**Software Development Cycle**

1. Explain SDLC at a high level

Software Development Life Cycle, or SDLC is a process used to develop software. There are different stages or phases within the software development life cycle and in each phase, different activities take place.

SDLC creates a structure for the development teams to be able to design, create and deliver high quality software by defining various tasks that need to happen The life cycle defines a methodology for improving the quality of software and the overall development process.

The intent of a SDLC process it to help produce a product that is cost-efficient, effective, and of high quality.

There are different phases of SDLC. They are:

* 1. Requirement Analysis
  2. Design
  3. Implementation
  4. Testing
  5. Deployment and Maintenance

1. What is waterfall and why it is still relevant?

Waterfall model is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase.

Waterfall is still relevant because of the few advantages which cannot be achieved by other methodology and they are:

* 1. Adapts to Shifting Teams: While not necessarily specific to the waterfall model only, using a waterfall method does allow the project as a whole to maintain a more detailed, robust scope and design structure due to all the upfront planning and documentation stages. This is particularly well suited to large teams that may see members come and go throughout the life cycle of the project, allowing the burden of design to be placed on the core documentation and less on any individual team member.
  2. Forces Structured Organization: While some may argue this is a burden rather than a benefit, the fact remains that the waterfall model forces the project, and even the organization building said project, to be extraordinarily disciplined in its design and structure. Most sizable projects will, by necessity, include detailed procedures to manage every aspect of the project, from design and development to testing and implementation.
  3. Allows for Early Design Changes: While it can be difficult to make design changes later in the process, the waterfall approach lends itself well to alterations early in the life cycle. This is great when fleshing out the specification documents in the first couple stages with the development team and clients, as alterations can be made immediately and with minimal effort, since no coding or implementation has actually taken place up to that point.
  4. Suited for Milestone-Focused Development: Due to the inherent linear structure of a waterfall project, such applications are always well-suited for organizations or teams that work well under a milestone- and date-focused paradigm. With clear, concrete, and well understood stages that everyone on the team can understand and prepare for, it is relatively simple to develop a time line for the entire process and assign particular markers and milestones for each stage and even completion. This isn’t to suggest software development isn’t often rife with delays (since it is), but waterfall is befitting the kind of project that needs deadlines.

1. Explain Agile Model with a use case and the role of SCRUM in that?

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

One example of a use case in Agile can be like:

A guest places an order for concierge assistance. The system forwards the request to the concierge. The concierge sends a personal note of confirmation to the guest's email address.

1. Who is Scrum Master?

A scrum master is the facilitator for an agile development team. Scrum is a methodology that allows a team to self-organize and make changes quickly, in accordance with agile principles. The scrum master manages the process for how information is exchanged.

1. Differentiate between Product/Sprint Backlog

The agile product backlog in [Scrum](https://www.mountaingoatsoftware.com/agile/scrum) is a prioritized features list, containing short descriptions of all functionality desired in the product. When applying Scrum, it's not necessary to start a project with a lengthy, upfront effort to document all requirements.

A sprint backlog is the set of items that a cross-functional product team selects from its product backlog to work on during the upcoming sprint.

1. What is Epic & Story?

Epic: An epic is a large user story that cannot be delivered as defined within a single iteration or is large enough that it can be split into smaller user stories.

User Story: They are one of the primary development artifacts for Scrum and Extreme Programming (XP) project teams. A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

1. What is called Velocity in SCRUM?

Velocity is an extremely simple, powerful method for accurately measuring the rate at which scrum development teams consistently deliver business value. It is an indication of the average amount of product backlog turned into an Increment of product during a sprint by a scrum team, tracked by the Development Team for use within the Scrum Team.

1. Explain the SCRUM ceremonies

* Organic backlog
* Sprint Planning
* Sprints- Short spans in which products are planned, developed, reviewed, and released. They are projects within the projects.
* Sprint Execution
* Stand up/Daily Scrum- A daily meeting in which contributors and managers discuss what work was done yesterday, what they’re working on today, and any questions that come up.
* Sprint Review
* Sprint Retrospective- An Agile team manages itself, but there are built-in measures to make sure work is being delivered at a consistent quality. Peer review and reviews by managers occur before tasks get completed and after the sprint is over.

