **CONNECT:** The CONNECT method establishes a tunnel to the server identified by the target resource.
* **OPTIONS:** The OPTIONS method is used to describe the communication options for the target resource
* **TRACE:** The TRACE method performs a message loop-back test along the path to the target resource.
* **PATCH:** The PATCH method is used to apply partial modifications to a resource.

1. **What are the differences between GET and POST methods?**

The differences between GET and POST methods are as follows:

* GET requests can be cached whereas POST requests are never cached
* GET requests remain in the browser history whereas POST requests do not remain in the browser history
* GET requests have length restrictions whereas POST requests have no restrictions on data length
* GET requests are only used to request data (not modify) whereas POST requests are used to post data
* GET requests can be bookmarked whereas POST requests cannot be bookmarked

1. **What is status code in HTTP?**

HTTP Status codes are issued by a server in response to a client's request

made to the server

1. **What are the header fields in HTTP?**

HTTP header fields provide required information about the request or response, or about the object sent in the message body

There are four types of HTTP message headers:

* **General-header:** These header fields have general applicability for both request and response messages.
* **Client Request-header:** These header fields have applicability only for request messages.
* **Server Response-header:** These header fields have applicability only for response messages.
* **Entity-header:** These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.

1. **What is URI?**

A Uniform Resource Identifier (URI) is a string of characters that unambiguously identifies a particular resource.

1. **What are Idempotent methods and why do we call them?**

An idempotent HTTP method isa HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Again, this only applies to the result,

not the result itself.

1. **Explain HTTP Request & Response Messages**

HTTP requests, and responses, share similar structure and are composed of:

1. A *start-line* describing the requests to be implemented, or its status of whether successful or a failure. This start-line is always a single line.
2. An optional set of *HTTP headers* specifying the request, or describing the body included in the message.
3. A blank line indicating all meta-information for the request has been sent.
4. An optional *body* containing data associated with the request (like content of an HTML form), or the document associated with a response. The presence of the body and its size is specified by the start-line and HTTP headers.
5. **What is Session State in HTTP?**

**HTTP sessions** is an industry standard feature that allows Web servers to maintain user identity and to store user-specific data during multiple request/response interactions between a client application and a Web application

1. **What is HTTPS?**

Hypertext Transfer Protocol Secure is an extension of the Hypertext Transfer Protocol. It is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security or, formerly, Secure Sockets Layer.

**Introduction to API**

1. **Explain REST and RESTFUL?**

Representational state transfer (REST) is a style of software architecture. As described in a dissertation by Roy Fielding, REST is an "architectural style" that basically exploits the existing technology and protocols of the Web. RESTful is typically used to refer to web services implementing such an architecture.

1. **Mention what are the HTTP methods supported by REST?**

The REST supports a subset of HTTP methods to manipulate the requests and responses processed by GlassFish ESB. Certain HTTP methods, such as GET and HEAD, do not make any changes to the resource or to the message and are thus considered to be “safe”. Other method are idempotent, which means that the results of multiple identical request are the same as for a single request. All supported methods except POST fall under this category.

The following subset of HTTP methods are supported for the REST BC:

* **GET:** The GET method retrieves specific information from the server as identified by the request URI.
* **PUT**: The PUT method requests that the message body sent with the request be stored under the location provided in the HTTP message.
* **DELETE**: The DELETE method deletes the specified resources.
* **POST**: The POST method modifies data on the server from which a request was sent.
* **HEAD**: The HEAD method is similar to the GET method except the message body is not returned in the response. The response only includes metainformation, such as a response code or corresponding headers.

1. **Explain the architectural style for creating web API?**

REST is an architectural style for services. The REST (Representational State Transfer) style is a set of software engineering practices that contains constraints that should be used in order to create web services in distributed hypermedia systems. REST is not a tool and neither is it a language; in fact, REST is agnostic of protocols, components, and languages.

