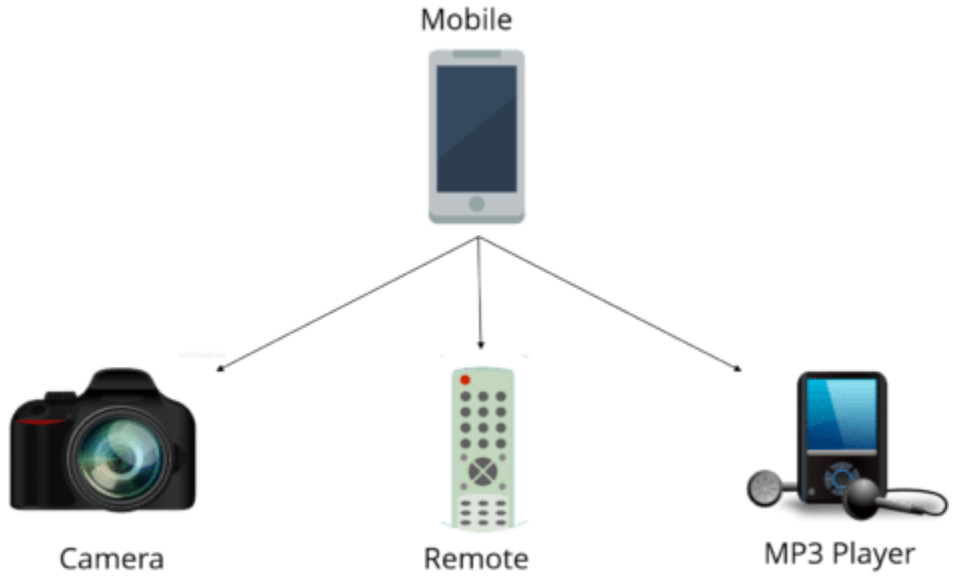
Java OOPS Interview Questions

Q1. What is Polymorphism?

Polymorphism is briefly described as “one interface, many implementations”. Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form. There are two types of polymorphism:



1. Compile time polymorphism
2. Run time polymorphism

Compile time polymorphism is method overloading whereas Runtime time polymorphism is done using inheritance and interface.

Q2. What is runtime polymorphism or dynamic method dispatch?

In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. Let’s take a look at the example below to understand it better.

```
class Car {
void run()
{
System.out.println(&ldquo;car is running&rdquo;);
```

```
}
}
class Audi extends Car {
void run()
{
System.out.prinltn(&ldquo;Audi is running safely with 100km&rdquo;);
}
public static void main(String args[])
{
Car b= new Audi(); //upcasting
b.run();
}
}
```

Q3. What is abstraction in Java?

Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user. Thus you can say that abstraction in Java is the process of hiding the implementation details from the user and revealing only the functionality to them. Abstraction can be achieved in two ways:

1. **Abstract Classes** (0-100% of abstraction can be achieved)
2. **Interfaces** (100% of abstraction can be achieved)

Q4. What do you mean by an interface in Java?

An interface in Java is a blueprint of a class or you can say it is a collection of abstract methods and static constants. In an interface, each method is public and abstract but it does not contain any constructor. Thus, interface basically is a group of related methods with empty bodies. Example:

```
public interface Animal {
  public void eat();
  public void sleep();
  public void run();
}
```

Q5. What is the difference between abstract classes and interfaces?

Abstract Class	Interfaces
An abstract class can provide complete, default code and/or just the details that have to be overridden	An interface cannot provide any code at all, just the signature
In the case of an abstract class, a class may extend only one abstract class	A Class may implement several interfaces
An abstract class can have non-abstract methods	All methods of an Interface are abstract
An abstract class can have instance variables	An Interface cannot have instance variables
An abstract class can have any visibility: public, private, protected	An Interface visibility must be public (or) none
If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly	If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method
An abstract class can contain constructors	An Interface cannot contain constructors
Abstract classes are fast	Interfaces are slow as it requires extra indirection to find the corresponding method in the actual class

Q6. What is inheritance in Java?

Inheritance in Java is the concept where the properties of one class can be inherited by the other. It helps to reuse the code and establish a relationship between different classes. Inheritance is performed between two types of classes:

1. Parent class (Super or Base class)
2. Child class (Subclass or Derived class)

A class which inherits the properties is known as Child Class whereas a class whose properties are inherited is known as Parent class.

Q7. What are the different types of inheritance in Java?

Java supports four types of inheritance which are:

1. **Single Inheritance:** In single inheritance, one class inherits the properties of another i.e there will be only one parent as well as one child class.
2. **Multilevel Inheritance:** When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent

class but at different levels, such type of inheritance is called Multilevel Inheritance.