1. What is grooming?

Backlog grooming, also referred to as backlog refinement or story time, is a recurring event for agile product development teams. The primary purpose of a backlog grooming session is to ensure the next few sprints worth of user stories in the product backlog are prepared for sprint planning. Regular backlog grooming sessions also help ensure the right stories are prioritized and that the product backlog does not become a black hole.

1. How Jira board is effective in SCRUM?

Jira Board is very effective in agile methodology. It helps the scrum master to keep track of the user stories and also check whether they will be completed in the present sprint.

1. Differentiate between SCRUM & Waterfall

|  |  |
| --- | --- |
| **Scrum SDLC** | **Waterfall SDLC** |
| It includes customers and stakeholders at each phase. | This keeps the customer at bay.  By the time the result is near. |
| Scrum development saves time and money by reviewing regular sprints in the development process. | It may take extra time as reviewing is done at the result only, if found inappropriate then the process is back to level 1. |
| Work is divided into teams as an individual responsibility. | Work is divided into phases. The team works closely. |
| Scrum takes feedback from the product owner and stakeholders. The customer is kept in the loop and constantly taken his word throughout the process of development. | The required documentation is done at the initial stage. Proper documentation takes place during the requirement phase only. |
| Scrum development process works well for difficult and complex projects. | Waterfall model works well with smaller projects. |
| It has no defined stages. | Waterfall model has clear and defined stages to work on the project. |
| Scrum welcome changes at an early and late stage during development. | It welcomes changes only at the requirement phase. There is not the liberty of making changes at later stages. |
| The development process is divided among the team as an individual, it does not wait for the previous stage to get completed. | Phases and processes are completed one at a time. |
| It divides its work into sprints and then assigned according to team members. | It divides its work into stages and process continues one after the other. |
| Working software is shown to the customer at an early stage. Which is why changes are welcomed. | Working software is produced at the delivery time only to the customer. |
| The customer is kept informed about every step taking place in project development. | The customer will contact only at the delivery date. |

1. Explain the responsibilities of Product owner.

The product owner acts as bridge between the customer and the development team. He is also responsible for managing the product backlog and even prioritize the requirements gicen by the customer.

Client-server

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

Some of the characteristics of the Client-Server Architectures are:

* The client-server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.
* The client-server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.
* A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.
* Denial of service attacks hindera servers ability to respond to authentic client requests by inundating it with false requests.
* An example of a client-server computing system is a web server. It returns the web pages to the clients that they request them.

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

The basic function of a web server is to host websites and to deliver web content from its hosted websites over the internet. During the delivery of web pages, web servers follow a network protocol known as hyper text transfer protocol (HTTP).

1. What is the role of the presentation layer?

The different functions of the presentation layer are:

* **Translation:** Before being transmitted, information in the form of characters and numbers should be changed to bit streams. The presentation layer is responsible for interoperability between encoding methods as different computers use different encoding methods. It translates data between the formats the network requires and the format the computer.
* **Encryption:** It carries out encryption at the transmitter and decryption at the receiver.
* **Compression:** It carries out data compression to reduce the bandwidth of the data to be transmitted. The primary role of Data compression is to reduce the number of bits to be 0transmitted. It is important in transmitting multimedia such as audio, video, text etc.

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

Every architecture claims that they are more secure. It Is the responsibility of the software engineer to determine whether the architecture is secure. To some extent, this architecture is secure. But it takes one good hacker to break into this system. Security can be achieved using different techniques. For example say cryptography. If the packets sent across the client and server could encrypted with a secure key, it makes the hacker to hack into the system.

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

Database server is a system which holds the database and all the data. It is also responsible for allowing access to clients to retrieve data from the database.

1. What are Super Servers in client server environment?

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

A two-tier architecture is a database architecture where presentation layer runs on a client and .data is stored on a Server. 3-tier architecture is consisting of the Presentation layer (PC, Tablet, Mobile, etc.), the Application layer (server) and Database Server.

1. What is a File server?

A file server is a central server in a computer network that provides file systems or at least parts of a file system to connected clients. File servers therefore offer users a central storage place for files on internal data media, which is accessible to all authorized clients.

**SOA and MicroServices**

* 1. What are the main benefits of SOA?

The Key benefits of SOA are:

* Improved information flow
* Ability to expose internal functionality
* Organizational flexibility
  1. How can you achieve loose coupling in SOA?