It is important to say that REST is an architectural style and not a toolkit. REST provides a set of design rules in order to create stateless services that are shown as resources and, in some cases, sources of specific information such as data and functionality. The identification of each resource is performed by its unique **Uniform Resource Identifier** (**URI**). REST describes simple interfaces that transmit data over a standardized interface such as HTTP and HTTPS without any additional messaging layer, such as **Simple Object Access Protocol** (**SOAP**).

1. **Explain the RESTFul Web Service?**

**RESTful web services** are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using **Uniform Resource Identifiers (URIs)**, typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

1. **Explain what is a “Resource” in REST?**

The fundamental concept in any RESTful API is the resource. A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. ... In this case, we refer to these resources as singleton resources. Collections are themselves resources as well.

1. **Which protocol is used by RESTful web services?**

RESTful web services use a famous web protocol i.e. HTTP protocol. This serves as a medium of data communication between client and server. HTTP standard methods are used to access resources in RESTful web service architecture.

1. **What is messaging in RESTful web services?**

Advertisements. RESTful Web Services make use of HTTP protocols as a medium of communication between client and server. A client sends a message in form of a HTTP Request and the server responds in the form of an HTTP Response. This technique is termed as Messaging.

1. **State the core components of an HTTP Request?**

There are 5 major components for HTTP Request.

* Verb − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
* URI − Uniform Resource Identifier (URI) to identify the resource on server.
* HTTP Version − Indicate HTTP version, for example HTTP v1.1 .
* Request Header − Contains metadata for the HTTP Request message as key-value pairs. For example, client ( or browser) type, format supported by client, format of message body, cache settings etc.
* Request Body − Message content or Resource representation

1. **State the core components of an HTTP response?**

There are 5 major components for HTTP Response.

* Verb − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
* URI − Uniform Resource Identifier (URI) to identify the resource on server.
* HTTP Version − Indicate HTTP version, for example HTTP v1.1 .
* Request Header − Contains metadata for the HTTP Request message as key-value pairs. For example, client ( or browser) type, format supported by client, format of message body, cache settings etc.
* Request Body − Message content or Resource representation

1. **What do you understand about payload in RESTFul web service?**

The payload is the part of that response that is communicating directly to you. In REST APIs this is usually some JSON formatted data. ... The payload is the part of that response that is communicating directly to you. In REST APIs this is usually some JSON formatted data.

1. **Explain the caching mechanism?**

When a caching mechanism is in place, it helps improve delivery speed by storing a copy of the asset you requested and later accessing the cached copy instead of the original. ... Otherwise, if the file is stored in cache, this results in a cache hit response and the asset is delivered from cache.

1. **List the main differences between SOAP and REST?**

The differences are as follows:

* **SOAP is a protocol. REST is an architectural style.** An API is designed to expose certain aspects of an application’s business logic on a server, and SOAP uses a service interface to do this while REST uses URIs.
* **REST APIs access a resource for data (a URI); SOAP APIs perform an operation.** REST is an architecture that’s more data-driven; SOAP is a standardized protocol for transferring structured information that’s more function-driven.
* **REST permits many different data formats** including plain text, [HTML, XML](https://www.upwork.com/hiring/development/the-basics-of-web-development/), and JSON, which is a great fit for data and yields more browser compatibility; SOAP only uses XML.
* **Security is handled differently.** SOAP supports WS-Security, which is great at the transport level and a bit more comprehensive than SSL, and more ideal for integration with enterprise-level security tools. Both support SSL for end-to-end security, and REST can use the secure version of the HTTP protocol, HTTPS.
* **SOAP requires more bandwidth; REST requires fewer resources** (depending on the API). There’s a little more overhead with SOAP out of the gate, on account of the envelope-style of payload transport. Because REST is used primarily for web services, its being lightweight is an advantage in those scenarios.
* **REST calls can be cached, SOAP-based calls cannot be cached.** Data can be marked as cacheable, which means it can be reused by the browser later without having to initiate another request back to the server. This saves time and resources.
* **An API is built to handle your app’s payload, and REST and SOAP do this differently.** A payload is data sent over the internet, and when a payload is “heavy,” it requires more resources. REST tends to use HTTP and JSON, which lighten the payload; SOAP relies more on XML.

