Q12:State the significance of public, private, protected, default modifiers both singly and in combination and state the effect of package relationships on declared items qualified by these modifiers.

A12:*public :* Public class is visible in other packages, field is visible everywhere (class must be public too)
private : Private variables or methods may be used only by an instance of the same class that declares the variable or method, A private feature may only be accessed by the class that owns the feature.
protected : Is available to all classes in the same package and also available to all subclasses of the class that owns the protected feature.This access is provided even to subclasses that reside in a different package from the class that owns the protected feature.
default :What you get by default ie, without any access modifier (ie, public private or protected).It means that it is visible to all within a particular package.

Q14:What is static in java?

A14: Static means one per class, not one for each object no matter how many instance of a class might exist. This means that you can use them without creating an instance of a class.Static methods are implicitly final, because overriding is done based on the type of the object, and static methods are attached to a class, not an object. A static method in a superclass can be shadowed by another static method in a subclass, as long as the original method was not declared final. However, you can't override a static method with a nonstatic method. In other words, you can't change a static method into an instance method in a subclass.

Q15:What is final?

A15:A final class can't be extended ie., final class may not be subclassed. A final method can't be overridden when its class is inherited. You can't change value of a final variable (is a constant).

Q22:How can one prove that the array is not null but empty using one line of code?

A22: Print args.length. It will print 0. That means it is empty. But if it would have been null then it would have thrown a NullPointerException on attempting to print args.length.

Q23:What environment variables do I need to set on my machine in order to be able to run Java programs?

A23:CLASSPATH and PATH are the two variables.

Q24:Can an application have multiple classes having main method?

A24: Yes it is possible. While starting the application we mention the class name to be run. The JVM will look for the Main method only in the class whose name you have mentioned. Hence there is not conflict amongst the multiple classes having main method.

Q25:Can I have multiple main methods in the same class?

A25: No the program fails to compile. The compiler says that the main method is already defined in the class.

Q26:Do I need to import java.lang package any time? Why ?

A26: No. It is by default loaded internally by the JVM.

Q27:Can I import same package/class twice? Will the JVM load the package twice at runtime?

A27: One can import the same package or same class multiple times. Neither compiler nor JVM complains abt it. And the JVM will internally load the class only once no matter how many times you import the same class.

Q28:What are Checked and UnChecked Exception?

A28:A checked exception is some subclass of Exception (or Exception itself), excluding class RuntimeException and its subclasses.
Making an exception checked forces client programmers to deal with the possibility that the exception will be thrown. eg, IOException thrown by java.io.FileInputStream's read() method·
Unchecked exceptions are RuntimeException and any of its subclasses. Class Error and its subclasses also are unchecked. With an unchecked exception, however, the compiler doesn't force client programmers either to catch the exception or declare it in a throws clause. In fact, client programmers may not even know that the exception could be thrown. eg, StringIndexOutOfBoundsException thrown by String's charAt() method· Checked exceptions must be caught at compile time. Runtime exceptions do not need to be. Errors often cannot be.

Q29:What is Overriding?

A29: When a class defines a method using the same name, return type, and arguments as a method in its superclass, the method in the class overrides the method in the superclass.
When the method is invoked for an object of the class, it is the new definition of the method that is called, and not the method definition from superclass. Methods may be overridden to be more public, not more private.

Q30:What are different types of inner classes?

A30: *Nested top-level classes*, *Member classes, Local classes, Anonymous classes*

Nested top-level classes- If you declare a class within a class and specify the static modifier, the compiler treats the class just like any other top-level class. Any class outside the declaring class accesses the nested class with the declaring class name acting similarly to a package. eg, outer.inner. Top-level inner classes implicitly have access only to static variables.There can also be inner interfaces. All of these are of the nested top-level variety.

Member classes - Member inner classes are just like other member methods and member variables and access to the member class is restricted, just like methods and variables. This means a public member class acts similarly to a nested top-level class. The primary difference between member classes and nested top-level classes is that member classes have access to the specific instance of the enclosing class.

Local classes - Local classes are like local variables, specific to a block of code. Their visibility is only within the block of their declaration. In order for the class to be useful beyond the declaration block, it would need to implement a more publicly available interface.Because local classes are not members, the modifiers public, protected, private, and static are not usable.

Anonymous classes - Anonymous inner classes extend local inner classes one level further. As anonymous classes have no name, you cannot provide a constructor.

Q31: Are the imports checked for validity at compile time? e.g. will the code containing an import such as java.lang.ABCD compile?

A31: Yes the imports are checked for the semantic validity at compile time. The code containing above line of import will not compile. It will throw an error saying,can not resolve symbol
symbol : class ABCD
location: package io
import java.io.ABCD;

Q32:Does importing a package imports the subpackages as well? e.g. Does importing com.MyTest.* also import com.MyTest.UnitTests.*?

A32: No you will have to import the subpackages explicitly. Importing com.MyTest.* will import classes in the package MyTest only. It will not import any class in any of it's subpackage.

Q33:What is the difference between declaring a variable and defining a variable?

A33: In declaration we just mention the type of the variable and it's name. We do not initialize it. But defining means declaration + initialization.
e.g String s; is just a declaration while String s = new String ("abcd"); Or String s = "abcd"; are both definitions.

Q34: What is the default value of an object reference declared as an instance variable?

A34: null unless we define it explicitly.

Q35: Can a top level class be private or protected?

A35: No. A top level class can not be private or protected. It can have either "public" or no modifier. If it does not have a modifier it is supposed to have a default access.If a top level class is declared as private the compiler will complain that the "modifier private is not allowed here". This means that a top level class can not be private. Same is the case with protected.

Q36: What type of parameter passing does Java support?

A36: In Java the arguments are always passed by value .

37: Primitive data types are passed by reference or pass by value?

A37: Primitive data types are passed by value.

Q38: Objects are passed by value or by reference?

A38: Java only supports pass by value. With objects, the object reference itself is passed by value and so both the original reference and parameter copy both refer to the same object .

Q39: What is serialization?

A39: Serialization is a mechanism by which you can save the state of an object by converting it to a byte stream.

Q40: How do I serialize an object to a file?

A40: The class whose instances are to be serialized should implement an interface Serializable. Then you pass the instance to the ObjectOutputStream which is connected to a fileoutputstream. This will save the object to a file.

Q41: Which methods of Serializable interface should I implement?

A41: The serializable interface is an empty interface, it does not contain any methods. So we do not implement any methods.

Q42: How can I customize the seralization process? i.e. how can one have a control over the serialization process?

A42: Yes it is possible to have control over serialization process. The class should implement Externalizable interface. This interface contains two methods namely readExternal and writeExternal. You should implement these methods and write the logic for customizing the serialization process.

Q43: What is the common usage of serialization?

A43: Whenever an object is to be sent over the network, objects need to be serialized. Moreover if the state of an object is to be saved, objects need to be serilazed.

Q44: What is Externalizable interface?

A44: Externalizable is an interface which contains two methods readExternal and writeExternal. These methods give you a control over the serialization mechanism. Thus if your class implements this interface, you can customize the serialization process by implementing these methods.

Q45: When you serialize an object, what happens to the object references included in the object?

A45: The serialization mechanism generates an object graph for serialization. Thus it determines whether the included object references are serializable or not. This is a recursive process. Thus when an object is serialized, all the included objects are also serialized alongwith the original object.

Q46: What one should take care of while serializing the object?

A46: One should make sure that all the included objects are also serializable. If any of the objects is not serializable then it throws a NotSerializableException.

Q47: What happens to the static fields of a class during serialization?

A47: There are three exceptions in which serialization doesn’t necessarily read and write to the stream. These are
1. Serialization ignores static fields, because they are not part of ay particular object state.
2. Base class fields are only handled if the base class itself is serializable.
3. Transient fields

Q48: Does Java provide any construct to find out the size of an object?

A48: No there is not sizeof operator in Java. So there is not direct way to determine the size of an object directly in Java.

Q49: Give a simplest way to find out the time a method takes for execution without using any profiling tool?

A49: Read the system time just before the method is invoked and immediately after method returns. Take the time difference, which will give you the time taken by a method for execution.

To put it in code...

```
long start = System.currentTimeMillis ();
method ();
long end = System.currentTimeMillis ();

System.out.println ("Time taken for execution is” + (end - start));
```

Remember that if the time taken for execution is too small, it might show that it is taking zero milliseconds for execution. Try it on a method which is big enough, in the sense the one which is doing considerable amout of processing.

Q50: What are wrapper classes?

A50: Java provides specialized classes corresponding to each of the primitive data types. These are called wrapper classes. They are e.g. Integer, Character, Double etc.

Q51: Why do we need wrapper classes?

A51: It is sometimes easier to deal with primitives as objects. Moreover most of the collection classes store objects and not primitive data types. And also the wrapper classes provide many utility methods also. Because of these reasons we need wrapper classes. And since we create instances of these classes we can store them in any of the collection classes and pass them around as a collection. Also we can pass them around as method parameters where a method expects an object.

Q52: What are checked exceptions?

A52: Checked exceptions are those which the Java compiler forces you to catch. e.g. IOException are checked Exceptions.

Q53: What are runtime exceptions?

A53: Runtime exceptions are those exceptions that are thrown at runtime because of either wrong input data or because of wrong business logic etc. These are not checked by the compiler at compile time.

Q54: What is the difference between error and an exception?

A54: An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors and you can not repair them at runtime. While exceptions are conditions that occur because of bad input etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most of the cases it is possible to recover from an exception (probably by giving user a feedback for entering proper values etc.).

Q55: How to create custom exceptions?

A56: Your class should extend class Exception, or some more specific type thereof.

Q57: If I want an object of my class to be thrown as an exception object, what should I do?

A57: The class should extend from Exception class. Or you can extend your class from some more precise exception type also.

Q58: If my class already extends from some other class what should I do if I want an instance of my class to be thrown as an exception object?

A58: One can not do anything in this scenario because Java does not allow multiple inheritances and does not provide any exception interface as well.

Q59: How does an exception permeate through the code?

A59: An unhandled exception moves up the method stack in search of a matching When an exception is thrown from a code which is wrapped in a try block followed by one or more catch blocks, a search is made for matching catch block. If a matching type is found then that block will be invoked. If a matching type is not found then the exception moves up the method stack and reaches the caller method. Same procedure is repeated if the caller method is included in a try catch block. This process continues until a catch block handling the appropriate type of exception is found. If it does not find such a block then finally the program terminates.

Q60: What are the different ways to handle exceptions?

A60: There are two ways to handle exceptions,
1. By wrapping the desired code in a try block followed by a catch block to catch the exceptions.
2. List the desired exceptions in the throws clause of the method and let the caller of the method handle those exceptions.

Q61: What is the basic difference between the 2 approaches to exception handling?
1) try catch block and

2) specifying the candidate exceptions in the throws clause
When should you use which approach?
A61: In the first approach as a programmer of the method, you yourself are dealing with the exception. This is fine if you are in a best position to decide should be done in case of an exception, Whereas if it is not the responsibility of the method to deal with its own exceptions, then do not use this approach. In this case use the second approach. In the second approach we are forcing the caller of the method to catch the exceptions that the method is likely to throw. This is often the approach library creator's use. They list the exception in the throws clause and we must catch them. You will find the same approach throughout the java libraries we use.

Q62: Is it necessary that each try block must be followed by a catch block?

A62: It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block OR a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

Q63: If I write return at the end of the try block, will the finally block still execute?

A63: Yes even if you write return as the last statement in the try block and no exception occurs, the finally block will execute. The finally block will execute and then the control return.

Q64: If I write System. exit(0); at the end of the try block, will the finally block still execute?
A64: No in this case the finally block will not execute because when you say System.exit (0); the control immediately goes out of the program, and thus finally never executes.

Q65: How are Observer and Observable used?

A65: Objects that subclass the Observable class maintains a list of observers. When an Observable object is updated it invokes the update () method of each of its observers to notify the observers that it has changed state. The Observer interface is implemented by objects that observe Observable objects.

Q66: What is synchronization and why is it important?

A66: With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

Q67: How does Java handle integer overflows and underflows?

A67: It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

Q68: Does garbage collection guarantee that a program will not run out of memory?

A68: Garbage collection does not guarantee that a program will not run out of memory. It is possible for programs to use up memory resources faster than they are garbage collected. It is also possible for programs to create objects that are not subject to garbage collection
.

Q69: What is the difference between preemptive scheduling and time slicing?

A69: Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

Q70: When a thread is created and started, what is its initial state?

A70: A thread is in the ready state after it has been created and started.

Q71: What is the purpose of finalization?

A71: The purpose of finalization is to give an unreachable object the opportunity to perform any cleanup processing before the object is garbage collected.