3. **Hierarchical Inheritance:** When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.
4. **Hybrid Inheritance:** Hybrid inheritance is a combination of two *or more types* of inheritance.

Q8. What is method overloading and method overriding?

Method Overloading :

- In Method Overloading, Methods of the same class shares the same name but each method must have a different number of parameters or parameters having different types and order.
- Method Overloading is to "add" or "extend" more to the method's behavior.
- It is a compile-time polymorphism.
- The methods must have a different signature.
- It may or may not need inheritance in Method Overloading.

Let's take a look at the example below to understand it better.

```
class Adder {
Static int add(int a, int b)
{
return a+b;
}
Static double add( double a, double b)
{
return a+b;
}
public static void main(String args[])
{
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}}
```

Method Overriding:

- In Method Overriding, the subclass has the same method with the same name and exactly the same number and type of parameters and same return type as a superclass.
- Method Overriding is to “Change” existing behavior of the method.
- It is a run time polymorphism.
- The methods must have the same signature.
- It always requires inheritance in Method Overriding.

Let’s take a look at the example below to understand it better.

```
class Car {
void run(){
System.out.println(&ldquo;car is running&rdquo;);
}
Class Audi extends Car{
void run()
{
System.out.prinltn("Audi is running safely with 100km");
}
public static void main( String args[])
{
Car b=new Audi();
b.run();
}
}
```

Q9. Can you override a private or static method in Java?

You cannot override a private or static method in Java. If you create a similar method with the same return type and same method arguments in child class then it will hide the superclass method; this is known as method hiding. Similarly, you cannot override a private method in subclass because it’s not accessible there. What you can do is create another private method with the same name in the child class. Let’s take a look at the example below to understand it better.

```
class Base {
private static void display() {
System.out.println("Static or class method from Base");
}
```

```
public void print() {
System.out.println("Non-static or instance method from Base");
}
class Derived extends Base {
private static void display() {
System.out.println("Static or class method from Derived");
}
public void print() {
System.out.println("Non-static or instance method from Derived");
}
public class test {
public static void main(String args[])
{
Base obj= new Derived();
obj1.display();
obj1.print();
}
}
```

Q10. What is multiple inheritance? Is it supported by Java?

If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

The problem with multiple inheritance is that if multiple parent classes have the same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.

Therefore, Java doesn't support multiple inheritance. The problem is commonly referred to as Diamond Problem.

In case you are facing any challenges with these java interview questions, please comment on your problems in the section below.

Q11. What is encapsulation in Java?

Encapsulation is a mechanism where you bind your data(variables) and code(methods) together as a single unit. Here, the data is hidden from the outer world and can be accessed only via current class methods. This helps in protecting

the data from any unnecessary modification. We can achieve encapsulation in Java by:

- Declaring the variables of a class as private.
- Providing public setter and getter methods to modify and view the values of the variables.

Q12. What is an association?

Association is a relationship where all object have their own lifecycle and there is no owner. Let's take the example of Teacher and Student. Multiple students can associate with a single teacher and a single student can associate with multiple teachers but there is no ownership between the objects and both have their own lifecycle. These relationships can be one to one, one to many, many to one and many to many.

Q13. What do you mean by aggregation?

An aggregation is a specialized form of Association where all object has their own lifecycle but there is ownership and child object can not belong to another parent object. Let's take an example of Department and teacher. A single teacher can not belong to multiple departments, but if we delete the department teacher object will not destroy.

Q14. What is composition in Java?

Composition is again a specialized form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object does not have their lifecycle and if parent object deletes all child object will also be deleted. Let's take again an example of a relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different houses if we delete the house room will automatically delete.

Q15. What is a marker interface?

A Marker interface can be defined as the interface having no data member and member functions. In simpler terms, an empty interface is called the Marker interface. The most common examples of Marker interface in Java are Serializable, Cloneable etc. The marker interface can be declared as follows.

```
public interface Serializable{
}
```

Q16. What is object cloning in Java?

Object cloning in Java is the process of creating an exact copy of an object. It basically means the ability to create an object with a similar state as the original object. To achieve this, Java provides a method **clone()** to make use of this functionality. This method creates a new instance of the class of the current object and then initializes all its fields with the exact same contents of corresponding fields. To object clone(), the marker interface **java.lang.Cloneable** must be implemented to avoid any runtime exceptions. One thing you must note is Object clone() is a protected method, thus you need to override it.

Q17. What is a copy constructor in Java?