1. **Enlist advantages and disadvantages of ‘Statelessness’.**

The following are some of the advantages:

* As the server does not need to manage any session, deploying the services to any number of servers is possible, and so scalability will never be a problem
* No states equals less complexity; no session (state) synchronize logic to handle at the server side
* As the service calls (requests) can be cached by the underlying application, the statelessness constraint brings down the server's response time, that is, it improves performance with regard to response time
* Seamless integration/implementation with HTTP protocols is possible as HTTP is itself a stateless protocol

The disadavantage is web services need to get extra information in each request and then interpret to get the client's state in case the client interactions are to be taken care of.

**Object Oriented Programming Fundamentals**

1. **What is the main difference between a class and an object?**

A class is a blueprint from which you can create the instance, i.e., objects. An object is the instance of the class, which helps programmers to use variables and methods from inside the class. A class is used to bind data as well as methods together as a single unit. object acts as a variable of the class.

1. **What is Encapsulation? Explain with a used case**

In object-oriented computer programming languages, the notion of encapsulation refers to the bundling of data, along with the methods that operate on that data, into a single unit. Many programming languages use encapsulation frequently in the form of classes.

A class is a program-code-template that allows developers to create an object that has both variables (data) and behaviors (functions or methods). A class is an example of encapsulation in that it consists of data and methods that have been bundled into a single unit.

1. **What is Polymorphism? Explain with a used case.**

**Polymorphism** is the ability of an object to take on many forms. The most common **use** of **polymorphism** in OOP occurs when a parent class reference is **used** to refer to a child class object. Any Java object that can pass more than one IS-A test is considered to be **polymorphic**

**Real life example of polymorphism:** A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

1. **Explain Overriding & Overloading and its advantages.**

Overloading occurs when two or more methods in one class have the same method name but different parameters.

Overriding allows a child class to provide a specific implementation of a method that is already provided its parent class.

**Advantages:**

* The one main advantage of these overriding and [overloading is time-saving](https://www.educba.com/python-overloading/).
* Save memory space.
* The readability of the code is increased.
* Here, for [function overloading concept](https://www.educba.com/function-overloading-in-c-plus-plus/), we can use different same function names for different operations eliminating the use of different function names.
* Flexibility and maintainability of code become easier.
* In the case of overriding, the child class can have functions of parent class and can even have its own implementation of that function.
* The objects can be referenced and the functions of both the parent class and child class can be accessed by the child class.

1. **What is Inheritance and different types of inheritance? Explain with a used case.**

**Inheritance** is a mechanism in which one class acquires the property of another class. For example, a child inherits the traits of his/her parents. With inheritance, we can reuse the fields and methods of the existing class. Hence, inheritance facilitates Reusability and is an important concept of OOPs.

Below are Various types of inheritance in Java. We will see each one of them one by one with the help of examples and flow diagrams.

### 1) Single Inheritance

**Single inheritance** is damn easy to understand. When a class extends another one class only then we  call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a **parent class** of B and B would be  a **child class** of A.

### [Single Inheritance](https://beginnersbook.com/wp-content/uploads/2013/05/Single-Inheritance.png)

**Single Inheritance example program in Java**

Class A

{

public void methodA()

{

System.out.println("Base class method");

}

}

Class B extends A

{

public void methodB()

{

System.out.println("Child class method");

}

public static void main(String args[])

{

B obj = new B();

obj.methodA(); //calling super class method

obj.methodB(); //calling local method

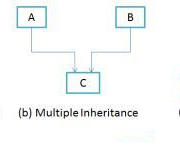
}

}

### 2) Multiple Inheritance

“**Multiple Inheritance**” refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with “multiple inheritance” is that the derived class will have to manage the dependency on two base classes.



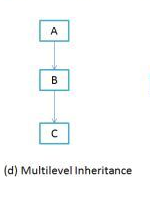
[](https://beginnersbook.com/wp-content/uploads/2013/05/Multiple-Inheritance.png)

Note 1: Multiple Inheritance is very rarely used in software projects. Using Multiple inheritance often leads to problems in the hierarchy. This results in unwanted complexity when further extending the class.

