Java Interview Questions

Q: What is the difference between an Interface and an Abstract class?
A: An abstract class can have instance methods that implement a default behavior. An Interface can only declare constants and instance methods, but cannot implement default behavior and all methods are implicitly abstract. An interface has all public members and no implementation. An abstract class is a class which may have the usual flavors of class members (private, protected, etc.), but has some abstract methods.
.

Q: What is the purpose of garbage collection in Java, and when is it used?
A: The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused. A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used.

Q: Describe synchronization in respect to multithreading.
A: With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

Q: Explain different way of using thread?
A: The thread could be implemented by using runnable interface or by inheriting from the Thread class. The former is more advantageous, 'cause when you are going for multiple inheritance..the only interface can help.

Q: What are pass by reference and passby value?
A: Pass By Reference means the passing the address itself rather than passing the value. Passby Value means passing a copy of the value to be passed.

Q: What is HashMap and Map?
A: Map is Interface and Hashmap is class that implements that.

Q: Difference between HashMap and HashTable?
A: The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesnt allow). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is unsynchronized and Hashtable is synchronized.

Q: Difference between Vector and ArrayList?
A: Vector is synchronized whereas arraylist is not.

Q: Difference between Swing and Awt?
A: AWT are heavy-weight componenets. Swings are light-weight components. Hence swing works faster than AWT.

Q: What is the difference between a constructor and a method?
A: A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator.
A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

Q: What is an Iterator?
A: Some of the collection classes provide traversal of their contents via a java.util.Iterator interface. This interface allows you to walk through a collection of objects, operating on each object in turn. Remember when using Iterators that they contain a snapshot of the collection at the time the Iterator was obtained; generally it is not advisable to modify the collection itself while traversing an Iterator.

Q: 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.
A: ***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.

Q: What is an abstract class?
A: Abstract class must be extended/subclassed (to be useful). It serves as a template. A class that is abstract may not be instantiated (ie, you may not call its constructor), abstract class may contain static data. Any class with an abstract method is automatically abstract itself, and must be declared as such.
A class may be declared abstract even if it has no abstract methods. This prevents it from being instantiated.

Q: What is static in java?
A: 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.

Q: What is final?
A: 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).

Q: What if the main method is declared as private?
A: The program compiles properly but at runtime it will give "Main method not public." message.

Q: What if the static modifier is removed from the signature of the main method?
A: Program compiles. But at runtime throws an error "NoSuchMethodError".

Q: What if I write static public void instead of public static void?
A: Program compiles and runs properly.

Q: What if I do not provide the String array as the argument to the method?
A: Program compiles but throws a runtime error "NoSuchMethodError".

Q: What is the first argument of the String array in main method?
A: The String array is empty. It does not have any element. This is unlike C/C++ where the first element by default is the program name.

Q: If I do not provide any arguments on the command line, then the String array of Main method will be empty or null?
A: It is empty. But not null.

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

A: 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.

Q: What environment variables do I need to set on my machine in order to be able to run Java programs?
A: CLASSPATH and PATH are the two variables.

Q: Can an application have multiple classes having main method?
A: 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.

Q: Can I have multiple main methods in the same class?
A: No the program fails to compile. The compiler says that the main method is already defined in the class.

Q: Do I need to import java.lang package any time? Why ?
A: No. It is by default loaded internally by the JVM.

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

Q: What are Checked and UnChecked Exception?
A: 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.

Q: What is Overriding?

A: 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.

Q: What are different types of inner classes?
A: *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.

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

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

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

A: 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.

Q: What is the default value of an object reference declared as an instance variable?
A: null unless we define it explicitly.

Q: Can a top level class be private or protected?
A: 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.

Q: What type of parameter passing does Java support?
A: In Java the arguments are always passed by value .

Q: Primitive data types are passed by reference or pass by value?
A: Primitive data types are passed by value.

Q: Objects are passed by value or by reference?
A: 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 .