One strategy for achieving loose coupling is to use the service interface (the WSDL for a SOAP Web Service) to limit this dependency, hiding the service implementation from the consumer. Loose coupling can be addressed by encapsulating the service functionalities in a manner that limits the impact of changes to the implementation on the service interface.

* 1. Are web services and SOA the same?

Service Oriented Architecture, as the name says is an architectural concept which focuses on having different services communicating with each other to carry out a bigger job.

Thus, a web service is a basic building block in a SOA. When multiple services are combined, we have an application that falls under SOA.

* 1. What is reusable service?

me for one of SOA’s best-known principles: reusability. It is certainly one of the most obvious benefits of service orientation. It is surely the easiest to explain and understand. It is often the easiest to take on by those approaching SOA. In fact, reusability is part of the history of software development since its inception, although at different scales, so more or less, sounds to us all enough.

* 1. What are the disadvantages of SOA?

Some of the disadvantages of SOA are:

* Stand alone
* Short lived applications
* Applications in which one way asynchronous communication is necessary, and where loose coupling is considered undesirable and unnecessary.
* Homogenous application environments, like, for instance, an environment wherein all applications were built utilizing J2EE components. In these instances, it is not a good idea to introduce XML over HTTP for inter-component communications rather than utilizing Java remote method invocation.
  1. What is ESB and where does it fit in?

An enterprise service bus (ESB for short) refers to software architecture that allows for the integration of enterprise applications and services, such as middleware infrastructure platforms.

ESB helps in B2B Integration.

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

In SOA it is not necessary to build a system from scratch.

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

urely 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

9. List down the advantages of Microservices Architecture

* **Improved fault isolation**: Larger applications can remain mostly unaffected by the failure of a single module.
* **Eliminate vendor or technology lock-in**: Microservices provide the flexibility to try out a new technology stack on an individual service as needed. There won’t be as many dependency concerns and rolling back changes becomes much easier. With less code in play, there is more flexibility.
* **Ease of understanding:**With added simplicity, developers can better understand the functionality of a service.
* **Smaller and faster deployments**: Smaller codebases and scope = quicker deployments, which also allow you to start to explore the benefits of Continuous Deployment.
* **Scalability**: Since your services are separate, you can more easily scale the most needed ones at the appropriate times, as opposed to the whole application. When done correctly, this can impact cost savings.

10. What are the best practices to design Microservices?

* Domain Driven Design
* Database per Microservice
* Micro Frontends
* Continuous Delivery
* Observability
* Unified Tech Stack
* Asynchronous Communication
* Microservice First
* Infrastructure over Libraries
* Organizational Considerations

11. How does Microservice Architecture work?

* 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

12. What are the pros and cons of Microservice Architecture?

Pros:

* Improved fault isolation
* Eliminate vendor or technology lock-in
* Ease of understanding
* Smaller and faster deployments
* Scalability

Cons:

* Communication between services is complex
* More services equals more resources
* Global testing is difficult
* Debugging problems can be harder
* Deployment challengers
* Large vs small product companies

13. What is the difference between Monolithic, SOA and Microservices Architecture?

Microservices:

* Designed to host services which can function independently
* Typically does not involve component sharing
* Fine Grained sevices
* Each service can have an independent data storage
* Requires collaboration between teams

SOA:

* Designed to share resources across services
* Frequently involves component sharing
* Larger, more modular services
* Involves sharing data storage between services

Monolithic:

* Fewer Cross-cutting Concerns
* Less Operational Overhead
* Performance
* Tightly Coupled
* Harder To Understand

14. What are the challenges you face while working Microservice Architectures?

* Managing Microservices
* Monitoring
* Embracing DevOps Culture
* Fault Tolerance
* Testing
* Design With Failure in Mind
* Cyclic Dependencies

15. What are the characteristics of Microservices?

* Componentization via Services
* Organized around Business Capabilities
* Products not Projects
* Smart endpoints and dumb pipes
* Decentralized Governance
* Decentralized Data Management
* Infrastructure Automation
* Design for failure

**HTTP Fundamentals**

1. What are the basic Features of HTTP?

* It is the protocol that allows web servers and browsers to exchange data over the web.
* It is a request response protocol.
* It uses the reliable TCP connections by default on TCP port 80.
* It is stateless means each request is considered as the new request. In other words, server doesn't recognize the user by default.