Q72: What is the Locale class?

A72: The Locale class is used to tailor program output to the conventions of a particular geographic, political, or cultural region.

Q73: What is the difference between a while statement and a do statement?

A73: A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will always execute the body of a loop at least once.

Q74: What is the difference between static and non-static variables?

A74: A static variable is associated with the class as a whole rather than with specific instances of a class. Non-static variables take on unique values with each object instance.

Q75: How are this() and super() used with constructors?

A75: This() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

Q76: What are synchronized methods and synchronized statements?

A76: Synchronized methods are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class. Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

Q77: What is daemon thread and which method is used to create the daemon thread?

A77: Daemon thread is a low priority thread which runs intermittently in the back ground doing the garbage collection operation for the java runtime system. setDaemon method is used to create a daemon thread.

Q78: Can applets communicate with each other?

A78: At this point in time applets may communicate with other applets running in the same virtual machine. If the applets are of the same class, they can communicate via shared static variables. If the applets are of different classes, then each will need a reference to the same class with static variables. In any case the basic idea is to pass the information back and forth through a static variable.

An applet can also get references to all other applets on the same page using the getApplets() method of java.applet.AppletContext. Once you get the reference to an applet, you can communicate with it by using its public members.

It is conceivable to have applets in different virtual machines that talk to a server somewhere on the Internet and store any data that needs to be serialized there. Then, when another applet needs this data, it could connect to this same server. Implementing this is non-trivial.

Q79: What are the steps in the JDBC connection?

A79: While making a JDBC connection we go through the following steps :

Step 1 : Register the database driver by using :

Class.forName(\" driver classs for that specific database\");

Step 2 : Now create a database connection using :

Connection con = DriverManager.getConnection(url,username,password);

Step 3: Now Create a query using :

Statement stmt = Connection.Statement(\"select * from TABLE NAME\");

Step 4 : Exceute the query :

stmt.exceuteUpdate();

Q80: How does a try statement determine which catch clause should be used to handle an exception?

A80: When an exception is thrown within the body of a try statement, the catch clauses of the try statement are examined in the order in which they appear. The first catch clause that is capable of handling the exceptionis executed. The remaining catch clauses are ignored.

Q81: Can an unreachable object become reachable again?

A81: An unreachable object may become reachable again. This can happen when the object's finalize() method is invoked and the object performs an operation which causes it to become accessible to reachable objects.

Q82: What method must be implemented by all threads?

A82: All tasks must implement the run() method, whether they are a subclass of Thread or implement the Runnable interface.

Q83: What are synchronized methods and synchronized statements?

A83: Synchronized methods are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class. Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

Q84: What is Externalizable?

A84: 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)

Q85: What modifiers are allowed for methods in an Interface?

A85: Only public and abstract modifiers are allowed for methods in interfaces.

Q86: What are some alternatives to inheritance?

A86: Delegation is an alternative to inheritance. Delegation means that you include an instance of another class as an instance variable, and forward messages to the instance. It is often safer than inheritance because it forces you to think about each message you forward, because the instance is of a known class, rather than a new class, and because it doesn't force you to accept all the methods of the super class: you can provide only the methods that really make sense. On the other hand, it makes you write more code, and it is harder to re-use (because it is not a subclass).

Q87: What does it mean that a method or field is "static"?

A87: Static variables and methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class.

Static methods can be referenced with the name of the class rather than the name of a particular object of the class (though that works too). That's how library methods like System.out.println() work out is a static field in the java.lang.System class.

88: What is the difference between preemptive scheduling and time slicing?

A88: Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

Q89: What is the catch or declare rule for method declarations?

A89: If a checked exception may be thrown within the body of a method, the method must either catch the exception or declare it in its throws clause.

Q90: Is Empty .java file a valid source file?

A90: Yes, an empty .java file is a perfectly valid source file.

Q91: Can a .java file contain more than one java classes?

A91: Yes, a .java file contain more than one java classes, provided at the most one of them is a public class.

Q92: Is String a primitive data type in Java?

A92: No String is not a primitive data type in Java, even though it is one of the most extensively used object. Strings in Java are instances of String class defined in java.lang package.

Q93: Is main a keyword in Java?

A93: No, main is not a keyword in Java.

Q94: Is next a keyword in Java?

A94: No, next is not a keyword.

95: Is delete a keyword in Java?

A95: No, delete is not a keyword in Java. Java does not make use of explicit destructors the way C++ does.

Q96: Is exit a keyword in Java?

A96: No. To exit a program explicitly you use exit method in System object.

Q97: What happens if you dont initialize an instance variable of any of the primitive types in Java?

A97: Java by default initializes it to the default value for that primitive type. Thus an int will be initialized to 0, a boolean will be initialized to false.

Q98: What will be the initial value of an object reference which is defined as an instance variable?

A98: The object references are all initialized to null in Java. However in order to do anything useful with these references, you must set them to a valid object, else you will get NullPointerExceptions everywhere you try to use such default initialized references.

Q99: What are the different scopes for Java variables?

A99: The scope of a Java variable is determined by the context in which the variable is declared. Thus a java variable can have one of the three scopes at any given point in time.
1. Instance : - These are typical object level variables, they are initialized to default

values at the time of creation of object, and remain accessible as long as the object accessible.
2. Local : - These are the variables that are defined within a method. They remain accessbile only during the course of method excecution. When the method finishes execution, these variables fall out of scope.
3. Static: - These are the class level variables. They are initialized when the class is loaded in JVM for the first time and remain there as long as the class remains loaded. They are not tied to any particular object instance.

Q100: What is the default value of the local variables?

A100: The local variables are not initialized to any default value, neither primitives nor object references. If you try to use these variables without initializing them explicitly, the java compiler will not compile the code. It will complain abt the local varaible not being initilized..

Q101: How many objects are created in the following piece of code?
MyClass c1, c2, c3;
c1 = new MyClass ();
c3 = new MyClass ();

A101: Only 2 objects are created, c1 and c3. The reference c2 is only declared and not initialized.

Q102: Can a public class MyClass be defined in a source file named YourClass.java?

A102: No the source file name, if it contains a public class, must be the same as the public class name itself with a .java extension.

Q103:Can main method be declared final?
A103: Yes, the main method can be declared final, in addition to being public static.

Q104: What will be the output of the following statement?
System.out.println ("1" + 3);

A104: It will print 13.

Q105: What will be the default values of all the elements of an array defined as an instance variable?

A105: If the array is an array of primitive types, then all the elements of the array will be initialized to the default value corresponding to that primitive type. e.g. All the elements of an array of int will be initialized to 0, while that of boolean type will be initialized to false. Whereas if the array is an array of references (of any type), all the elements will be initialized to null.

Q 106) what Should I use ServerSocket or DatagramSocket in my applications?

DatagramSocket allows a server to accept UDP packets, whereas ServerSocket allows an application to accept TCP connections. It depends on the protocol you're trying to implement. If you're creating a new protocol, here's a few tips

- DatagramSockets communciate using UDP packets. These packets don't guarantee delivery - you'll need to handle missing packets in your client/server
- ServerSockets communicate using TCP connections. TCP guarantees delivery, so all you need to do is have your applications read and write using a socket's InputStream and OutputStream.

Q 107) How do I get the IP address of a machine from its hostname?

The InetAddress class is able to resolve IP addresses for you. Obtain an instance of InetAddress for the machine, and call the getHostAddress() method, which returns a string in the xxx.xxx.xxx.xxx address form.

```
InetAddress inet = InetAddress.getByName("www.davidreilly.com");
System.out.println ("IP  : " + inet.getHostAddress());
```

Q 108) How do I perform a hostname lookup for an IP address?

The InetAddress class contains a method that can return the domain name of an IP address. You need to obtain an InetAddress class, and then call its getHostName() method. This will return the hostname for that IP address. Depending on the platform, a partial or a fully qualified hostname may be returned.

```
InetAddress inet = InetAddress.getByName("209.204.220.121");
System.out.println ("Host: " + inet.getHostName());
```

Q 109) How can I find out who is accessing my server?

If you're using a DatagramSocket, every packet that you receive will contain the address and port from which it was sent.

```
DatagramPacket packet = null;

// Receive next packet
myDatagramSocket.receive ( packet );

// Print address + port
System.out.println ("Packet from : " +
         packet.getAddress().getHostAddress() + ':' + packet.getPort());
```

If you're using a ServerSocket, then every socket connection you accept will contain similar information. The Socket class has a getInetAddress() and getPort() method which will allow you to find the same information.

```
Socket mySock = myServerSocket.accept();

// Print address + port
System.out.println ("Connection from : " +
         mySock.getInetAddress().getHostAddress() + ':' + mySock.getPort());
```

Q 110) How can I find out the current IP address for my machine?

The InetAddress has a static method called getLocalHost() which will return the current address of the local machine. You can then use the getHostAddress() method to get the IP address.

```
InetAddress local = InetAddress.getLocalHost();

// Print address
System.out.println ("Local IP : " + local.getHostAddress());
```

Q 111) Why can't my applet connect via sockets, or bind to a local port?

Applets are subject to heavy security constraints when executing under the control of a browser. Applets are unable to access the local file-system, to bind to local ports, or to connect to a computer via sockets other than the computer from which the applet is loaded. While it may seem to be an annoyance for developers, there are many good reasons why such tight constraints are placed on applets. Applets could bind to well known ports, and service network clients without authorization or consent. Applets executing within firewalls could obtain privileged information, and then send it across the network. Applets could even be infected by viruses, such as the Java StrangeBrew strain. Applets might become infected without an applet author's knowledge and then send information back that might leave hosts vulnerable to attack.

Signed applets may be allowed greater freedom by browsers than unsigned applets, which could be an option. In cases where an applet must be capable of network communication, HTTP can be used as a communication mechanism. An applet could communicate via java.net.URLConnection with a CGI script, or a Java servlet. This has an added advantage - applets that use the URLConnection will be able to communicate through a firewall.

Q 112) What are socket options, and why should I use them?

Socket options give developers greater control over how sockets behave. Most socket behavior is controlled by the operating system, not Java itself, but as of JDK1.1, you can control several socket options, including SO_TIMEOUT, SO_LINGER, TCP_NODELAY, SO_RCVBUF and SO_SNDBUF.

These are advanced options, and many programmers may want to ignore them. That's OK, but be aware of their existence for the future. You might like to specify a timeout for read operations, to control the amount of time a connection will linger for before a reset is sent, whether Nagle's algorithm is enabled/disabled, or the send and receive buffers for datagram sockets.

Q 113) When my client connects to my server, why does no data come out?

This is a common problem, made more difficult by the fact that the fault may lie in either the client, or the server, or both. The first step is to try and isolate the cause of the problem, by checking whether the server is responding correctly.

If you're writing a TCP service, then you can telnet to the port the server uses, and check to see if it is responding to data. If so, then the fault is more than likely in the client, and if not, you've found your problem. A debugger can be very helpful in tracking down the precise location of server errors. You could try jdb, which comes with JDK, or use an IDE's debugger like Visual J++ or Borland JBuilder.

If your fault looks like it is in the client, then it can often be caused by buffered I/O. If you're using a buffered stream, or a writer (such as PrintWriter), you may need to manually flush the data. Otherwise, it will be queued up but not sent, causing both client and server to stall. The problem can even be intermittent, as the buffer will flush sometimes (when it becomes full) but not other times.

Q 114)Name the containers which uses Border Layout as their default layout?
Ans:Containers which uses Border Layout as their default are: window, Frame and Dialog classes

Q 115)What are three ways in which a thread can enter the waiting state?
A thread can enter the waiting state by invoking its sleep() method, by blocking on IO, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's wait() method. It can also enter the waiting state by invoking its (deprecated) suspend() method.

Q 116)What is the preferred size of a component?
The preferred size of a component is the minimum component size that will allow the component to display normally.

Q 117)Does it matter in what order catch statements for FileNotFoundException and IOExceptipon are written?
Yes, it does. The FileNoFoundException is inherited from the IOException. Exception's subclasses have to be caught first.

Q 118)What is the purpose of the wait(), notify(), and notifyAll() methods?
The wait(),notify(), and notifyAll() methods are used to provide an efficient way for threads to communicate each other.