Note 2: Most of the new OO languages like **Small Talk, Java, C# do not support Multiple inheritance**. Multiple Inheritance is supported in C++.

### 3) Multilevel Inheritance

**Multilevel inheritance** refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A. For more details and example refer – [Multilevel inheritance in Java](https://beginnersbook.com/2013/12/multilevel-inheritance-in-java-with-example/).

[](https://beginnersbook.com/wp-content/uploads/2013/05/Multilevel-Inheritance.png)

**Multilevel Inheritance example program in Java**

Class X

{

public void methodX()

{

System.out.println("Class X method");

}

}

Class Y extends X

{

public void methodY()

{

System.out.println("class Y method");

}

}

Class Z extends Y

{

public void methodZ()

{

System.out.println("class Z method");

}

public static void main(String args[])

{

Z obj = new Z();

obj.methodX(); //calling grand parent class method

obj.methodY(); //calling parent class method

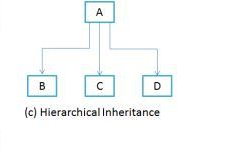
obj.methodZ(); //calling local method

}

}

### 4) Hierarchical Inheritance

In such kind of inheritance one class is inherited by many**sub classes**. In below example class B,C and D **inherits** the same class A. A is **parent class (or base class)** of B,C & D. Read More at – [Hierarchical Inheritance in java with example program](https://beginnersbook.com/2013/10/hierarchical-inheritance-java-program/).

[](https://beginnersbook.com/wp-content/uploads/2013/05/Hierarchical-Inheritance.png)

### 5) Hybrid Inheritance

In simple terms you can say that Hybrid inheritance is a combination of**Single** and **Multiple inheritance.** A typical flow diagram would look like below. A hybrid inheritance can be achieved in the java in a same way as multiple inheritance can be!! Using interfaces. yes you heard it right. By using **interfaces** you can have multiple as well as **hybrid inheritance** in Java.

1. **What is an abstract class?**

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

1. **What is an interface and how multiple inheritance is achieved with this.**

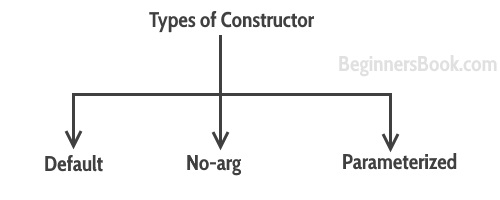
An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces

1. **What are the access modifiers?**

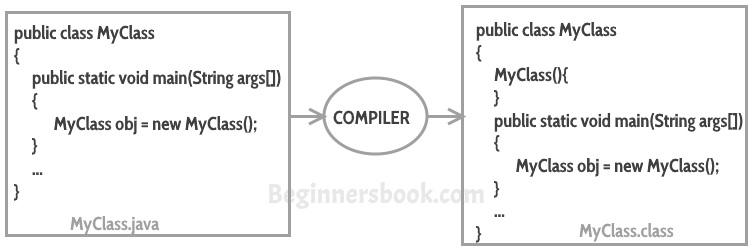
Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components. The access modifiers are public, private and protected.

1. **What are the various types of constructors?**

Constructor is a block of code that initializes the newly created object. A constructor resembles an instance method in java but it’s not a method as it doesn’t have a return type. In short constructor and method are different(More on this at the end of this guide). People often refer constructor as special type of method in Java.

here are three types of constructors: Default, No-arg constructor and Parameterized.  


### Default constructor

If you do not implement any constructor in your class, Java compiler inserts a [default constructor](https://beginnersbook.com/2014/01/default-constructor-java-example/) into your code on your behalf. This constructor is known as default constructor. You would not find it in your source code(the java file) as it would be inserted into the code during compilation and exists in .class file. This process is shown in the diagram below:  


If you implement any constructor then you no longer receive a default constructor from Java compiler.

### no-arg constructor:

Constructor with no arguments is known as **no-arg constructor**. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.