2. What are request methods in HTTP?

* GET
* PUT
* POST
* DELETE
* CONNECT
* HEAD
* OPTIONS
* TRACE

3. What are the differences between GET and POST methods?

A GET request retrieves data from a web server by specifying parameters in the URL portion of the request.

The POST method is used when you want to send some data to the server, for example, file update, form data, etc.

4. What is status code in HTTP?

* 1xx- Informational
* 2xx- Success
* 3xx- Redirection
* 4xx- Client Error
* 5xx- Server error

5. 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.

6. What is URI?

Uniform Resource Identifier- A URI is an identifier of a specific resource. Like a page, or book, or a document.

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

Idempotency is a property of HTTP methods.

A request method is considered idempotent if the intended effect on the server of multiple identical requests with that method is the same as the effect for a single such request. And it's worthwhile to mention that idempotency is about the effect produced on the state of the resource on the server and not about the response status code received by the client.

8. Explain HTTP Request & Response Messages

A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

9. What is Session State in HTTP?

Session state is a method to keep track of the user session during the http request.

10. What is HTTPS?

HTTPS is a secure version of HTTP

**Introduction to API**

1. Explain REST and RESTFUL?

REST means a server that exchanges JSON documents with a client over HTTP. Not only is that not a complete definition, but it’s also not always true. The REST specification doesn’t require HTTP or JSON.

RESTFUL is service which implements REST.

2. Mention what are the HTTP methods supported by REST?

* GET
* PUT
* POST
* HEAD
* DELETE

3. Explain the architectural style for creating web API?

* RESTful API Style
* REST-like API Style
* GraphQL API Style
* Falcor
* RPC Style
* SOAP (RPC API Style)
* gRPC (RPC API Style)

4. Explain the RESTFul Web Service?

RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

5. Explain what is a “Resource” in REST?

A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. It is similar to an object instance in an object-oriented programming language, with the important difference that only a few standard methods are defined for the resource (corresponding to the standard HTTP GET, POST, PUT and DELETE methods), while an object instance typically has many methods.

6. Which protocol is used by RESTful web services?

HTTP Protocol

7. What is messaging in RESTful web services?

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. These messages contain message data and metadata i.e. information about message itself

8. State the core components of an HTTP Request?

* Verb: Includes methods like GET, PUT, POST, etc.
* Uniform Resource Identifier for identifying the resources available on the server.
* HTTP Version for specifying the HTTP version.
* HTTP Request header for containing the information about the data.
* HTTP Request body that contains the representation of the resources in use

9. State the core components of an HTTP response?

* Request Code: This contains various codes which determine the status of the server response.
* HTTP Version for specifying the HTTP version.
* HTTP Response header for containing the information about the data.
* HTTP Response body that contains the representation of the resources in use.

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

When you send API requests you are trying to get a service to 'do something.' The service will then tell you what it did or give you what it asked for. The payload is the part of that response that is communicating directly to you. In REST APIs this is usually some JSON formatted data.

11. Explain the caching mechanism?

Caching is the ability to store copies of frequently accessed data in several places along the request-response path. When a consumer requests a resource representation, the request goes through a cache or a series of caches (local cache, proxy cache, or reverse proxy) toward the service hosting the resource. If any of the caches along the request path has a fresh copy of the requested representation, it uses that copy to satisfy the request. If none of the caches can satisfy the request, the request travels all the way to the service (or origin server as it is formally known).

12. List the main differences between SOAP and REST?

|  |  |
| --- | --- |
| SOAP | REST |
| SOAP stands for Simple Object Access Protocol | REST stands for Representational State Transfer |
| SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service. | REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being  Client Server  Stateless  Cacheable  Layered System  Uniform Interface |
| SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. | REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern. |

13. Enlist advantages and disadvantages of ‘Statelessness’.

Advantages of Statelessness

Following are the benefits of statelessness in RESTful Web Services −

* Web services can treat each method request independently.
* Web services need not maintain the client's previous interactions. It simplifies the application design.
* As HTTP is itself a statelessness protocol, RESTful Web Services work seamlessly with the HTTP protocols.

Disadvantages of Statelessness

Following are the disadvantages of statelessness in RESTful Web Services −

* 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?**

**Class**:

A class is a blueprint from which you can create the instance, i.e., objects.

A class is used to bind data as well as methods together as a single unit.

Classes have logical existence.

A class doesn't take any memory spaces when a programmer creates one.