Q 119)What is the difference between yielding and sleeping?
When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep() method, it returns to the waiting state

Q 120)What is the difference between Process and Thread?
A process can contain multiple threads. In most multithreading operating systems, a process gets its own memory address space; a thread doesn't. Threads typically share the heap belonging to their parent process. For instance, a JVM runs in a single process in the host O/S. Threads in the JVM share the heap belonging to

that process; that's why several threads may access the same object. Typically, even though they share a common heap, threads have their own stack space. This is how one thread's invocation of a method is kept separate from another's. This is all a gross oversimplification, but it's accurate enough at a high level. Lots of details differ between operating systems. Process vs. Thread A program vs. similar to a sequential program an run on its own vs. Cannot run on its own Unit of allocation vs. Unit of execution Have its own memory space vs. Share with others Each process has one or more threads vs. Each thread belongs to one process Expensive, need to context switch vs. Cheap, can use process memory and may not need to context switch More secure. One process cannot corrupt another process vs. Less secure. A thread can write the memory used by another thread

Q 121)Can an inner class declared inside of a method access local variables of this method?
It's possible if these variables are final.

Q 122)What is a native method?
A native method is a method that is implemented in a language other than Java.

Q 123)How many methods in Object class?
This question is not asked to test your memory. It tests you how well you know Java. Ten in total.
clone()
equals() & hashcode()
getClass()
finalize()
wait() & notify()
toString()

Q 124)What is JDBC Driver interface?
The JDBC Driver interface provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each vendor driver must provide implementations of the java.sql.Connection,Statement,PreparedStatement, CallableStatement, ResultSet and Driver.

Q 125)What are four types of JDBC driver?
Type 1 Drivers

Bridge drivers such as the jdbc-odbc bridge. They rely on an intermediary such as ODBC to transfer the SQL calls to the database and also often rely on native code. It is not a serious solution for an application
Type 2 Drivers

Use the existing database API to communicate with the database on the client. Faster than Type 1, but need native code and require additional permissions to work in an applet. Client machine requires software to run.
Type 3 Drivers

JDBC-Net pure Java driver. It translates JDBC calls to a DBMS-independent network protocol, which is then translated to a DBMS protocol by a server. Flexible. Pure Java and no native code.
Type 4 Drivers

Native-protocol pure Java driver. It converts JDBC calls directly into the network protocol used by DBMSs. This allows a direct call from the client machine to the

DBMS server. It doesn't need any special native code on the client machine. Recommended by Sun's tutorial, driver type 1 and 2 are interim solutions where direct pure Java drivers are not yet available. Driver type 3 and 4 are the preferred way to access databases using the JDBC API, because they offer all the advantages of Java technology, including automatic installation. For more info, visit Sun JDBC page

Q 126)Which type of JDBC driver is the fastest one?
JDBC Net pure Java driver(Type IV) is the fastest driver because it converts the jdbc calls into vendor specific protocol calls and it directly interacts with the database.

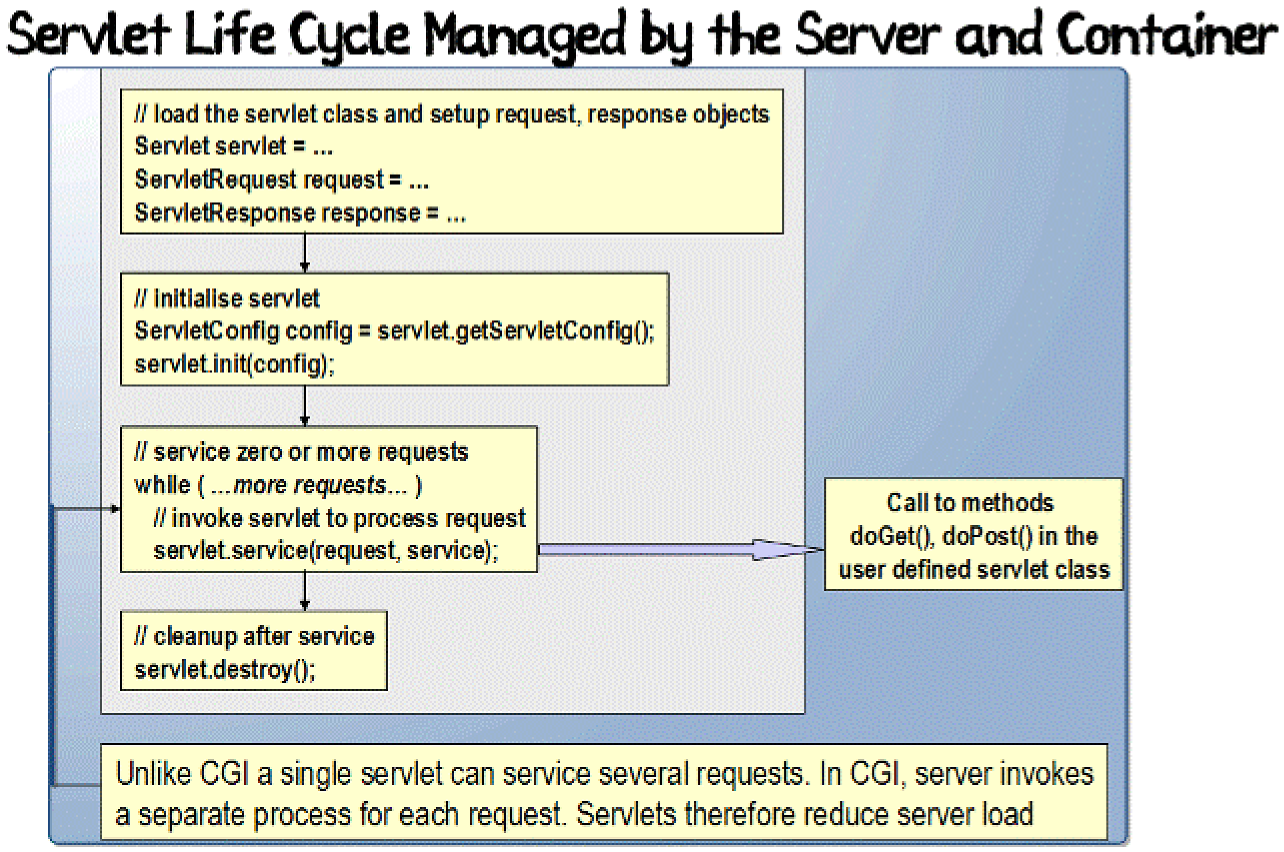
Q 127) What are the advantages of Servlet over CGI?
Servlets have several advantages over CGI:

A Servlet does not run in a separate process. This removes the overhead of creating a new process for each request.
A Servlet stays in memory between requests. A CGI program (and probably also an extensive runtime system or interpreter) needs to be loaded and started for each CGI request.
There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.
Several web.xml conveniences
A handful of removed restrictions
Some edge case clarifications

Q 128) What are the phases of the servlet life cycle?
The life cycle of a servlet consists of the following phases:

Servlet class loading: For each servlet defined in the deployment descriptor of the Web application, the servlet container locates and loads a class of the type of the servlet. This can happen when the servlet engine itself is started, or later when a client request is actually delegated to the servlet.
Servlet instantiation: After loading, it instantiates one or more object instances of the servlet class to service the client requests.
Initialization (call the init method) : After instantiation, the container initializes a servlet before it is ready to handle client requests. The container initializes the servlet by invoking its init() method, passing an object implementing the ServletConfig interface. In the init() method, the servlet can read configuration parameters from the deployment descriptor or perform any other one-time activities, so the init() method is invoked once and only once by the servlet container.
Request handling (call the service method) : After the servlet is initialized, the container may keep it ready for handling client requests. When client requests arrive, they are delegated to the servlet through the service() method, passing the request and response objects as parameters. In the case of HTTP requests, the request and response objects are implementations of HttpServletRequest and HttpServletResponse respectively. In the HttpServlet class, the service() method invokes a different handler method for each type of HTTP request, doGet() method for GET requests, doPost() method for POST requests, and so on.
Removal from service (call the destroy method) : A servlet container may decide to remove a servlet from service for various reasons, such as to conserve memory resources. To do this, the servlet container calls the destroy() method on the servlet. Once the destroy () method has been called, the servlet may not service any more client requests. Now the servlet instance is eligible for garbage collection

The life cycle of a servlet is controlled by the container in which the servlet has been deployed.


Servlet Life Cycle Managed by the Server and Container
// load the servlet class and setup request, response objects
Servlet servlet = ...
ServletRequest request = ...
ServletResponse response = ...
// Initialise servlet
ServletConfig config = servlet.getServletConfig();
servlet.init(config);
// service zero or more requests
while (...more requests...)
// invoke servlet to process request
servlet.service(request, service);
// cleanup after service
servlet.destroy();
Call to methods doGet(), doPost() in the user defined servlet class
Unlike CGI a single servlet can service several requests. In CGI, server invokes a separate process for each request. Servlets therefore reduce server load

Q 129) Why do we need a constructor in a servlet if we use the init method?

Even though there is an init method in a servlet which gets called to initialize it, a constructor is still required to instantiate the servlet. Even though you as the developer would never need to explicitly call the servlet's constructor, it is still being used by the container (the container still uses the constructor to create an instance of the servlet). Just like a normal POJO (plain old java object) that might have an init method, it is no use calling the init method if you haven't constructed an object to call it on yet.

Q 130) How the servlet is loaded?

A servlet can be loaded when:

First request is made.
Server starts up (auto-load).
There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.
Administrator manually loads.

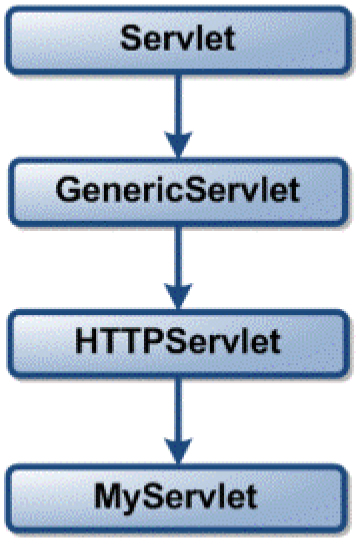
Q 131)How a Servlet is unloaded?

A servlet is unloaded when:

Server shuts down.

Administrator manually unloads.

Q 132) What is Servlet interface?
The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly by extending a class that implements it.



Note: Most Servlets, however, extend one of the standard implementations of that interface, namely `javax.servlet.GenericServlet` and `javax.servlet.http.HttpServlet`.

Q 133)What is the GenericServlet class?
GenericServlet is an abstract class that implements the Servlet interface and the ServletConfig interface. In addition to the methods declared in these two interfaces, this class also provides simple versions of the lifecycle methods init and destroy, and implements the log method declared in the ServletContext interface.
Note: This class is known as generic servlet, since it is not specific to any protocol.

Q 134) What's the difference between GenericServlet and HttpServlet?

GenericServlet	HttpServlet
The GenericServlet is an abstract class that is extended by HttpServlet to provide HTTP protocol-specific methods.	An abstract class that simplifies writing HTTP servlets. It extends the GenericServlet base class and provides an framework for handling the HTTP protocol.
The GenericServlet does not include protocol-specific methods for handling request parameters, cookies, sessions and setting response headers.	The HttpServlet subclass passes generic service method requests to the relevant doGet() or doPost() method.
GenericServlet is not specific to any protocol.	HttpServlet only supports HTTP and HTTPS protocol.

Q 135) Why HttpServlet is declared abstract?
The HttpServlet class is declared abstract because the default implementations of the main service methods do nothing and must be overridden. This is a convenience implementation of the Servlet interface, which means that developers do not need to implement all service methods. If your servlet is required to handle `doGet ()` requests for example, there is no need to write a `doPost()` method too.

Q 136) Can servlet have a constructor?

One can definitely have constructor in servlet. Even you can use the constructor in servlet for initialization purpose, but this type of approach is not so common. You can perform common operations with the constructor as you normally do. The only thing is that you cannot call that constructor explicitly by the new keyword as we normally do. In the case of servlet, servlet container is responsible for instantiating the servlet, so the constructor is also called by servlet container only.

Q 137) What are the types of protocols supported by HttpServlet?

It extends the GenericServlet base class and provides a framework for handling the HTTP protocol. So, HttpServlet only supports HTTP and HTTPS protocol.

Q 138) What is the difference between doGet () and doPost ()?

#	doGet()	doPost()
1	In doGet () the parameters are appended to the URL and sent along with header information.	In doPost (), on the other hand will (typically) send the information through a socket back to the webserver and it won't show up in the URL bar.
2	The amount of information you can send back using a GET is restricted as URLs can only be 1024 characters.	You can send much more information to the server this way - and it's not restricted to textual data either. It is possible to send files and even binary data such as serialized Java objects!
3	doGet () is a request for information; it does not (or should not) change anything on the server. (doGet() should be idempotent)	doPost() provides information (such as placing an order for merchandise) that the server is expected to remember
4	Parameters are not encrypted	Parameters are encrypted
5	doGet() is faster if we set the response content length since the same connection is used. Thus increasing the performance	doPost() is generally used to update or post some information to the server.doPost is slower compared to doGet since doPost does not write the content length
6	doGet() should be idempotent. i.e. doget should be able to be repeated safely many times	This method does not need to be idempotent. Operations requested through POST can have side effects for which the user can be held accountable.
7	doGet() should be safe without any side effects for which user is held responsible	This method does not need to be either safe
8	It allows bookmarks.	It disallows bookmarks.