Q: What is serialization?
A: Serialization is a mechanism by which you can save the state of an object by converting it to a byte stream.

Q: How do I serialize an object to a file?
A: 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.

Q: Which methods of Serializable interface should I implement?
A: The serializable interface is an empty interface, it does not contain any methods. So we do not implement any methods.

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

Q: What is the common usage of serialization?
A: 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.

Q: What is Externalizable interface?
A: 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.

Q: When you serialize an object, what happens to the object references included in the object?
A: 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 obect.

Q: What one should take care of while serializing the object?
A: 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.

Q: What happens to the static fields of a class during serialization?
A: There are three exceptions in which serialization doesnot necessarily read and write to the stream. These are
1. Serialization ignores static fields, because they are not part of ay particular state state.
2. Base class fields are only hendled if the base class itself is serializable.
3. Transient fields.

Q: Does Java provide any construct to find out the size of an object?
A: No there is not sizeof operator in Java. So there is not direct way to determine the size of an object directly in Java.

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

A: 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.

Q: What are wrapper classes?
A: 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.

Q: Why do we need wrapper classes?
A: 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 resons 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.

Q: What are checked exceptions?
A: Checked exception are those which the Java compiler forces you to catch. e.g. IOException are checked Exceptions.

Q: What are runtime exceptions?
A: 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.

Q: What is the difference between error and an exception?
A: 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.).

Q: How to create custom exceptions?
A: Your class should extend class Exception, or some more specific type thereof.

Q: If I want an object of my class to be thrown as an exception object, what should I do?
A: The class should extend from Exception class. Or you can extend your class from some more precise exception type also.

Q: 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?
A: One can not do anytihng in this scenarion. Because Java does not allow multiple inheritance and does not provide any exception interface as well.

Q: How does an exception permeate through the code?
A: 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.

Q: What are the different ways to handle exceptions?
A: 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. and
2. List the desired exceptions in the throws clause of the method and let the caller of the method hadle those exceptions.

Q: 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?
A: In the first approach as a programmer of the method, you urself 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 it's 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 creators 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.

Q: Is it necessary that each try block must be followed by a catch block?
A: 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.

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

Q: If I write System.exit (0); at the end of the try block, will the finally block still execute?
A: 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.

Q: How are Observer and Observable used?
A: Objects that subclass the Observable class maintain 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.

Q: What is synchronization and why is it important?
A: 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.

Q: How does Java handle integer overflows and underflows?
A: It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

Q: Does garbage collection guarantee that a program will not run out of memory?
A: 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
.

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

A: 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.

Q: When a thread is created and started, what is its initial state?
A: A thread is in the ready state after it has been created and started.

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

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

Q: What is the difference between a while statement and a do statement?
A: 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.

Q: What is the difference between static and non-static variables?
A: 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.

Q: How are this() and super() used with constructors?
A: This() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

Q: What are synchronized methods and synchronized statements?
A: 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.

Q: What is daemon thread and which method is used to create the daemon thread?
A: 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.

Q: Can applets communicate with each other?

A: 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.

Q: What are the steps in the JDBC connection?

A: 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();

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

A: 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.

Q: Can an unreachable object become reachable again?
A: 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.

Q: What method must be implemented by all threads?
A: All tasks must implement the run() method, whether they are a subclass of Thread or implement the Runnable interface.

Q: What are synchronized methods and synchronized statements?
A: 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.

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

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

Q: What are some alternatives to inheritance?
A: 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).

Q: What does it mean that a method or field is "static"?
A: 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.

Q: What is the difference between preemptive scheduling and time slicing?
A: 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.

Q: What is the catch or declare rule for method declarations?
A: 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.

Q: Is Empty .java file a valid source file?
A: Yes, an empty .java file is a perfectly valid source file.

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

Q: Is String a primitive data type in Java?
A: 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.

Q: Is main a keyword in Java?
A: No, main is not a keyword in Java.