The class has to be declared only once.

**Object**:

An object is the instance of the class, which helps programmers to use variables and methods from inside the class.

Object acts like a variable of the class.

Objects have a physical existence.

An object takes memory when a programmer creates one.

Objects can be declared several times depending on the requirement.

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

Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.

We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

The Java Bean class is the example of a fully encapsulated class.

//A Java class which is a fully encapsulated class.

//It has a private data member and getter and setter methods.

package com.javatpoint;

public class Student{

//private data member

private String name;

//getter method for name

public String getName(){

return name;

}

//setter method for name

public void setName(String name){

this.name=name

}

}

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

Polymorphism is an object-oriented programming concept that refers to the ability of a variable, function or object to take on multiple forms. A language that features polymorphism allows developers to program in the general rather than program in the specific.

class Overload

{

void demo (int a)

{

System.out.println ("a: " + a);

}

void demo (int a, int b)

{

System.out.println ("a and b: " + a + "," + b);

}

double demo(double a) {

System.out.println("double a: " + a);

return a\*a;

}

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

Overriding and Overloading are two very important concepts in Java. They are confusing for Java novice programmers. This post illustrates their differences by using two simple examples.

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

Overriding means having two methods with the same method name and parameters (i.e., method signature). One of the methods is in the parent class and the other is in the child class. Overriding allows a child class to provide a specific implementation of a method that is already provided its parent class.

Here are some important facts about Overriding and Overloading:

1). The real object type in the run-time, not the reference variable's type, determines which overridden method is used at runtime. In contrast, reference type determines which overloaded method will be used at compile time.

2). Polymorphism applies to overriding, not to overloading.

3). Overriding is a run-time concept while overloading is a compile-time concept.

The one main advantage of these overriding and overloading is time-saving.

Save memory space.

The readability of the code is increased.

Here, for function overloading concept, 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 of acquiring the features and behaviors of a class by another class. The class whose members are inherited is called the base class, and the class that inherits those members is called the derived class. Inheritance implements the IS-A relationship.

OOPs support the six different types of inheritance as given below :

Single inheritance

Multi-level inheritance

Multiple inheritance

Multipath inheritance

Hierarchical Inheritance

Hybrid Inheritance

class Doctor {

void Doctor\_Details() {

System.out.println("Doctor Details...");

}

}

class Surgeon extends Doctor {

void Surgeon\_Details() {

System.out.println("Surgen Detail...");

}

}

public class Hospital {

public static void main(String args[]) {

Surgeon s = new Surgeon();

s.Doctor\_Details();

s.Surgeon\_Details();

}

}

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

An abstract class is a template definition of methods and variables of a class (category of objects) that contains one or more abstracted methods. Abstract classes are used in all object-oriented programming (OOP) languages, including Java (see Java abstract class), C++, C# and VB.NET. Objects or classes may be abstracted, which means that they are summarized into characteristics that are relevant to the current program’s operation.

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

An interface is a collection of declaration of methods. These methods do not contain an implementation in the interface. The implementation is provided by the respective class that implements the interface. Thus, a single declaration of a method in an interface is usable by different classes to provide separate implementations.

class A{}

interface IB{}

interface IC{}

class D extends A implements IB, IC{

}

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

Access modifiers define the access privileges of classes, interfaces, constructors, methods, and data members. Access modifiers consist of public, private, and protected. If no modifier is present, the default access of package-private is used.

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

A constructor is a special type of method. It is a block of codes similar to the methods. There are mainly two types of constructors in JAVA.

Types of constructors in java are:

**1. Default Constructor (without arguments):**

The constructor which doesn't have any parameter is called as "Default Constructor".

**2. Parameterized Constructor (with arguments) :**

The constructor having parameters (arguments) is called "Parameterized Constructor".

A constructor is called when an instance of the object is created, and memory is allocated for the object.

1. **What is ‘this’ pointer?**

this is a pointer to an instance of its class and available to all non-static member functions.

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

Static Binding: The binding which can be resolved at compile time by compiler is known as static or early binding. Binding of all the static, private and final methods is done at compile-time .

Dynamic Binding: In Dynamic binding compiler doesn’t decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have same method

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

We cannot create an instance of an abstract class instead we 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. But an abstract class has a protected constructor (by default) and you can declare your own constructor also allowing derived types (i.e classes) to initialize it member variables so that those (member variables) would be accessible to their derived types through inheritance.