Q139) When to use doGet() and when doPost()?

Always prefer to use GET (As because GET is faster than POST), except mentioned in the following reason:

If data is sensitive
Data is greater than 1024 characters
If your application don't need bookmarks.

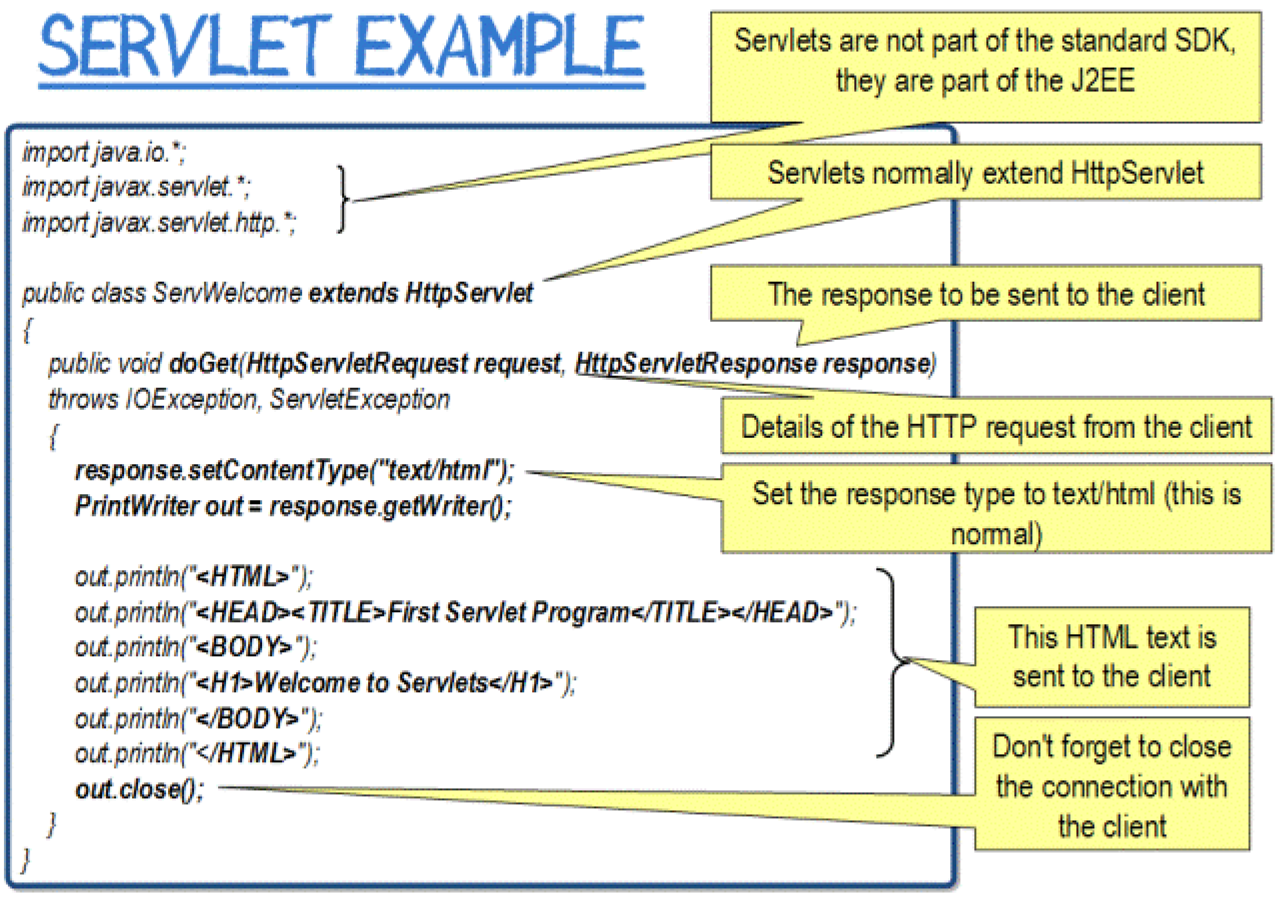
Q 140) How do I support both GET and POST from the same Servlet?

The easy way is, just support POST, then have your doGet method call your doPost method:

```
 public void doGet(HttpServletRequest request,
HttpServletResponse response)
                              throws ServletException, IOException
{
    doPost(request, response);
}
```

Q 141) Should I override the service () method?
We never override the service method, since the HTTP Servlets have already taken care of it. The default service function invokes the doXXX() method corresponding to the method of the HTTP request. For example, if the HTTP request method is GET, doGet () method is called by default. A servlet should override the doXXX() method for the HTTP methods that servlet supports. Because HTTP service method checks the request method and calls the appropriate handler method, it is not necessary to override the service method itself. Only override the appropriate doXXX() method.

Q 142) How the typical servlet code look like?



```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServWelcome extends HttpServlet
{
  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws IOException, ServletException
  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<HTML>");
    out.println("<HEAD><TITLE>First Servlet Program</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<H1>Welcome to Servlets</H1>");
    out.println("</BODY>");
    out.println("</HTML>");
    out.close();
  }
}
```

SERVLET EXAMPLE
Servlets are not part of the standard SDK, they are part of the J2EE
Servlets normally extend HttpServlet
The response to be sent to the client
Details of the HTTP request from the client
Set the response type to text/html (this is normal)
This HTML text is sent to the client
Don't forget to close the connection with the client

Q 143) What is a servlet context object?
A servlet context object contains the information about the Web application of which the servlet is a part. It also provides access to the resources common to all the servlets in the application. Each Web application in a container has a single servlet context associated with it.

Q 144) What are the differences between the ServletConfig interface and the ServletContext interface?

ServletConfig	ServletContext
The ServletConfig interface is implemented by the servlet container in order to pass configuration information to a servlet. The server passes an object that implements the ServletConfig interface to the servlet's init() method.	A ServletContext defines a set of methods that a servlet uses to communicate with its servlet container.
There is one ServletConfig parameter per servlet.	There is one ServletContext for the entire webapp and all the servlets in a webapp share it.
The param-value pairs for ServletConfig object are specified in the <init-param> within the <servlet> tags in the web.xml file	The param-value pairs for ServletContext object are specified in the <context-param> tags in the web.xml file.

Q 145) What's the difference between forward () and sendRedirect () methods?

forward()	sendRedirect()
A forward is performed internally by the servlet.	A redirect is a two step process, where the web application instructs the browser to fetch a second URL, which differs from the original.
The browser is completely unaware that it has taken place, so its original URL remains intact.	The browser, in this case, is doing the work and knows that it's making a new request.
Any browser reload of the resulting page will simple repeat the original request, with the original URL	A browser reloads of the second URL, will not repeat the original request, but will rather fetch the second URL.
Both resources must be part of the same context (Some containers make provisions for cross-context communication but this tends not to be very portable)	This method can be used to redirect users to resources that are not part of the current context, or even in the same domain.
Since both resources are part of same context, the original request context is retained	Because this involves a new request, the previous request scope objects, with all of its parameters and attributes are no longer available after a redirect. (Variables will need to be passed by via the session object).
Forward is marginally faster than redirect.	sendRedirect() is marginally slower than a forward, since it requires two browser requests, not one.

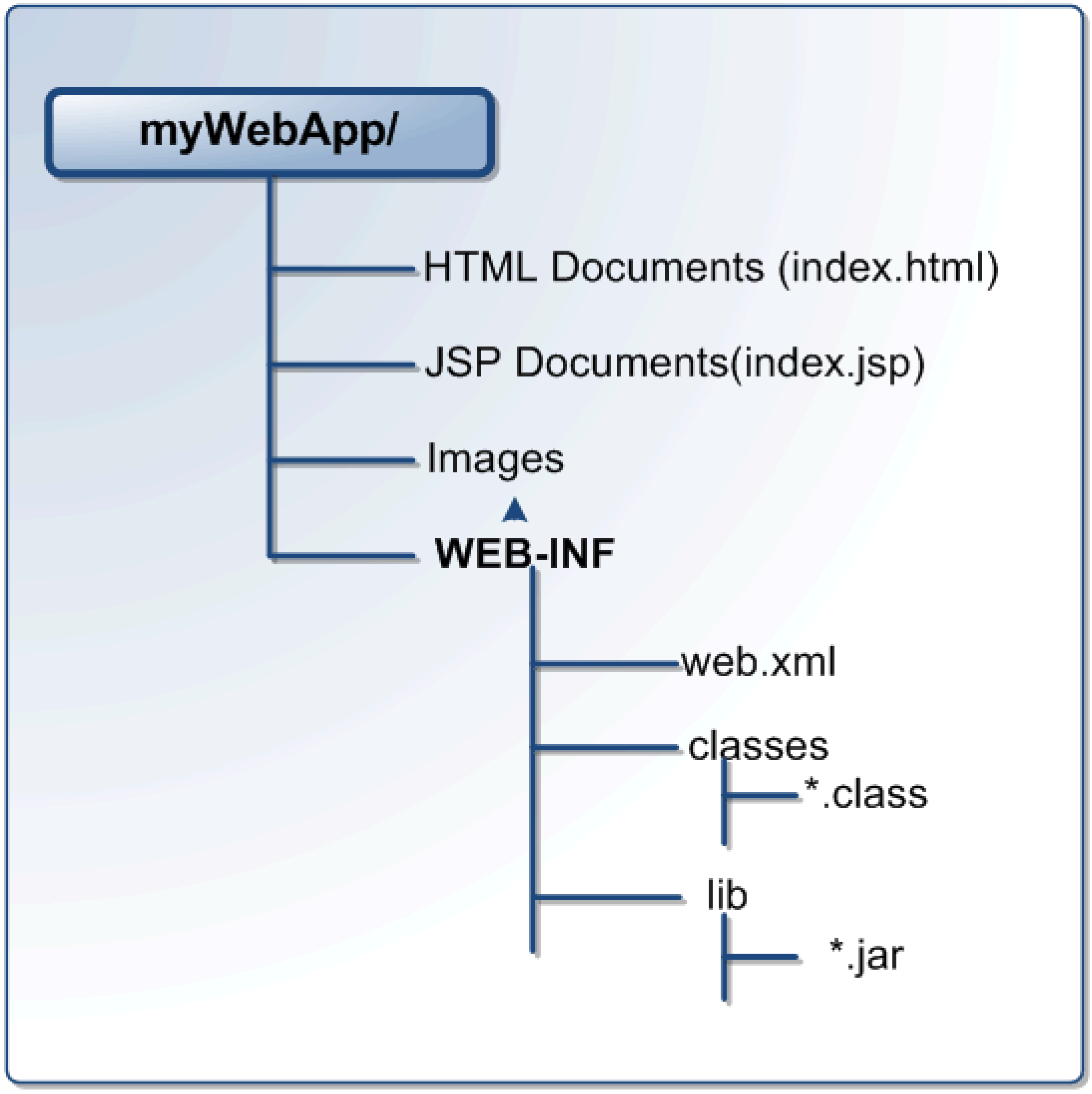
Q 146) What is the difference between the include() and forward() methods?

include()	forward()
The `RequestDispatcher include()` method inserts the the contents of the specified resource directly in the flow of the servlet	The `RequestDispatcher forward()` method is used to show a different resource in place of the servlet that was originally called.

response, as if it were part of the calling servlet.	
If you include a servlet or JSP document, the included resource must not attempt to change the response status code or HTTP headers, any such request will be ignored.	The forwarded resource may be another servlet, JSP or static HTML document, but the response is issued under the same URL that was originally requested. In other words, it is not the same as a redirection.
The `include()` method is often used to include common "boilerplate" text or template markup that may be included by many servlets.	The `forward()` method is often used where a servlet is taking a controller role; processing some input and deciding the outcome by returning a particular response page.

Q 147) What's the use of the servlet wrapper classes??
The `HttpServletRequestWrapper` and `HttpServletResponseWrapper` classes are designed to make it easy for developers to create custom implementations of the servlet request and response types. The classes are constructed with the standard `HttpServletRequest` and `HttpServletResponse` instances respectively and their default behaviour is to pass all method calls directly to the underlying objects.

Q 148) What is the directory structure of a WAR file?



Q 149) What is a deployment descriptor?

A deployment descriptor is an XML document with an .xml extension. It defines a component's deployment settings. It declares transaction attributes and security authorization for an enterprise bean. The information provided by a deployment descriptor is declarative and therefore it can be modified without changing the source code of a bean.
The JavaEE server reads the deployment descriptor at run time and acts upon the component accordingly.