Q: Is next a keyword in Java?
A: No, next is not a keyword.

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

Q: Is exit a keyword in Java?
A: No. To exit a program explicitly you use exit method in System object.

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

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

Q: What are the different scopes for Java variables?
A: 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.

Q: What is the default value of the local variables?
A: 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..

Q: How many objects are created in the following piece of code?
```
MyClass c1, c2, c3;
c1 = new MyClass ();
c3 = new MyClass ();
```
A: Only 2 objects are created, c1 and c3. The reference c2 is only declared and not initialized.

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

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

Q: What will be the output of the following statement?
System.out.println ("1" + 3);
A: It will print 13.

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

1. What is the difference between private, protected, and public?

These keywords are for allowing privileges to components such as java methods and variables.
Public: accessible to all classes
Private: accessible only to the class to which they belong
Protected: accessible to the class to which they belong and any subclasses.
Access specifiers are keywords that determines the type of access to the member of a class. These are:
* Public
* Protected
* Private
* Defaults

2. What's the difference between an interface and an abstract class? Also discuss the similarities. (Very Important)

Abstract class is a class which contain one or more abstract methods, which has to be implemented by sub classes. Interface is a Java Object containing method declaration and doesn't contain implementation. The classes which have implementing the Interfaces must provide the method definition for all the methods
Abstract class is a Class prefix with a abstract keyword followed by Class definition. Interface is a Interface which starts with interface keyword.
Abstract class contains one or more abstract methods. where as Interface contains all abstract methods and final declarations
Abstract classes are useful in a situation that Some general methods should be implemented and specialization behavior should be implemented by child classes. Interfaces are useful in a situation that all properties should be implemented.

Differences are as follows:

* Interfaces provide a form of multiple inheritance. A class can extend only one other class.
* Interfaces are limited to public methods and constants with no implementation. Abstract classes can have a partial implementation, protected parts, static methods, etc.
* A Class may implement several interfaces. But in case of abstract class, a class may extend only one abstract class.

* Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. Abstract classes are fast.

Similarities:

* Neither Abstract classes or Interface can be instantiated.

How to define an Abstract class?
A class containing abstract method is called Abstract class. An Abstract class can't be instantiated.
Example of Abstract class:

```
abstract class testAbstractClass {
   protected String myString;
   public String getMyString() {
   return myString;
}
public abstract string anyAbstractFunction();
}
```

How to define an Interface?
Answer: In Java Interface defines the methods but does not implement them. Interface can include constants. A class that implements the interfaces is bound to implement all the methods defined in Interface.
Example of Interface:

```
public interface sampleInterface {
   public void functionOne();
   public long CONSTANT_ONE = 1000;
}
```

3. Question: How you can force the garbage collection?

Garbage collection automatic process and can't be forced. You could request it by calling System.gc(). JVM does not guarantee that GC will be started immediately.

Garbage collection is one of the most important feature of Java, Garbage collection is also called automatic memory management as JVM automatically removes the unused variables/objects (value is null) from the memory. User program can't directly free the object from memory, instead it is the job of the garbage collector to automatically free the objects that are no longer referenced by a program. Every class inherits finalize() method from java.lang.Object, the finalize() method is called by garbage collector when it determines no more references to the object exists. In Java, it is good idea to explicitly assign null into a variable when no more in use. I Java on calling System.gc() and Runtime.gc(), JVM tries to recycle the unused objects, but there is no guarantee when all the objects will garbage collected.

4. What's the difference between constructors and normal methods?

Constructors must have the same name as the class and can not return a value. They are only called once while regular methods could be called many times and it can return a value or can be void.

5. Can you call one constructor from another if a class has multiple constructors

Yes. Use this() to call a constructor from an other constructor.

6. Explain the usage of Java packages.

This is a way to organize files when a project consists of multiple modules. It also helps resolve naming conflicts when different packages have classes with the same names. Packages access level also allows you to protect data from being used by the non-authorized classes.