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

Inheritance is the OOPS concept that can be used as reuse mechanism.

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

Objects logic are done by classes for example ,by phone we can call,Bluetooth ,take photo etc. those every logic

will be divide as classes.When we are creating class we need consider about SOLID principle.

Example for object and class.

public class Phone

{

public string Name{get; set;}

public int IMINumber{get; set;}

public string Type{get; set;}

public void Dial

{

//implemention of function

}

public void SendMassage

{

//implemention of function

}

}

**Abstraction**

Abstraction means only show relevant data and details rest of others are hide.this is the most important pillar in OOP. This is mostly done by interfaces rather than abstract class.

**Interface**

Abstraction is done by classes or interface.

Abstract classes may have not implemented methods.

**Abstract class**

Abstract classes may have implemented methods.

We can generate tv by remote ,here remote is the interface between tv and man.

When interface and when abstract classes

So if you need multiple inheritance and a clear blueprint than has only the design and not the implementation; you go for interface. If you don’t need multiple inheritance but you need a mix of design plan and pre-implementation then abstract class is your choice.

Some real world example

1.When we are making a call it only concatenate about the numbers and display that in screen,we really do not know

how this connect with other number.

2. We send data from Bluetooth we really do not know how it connect with other phone or devices.

**Encapsulation.**

Both Abstraction and Encapsulation works hand in hand because abstraction which have to show in a level that particular

details access by encapsulation.

In real time we are using Encapsulation for security purpose.

Example

1.When we turn on the Bluetooth we can assess to connect to other phone but not access to call or send sms from that phone that level is hidden from encapsulation(some of abstraction is available).

2.If i connect with kajas phone and kajas connect with sumanie phone ,but i do not have access to connect with sumanie is phone through kajas phone.this kind of hidden will handle by encapsulation.

This are handle by access specifier like public,private,protected and internal.

**Polymorphism**

Here is an example of polymorphism.

public class Samsung : Mobile

{

public void Message()

{

Console.WriteLine("Text message sent");

}

public void Message(string MessageType)

{

Console.WriteLine("Change the message type " + MessageType + " Type");

}

}

**Inheritance**

Creating a new class (sub class) from base/super cass is called inheritance.

When we inherit all the features from base/super class will be in in the sub class.

There are mainly 4 types of inheritance:

Single level inheritance

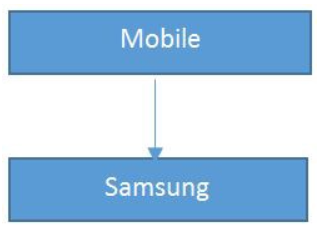
Multi-level inheritance

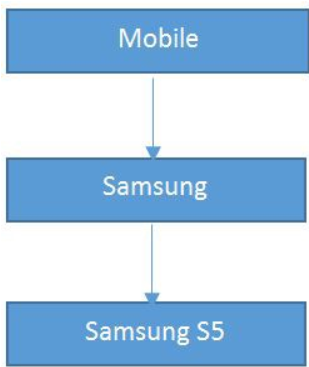
Hierarchical inheritance

Hybrid inheritance

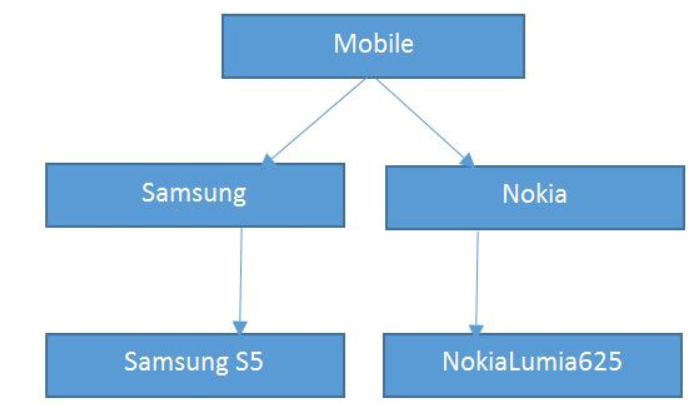
Multiple inheritance

Single level inheritance

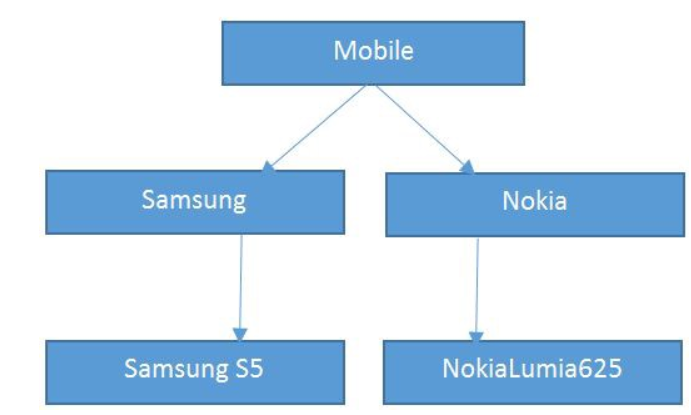


Multi-level inheritance

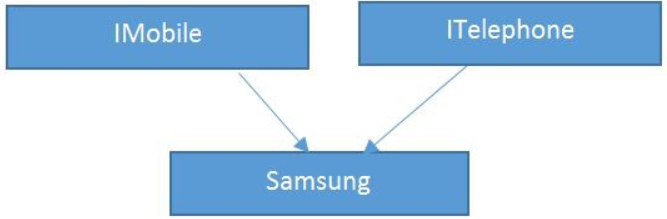
Hierarchical inheritance



Hybrid inheritance.



Multiple inheritance



**Unit Testing and Junit**

1. What is unit testing?

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

2. What is the difference between manual testing and automated testing?

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Manual Testing** | **Automation Testing** |
| **Accuracy** | Manual Testing shows lower accuracy due to the higher possibilities of human errors. | Automation Testing depicts a higher accuracy due to computer-based testing eliminating the chances of errors |
| **Testing at Scale** | Manual Testing needs time when testing is needed at a large scale. | Automation Testing easily performs testing at a large scale with the utmost efficiency. |
| **Turnaround time** | Manual Testing takes more time to complete a cycle of testing, and thus the turnaround time is higher. | Automation Testing completes a cycle of testing within record time and thus the turnaround time is much lower. |
| **Cost Efficiency** | Manual Testing needs more cost as it involves the hiring of expert professionals to perform testing. | Automation Testing saves costs incurred as once the software infrastructure is integrated, it works for a long time. |
| **User Experience** | Manual Testing ensures a high-end User Experience to the end user of the software, as it requires human observation and cognitive abilities. | Automation Testing cannot guarantee a good User Experience since the machine lacks human observation and cognitive abilities. |
| **Areas of Specialization** | Manual Testing should be used to perform Exploratory Testing, Usability Testing and Ad-hoc Testing to exhibit the best results. | Automation Testing should be used to perform Regression Testing, Load Testing, Performance Testing and Repeated Execution for best results. |
| **User Skills** | Users must have the ability to mimic user behavior and build test plans to cover all the scenarios. | Users must be highly skilled at programming and scripting to build test cases and automate as many scenarios as possible. |

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

Yes it is necessary. Writing test case for every logic confirms that this logic can work for every logic.

4. What are the features of JUnit?

* 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.

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

JUNIT ANNOTATIONS is a special form of syntactic meta-data that can be added to Java source code for better code readability and structure. Variables, parameters, packages, methods and classes can be annotated. Annotations were introduced in Junit4, which makes Java code more readable and simple. This is the big difference between Junit3 and Junit4 that Junit4 is annotation based.

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.junit.Assert which extends java.lang.Object class.

7. What is Code Coverage?

Code coverage checks how many percentage of the program has covered with test functions which has been written by the tester. The best practice is to get 100% coverage for the code.

8. What are the best practices to perform Unit Testing?

Arrange, Act, Assert

One Assert Per Test Method

Avoid Test Interdependence

Keep It Short, Sweet, and Visible

Recognize Test Setup Pain as a Smell

Add Them to the Build

9. What is Mocking?

Mocking is generating pseudo-objects that simulate real objects behaviour for tests

**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.

2. What is the difference between GIT and SVN?

SVN allows you to check out sub-trees (or branches) only whereas Git requires you to check out the entire repository as a unit. This is because there is a .svn in each one of your folders while git only has one .git at the top level parent directory.