Q 150) What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?

ServletRequest.getRequestDispatcher(String path)	ServletContext.getRequestDispatcher(String path)
The `getRequestDispatcher(String path)` method of `javax.servlet.ServletRequest` interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root.	The `getRequestDispatcher(String path)` method of `javax.servlet.ServletContext` interface cannot accept relative paths. All path must start with a "/" and are interpreted as relative to current context root.

Q 151) What is preinitialization of a servlet?

A container does not initialize the servlets as soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

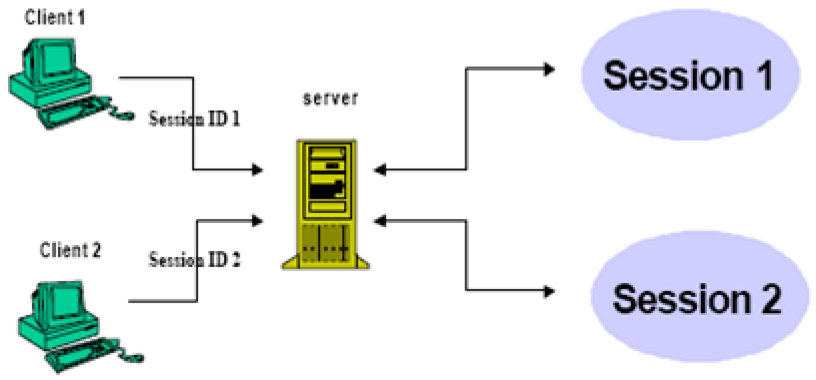
Q 152)What is the <load-on-startup> element?

The `<load-on-startup>` element of a deployment descriptor is used to load a servlet file when the server starts instead of waiting for the first request. It is also used to specify the order in which the files are to be loaded. The <load-on-startup> element is written in the deployment descriptor as follows:

```
<servlet>

   <servlet-name>ServletName</servlet-name>

   <servlet-class>ClassName</servlet-class>

   <load-on-startup>1</load-on-startup>

</servlet>
```

Q153) What is session?

A session refers to all the requests that a single client might make to a server in the course of viewing any pages associated with a given application. Sessions are specific to both the individual user and the application. As a result, every user of an application has a separate session and has access to a separate set of session variables.



Q154) What is Session Tracking?

Session tracking is a mechanism that servlets use to maintain state about a series of requests from the same user (that is, requests originating from the same browser) across some period of time.

Q155) What is the need of Session tracking in web application?

HTTP is a stateless protocol i.e., every request is treated as new request. For web applications to be more realistic they have to retain information across multiple requests. Such information which is part of the application is referred as "state". To keep track of this state we need session tracking.

Typical example: Putting things one at a time into a shopping cart, then checking out--each page request must somehow be associated with previous requests.

Q156) What are the types of Session Tracking?

Sessions need to work with all web browsers and take into account the users security preferences. Therefore there are a variety of ways to send and receive the identifier:

URL rewriting: URL rewriting is a method of session tracking in which some extra data (session ID) is appended at the end of each URL. This extra data identifies the session. The server can associate this session identifier with the data it has stored about that session. This method is used with browsers that do not support cookies or where the user has disabled the cookies.
Hidden Form Fields: Similar to URL rewriting. The server embeds new hidden fields in every dynamically generated form page for the client. When the client submits the form to the server the hidden fields identify the client.
Cookies: Cookie is a small amount of information sent by a servlet to a Web browser. Saved by the browser, and later sent back to the server in subsequent requests. A cookie has a name, a single value, and optional attributes. A cookie's value can uniquely identify a client.
Secure Socket Layer (SSL) Sessions: Web browsers that support Secure Socket Layer communication can use SSL's support via HTTPS for generating a unique session key as part of the encrypted conversation.

Q157) How do I use cookies to store session state on the client?

In a servlet, the HttpServletResponse and HttpServletRequest objects passed to method HttpServlet.service() can be used to create cookies on the client and use cookie information transmitted during client requests. JSPs can also use cookies, in scriptlet code or, preferably, from within custom tag code.

To set a cookie on the client, use the addCookie() method in class HttpServletResponse. Multiple cookies may be set for the same request, and a single cookie name may have multiple values.
To get all of the cookies associated with a single HTTP request, use the getCookies() method of class HttpServletRequest

Q158) What are some advantages of storing session state in cookies?

Cookies are usually persistent, so for low-security sites, user data that needs to be stored long-term (such as a user ID, historical information, etc.) can be maintained easily with no server interaction.
For small- and medium-sized session data, the entire session data (instead of just the session ID) can be kept in the cookie.

Q159) What are some disadvantages of storing session state in cookies?

Cookies are controlled by programming a low-level API, which is more difficult to implement than some other approaches.
All data for a session are kept on the client. Corruption, expiration or purging of cookie files can all result in incomplete, inconsistent, or missing information.
Cookies may not be available for many reasons: the user may have disabled them, the browser version may not support them, the browser may be behind a firewall that filters cookies, and so on. Servlets and JSP pages that rely exclusively on cookies for client-side session state will not operate properly for all clients. Using cookies, and then switching to an alternate client-side session state strategy in cases where cookies aren't available, complicates development and maintenance.
Browser instances share cookies, so users cannot have multiple simultaneous sessions.
Cookie-based solutions work only for HTTP clients. This is because cookies are a feature of the HTTP protocol. Notice that the while package `javax.servlet.http` supports session management (via class `HttpSession`), package `javax.servlet` has no such support.

Q160) What is URL rewriting?

URL rewriting is a method of session tracking in which some extra data is appended at the end of each URL. This extra data identifies the session. The server can associate this session identifier with the data it has stored about that session.

Every URL on the page must be encoded using method `HttpServletResponse`.encodeURL(). Each time a URL is output, the servlet passes the URL to encodeURL(), which encodes session ID in the URL if the browser isn't accepting cookies, or if the session tracking is turned off.
E.g., http://abc/path/index.jsp;jsessionid=123465hfhs

Advantages

URL rewriting works just about everywhere, especially when cookies are turned off.
Multiple simultaneous sessions are possible for a single user. Session information is local to each browser instance, since it's stored in URLs in each page being displayed. This scheme isn't foolproof, though, since users can start a new browser instance using a URL for an active session, and confuse the server by interacting with the same session through two instances.
Entirely static pages cannot be used with URL rewriting, since every link must be dynamically written with the session state. It is possible to combine static and dynamic content, using (for example) templating or server-side includes. This limitation is also a barrier to integrating legacy web pages with newer, servlet-based pages.

Disadvantages

Every URL on a page which needs the session information must be rewritten each time a page is served. Not only is this expensive computationally, but it can greatly increase communication overhead.
URL rewriting limits the client's interaction with the server to HTTP GETs, which can result in awkward restrictions on the page.
URL rewriting does not work well with JSP technology.
If a client workstation crashes, all of the URLs (and therefore all of the data for that session) are lost.

Q 161) How can an existing session be invalidated?

An existing session can be invalidated in the following two ways:

Setting timeout in the deployment descriptor: This can be done by specifying timeout between the `<session-timeout>`tags as follows:

```
<session-config>
       <session-timeout>10</session-timeout>
</session-config>
```

This will set the time for session timeout to be ten minutes.

Setting timeout programmatically: This will set the timeout for a specific session. The syntax for setting the timeout programmatically is as follows:

```
public void setMaxInactiveInterval(int interval)
```

The `setMaxInactiveInterval()` method sets the maximum time in seconds before a session becomes invalid.

Note: Setting the inactive period a*s negative(-1), makes the container stop tracking session, i.e, session never expires.*

Q162) How can the session in Servlet can be destroyed?

An existing session can be destroyed in the following two ways:

Programatically : Using `session.invalidate()` method, which makes the container abandon the session on which the method is called.
When the server itself is shutdown.

Q163) A client sends requests to two different web components. Both of the components access the session. Will they end up using the same session object or different session ?

Creates only one session i.e., they end up with using same session .

Sessions is specific to the client but not the web components. And there is a 1-1 mapping between client and a session.

Q164) What is servlet lazy loading?

A container doesn't initialize the servlets ass soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading.
The servlet specification defines the <load-on-startup> element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up.
The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

Q165) What is Servlet Chaining?

Servlet Chaining is a method where the output of one servlet is piped into a second servlet. The output of the second servlet could be piped into a third servlet, and so on. The last servlet in the chain returns the output to the Web browser.

Q166) How are filters?

Filters are Java components that are used to intercept an incoming request to a Web resource and a response sent back from the resource. It is used to abstract any useful information contained in the request or response. Some of the important functions performed by filters are as follows:

Security checks
Modifying the request or response
Data compression
Logging and auditing
Response compression

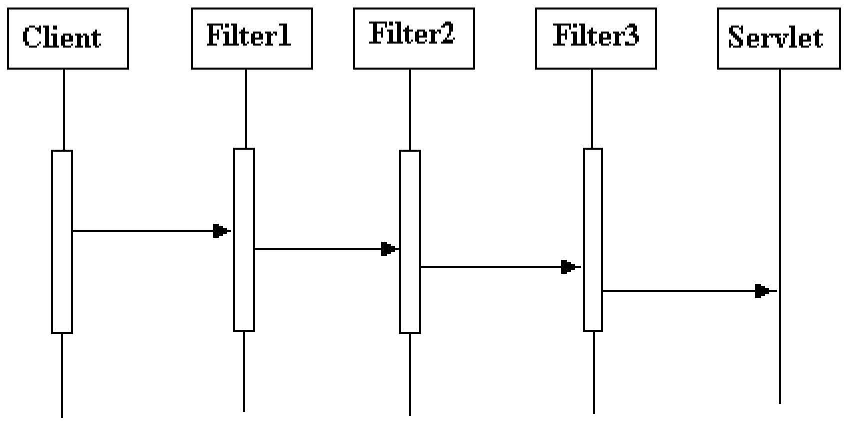
Filters are configured in the deployment descriptor of a Web application. Hence, a user is not required to recompile anything to change the input or output of the Web application.

Q167) What are the functions of an intercepting filter?

The functions of an intercepting filter are as follows:

It intercepts the request from a client before it reaches the servlet and modifies the request if required.

It intercepts the response from the servlet back to the client and modifies the request if required.
There can be many filters forming a chain, in which case the output of one filter becomes an input to the next filter. Hence, various modifications can be performed on a single request and response.



Q168) What are the functions of the Servlet container?
The functions of the Servlet container are as follows:

Lifecycle management: It manages the life and death of a servlet, such as class loading, instantiation, initialization, service, and making servlet instances eligible for garbage collection.
Communication support: It handles the communication between the servlet and the Web server.
Multithreading support: It automatically creates a new thread for every servlet request received. When the Servlet service () method completes, the thread dies.
Declarative security: It manages the security inside the XML deployment descriptor file.
JSP support: The container is responsible for converting JSPs to servlets and for maintaining them.

Q 169)Explain the life cycle methods of a Servlet.

A: The javax.servlet.Servlet interface defines the three methods known as life-cycle method.
public void init(ServletConfig config) throws ServletException
public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
public void destroy()
First the servlet is constructed, then initialized wih the ***init()*** method.
Any request from client are handled initially by the ***service()*** method before delegating to the ***doXxx()*** methods in the case of HttpServlet.

The servlet is removed from service, destroyed with the ***destroy()*** methid, then garbaged collected and finalized.

Q 170)What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?
A: The getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to,

which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root.

The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accepts relative paths. All path must sart with a "/" and are interpreted as relative to curent context root.

Q 171)Explain the directory structure of a web application.
A: The directory structure of a web application consists of two parts.
A private directory called WEB-INF
A public resource directory which contains public resource folder.

WEB-INF folder consists of
1. web.xml
2. classes directory
3. lib directory

Q 172) What are the common mechanisms used for session tracking?

A: Cookies
SSLsessions
URL- rewriting

Q 173) Explain ServletContext.
A: ServletContext interface is a window for a servlet to view it's environment. A servlet can use this interface to get information such as initialization parameters for the web application or servlet container's version. Every web application has one and only one ServletContext and is accessible to all active resource of that application.

Q 174) What is preinitialization of a servlet?

A: A container doesn't initialize the servlets as soon as it starts up, it initializes a servlet when it receives a request for that servlet first time. This is called lazy loading. The servlet specification defines the <load-on-startup> element, which can be specified in the deployment descriptor to make the servlet container load and initialize the servlet as soon as it starts up. The process of loading a servlet before any request comes in is called preloading or preinitializing a servlet.

Q 175) What is the difference between Difference between doGet() and doPost()?