7. Explain in your own words the "bottom line" benefits of the use of an interface.

The interface makes it possible for a method in one class to invoke methods on objects of other classes, without the requirement to know the true class of those objects, provided that those objects are all instantiated from classes that implement one or more specified interfaces. In other words, objects of classes that implement specified interfaces can be passed into methods of other objects as the generic type Object, and the methods of the other objects can invoke methods on the incoming objects by first casting them as the interface type.

8. What are some advantages and disadvantages of Java Sockets?

Some advantages of Java Sockets:
Sockets are flexible and sufficient. Efficient socket based programming can be easily implemented for general communications. Sockets cause low network traffic. Unlike HTML forms and CGI scripts that generate and transfer whole web pages for each new request, Java applets can send only necessary updated information.

Some disadvantages of Java Sockets:
Security restrictions are sometimes overbearing because a Java applet running in a Web browser is only able to establish connections to the machine where it came from, and to nowhere else on the network Despite all of the useful and helpful Java features, Socket based communications allows only to send packets of raw data between applications. Both the client-side and server-side have to provide mechanisms to make the data useful in any way.

9. Explain the usage of the keyword transient?

Transient keyword indicates that the value of this member variable does not have to be serialized with the object. When the class will be de-serialized, this variable will be initialized with a default value of its data type (i.e. zero for integers).

10. What's the difference between the methods sleep() and wait()

The code sleep(1000); puts thread aside for exactly one second. The code wait(1000), causes a wait of up to one second. A thread could stop waiting earlier if it receives the notify() or notifyAll() call. The method wait() is defined in the class Object and the method sleep() is defined in the class Thread.

11. What would you use to compare two String variables - the operator == or the method equals()?

I'd use the method equals() to compare the values of the Strings and the == to check if two variables point at the same instance of a String object.

12. Why would you use a synchronized block vs. synchronized method?

Synchronized blocks place locks for shorter periods than synchronized methods.

13. What access level do you need to specify in the class declaration to ensure that only classes from the same directory can access it?

You do not need to specify any access level, and Java will use a default package access level.

14. Can an inner class declared inside of a method access local variables of this method?

It's possible if these variables are final.

15. What can go wrong if you replace && with & in the following code:
String a=null; if (a!=null && a.length()>10) {...}

A single ampersand here would lead to a NullPointerException.

16. What's the main difference between a Vector and an ArrayList?

Java Vector class is internally synchronized and ArrayList is not synchronized.

17. Describe the wrapper classes in Java.

Wrapper class is wrapper around a primitive data type. An instance of a wrapper class contains, or wraps, a primitive value of the corresponding type.

Following table lists the primitive types and the corresponding wrapper classes:

Primitive	Wrapper
boolean	- java.lang.Boolean
byte	- java.lang.Byte
char	- java.lang.Character
double	- java.lang.Double
float	- java.lang.Float
int	- java.lang.Integer
long	- java.lang.Long
short	- java.lang.Short
void	- java.lang.Void

18. How could Java classes direct program messages to the system console, but error messages, say to a file?

The class System has a variable out that represents the standard output, and the variable err that represents the standard error device. By default, they both point at the system console. This how the standard output could be re-directed:
Stream st = new Stream(new FileOutputStream("output.txt")); System.setErr(st); System.setOut(st);

19. How do you know if an explicit object casting is needed?

If you assign a superclass object to a variable of a subclass's data type, you need to do explicit casting. For example:
Object a; Customer b; b = (Customer) a;

20. When you assign a subclass to a variable having a supeclass type, the casting is performed automatically. Can you write a Java class that could be used both as an applet as well as an application?

Yes. Add a main() method to the applet.

21. If a class is located in a package, what do you need to change in the OS environment to be

able to use it?

