JAVA INTERVIEW QUESTIONS:

Q) What is Encapsulation?
Answer: Encapsulation provides objects with the ability to hide their internal characteristics and behavior. Each object provides a number
of methods, which can be accessed by other objects and change its internal data. In Java, there are three access modifiers: public, private and protected. Each modifier imposes different access rights to other classes, either in the same or in external packages.
Some of the advantages of using encapsulation are listed below:
• The internal state of every objected is protected by hiding its attributes.
• It increases usability and maintenance of code, because the behavior of an object can be independently changed or extended.
• It improves modularity by preventing objects to interact with each other, in an undesired way.
You can refer to our tutorial here for more details and examples on encapsulation.

Q) What is Polymorphism?
Answer: Polymorphism is the ability of programming languages to present the same interface for differing underlying data types. A polymorphic type is a type whose operations can also be applied to values of some other type.

Q) What is Inheritance?
Answer: Inheritance provides an object with the ability to acquire the fields and methods of another class, called base class. Inheritance provides re-usability of code and can be used to add additional features to an existing class, without modifying it.

Q) What is Abstraction?
Answer: Abstraction is the process of separating ideas from specific instances and thus, develop classes in terms of their own functionality, instead of their implementation details. Java supports the creation and existence of abstract classes that expose interfaces, without including the actual implementation of all methods. The abstraction technique aims to separate the implementation details of a class from its behavior.

Q) Differences between Abstraction and Encapsulation
Answer: Abstraction and encapsulation are complementary concepts. On the one hand, abstraction focuses on the behavior of an object. On the other hand, encapsulation focuses on the implementation of an object's behavior. Encapsulation is usually achieved by hiding information about the internal state of an object and thus, can be seen as a strategy used in order to provide abstraction.

Q) Can there be an abstract class with no abstract methods in it?
Answer: Yes

Q) Can an Interface be final?
Answer: No

Q) Can an Interface have an inner class?
Answer: Yes.
Example

```
public interface abc {
static int i=0;
void dd();
class a1 {
a1() {
int j;
System.out.println(“in interfia”);
};
public static void main(String a1[])
{
System.out.println(“in interfia”);
}
}
}
```

Q) Can there be an abstract class with no abstract methods in it?
Answer: Yes

Q) Can an Interface be final?
Answer: No

Q) Can we define private and protected modifiers for variables in interfaces?
Answer: No

Q) What is the query used to display all tables names in SQL Server (Query analyzer)?
Answer: select * from information_schema.tables

Q) What is Externalizable?
Answer: Externalizable is an Interface that extends Serializable Interface. And sends data into Streams in Compressed Format. It has two methods, writeExternal(ObjectOuput out) and readExternal(ObjectInput in)

Q) What modifiers are allowed for methods in an Interface?
Answer: Only public and abstract modifiers are allowed for methods in interfaces.

Q) What is a local, member and a class variable?
Answer: Variables declared within a method are “local” variables. Variables declared within the class i.e not within any methods are “member” variables (global variables). Variables

declared within the class i.e not within any methods and are defined as “static” are class variables

Q) How many types of JDBC Drivers are present and what are they?
Answer: There are 4 types of JDBC Drivers
Type 1: JDBC-ODBC Bridge Driver
Type 2: Native API Partly Java Driver
Type 3: Network protocol Driver
Type 4: JDBC Net pure Java Driver

Q) Can we implement an interface in a JSP?
Answer: No

Q) What is the difference between ServletContext and PageContext?
Answer:
ServletContext: Gives the information about the container
PageContext: Gives the information about the Request

Q) What is the difference in using request.getRequestDispatcher() and context.getRequestDispatcher()?
Answer: request.getRequestDispatcher(path): In order to create it we need to give the relative path of the resource context.getRequestDispatcher(path): In order to create it we need to give the absolute path of the resource.

Q) How to pass information from JSP to included JSP?
Answer: Using <%jsp:param> tag.

Q) What is the difference between directive include and jsp include?
Answer: <%@ include> : Used to include static resources during translation time.
: Used to include dynamic content or static content during runtime.

Q) What is the difference between RequestDispatcher and sendRedirect?
Answer: RequestDispatcher: server-side redirect with request and response objects.
sendRedirect : Client-side redirect with new request and response objects.

Q) How does JSP handle runtime exceptions?
Answer: Using errorPage attribute of page directive and also we need to specify isErrorPage=true if the current page is intended to URL redirecting of a JSP.

Q) How do you delete a Cookie within a JSP?
Answer:
Cookie mycook = new Cookie(“name”,”value”);
response.addCookie(mycook);
Cookie killmycook = new Cookie(“mycook”,”value”);

killmycook.setMaxAge(0);
killmycook.setPath("/");
killmycook.addCookie(killmycook);

Q) What is Function Overriding and Overloading in Java ?
Answer: Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

Q) What is a Constructor, Constructor Overloading in Java and Copy-Constructor
Answer: A constructor gets invoked when a new object is created. Every class has a constructor. In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class. The constructor overloading is similar to method overloading in Java. Different constructors can be created for a single class. Each constructor must have its own unique parameter list. Finally, Java does support copy constructors like C++, but the difference lies in the fact that Java doesn't create a default copy constructor if you don't write your own.

Q) Does Java support multiple inheritance ?
Answer: No, Java does not support multiple inheritance. Each class is able to extend only on one class, but is able to implement more than one interfaces. That means Multiple inheritance can be achieved using interface.

Q) What is the difference between an Interface and an Abstract class ?
Answer: Java provides and supports the creation both of abstract classes and interfaces. Both implementations share some common
characteristics, but they differ in the following features:
• All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and nonabstract methods.
• A class may implement a number of Interfaces, but can extend only one abstract class.
• In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
• Abstract classes can implement interfaces without even providing the implementation of interface methods.
• Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
• Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
• An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

Q) What are pass by reference and pass by value ?
Answer: When an object is passed by value, this means that a copy of the object is passed. Thus, even if changes are made to that object, it doesn't affect the original value. When an object is passed by reference, this means that the actual object is not passed, rather a

reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.