A: A doGet() method is limited with 2k of data to be sent, and doPost() method doesn't have this limitation. A request string for doGet() looks like the following:
http://www.allapplabs.com/svt1?p1=v1&p2=v2&...&pN=vN
doPost() method call doesn't need a long text tail after a servlet name in a request. All parameters are stored in a request itself, not in a request string, and it's impossible to guess the data transmitted to a servlet only looking at a request string.

Q176) What is the difference between HttpServlet and GenericServlet?

A: A GenericServlet has a service() method aimed to handle requests. HttpServlet extends GenericServlet and adds support for doGet(), doPost(), doHead() methods (HTTP 1.0) plus doPut(), doOptions(), doDelete(), doTrace() methods (HTTP 1.1). Both these classes are abstract.

Q177) What is the difference between ServletContext and ServletConfig?

Ans: ServletContext: Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.The ServletContext object is contained within the ServletConfig object, which the Web server provides the servlet when the servlet is initialized

ServletConfig: The object created after a servlet is instantiated and its default constructor is read. It is created to pass initialization information to the servlet.

Q178) What is an output comment?

Ans:A comment that is sent to the client in the viewable page source.The JSP engine handles an output comment as uninterpreted HTML text, returning the comment in the HTML output sent to the client. You can see the comment by viewing the page source from your Web browser.

JSP Syntax

```
<!-- comment [ <%= expression %> ] -->
```

Example 1

```
<!-- This is a commnet sent to client on
<%= (new java.util.Date()).toLocaleString() %>
-->
```

Displays in the page source:

```
<!-- This is a comment sent to client on January 24, 2004 -->
```

Q179) What is a Hidden Comment?

A: A comment that documents the JSP page but is not sent to the client. The JSP engine ignores a hidden comment, and does not process any code within hidden comment tags. A hidden comment is not sent to the client, either in the displayed JSP page or the HTML page source. The hidden comment is useful when you want to hide or "comment out" part of your JSP page.

You can use any characters in the body of the comment except the closing --%> combination. If you need to use --%> in your comment, you can escape it by typing --%\>.

JSP Syntax

```
<%-- comment -- %>
```

Examples

```
<%@ page language="java" %>
<html>
```

```
<head><title>A Hidden Comment </title></head>
<body>
<%-- This comment will not be visible to the colent in the page source --%>
</body>
</html>
```

Q180) What is an Expression?

A: An expression tag contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file. Because the value of an expression is converted to a String, you can use an expression within text in a JSP file. Like

```
<%= someexpression %>
<%= (new java.util.Date()).toLocaleString() %>
```

You cannot use a semicolon to end an expression

Q181) What is a Declaration?

A: A declaration declares one or more variables or methods for use later in the JSP source file.

A declaration must contain at least one complete declarative statement. You can declare any number of variables or methods within one declaration tag, as long as they are separated by semicolons. The declaration must be valid in the scripting language used in the JSP file.

```
<%! somedeclarations %>
<%! int i = 0; %>
<%! int a, b, c; %>
```

Q182)What is a Scriptlet?

Ans: A scriptlet can contain any number of language statements, variable or method declarations, or expressions that are valid in the page scripting language.Within scriptlet tags, you can

1. Declare variables or methods to use later in the file (see also Declaration).

2. Write expressions valid in the page scripting language (see also Expression).

3. Use any of the JSP implicit objects or any object declared with a <jsp:useBean> tag.

You must write plain text, HTML-encoded text, or other JSP tags outside the scriptlet.

Scriptlets are executed at request time, when the JSP engine processes the client request. If the scriptlet produces output, the output is stored in the out object, from which you can display it.

Q183)What are implicit objects? List them?

A: Certain objects those are available for the use in JSP documents without being declared first. These objects are parsed by the JSP engine and inserted into the generated servlet. The implicit objects re listed below

- request
- response
- pageContext
- session
- application
- out
- config
- page
- exception

Q184) What is the Difference between forward and sendRedirect?

A: When you invoke a forward request, the request is sent to another resource on the server, without the client being informed that a different resource is going to process the request. This process occurs completely within the web container. When a **sendRedirtect** method is invoked, it causes the web container to return to the browser indicating that a new URL should be requested. Because the browser issues a completely new request any object that are stored as request attributes before the redirect occurs will be lost. This extra round trip a redirect is slower than forward.

Q185) What are the different scope values for the <jsp:useBean>?

A: The different scope values for <jsp:useBean> are

1. page
2. request
3. session
4. application

Q186) Explain the life-cycle methods in JSP?

A: The generated servlet class for a JSP page implements the HttpJspPage interface of the javax.servlet.jsp package. HttpJspPage interface extends the JspPage interface which inturn extends the Servlet interface of the javax.servlet package. The generated servlet class thus implements all the methods of these three interfaces. The JspPage interface declares only two mehtods - *jspInit()* and *jspDestroy()* that must be implemented by all JSP pages regardless of the client-server protocol. However the JSP specification has provided the HttpJspPage interfaec specifically for the JSP pages serving HTTP requests. This interface declares one method *_jspService()*.
The jspInit()- The container calls the jspInit() to initialize te servlet instance.It is called before any other method, and is called only once for a servlet instance.
The _jspservice()- The container calls the _jspservice() for each request, passing it the request and the response objects.
The jspDestroy()- The container calls this when it decides take the instance out of service. It is the last method called n the servlet instance.

Q187) What is the servlet?
Ans: Servlet is a server side programming language Which takes request in form of Http and gives response in the form of Http.

Q188)What is the architecture of servlet package?
Ans: Servlet Interface is the central abstraction. All servlets implements this Servlet

Interface either directly or indirectly (may implement or extend Servlet Interfaces sub classes or sub interfaces)

Servlet
|
Generic Servlet
|
HttpServlet (Class) -- we will extend this class to handle GET / PUT HTTP requests
|
MyServlet

Q189)What is the difference between HttpServlet and GenericServlet?
Ans: A GenericServlet has a service() method to handle requests.HttpServlet extends GenericServlet added new methods
doGet()
doPost()
doHead()
doPut()
doOptions()
doDelete()
doTrace() methods
Both these classes are abstract.

Q190) What's the difference between servlets and applets?
Ans: Servlets executes on Servers. Applets executes on browser. Unlike applets, however, servlets have no graphical user interface.

Q191) What are the uses of Servlets?
Ans: A servlet can handle multiple requests concurrently, and can synchronize requests. Servlets can forward requests to other servers and servlets. Thus servlets can be used to balance load among several servers.

Q192) When doGET() method will going to execute?
Ans: When we specified method='GET' in HTML
Example : < form name='SSS' method='GET'>

Q193) When doPOST() method will going to execute?
Ans: When we specified method='POST' in HTML
< form name='SSS' method='POST' >

Q194) What is the difference between Difference between doGet() and doPost()?
Ans: GET Method Using get method we can able to pass 2K data from HTML
All data we are passing to Server will be displayed in URL (request string).This is used to sent small size data.
POST Method: In this method we do not have any size limitation. All data passed to server will be hidden; User cannot able to see this info on the browser. .The data is send into small packets. This used to send large amount of data.

Q195) What is the servlet life cycle?
Ans: When first request came in for the servlet, Server will invoke **init ()** method of the

servlet. There after if any user request the servlet program, Server will directly executes the **service ()** method. When Server want to remove the servlet from pool, then it will execute the **destroy ()** method. The **init()** and **destroy()** methods are called at only one time. But **service ()** method can call at each request. The service () method creates a separate thread for each request and gives response according to request.

Q196) Which code line must be set before any of the lines that use the PrintWriter?
Ans:setContentType() method must be set.

Q197) Which protocol will be used by browser and servlet to communicate?
Ans: HTTP protocol.

Q198) In how many ways we can track the sessions?
Ans: There are four ways to track session:
Method 1) By URL rewriting
Method 2) Using Session object
Getting Session form **HttpServletRequest** object
HttpSession session = request.**getSession(true)**;

Get a Value from the session
session.getValue(session.getId());

Adding values to session
cart = new Cart();
session.putValue(session.getId(), cart);

At the end of the session, we can inactivate the session by using the following command
session.invalidate();
Method 3) Using cookies
Method 4) Using hidden fields

Q199)How Can You invoke other web resources (or other servelt / jsp) ?
Ans: Servelt can invoke other Web resources in two ways: indirect and direct.
Indirect Way : Servlet will return the resultant HTML to the browser which will point to another Servlet (Web resource)
Direct Way: We can call another Web resource (Servelt / Jsp) from Servelt program itself, by using **RequestDispatcher** object.
You can get this object using getRequestDispatcher("URL") method. You can get this object from either a request or a Context.
Example:
RequestDispatcher dispatcher = request.getRequestDispatcher("/jspsample.jsp");
if (dispatcher != null)
dispatcher.forward(request, response);
}

Q 200) How Can you include other Resources in the Response?
Ans: Using include method of a **RequestDispatcher** object.Included WebComponent (Servlet / Jsp) cannot set headers or call any method (for example, setCookie) that affects the headers of the response.
Example :

```
 RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/banner");
if (dispatcher != null)
dispatcher.include(request, response);
}
```

Q 201)What is the difference between the getRequestDispatcher(String path) ServletRequest interface and ServletContext interface?
Ans: The **getRequestDispatcher(String path)** method of **ServletRequest** interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root.

The **getRequestDispatcher(String path)** method of ServletContext interface cannot accepts relative paths. All path must sart with a "/" and are interpreted as relative to curent context root. If the resource is not available, or if the server has not implemented a RequestDispatcher object for that type of resource, getRequestDispatcher will return null. Your servlet should be prepared to deal with this condition.

Q 202) What is the use of ServletContext?
Ans: Using ServletContext, We can access data from its environment. Servlet context is common to all Servlets so all Servlets share the information through ServeltContext.

Q 203) Is there any way to generate PDF'S dynamically in servlets?
Ans: need to use iText. A open source library for java. Please refer sourceforge site for sample servlet examples.

Q 204) What is the difference between using getSession(true) and getSession(false) methods?
Ans:getSession(true): This method will check whether already a session is existing for the user. If a session is existing, it will return that session object, otherwise it will create new session object and return taht object.
getSession(false): This method will check existence of session. If session exists, then it returns the reference of that session object, if not, this methods will return null.

EJB FAQ

Q 205)What is the Java^{TM} 2 Platform, Enterprise Edition (J2EE)?

Java 2 Platform, Enterprise Edition (J2EE) is a platform that enables solutions for developing, deploying and managing multi-tier server-centric applications. J2EE utilizes Java 2 Platform, Standard Edition to extend a complete, stable, secure, fast Java platform to the enterprise level. It delivers value to the enterprise by enabling a platform which significantly reduces the cost and complexity of developing multi-tier solutions, resulting in services that can be rapidly deployed and easily enhanced.

Q 206) What are the main benefits of J2EE?

J2EE provides the following:

1. A unified platform for building, deploying and managing enterprise-class software without locking users into a vendor specific-architecture and saves IT time.
2. A platform that will allow enterprise-class application the ability to run anywhere.
3. A platform with a complete range of readily available enterprise-class services.
4. A single easy-to-learn blueprint programming model for J2EE.
5. A platform that is built upon and leverages existing IT investments and guarantees that enterprise-class software will work on multiple platforms.

Q 207) What technologies are included in J2EE?

The primary technologies in J2EE are: Enterprise JavaBeansTM, JavaServer PagesTM, servlets, the Java Naming and Directory InterfaceTM (JNDI), the Java Transaction API (JTA), CORBA, and the JDBCTM data access API.

Q 208) How does J2EE relate to Enterprise JavaBeans technology?

Enterprise JavaBeans (EJB) technology is the basis of J2EE. EJB technology provides the scaleable architecture for executing business logic in a distributed computing environment. J2EE makes the life of an enterprise developer easier by combining the EJB component architecture with other enterprise technologies to solutions on the Java platform for seamless development and deployment of server side applications.

Q 209) Who needs J2EE?

ISVs need J2EE because it gives them a blueprint for providing a complete enterprise computing solution on the Java platform. Enterprise developers need J2EE because writing distributed business applications is hard, and they need a high-productivity solution that allows them to focus only on writing their business logic and having a full range of enterprise-class services to rely on, like transactional distributed objects, message oriented middleware, and naming and directory services.

Q 210)What is the purpose of the Reference Implementation?

The purpose of the reference implementation is to validate the specifications. In short, it is to prove that the specifications can be implemented.

Q 211) Why don't you allow the binary reference implementation to be deployed or redistributed?

We do not allow the binary reference implementation to be deployed or redistributed at the request of our partners. The J2EE reference implementation is essentially a full-featured application server. To make it available on the market would provide a product that competes with the companies that we want to adopt the technology. In this light, we set up the licensing terms to honor this request.