You need to add a directory or a jar file that contains the package directories to the CLASSPATH environment variable. Let's say a class Employee belongs to a package com.xyz.hr; and is located in the file c:\dev\com\xyz\hr\Employee.javIn this case, you'd need to add c:\dev to the variable CLASSPATH. If this class contains the method main(), you could test it from a command prompt window as follows:
c:\>java com.xyz.hr.Employee

22. What's the difference between J2SDK 1.5 and J2SDK 5.0?

There's no difference, Sun Microsystems just re-branded this version.

23. 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.

24. Name the containers which uses Border Layout as their default layout?

Containers which uses Border Layout as their default are: window, Frame and Dialog classes.

**25. You are planning to do an indexed search in a list of objects. Which of the two Java collections should you use:
ArrayList or LinkedList?**

ArrayList

26. When should the method invokeLater()be used?

This method is used to ensure that Swing components are updated through the event-dispatching thread.

27. How can a subclass call a method or a constructor defined in a superclass?

Use the following syntax: super.myMethod(); To call a constructor of the superclass, just write super(); in the first line of the subclass's constructor.

28. What do you understand by Synchronization?

Synchronization is a process of controlling the access of shared resources by the multiple threads in such a manner that only one thread can access one resource at a time. In non synchronized multithreaded application, it is possible for one thread to modify a shared object while another thread is in the process of using or updating the object's value. Synchronization prevents such type of data corruption.
E.g. Synchronizing a function:

```
public synchronized void Method1 () {
   // Appropriate method-related code.
}
```

E.g. Synchronizing a block of code inside a function:

```
public myFunction (){
   synchronized (this) {
   // Synchronized code here.
```

```
  }
}
```

29. What's the difference between a queue and a stack?

Stacks works by last-in-first-out rule (LIFO), while queues use the FIFO rule

30. You can create an abstract class that contains only abstract methods. On the other hand, you can create an interface that declares the same methods. So can you use abstract classes instead of interfaces?

Sometimes. But your class may be a descendent of another class and in this case the interface is your only option.

31. If you're overriding the method equals() of an object, which other method you might also consider?

hashCode()

32. What is Collection API?

The Collection API is a set of classes and interfaces that support operation on collections of objects. These classes and interfaces are more flexible, more powerful, and more regular than the vectors, arrays, and hashtables if effectively replaces.
Example of classes: HashSet, HashMap, ArrayList, LinkedList, TreeSet and TreeMap.
Example of interfaces: Collection, Set, List and Map.

33. How would you make a copy of an entire Java object with its state?

Have this class implement Cloneable interface and call its method clone().

34. How can you minimize the need of garbage collection and make the memory use more effective?

Use object pooling and weak object references.

35. There are two classes: A and B. The class B need to inform a class A when some important event has happened. What Java technique would you use to implement it?

If these classes are threads I'd consider notify() or notifyAll(). For regular classes you can use the Observer interface.

36. Explain the Encapsulation principle.

Encapsulation is a process of binding or wrapping the data and the codes that operates on the data into a single entity. This keeps the data safe from outside interface and misuse. One way to think about encapsulation is as a protective wrapper that prevents code and data from being arbitrarily accessed by other code defined outside the wrapper.

37. Explain the Inheritance principle.

Inheritance is the process by which one object acquires the properties of another object.

38. Explain the Polymorphism principle.

The meaning of Polymorphism is something like one name many forms. Polymorphism enables one entity to be used as as general category for different types of actions. The specific action is determined by the exact nature of the situation. The concept of polymorphism can be explained as "one interface, multiple methods".
From a practical programming viewpoint, polymorphism exists in three distinct forms in Java:

* Method overloading
* Method overriding through inheritance
* Method overriding through the Java interface

39. Is Iterator a Class or Interface? What is its use?

Iterator is an interface which is used to step through the elements of a Collection.