Although you may see some people claim that that default and no-arg constructor is same but in fact they are not, even if you write **public Demo() { }** in your class Demo it cannot be called default constructor since you have written the code of it.

#### Example: no-arg constructor

class Demo

{

public Demo()

{

System.out.println("This is a no argument constructor");

}

public static void main(String args[]) {

new Demo();

}

}

Output:  
This is a no argument constructor

### Parameterized constructor

Constructor with arguments(or you can say parameters) is known as [Parameterized constructor](https://beginnersbook.com/2014/01/parameterized-constructor-in-java-example/).

#### Example: parameterized constructor

In this example we have a parameterized constructor with two parameters id and name. While creating the objects obj1 and obj2 I have passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

public class Employee {

int empId;

String empName;

//parameterized constructor with two parameters

Employee(int id, String name){

this.empId = id;

this.empName = name;

}

void info(){

System.out.println("Id: "+empId+" Name: "+empName);

}

public static void main(String args[]){

Employee obj1 = new Employee(10245,"Chaitanya");

Employee obj2 = new Employee(92232,"Negan");

obj1.info();

obj2.info();

}

}

**Output:**

Id: 10245 Name: Chaitanya

Id: 92232 Name: Negan

#### Example2: parameterized constructor

In this example, we have two constructors, a default constructor and a parameterized constructor. When we do not pass any parameter while creating the object using new keyword then default constructor is invoked, however when you pass a parameter then parameterized constructor that matches with the passed parameters list gets invoked.

class Example2

{

private int var;

//default constructor

public Example2()

{

this.var = 10;

}

//parameterized constructor

public Example2(int num)

{

this.var = num;

}

public int getValue()

{

return var;

}

public static void main(String args[])

{

Example2 obj = new Example2();

Example2 obj2 = new Example2(100);

System.out.println("var is: "+obj.getValue());

System.out.println("var is: "+obj2.getValue());

}

}

**Output:**

var is: 10

var is: 100

1. **What is ‘this’ pointer?**

this is a keyword in **Java**. It can be used inside the method or constructor of a class. It(this) works as a reference to the current object, whose method or constructor is being invoked. This keyword can be used to refer to any member of the current object from within an instance method or a constructor

1. **What is static and dynamic Binding?**

Static binding uses Type information for binding while Dynamic binding uses Objects to resolve binding. Overloaded methods are resolved (deciding which method to be called when there are multiple methods with same name) using static binding while overridden methods using dynamic binding, i.e, at run time.

1. **How many instances can be created for an abstract class and why?**

No you can't create an instance for an abstract class, instead you can create instance of all other classes extending that abstract class. Because it's abstract and an object is concrete. An abstract class is sort of like a template, or an empty/partially empty structure, you have to extend it and build on it before you can use it.

1. **Which OOPS concept is used as a reuse mechanism and explain with a use case**

Reusability in OOP achieves through the features of C++ where it possible to extend or reuse the properties of parent class or superclass or base class in a subclass and in addition to that, adding extra more features or data members in the subclass or child class or derived class.

1. **Please identify one practical scenario for each pillar of OOPs.**

* Abstraction allows us to expose limited data and functionality of objects publicly and hide the actual implementation. It is the most important pillar in OOPS. In our example of Mobile class and objects like Nokia, Samsung, IPhone.

Some features of mobiles,

1. Dialing a number call some method internally which concatenate the numbers and displays it on screen but what is it doing we don’t know.
2. Clicking on green button actual send signals to calling person's mobile but we are unaware of how it is doing.

This is called abstraction. In classes, we can create methods that can be called and used by the users of the class but users will have no idea what these methods do internally.

* Encapsulation is defined as the process of enclosing one or more details from outside world through access right. It says how much access should be given to particular details. Both Abstraction & Encapsulation works hand in hand because Abstraction says what details to be made visible and Encapsulation provides the level of access right to that visible details. i.e. – It implements the desired level of abstraction.

Talking about Bluetooth which we usually have it in our mobile. When we switch on a Bluetooth, I am able to connect to another mobile or bluetooth enabled devices but I'm not able to access the other mobile features like dialing a number, accessing inbox etc. This is because, Bluetooth feature is given some level of abstraction.