3. What are the advantages of using GIT?

* Distributed model
* Branching and merging are easy
* Workflow is flexible
* Data integrity is assured
* Fast
* Staging area
* Free

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

Staging Area: Make sure your commits have logically grouped changes and not everything else you are working on.

Index: The Git index is used as a staging area between your working directory and your repository

5. 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.

6. What is the function of git clone?

git clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository.

7. How can you create a repository in Git?

git init <project directory>

8. What is the purpose of branching in GIT?

Branches helps you to work independently without affecting the master.

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

git remote is used to refer to a remote repository or your central repository.

git clone is used to copy or clone a different repository.

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

The git diff is a multi-function Git command, which is used to compare changes committed in Git.

11. Explain what the commit message is?

Commit message help you to add the changes made in your local system to the repo.

12. 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.

13. What is Rebasing

The rebase command integrates changes from one branch into another. It is an alternative to the better known "merge" command.

**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

2. Explain what is POM and its significance

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

3. 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.

4. 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.

5. List out what are the build phases in Maven?

* 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

6. Mention the three build lifecycle of Maven?

* default: the main life cycle as it's responsible for project deployment
* clean: to clean the project and remove all files generated by the previous build
* site: to create the project's site documentation

7. List out what are the aspects does Maven Manages?

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

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

A repository in Maven holds build artifacts and dependencies of varying types. There are exactly two types of repositories: local and remote. The local repository is a directory on the computer where Maven runs. It caches remote downloads and contains temporary build artifacts that you have not yet released.

9. Explain how you can exclude dependency?

Exclusions are set on a specific dependency in your POM, and are targeted at a specific groupId and artifactId. When you build your project, that artifact will not be added to your project's classpath by way of the dependency in which the exclusion was declared.

10. For POM what are the minimum required elements?

* project root.
* modelVersion
* groupId
* artifactId
* version

**CI/CD**

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

DevOps is a software development method which focuses on communication, integration, and collaboration among IT professionals to enables rapid deployment of products.

Agile Methodology involves continuous iteration of development and testing in the SDLC process. This software development method emphasizes on iterative, incremental, and evolutionary development.

Agile development process breaks the product into smaller pieces and integrates them for final testing. It can be implemented in many ways, including scrum, kanban, scrum, XP, etc.

2. What is the need for DevOps?

DevOps is a culture that promotes collaboration between Development and Operations Team. This allows deploying code to production faster and in an automated way. It helps to increases an organization's speed to deliver application and services. It can be defined as an alignment of development and IT operation.

3. What are the advantages of DevOps?

* Reduce the implementation time of new services from months to minutes
* 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
* Increase the rate of success for digitalization strategies and transformation projects
* Ensure that money invested in cloud infrastructure, analytics and data management are not wasted

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

In the context of the lecture about the DevOps, Devops made it easier for integration. Earlier there was other engineers to integrate the project. With the help of devops,now continuous integration is taking places in industries.

5. What are the success factors for Continuous Integration?

With help of Contiuos Integration,

* The work force could be reduced.
* Confusion with developer and integration engineer is reduced.
* Made integration more easy and hazzle free.

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

With Continuous Deployment, every change that is made is automatically deployed to production. This approach works well in enterprise environments where you plan to use the user as the actual tester and it can be quicker to release.

Continuous Integration is merging all code from all developers to one central branch of the repo many times a day trying to avoid conflicts in the code in the future. The concept here is to have multiple devs on a project to keep the main branch of the repo to the most current form of the source code, so each dev can check out or pull from the latest code to avoid conflicts.

Continuous Delivery is a small build cycle with short sprint where the aim is to keep the code in a deployable state at any given time. This does not mean the code or project is 100% complete, but the feature sets that are available are vetted, tested, debugged and ready to deploy, although you may not deploy at that moment.

7. 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.

8. 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.

One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily. As each change introduced is typically small, pinpointing the specific change that introduced a defect can be done quickly.

In recent years CI has become a best practice for software development and is guided by a set of key principles. Among them are revision control, build automation and automated testing.

Additionally, Continuous Deployment and Continuous Delivery have developed as best-practices for keeping your application deployable at any point or even pushing your main codebase automatically into production whenever new changes are brought into it. This allows your team to move fast while keeping high quality standards that can be checked automatically.

9. 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
* Implement Automation in Dashboards

10. 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
* Automate wherever possible
* Integrate and deliver continuously
* Achieve better results with monitoring & feedback