40. Explain the user defined Exceptions?

User defined Exceptions are the separate Exception classes defined by the user for specific purposed. An user defined can created by simply sub-classing it to the Exception class. This allows custom exceptions to be generated (using throw) and caught in the same way as normal exceptions.
Example:

```
class myCustomException extends Exception {
    / The class simply has to exist to be an exception
}
```

41. What is OOPS?

OOP is the common abbreviation for Object-Oriented Programming.
There are three main principals of oops which are called Polymorphism, Inheritance and Encapsulation.

39. Read the following program:

```
public class test {
public static void main(String [] args) {
    int x = 3;
    int y = 1;
    if (x = y)
        System.out.println("Not equal");
   else
        System.out.println("Equal");
  }
}
```

What is the result?
The output is “Equal”
B. The output in “Not Equal”
C. An error at " if (x = y)" causes compilation to fall.
D. The program executes but no output is show on console.
Answer: C
Answer: Transient variable can't be serialize. For example if a variable is declared as transient in a Serializable class and the class is written to an ObjectStream, the value of the variable can't be written

to the stream instead when the class is retrieved from the ObjectStream the value of the variable becomes null.

# Question: Name the containers which uses Border Layout as their default layout?
Answer: Containers which uses Border Layout as their default are: window, Frame and Dialog classes.

# Question: What do you understand by Synchronization?
Answer: Synchronization is a process of controlling the access of shared resources by the multiple threads in such a manner that only one thread can access one resource at a time. In non synchronized multithreaded application, it is possible for one thread to modify a shared object while another thread is in the process of using or updating the object's value. Synchronization prevents such type of data corruption.
E.g. Synchronizing a function:

```
public synchronized void Method1 () {
// Appropriate method-related code.
}
```

E.g. Synchronizing a block of code inside a function:

```
public myFunction (){
synchronized (this) {
// Synchronized code here.
}
}
```

Question: What is transient variable?
Answer: Transient variable can't be serialize. For example if a variable is declared as transient in a Serializable class and the class is written to an ObjectStream, the value of the variable can't be written to the stream instead when the class is retrieved from the ObjectStream the value of the variable becomes **null**.

Question: Name the containers which uses Border Layout as their default layout?
Answer: Containers which uses Border Layout as their default are: window, Frame and Dialog classes.

Question: What do you understand by Synchronization?
Answer: Synchronization is a process of controlling the access of shared resources by the multiple threads in such a manner that only one thread can access one resource at a time. In non synchronized multithreaded application, it is possible for one thread to modify a shared object while another thread is in the process of using or updating the object's value. Synchronization prevents such type of data corruption.
E.g. Synchronizing a function:

```
public synchronized void Method1 () {
    // Appropriate method-related code.
}
```

E.g. Synchronizing a block of code inside a function:

```
public myFunction (){
    synchronized (this) {
            // Synchronized code here.
        }
}
```

Question: What is Collection API?
Answer: The Collection API is a set of classes and interfaces that support operation on collections of objects. These classes and interfaces are more flexible, more powerful, and more regular than the vectors, arrays, and hashtables if effectively replaces.
Example of classes: `HashSet`, `HashMap`, `ArrayList`, `LinkedList`, `TreeSet` and `TreeMap`.
Example of interfaces: `Collection`, `Set`, `List` and `Map`.

Question: Is Iterator a Class or Interface? What is its use?
Answer: Iterator is an interface which is used to step through the elements of a Collection.

Question: What is similarities/difference between an Abstract class and Interface?
Answer: Differences are as follows:

- Interfaces provide a form of multiple inheritance. A class can extend only one other class.
- Interfaces are limited to public methods and constants with no implementation. Abstract classes can have a partial implementation, protected parts, static methods, etc.
- A Class may implement several interfaces. But in case of abstract class, a class may extend only one abstract class.
- Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. Abstract classes are fast.

Similarities:

- Neither Abstract classes or Interface can be instantiated.