Another point is when mobile A is connected with mobile B via Bluetooth whereas mobile B is already connected to mobile C then A is not allowed to connect C via B. This is because of accessibility restriction.

* Polymorphism can be defined as the ability of using the same name for doing different things. More precisely we say it as 'many forms of single entity'. This play a vital role in the concept of OOPS.

Let's say Samsung mobile has a 5MP camera available i.e. – it is having a functionality of CameraClick(). Now same mobile is having Panorama mode available in camera, so functionality would be same but with mode. This type is said to be Static polymorphism or Compile time polymorphism

* Inheritance is the ability to extend the functionality from base entity in new entity belonging to same group. This will help us to reuse the functionality which is already defined before and extend into a new entity.

Basic Mobile functionality is to send a message, dial and receive a call. So the brands of mobile is using this basic functionality by extending the mobile class functionality and adding their own new features to their respective brand.

**Unit Testing & Junit**

1. **What is unit testing?**

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

1. **What is the difference between manual testing and automated testing?**

In manual testing (as the name suggests), test cases are executed manually (by a human, that is) without any support from tools or scripts. But with automated testing, test cases are executed with the assistance of tools, scripts, and software

1. **Is it necessary to write the test case for every logic? If yes, why.**

Ideallyyou should do that, but not many people actually do that. The important thing is to write tests ideally before writing code, or right after you wrote the code. ... It is most important to test completely business logic and helper classes. Other parts of application are less important, but you can test some parts.

1. **What are the features of JUnit?**

Features of Junit are as follows:

* JUnit is an open source framework, which is used for writing and running tests.
* Provides annotations to identify test methods.
* Provides assertions for testing expected results.
* Provides test runners for running tests.
* JUnit tests allow you to write codes faster, which increases quality.
* JUnit is elegantly simple. It is less complex and takes less time.
* JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
* JUnit tests can be organized into test suites containing test cases and even other test suites.
* JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

1. **What are the important JUnit annotations? And its usage in coding.**

Here’re some basic JUnit annotations you should understand:

1. @BeforeClass – Run once before any of the test methods in the class, public static void
2. @AfterClass – Run once after all the tests in the class have been run, public static void
3. @Before – Run before @Test, public void
4. @After – Run after @Test, public void
5. @Test – This is the test method to run, public void
6. **What does Assert class?**

Assert is a method useful in determining Pass or Fail status of a test case, The assert methods are provided by the class org. ... Assert which extends java. lang. Object class.

1. **What is Code Coverage?**

Code coverage is a measurement of how many lines/blocks/arcs of your code are executed while the automated tests are running. Code coverage is collected by using a specialized tool to instrument the binaries to add tracing calls and run a full set of automated tests against the instrumented product

1. **What are the best practices to perform Unit Testing?**

* **Unit Tests** Should Be Trustworthy. ...
* **Unit Tests** Should Be Maintainable and Readable. ...
* **Unit Tests** Should Verify a Single-Use Case. ...
* **Unit Tests** Should Be Isolated. ...
* **Unit Tests** Should Be Automated. ...
* Use a **Good** Mixture of **Unit** and Integration **Tests**. ...
* **Unit Tests** Should Be Executed Within an Organized **Test Practice**

1. **What is Mocking?**

Mocking is creating objects that simulate the behavior of real objects. ... If you instead want to verify that the object under test writes some specific data to the database you will have to mock the database. Your test would then incorporate assertions about what was written to the database mock.

**GIT:**

1. **What is GIT and its significance in SDLC**

Git is a version control system for tracking changes in files and coordinating work on those files among multiple people. It is primarily used for source code management in software development. It is a distributed revision control system and is very useful to support software development workflows

1. **What is the difference between GIT and SVN?**

The difference between Git and SVN version control systems is that Git is a distributed version control system, whereas SVN is a centralized version control system. Git uses multiple repositories including a centralized repository and server, as well as some local repositories

1. **What are the advantages of using GIT?**

One of the biggest advantages of Git is its branching capabilities. Unlike centralized version control systems, Git branches are cheap and easy to merge. This facilitates the feature branch workflow popular with many Git users. Feature branches provide an isolated environment for every change to your codebase.