Q 212)Is XML supported in J2EE?

XML is an essential component in the J2EE platform. J2EE will provide a framework for business-to-business data interchange using XML. Currently, JavaServer Pages framework can be used to generate and consume XML between servers or between server and client. In addition, Enterprise JavaBeans component architecture uses XML to describe its deployment properties, giving Enterprise JavaBeans data portability in addition to its code portability.

Q 213) What is Enterprise JavaBeans ^{TM} (EJB)?

EJB, the widely-adopted server-side component architecture for Java^{TM} 2 Platform, Enterprise Edition (J2EE), enables rapid development of mission-critical application that are versatile, reusable and portable across middleware while protecting IT investment and preventing vendor lock-in.

Q 214) What is EJB technology's role in J2EE?

EJB technology is the core of J2EE. It enables developers to write reusable portable server-side business logic for the J2EE platform.

Q 215)Is EJB a product?

No, EJB is a specification, which defines the EJB component architecture and the interfaces between the Enterprise JavaBeans technology-enabled server and the component. Commercial vendors like IBM, BEA, Sun and Oracle are providing products that implement the EJB specification.

Q 216)What are the key features of the EJB technology?

- EJB components are server-side components written entirely in the Java programming language
- EJB components contain business logic only - no System-level programming
- System-level services (i.e. "plumbing") such as transactions, security, Life-cycle, threading, persistence, etc. are automatically managed for the EJB component by the EJB server
- EJB architecture is inherently transactional, distributed, portable, multi-tier, scalable and secure
- Components are declaratively customized at deployment time (Can customize: transactional behavior, security features, life-cycle, state management, persistence, etc.)
- EJB components are fully portable across any EJB server and any OS
- EJB architecture is wire-protocol neutral - Any protocol can be utilized: IIOP, JRMP, HTTP, DCOM, etc.

Q 217)What are the key benefits of the EJB technology?

- Rapid application development: The productivity benefits of writing components in the Java programming language - Java technology productivity and component reuse and outsourcing; Developer focuses on business logic only; Declarative customization (not programmatic)
- Broad industry adoption: Industry-wide collaboration on the spec yielded a superior architecture and ensured adoption; Choice and flexibility in server selection - no vendor lock-in; 3rd party components are widely usable across servers - ISVs don't need to choose vendor-specific development platform
- Application portability: Business logic that runs everywhere - Platform-independence and middleware independence
- Choice, not vendor lock-in: Architecture decisions are made at deployment, not at development; Inter-server portability - code can be deployed on any EJB server; Inter-server scalability - Servers can be transparently replaced to accommodate changing needs for service level, performance or security; Any wire protocol can be selected at deployment

Protection of IT investment: Wrap and embrace existing infrastructure, application and data stores; Existing middleware solutions are being adapted by the well-established vendors to support the EJB technology via a thin portability layer; Portable across multiple servers and databases; Serve multi-lingual clients

- Browsers, Java technology, ActiveX, or CORBA clients; EJB technology simplifies and enhances CORBA and DCOM.

Q 218)What platforms does EJB technology run on?

Applications that are based on EJB components are not only platform independent but also middleware independent! They can run on any OS and any middleware that support EJB.

Q 219)Who is EJB technology for?

EJB technology benefits a number of audiences:

- Enterprise customers that build and/or deploy EJB-based applications - gain development productivity, can choose from a wide selection of EJB servers, create business logic that runs everywhere and is architecture independent, all this while protecting their existing IT investment!
- ISVs and SIs that develop EJB components or applications based on EJB components - Invest in business logic that is widely deployable, across any OS and middleware, don't need to choose one vendor-specific server platform. Like enterprise customers they also benefit from productivity gains and architecture independence
- The EJB specification itself is mostly targeted at the EJB server vendors - It is the blueprint that instructs these vendors on how to build an EJB server that EJB components can execute on successfully

Q 220)Who is using EJB technology?

The adoption and momentum around EJB technology since its first public release in 3/98 has been phenomenal. Vendors across the board, from the smallest start-up to the largest corporation, are adopting this technology and building EJB servers, EJB tools and EJB components and applications. We are also seeing strong interest and early adoption among many enterprise customers such as: Visa, GTE, Instinet (Reuters), Qwest, Charles Schwab, NationsBank Montegomery and many others.

Q 221)Is EJB technology really going to fulfill the promise of true portability across middleware solutions?

Yes. EJB technology hides the details of the underlying middleware "plumbing", allowing developers to focus on writing "pure" business logic which does not contain server-specific code. The code can execute on top of any server implementing the EJB specification. Sun is providing a J2EE compliance program to guarantee portability across servers.

Q 221)What are the key features included in EJB 1.1?

- Mandatory support for entity beans which will improve the way EJB-based applications connect to back-end data sources such as relational databases and greatly simplify the process of application development by providing developers with a higher level of abstraction.
- Enhancements to the Deployment Descriptor, including support for XML text format and improved content organization, both of which will ease development and deployment of EJB-based applications
- Significant tightening of the specification which will greatly improve EJB-based application portability

Q 223)What are the key features expected to be included in EJB 2.0?

EJB 2.0 is currently in the Expert Group phase of the JCP specification development process. Based on the JSR (Java Specification Request) which has been approved, enhancements are expected in the following areas:

- JMS (Java Message Service) integration
- Improved persistence management
- Support for RMI/IIOP protocol for network interoperability
- Management of beans relationships
- Standardized container extensibility

Q 224)Doesn't EJB technology compete with CORBA?

No. In fact, EJB technology complements CORBA quite nicely. CORBA provides a great Standards-based infrastructure on which to build EJB servers. EJB technology makes it easier to build application on top of a CORBA infrastructure. Additionally, the recently released CORBA components specification refers to EJB as the architecture when building CORBA components in Java.

Q 225)What about RMI/IIOP? Is that part of EJB specification?

The EJB specification mandates that you must use the RMI interfaces to communicate with an EJB component. The EJB 1.1 specification does not mandate which wire protocol to use. However, EJB 2.0 is expected to required EJB implementations to Support RMI/IIOP.

Q 226)How does EJB compare with MTS (Microsoft Transaction Server)?

EJB is a widely endorsed and supported architecture. MTS is a proprietary product. EJB allows for choice. MTS allows only one choice, Microsoft's.

Q 227)Why should people want to use products based on EJB technology when they can get MTS for free with Windows NT?

Because EJB technology eliminates platform lock, allows a user to move their application to any EJB server, eliminates dependencies on hardware, OS and middleware.

Q 228)What are the design goals of the Enterprise JavaBeans^{TM} architecture?

The Enterprise JavaBeans specification defines a standard architecture for implementing the business logic of multi-tier applications as reusable components. In addition to Enterprise JavaBeans components, the architecture defines three other entities: servers, containers, and clients. This architecture incorporates several design goals:

- Enterprise JavaBeans servers are designed to wrap around legacy systems to provide fundamental services for containers and the components they contain.
- Enterprise JavaBeans containers are designed to handle details of component life-cycle, transaction, and security management. By interceding between clients and components at the method call level, containers can manage transactions that propagate across calls and components, and even across containers running on different servers and different machines. This mechanism simplifies development of both component and clients.
- Component developers are free to focus on business logic, since containers provide services automatically by interceding in component method calls. A simple set of callback interfaces are all that a developer needs to implement to participate in container provided services.
- A client's view of an Enterprise JavaBean remains the same regardless of the container it is deployed in. Any container in which an Enterprise JavaBean is deployed presents the same interfaces to the client. This extends to containers from different vendors, running

against different servers and different databases, on diverse systems on a network. This client transparency ensures wide scalability for multi-tier applications.

- Along with container managed transactions, the Enterprise JavaBeans architecture enables component- and client-managed transactions. Containers can participate in component or client initiated transactions to enforce transaction rules across method call and component boundaries. Components can also specify transaction types by method, enabling them to mix transaction types within a single object.
- A variety of Enterprise JavaBean attributes, including the default component transaction type, can be specified at either development or deployment time, and enforced through mechanisms built into the container architecture.

The Enterprise JavaBeans architecture is based on the Java programming language, so enterprise Beans take full advantage of the "write once, run anywhere™" standard.

Q 229) What is EJB?

Ans: Enterprise JavaBeans (EJB) technology is the server-side component architecture for the Java 2 Platform, Enterprise Edition (J2EE) platform. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.

Q 230) What are the different type of Enterprise JavaBeans ?

Ans There are 3 types of enterprise beans, namely: Session bean, Entity beans and Message driven beans.

Q 231) What is Session Bean ?

Ans: Session bean represents a single client inside the J2EE server. To access the application deployed in the server the client invokes methods on the session bean. The session bean performs the task shielding the client from the complexity of the business logic.

Session bean components implement the javax.ejb.SessionBean interface. Session beans can act as agents modeling workflow or provide access to special transient business services. Session beans do not normally represent persistent business concepts. A session bean corresponds to a client server session. The session bean is created when a client requests some query on the database and exists as long as the client server session exists.

Q232) What are different types of session bean?

Ans There are two types of session beans, namely: Stateful and Stateless.

Q233) What is a Stateful Session bean?

Ans Stateful session bean maintains the state of the conversation between the client and itself. When the client invokes a method on the bean the instance variables of the bean may contain a state but only for the duration of the invocation.

A stateful session bean is an enterprise bean (EJB component) that acts as a server-side extension of the client that uses it. The stateful session bean is created by a client and will work for only that client until the client connection is dropped or the bean is explicitly removed. The stateful session bean is EJB component that implements the javax.ejb.SessionBean interface and is deployed with the declarative attribute "stateful". Stateful session beans are called "stateful" because they maintain a conversational state with the client. In other words, they have state or instance fields that can be initialized and changed by the client with each method invocation. The bean can use the conversational state as it process business methods invoked by the client.

Q234) What is stateless session bean?

Ans Stateless session beans are of equal value for all instances of the bean. This means the container can assign any bean to any client, making it very scalable.

A stateless session bean is an enterprise bean that provides a stateless service to the client. Conceptually, the business methods on a stateless session bean are similar to procedural applications or static methods; there is no instance state, so all the data needed to execute the method is provided by the method arguments. The stateless session bean is an EJB component that implements the javax.ejb.SessionBean interface and is deployed with the declarative attribute "stateless". Stateless session beans are called "stateless" because they do not maintain conversational state specific to a client session. In other words, the instance fields in a stateless session bean do not maintain data relative to a client session. This makes stateless session beans very lightweight and fast, but also limits their behavior. Typically an application requires less number of stateless beans compared to stateful beans.

Q235) What is an Entity Bean?

Ans An entity bean represents a business object in a persistent storage mechanism. An entity bean typically represents a table in a relational database and each instance represents a row in the table. Entity bean differs from session bean by: persistence, shared access, relationship and primary key. T

Q236) What are different types of entity beans?
Ans There are two types of entity beans available. Container Managed Persistence (CMP) , Bean managed persistence (BMP).

Q237) What is CMP (Container Managed Persistence) ?

Ans: Term container-managed persistence means that the EJB container handles all database access required by the entity bean. The bean's code contains no database access (SQL) calls. As a result, the bean's code is not tied to a specific persistent storage mechanism (database). Because of this flexibility, even if you redeploy the same entity bean on different J2EE servers that use different databases, you won't need to modify or recompile the bean's code. So, your entity beans are more portable.

Q238) What is BMP (Bean managed persistence)?

Ans: Bean managed persistence (BMP) occurs when the bean manages its persistence. Here the bean will handle all the database access. So the bean's code contains the necessary SQLs calls. So it is not much portable compared to CMP. Because when we are changing the database we need to rewrite the SQL for supporting the new database.

Q239) What is abstract schema?
Ans: In order to generate the data access calls, the container needs information that you provide in the entity bean's abstract schema. It is a part of Deployment Descriptor. It is used to define the bean's persistent fields and relation ships.

Q240) When we should use Entity Bean?

Ans: When the bean represents a business entity, not a procedure. we should use an entity bean. Also when the bean's state must be persistent we should use an entity bean. If the bean instance terminates or if the J2EE server is shut down, the bean's state still exists in persistent storage (a database).

Q241) When to Use Session Beans?

Ans: At any given time, only one client has access to the bean instance. The state of the bean is not persistent, existing only for a short period (perhaps a few hours). The bean implements a web service. Under all the above circumstances we can use session beans.

Q242) When to use Stateful session bean?

Ans: The bean's state represents the interaction between the bean and a specific client. The bean needs to hold information about the client across method invocations. The bean mediates between the client and the other components of the application, presenting a simplified view to the client. Under all the above circumstances we can use a stateful session bean.

Q243) When to use a stateless session bean?