Question: How to define an Abstract class?
Answer: A class containing abstract method is called Abstract class. An Abstract

class can't be instantiated.
Example of Abstract class:

```
abstract class testAbstractClass {
   protected String myString;
   public String getMyString() {
      return myString;
      }
   public abstract string anyAbstractFunction();
}
```

Question: How to define an Interface?
Answer: In Java Interface defines the methods but does not implement them. Interface can include constants. A class that implements the interfaces is bound to implement all the methods defined in Interface.
Emaple of Interface:

```
public interface sampleInterface {
   public void functionOne();

   public long CONSTANT_ONE = 1000;
}
```

Question: Explain the user defined Exceptions?
Answer: User defined Exceptions are the separate Exception classes defined by the user for specific purposed. An user defined can created by simply sub-classing it to the Exception class. This allows custom exceptions to be generated (using throw) and caught in the same way as normal exceptions.
Example:

```
class myCustomException extends Exception {
    // The class simply has to exist to be an exception
}
```

Question: Explain the new Features of JDBC 2.0 Core API?
Answer: The JDBC 2.0 API includes the complete JDBC API, which includes both core and Optional Package API, and provides inductrial-strength database computing capabilities.
New Features in JDBC 2.0 Core API:

- Scrollable result sets- using new methods in the ResultSet interface allows programmatically move the to particular row or to a position relative to its current position
- JDBC 2.0 Core API provides the Batch Updates functionality to the java applications.
- Java applications can now use the ResultSet.updateXXX methods.
- New data types - interfaces mapping the SQL3 data types
- Custom mapping of user-defined types (UTDs)
- Miscellaneous features, including performance hints, the use of character streams, full precision for java.math.BigDecimal values, additional security, and support for time zones in date, time, and timestamp values.

Question: Explain garbage collection?
Answer: Garbage collection is one of the most important feature of Java. Garbage collection is also called automatic memory management as JVM automatically removes the unused variables/objects (value is null) from the memory. User program cann't directly free the object from memory, instead it is the job of the garbage collector to automatically free the objects that are no longer referenced by a program. Every class inherits **finalize()** method from **java.lang.Object**, the finalize() method is called by garbage collector when it determines no more references to the object exists. In Java, it is good idea to explicitly assign **null** into a variable when no more in use. I Java on
calling **System.gc()** and **Runtime.gc(),** JVM tries to recycle the unused objects, but there is no guarantee when all the objects will garbage collected.

Question: How you can force the garbage collection?
Answer: Garbage collection automatic process and can't be forced.

Question: What is OOPS?
Answer: OOP is the common abbreviation for Object-Oriented Programming.

Question: Describe the principles of OOPS.
Answer: There are three main principals of oops which are called Polymorphism, Inheritance and Encapsulation.

Question: Explain the Encapsulation principle.
Answer: Encapsulation is a process of binding or wrapping the data and the codes

that operates on the data into a single entity. This keeps the data safe from outside interface and misuse. One way to think about encapsulation is as a protective wrapper that prevents code and data from being arbitrarily accessed by other code defined outside the wrapper.

Question: Explain the Inheritance principle.
Answer: Inheritance is the process by which one object acquires the properties of another object.

Question: Explain the Polymorphism principle.
Answer: The meaning of Polymorphism is something like one name many forms. Polymorphism enables one entity to be used as as general category for different types of actions. The specific action is determined by the exact nature of the situation. The concept of polymorphism can be explained as "one interface, multiple methods".

Question: Explain the different forms of Polymorphism.
Answer: From a practical programming viewpoint, polymorphism exists in three distinct forms in Java:

- Method overloading
- Method overriding through inheritance
- Method overriding through the Java interface

Question: What are Access Specifiers available in Java?
Answer: Access specifiers are keywords that determines the type of access to the member of a class. These are:

- Public
- Protected
- Private
- Defaults

Question: Describe the wrapper classes in Java.
Answer: Wrapper class is wrapper around a primitive data type. An instance of a wrapper class contains, or wraps, a primitive value of the corresponding type.