1. **What is “Staging Area” or “Index” in GIT?**

The Git index is used as a staging area between your working directory and your repository. You can use the index to build up a set of changes that you want to commit together. When you create a commit, what is committed is what is currently in the index, not what is in your working directory.

1. **What is GIT stash?**

GIT stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on

1. **What is the function of git clone?**

GIT clone is primarily used to point to an existing repository and make a clone or copy of that repo at in a new directory, at another location. The original repository can be located on the local filesystem or on remote machine accessible supported protocols.

1. **How can you create a repository in Git?**

* **Create** a directory to contain the project.
* Go into the new directory.
* Type **git** init .
* Write some code.
* Type **git** add to add the files (see the typical use page).
* Type **git** commit

1. **What is the purpose of branching in GIT?**

In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.

1. **What is the difference between ‘git remote’ and ‘git clone’?**

git remote add just creates an entry in your git config that specifies a name for a particular URL. You must have an existing git repo to use this. git clone creates a new git repository by copying an existing one located at the URI you specify.

1. **What is the function of ‘git diff ’ in git?**

Diff command is used in git to track the difference between the changes made on a file. Since Git is a version control system, tracking changes are something very vital to it. Diff command takes two inputs and reflects the differences between them. It is not necessary that these inputs are files only.

1. **Explain what the commit message is?**

Commit messages can do exactly that and as a result, a commit message shows whether a developer is a good collaborator. If you haven't given much thought to what makes a great Git commit message, it may be the case that you haven't spent much time using git log and related tools

1. **Why is it advisable to create an additional commit rather than amending an existing commit?**

The amend operation will destroy the state that was previously saved in a commit. If it's just the commit message being changed then that's not an issue. But if the contents are being amended then chances of eliminating something important remains more.

1. **What is Rebasing**

In Git, the rebase command integrates changes from one branch into another. It is an alternative to the better known "merge" command. Most visibly, rebase differs from merge by rewriting the commit history in order to produce a straight, linear succession of commits.

**Maven Fundamentals**

1. **Explain what is Maven? How does it work?**

Maven is a popular open source build tool for enterprise Java projects, designed to take much of the hard work out of the build process. Maven uses a declarative approach, where the project structure and contents are described, rather then the task-based approach used in Ant or in traditional make files, for example.

1. **Explain what is POM and its significance**

POM stands for Project Object Model. It is fundamental unit of work in Maven. The POM contains information about the project and various configuration detail used by Maven to build the project(s).

1. **Explain what a Maven artifact is?**

An artifact is a file, usually a JAR, that gets deployed to a Maven repository. A Maven build produces one or more artifacts, such as a compiled JAR and a "sources" JAR. Each artifact has a group ID (usually a reversed domain name, like com. example. foo), an artifact ID (just a name), and a version string.

1. **List out the dependency scope in Maven?**

Maven dependency scope attribute is used to specify the visibility of a dependency, relative to the different lifecycle phases (build, test, runtime etc). Maven provides six scopes i.e. compile , provided , runtime , test , system , and import .

1. **List out what are the build phases in Maven?**

A Maven phase represents a stage in the Maven build lifecycle. Each phase is responsible for a specific task.

Here are some of the most important phases in the *default* build lifecycle:

* *validate:* check if all information necessary for the build is available
* *compile:* compile the source code
* *test-compile:* compile the test source code
* *test:* run unit tests
* *package:* package compiled source code into the distributable format (jar, war, …)
* *integration-test:* process and deploy the package if needed to run integration tests
* *install:* install the package to a local repository
* *deploy:* copy the package to the remote repository

1. **Mention the three build lifecycle of Maven?**

There are three built-in build lifecycles: default, clean and site. The default lifecycle handles your project deployment, the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

1. **List out what are the aspects does Maven Manages?**

Following are the aspects which maven manages:

* Build.
* Documentation.
* Reporting.
* Dependencies.
* SCMs.
* Releases.
* Distribution.
* Mailing **list**.