Ans: The bean's state has no data for a specific client. In a single method invocation, the bean performs a generic task for all clients. For example, you might use a stateless session bean to send an email that confirms an online order. The bean fetches from a database a set of read-only data that is often used by clients. Such a bean, for example, could retrieve the table rows that represent the products that are on sale this month. Under all the above circumstance we can use a stateless session bean.

Q244) Why Use EJB?

Ans: EJB helps in building enterprise applications simply. A developer of EJB needs to focus on business logic only. All other features like transaction; persistence etc. will be managed by the container. EJB provides developers architectural independence

Q245) What are the different methods of Entity Bean?

Ans: An entity bean consists of 4 type of methods: create methods, finder methods, remove methods and home methods

Q246) What are create methods of Entity Bean?

Ans: Create methods are used to create a new instance of a CMP entity bean. The create() method on bean's home interface returns an object of remote. ejbCreate(parameters) methods are used for creating Entity Bean instances according to the parameters specified and to some programmer-defined conditions. We can declare more than one create() methods in home interface, and each of which must have a corresponding ejbCreate() and ejbPostCreate() methods in the bean class. These creation methods are linked at run time, so that when a create() method is invoked on the home interface, the container delegates the invocation to the corresponding ejbCreate() and ejbPostCreate() methods on the bean class.

Q247) What are finder methods of Entity Bean?

Ans: Finder methods are used to query for specific entity beans. These are methods declared in home interface and begin with find. There are two kinds of finder methods, single-entity and multi-entity. Single-entity finder methods return a remote object that matches given find request. If no records found, this method throws an ObjectNotFoundException . The multi-entity finder methods return a collection (Enumeration or Collection type) of entities that match the find request. If no entities are found, finder returns an empty collection.

Q248) What are remove methods of Entity Bean ?

Ans: Remove methods allow the client to remove an Entity bean by specifying either Handle or a Primary Key of that Entity Bean.

Q249) What are home methods in Entity Bean?

Ans: Home methods are methods that are designed and implemented by a developer according to his/her needs. EJB specification doesn't have any requirements for home methods except they need to throw a RemoteException.

Q250) What are different callback methods in Entity beans?

Ans: The bean class implements a set of callback methods that allow the container to notify the events in its life cycle. The call back methods available in Entity Bean are
public void setEntityContext();
public void unsetEntityContext();
public void ejbLoad();
public void ejbStore();
public void ejbActivate();
public void ejbPassivate();
public void ejbRemove();

Q251)What is the use of setEntityContext in Entity bean?

Ans: The setEntityContext() method is used to set the EntityContext interface for that bean. The EntityContext contains information about the context under which bean is operating. EntityContext interface gives security information about caller. The EntityContext is set only once in the life time of an entity bean instance

Q252)What is the use of unsetEntityContext in Entity Bean?

Ans: The unsetEntityContext() method is called at the end of a bean's life cycle before the instance is unloaded from memory. It is used to dereference EntityContext and to perform any clean up operations if required.

Q253)What are ejbLoad and ejbStore methods of Entity Bean?

Ans:ejbLoad method is primarily used for data retrievals. ejbStore is used for updating data. Typically the container invokes ejbLoad before the first business method in a transaction and the ejbStore is invoked at the end of the transaction. ejbStore method will be invoked when we change some values in memory.

Q254)What are ejbActivate and ejbPassivate methods of Entity Bean?

Ans: The ejbPassivate() is invoked by the container before the beans is passivated and ejbActivate() is invoked by the container after the bean is activated. Passivation and activation is used to save resources. passivation means, dissociating a bean instance from its EJB object. Activation is the process of associating a bean with EJB object. Stateless session beans are never passivated.

Q255) What is the architecture of EJB?

Ans: Every EJB is having three classes. A home interface which acts as a factory of remote objects. A remote object which is used for client interaction and a bean object which contains all the business logic.

Q256) Can an Entity bean have zero create methods?

Ans: An entity bean ca have zero or more create methods. If there is no create methods we will not be able insert data to the database. So that

Q257) What is the default transaction attribute in EJB?

Ans: The default transaction attribute is 'supports'

Q258) What are the different transaction attributes ?

Ans There are six different transaction attributes available: Not Supported, Required, Supports, RequiresNew, Mandatory, Never.

Q259)What are the different kinds of enterprise beans?

Ans: Stateless session bean- An instance of these non-persistent EJBs provides a service without storing an interaction or conversation state between methods. Any instanc can be used for any client.

Stateful session bean- An instance of these non-persistent EJBs maintains state across methods and transactions. Each instance is associated with a particular client.

Entity bean- An instance of these persistent EJBs represents an object view of the data, usually rows in a database. They have a primary key as a unique identifier. Entity bean persistence can be either container-managed or bean-managed.

Message-driven bean- An instance of these EJBs is integrated with the Java Message Service (JMS) to provide the ability for message-driven beans to act as a standard JMS message consumer and perform asynchronous processing between the server and the JMS message producer.

Q260)What is Session Bean?

A:A session bean is a non-persistent object that implements some business logic running on the server. One way to think of a session object is as a logical extension of the client program that runs on the server.

Session beans are used to manage the interactions of entity and other session beans, access resources, and generally perform tasks on behalf of the client.

There are two basic kinds of session bean: stateless and stateful.

Stateless session beans are made up of business methods that behave like procedures; they operate only on the arguments passed to them when they are invoked. Stateless beans are called stateless because they are transient; they do not maintain business state between method invocations. Each invocation of a stateless business method is independent from previous invocations. Because stateless session beans are stateless, they are easier for the EJB container to manage, so they tend to process requests faster and use fewer resources.

Stateful session beans encapsulate business logic and state specific to a client. Stateful beans are called "stateful" because they do maintain business state between method invocations, held in memory and not persistent. Unlike stateless session beans, clients do not share stateful beans. When a client creates a stateful bean, that bean instance is dedicated to service only that client. This makes it possible to maintain conversational state, which is business state that can be shared by methods in the same stateful bean.

Q261)What is Entity Bean?

A:The entity bean is used to represent data in the database. It provides an object-oriented interface to data that would normally be accessed by the JDBC or some other back-end API. More than that, entity beans provide a component model that allows bean developers to focus their attention on the business logic of the bean, while the container takes care of managing persistence, transactions, and access control.

There are two basic kinds of entity beans: container-managed persistence (CMP) and bean-managed persistence (BMP).

Container-managed persistence beans are the simplest for the bean developer to create and the most difficult for the EJB server to support. This is because all the logic for synchronizing the bean's state with the database is handled automatically by the container. This means that the bean developer doesn't need to write any data access logic, while the EJB server is
supposed to take care of all the persistence needs automatically. With CMP, the container manages the persistence of the entity bean. Vendor tools are used to map the entity fields to the database and absolutely no database access code is written in the bean class.

The bean-managed persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container. The bean uses a database API to read and write its fields to the database, but the container tells it when to do each synchronization operation and manages the transactions for the bean automatically. Bean-managed persistence gives the bean developer the flexibility to perform persistence operations that are too complicated for the container or to use a data source that is not supported by the container.

Q262) What are the methods of Entity Bean? What is the difference between Container-Managed Persistent (CMP) bean and Bean-Managed Persistent (BMP) ?

A: Container-managed persistence beans are the simplest for the bean developer to create and the most difficult for the EJB server to support. This is because all the logic for synchronizing the bean's state with the database is handled automatically by the container. This means that the bean developer doesn't need to write any data access logic, while the EJB server is supposed to take care of all the persistence needs automatically. With CMP, the container manages the persistence of the entity bean. A CMP bean developer doesn't need to worry about JDBC code and transactions, because the Container performs database calls and transaction management instead of the programmer. Vendor tools are used to map the entity fields to the database and absolutely no database access code is written in the bean class. All table mapping is specified in the deployment descriptor. Otherwise, a BMP bean developer takes the load of linking an application and a database on his shoulders.

The bean-managed persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container. The bean uses a database API to read and write its fields to the database, but the container tells it when to do each synchronization operation and manages the transactions for the bean automatically. Bean-managed persistence gives the bean developer the flexibility to perform persistence operations that are too complicated for the container or to use a data source that is not supported by the container. BMP beans are not 100% database-independent, because they may contain database-specific code, but CMP beans are unable to perform complicated DML (data manipulation language) statements. EJB 2.0 specification introduced some new ways of querying database (by using the EJB QL - query language).

Q263) What are the callback methods in Entity beans?

A: The bean class defines create methods that match methods in the home interface and business methods that match methods in the remote interface. The bean class also implements a set of callback methods that allow the container to notify the bean of events in its life cycle. The callback methods are defined in the **javax.ejb.EntityBean** interface that is implemented by all entity beans.The EntityBean interface has the following definition. Notice that the bean class implements these methods.

```
public interface javax.ejb.EntityBean {

public void setEntityContext();
public void unsetEntityContext();
public void ejbLoad();
public void ejbStore();
public void ejbActivate();
public void ejbPassivate();
public void ejbRemove();
}
```

The **setEntityContext()** method provides the bean with an interface to the container called the EntityContext. The EntityContext interface contains methods for obtaining information about the context under which the bean is operating at any particular moment. The EntityContext interface is used to access security information about the caller; to determine the status of the current transaction or to force a transaction rollback; or to get a reference to the bean itself, its home, or its primary key. The EntityContext is set only once in the life of an entity bean instance, so its reference should be put into one of the bean instance's fields if it will be needed later.

The **unsetEntityContext()** method is used at the end of the bean's life cycle before the instance is evicted from memory to dereference the EntityContext and perform any last-minute clean-up.

The **ejbLoad()** and **ejbStore()** methods in CMP entities are invoked when the entity bean's state is being synchronized with the database. The ejbLoad() is invoked just after the container has refreshed the bean container-managed fields with its state from the database. The ejbStore() method is invoked just before the container is about to write the bean container-managed fields to the database. These methods are used to modify data as it's being synchronized. This is common when the data stored in the database is different than the data used in the bean fields.

The **ejbPassivate()** and **ejbActivate()** methods are invoked on the bean by the container just before the bean is passivated and just after the bean is activated, respectively. Passivation in entity beans means that the bean instance is disassociated with its remote reference so that the container can evict it from memory or reuse it. It's a resource conservation measure the container employs to reduce the number of instances in memory. A bean might be passivated if it hasn't been used for a while or as a normal operation performed by the container to maximize reuse of resources. Some containers will evict beans from memory, while others will reuse instances for other more active remote references. The ejbPassivate() and ejbActivate() methods provide the bean with a notification as to when it's about to be passivated (disassociated with the remote reference) or activated (associated with a remote reference).

Q264) What is software architecture of EJB?

A: Session and Entity EJBs consist of 4 and 5 parts respetively:

1. A remote interface (a client interacts with it),

2. A home interface (used for creating objects and for declaring business methods),

3. A bean objects (an object, which actually performs business logic and EJB-specific operations).

4. A deployment descriptor (an XML file containing all information required for maintaining the EJB) or a set of deployment descriptors (if you are using some container-specific features).

5. A Primary Key class - is only Entity bean specific.

Q265) Can Entity Beans have no create () methods?

A: Yes. In some cases the data is inserted NOT using Java application, so you may only need to retrieve the information, perform its processing, but not create your own information of this kind.

Q266) What is bean managed transaction?

A: If a developer doesn't want a Container to manage transactions, it's possible to implement all database operations manually by writing the appropriate JDBC code. This often leads to productivity increase, but it makes an Entity Bean incompatible with some databases and it enlarges the amount of code to be written. All transaction management is explicitly performed by a developer.

Q267) What are transaction attributes?

A: The transaction attribute specifies how the Container must manage transactions for a method when a client invokes the method via the enterprise bean’s home or component interface or when the method is invoked as the result of the arrival of a JMS message. (Sun's EJB Specification) Below is a list of transactional attributes:

1.NotSupported - transaction context is unspecified.

2. Required - bean's method invocation is made within a transactional context. If a client is not associated with a transaction, a new transaction is invoked automatically.

3. Supports - if a transactional context exists, a Container acts like the transaction attribute is Required, else - like NotSupported.

4. RequiresNew - a method is invoked in a new transaction context.

5. Mandatory - if a transactional context exists, a Container acts like the transaction attribute is Required, else it throws a javax.ejb.TransactionRequiredException.

6. Never - a method executes only if no transaction context is specified.

Q268) What are transaction isolation levels in EJB?

A: 1. Transaction_read_uncommitted- Allows a method to read uncommitted data from a DB(fast but not wise).

2. Transaction_read_committed- Guarantees that the data you are getting has been committed.

3. Transaction_repeatable_read - Guarantees that all reads of the database will be the same during the transaction (good for read and update operations).

4. Transaction_serializable- All the transactions for resource are performed serial.