Copy constructor is a member function that is used to initialize an object using another object of the same class. Though there is no need for copy constructor in Java since all objects are passed by reference. Moreover, Java does not even support automatic pass-by-value.

Q18. What is a constructor overloading in Java?

In Java, constructor overloading is a technique of adding any number of constructors to a class each having a different parameter list. The compiler uses the number of parameters and their types in the list to differentiate the overloaded constructors.

```
class Demo
{
int i;
public Demo(int a)
{
i=k;
}
public Demo(int a, int b)
{
//body
}
}
```

In case you are facing any challenges with these java interview questions, please comment on your problems in the section below. Apart from this Java Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for a structured training from edureka!

Servlets – Java Interview Questions

Q1. What is a servlet?

- Java Servlet is server-side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.
- The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.
- All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.
- Most of the times, web applications are accessed using HTTP protocol and thats why we mostly extend HttpServlet class. Servlet API hierarchy is shown in below image.

Q2. What are the differences between Get and Post methods?

Get	Post
Limited amount of data can be sent because data is sent in header.	Large amount of data can be sent because data
Not Secured because data is exposed in URL bar.	Secured because data is not exposed in URL b
Can be bookmarked	Cannot be bookmarked
Idempotent	Non-Idempotent
It is more efficient and used than Post	It is less efficient and used

Q3. What is Request Dispatcher?

RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in same application. We can also use this to include the content of another resource to the response.

There are two methods defined in this interface:

1.void forward()

2.void include()

Q4. What are the differences between forward() method and sendRedirect() methods?

forward() method	SendRedirect() method
forward() sends the same request to another resource.	sendRedirect() method sends new request alw uses the URL bar of the browser.
forward() method works at server side.	sendRedirect() method works at client side.
forward() method works within the server only.	sendRedirect() method works within and outs

Q5. What is the life-cycle of a servlet?

There are 5 stages in the lifecycle of a servlet:

1. Servlet is loaded
2. Servlet is instantiated
3. Servlet is initialized

4. Service the request
5. Servlet is destroyed

Q6. How does cookies work in Servlets?

- Cookies are text data sent by server to the client and it gets saved at the client local machine.
- Servlet API provides cookies support through javax.servlet.http.Cookie class that implements Serializable and Cloneable interfaces.
- HttpServletRequest getCookies() method is provided to get the array of Cookies from request, since there is no point of adding Cookie to request, there are no methods to set or add cookie to request.
- Similarly HttpServletResponse addCookie(Cookie c) method is provided to attach cookie in response header, there are no getter methods for cookie.

Q7. What are the differences between ServletContext vs ServletConfig?

The difference between ServletContext and ServletConfig in Servlets JSP is in below tabular format.

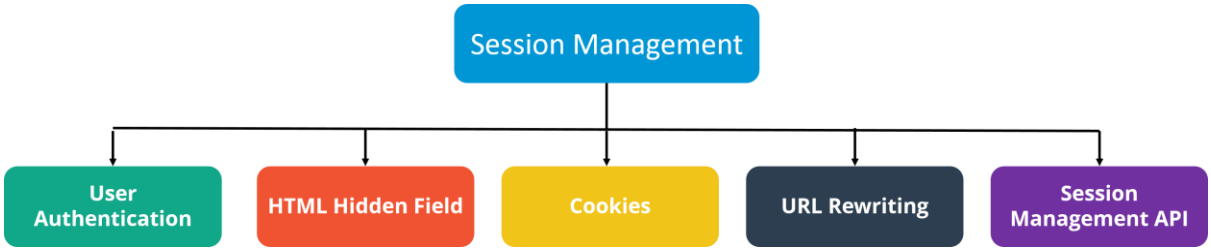
ServletConfig	ServletContext
Servlet config object represent single servlet	It represent whole web application running on p and common for all the servlet
Its like local parameter associated with particular servlet	Its like global parameter associated with whole
It's a name value pair defined inside the servlet section of web.xml file so it has servlet wide scope	ServletContext has application wide scope so de servlet tag in web.xml file.
getServletConfig() method is used to get the config object	getServletContext() method is used to get the c
for example shopping cart of a user is a specific to particular user so here we can use servlet config	To get the MIME type of a file or application se information is stored using servlet context objec

Q8. What are the different methods of session management in servlets?

Session is a conversational state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

Some of the common ways of session management in servlets are:

1. User Authentication
2. HTML Hidden Field
3. Cookies
4. URL Rewriting
5. Session Management API



Apart from this blog, if you want to get trained by professionals on this technology, you can opt for structured training from edureka! Click below to know more.