1. **Explain what a Maven Repository is? What are their types?**

A **maven repository** is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

1. Local Repository
2. Central Repository
3. Remote Repository
4. **Explain how you can exclude dependency?**

Following are the steps:

1. Right-click on the project.
2. Choose [delete]
3. Do NOT check "Delete project contents on disk"
4. Click OK.
5. Wait for it to complete.
6. Import your project to your workspace again (Existing Maven Projects) Now the unused JAR files are gone.
7. **For POM what are the minimum required elements?**

All POM files require the project element and three mandatory fields: groupId, artifactId, version. Projects notation in repository is groupId:artifactId:version.

**CI/CD**

1. **What are the fundamental differences between DevOps & Agile?**

**DevOps** is a practice of bringing development and operations teams together whereas **Agile** is an iterative approach that focuses on collaboration, customer feedback and small rapid releases. **DevOps** focuses on constant testing and delivery while the **Agile** process focuses on constant changes.

1. **What is the need for DevOps?**

)**DevOps** describes a culture and set of processes that bring development and operations teams together to complete software development. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches.

1. **What are the advantages of DevOps?**

Increase productivity of business and IT teams. Save costs on maintenance and upgrades, and eliminate unnecessary capital expenditure. Standardize processes for easy replication and faster delivery. Improve quality, reliability and reusability of all system components.

1. **Explain with a use case where DevOps can be used in industry/ real-life.**

Etsy is a peer-to-peer e-commerce website focused on handmade or vintage items and supplies, as well as unique factory-manufactured items. Etsy struggled with slow, painful site updates that frequently caused the site to go down. It affected sales for millions of Etsy’s users who sold goods through an online marketplace and risked driving them to the competitor.

With the help of a new technical management team, Etsy transitioned from its waterfall model, which produced four-hour full-site deployments twice weekly, to a more agile approach. Today, it has a fully automated deployment pipeline, and its continuous delivery practices have reportedly resulted in more than 50 deployments a day with fewer disruptions.

1. **What are the success factors for Continuous Integration?**

* Closing the Loop. At the center of a **continuous integration** strategy is an automated and repeatable build.
* Communicate Results. When a build fails, whether due to a compiler error or test failure, all team members should be notified.
* Making the Application Accessible.
* Conclusion.

1. **What are the differences between continuous integration, continuous delivery, and continuous deployment?**

### Continuous integration

Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer's changes are validated by creating a build and running automated tests against the build. By doing so, you avoid the integration hell that usually happens when people wait for release day to merge their changes into the release branch.

Continuous integration puts a great emphasis on testing automation to check that the application is not broken whenever new commits are integrated into the main branch.

### Continuous delivery

[Continuous delivery](https://www.atlassian.com/continuous-delivery) is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button.

In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

### Continuous deployment

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there isn't a Release Day anymore. Developers can focus on building software, and they see their work go live minutes after they've finished working on it.

1. **What role does the Quality Assurance (QA) team play in DevOps?**

QA further enables and directs teams to check the application for performance and quality before it is delivered. The role of QA in the current software development process is changing, where the customer comes into focus and the needs of the customer are brought into perspective while delivering the application

1. **Describe an efficient workflow for continuous integration**

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied.

1. **What are the best practices for DevOps implementation?**

* Have a Centralized Unit for DevOps.
* Shift Left With CI/CD.
* Implement Test Automation.
* Implement Continuous Deployment.
* Keep All Teams on the Same Page.
* Exercise Patience.
* Implement Automation in Dashboards.

1. **How will you approach when a project needs to implement DevOps?**

* Evaluate the need to **implement DevOps** practice.
* Break the organizational silos & encourage collaboration.
* Put Customer / end-user satisfaction at the center.
* Don't jump start, instead, start small and then scale up.
* Automate wherever possible.
* Select tools that are compatible with each other
* **Define performance reviews for team and an individua**
* **Ensure real-time visibility into the project**
* **Integrate and deliver continuously**
* **Achieve better results with monitoring & feedback**