Following table lists the primitive types and the corresponding wrapper classes:

Primitive	Wrapper
boolean	java.lang.Boolean
byte	java.lang.Byte
char	java.lang.Character
double	java.lang.Double
float	java.lang.Float
int	java.lang.Integer
long	java.lang.Long
short	java.lang.Short
void	java.lang.Void

Question: Read the following program:

```
public class test {
public static void main(String [] args) {
    int x = 3;
    int y = 1;
   if (x = y)
     System.out.println("Not equal");
  else
    System.out.println("Equal");
 }
}
```

What is the result?
A. The output is “Equal”
B. The output in “Not Equal”
C. An error at " if (x = y)" causes compilation to fall.
D. The program executes but no output is show on console.
Answer: C

Question: what is the class variables ?
Answer: When we create a number of objects of the same class, then each object will share a common copy of variables. That means that there is only one copy per

class, no matter how many objects are created from it. Class variables or static variables are declared with the static keyword in a class, but mind it that it should be declared outside outside a class. These variables are stored in static memory. Class variables are mostly used for constants, variable that never change its initial value. Static variables are always called by the class name. This variable is created when the program starts i.e. it is created before the instance is created of class by using new operator and gets destroyed when the programs stops. The scope of the class variable is same a instance variable. The class variable can be defined anywhere at class level with the keyword static. It initial value is same as instance variable. When the class variable is defined as int then it's initial value is by default zero, when declared boolean its default value is false and null for object references. Class variables are associated with the class, rather than with any object.

Question: What is the difference between the instanceof and getclass, these two are same or not ?
Answer: instanceof is a operator, not a function while getClass is a method of java.lang.Object class. Consider a condition where we use
if(o.getClass().getName().equals("java.lang.Math")){ }
This method only checks if the classname we have passed is equal to java.lang.Math. The class java.lang.Math is loaded by the bootstrap ClassLoader. This class is an abstract class.This class loader is responsible for loading classes. Every Class object contains a reference to the ClassLoader that defines. getClass() method returns the runtime class of an object. It fetches the java instance of the given fully qualified type name. The code we have written is not necessary, because we should not compare getClass.getName(). The reason behind it is that if the two different class loaders load the same class but for the JVM, it will consider both classes as different classes so, we can't compare their names. It can only gives the implementing class but can't compare a interface, but instanceof operator can.
The instanceof operator compares an object to a specified type. We can use it to test if an object is an instance of a class, an instance of a subclass, or an instance of a class that implements a particular interface. We should try to use instanceof operator in place of getClass() method. Remember instanceof opeator and getClass are not same. Try this example, it will help you to better understand the difference between the two.

```
Interface one{
}

Class Two implements one {
}
Class Three implements one {
}
```

```
public class Test {
public static void main(String args[]) {
one test1 = new Two();
one test2 = new Three();
System.out.println(test1 instanceof one); //true
System.out.println(test2 instanceof one); //true
System.out.println(Test.getClass().equals(test2.getClass())); //false
}
}
```

Q: What is Jakarta Struts Framework?
A: Jakarta Struts is open source implementation of MVC (Model-View-Controller) pattern for the development of web based applications. Jakarta Struts is robust architecture and can be used for the development of application of any size. Struts framework makes it much easier to design scalable, reliable Web applications with Java.

Q: What is ActionServlet?
A: The class *org.apache.struts.action.ActionServlet* is the called the ActionServlet. In the the Jakarta Struts Framework this class plays the role of controller. All the requests to the servergoes through the controller. Controller is responsible for handling all the requests.

Q: How you will make available any Message Resources Definitions file to the Struts Framework Environment?
A: Message Resources Definitions file are simple .properties files and these files contains the messages that can be used in the struts project. Message Resources Definitions files can be added to the struts-config.xml file through **<message-resources />** tag.
Example:
<message-resources parameter="MessageResources" />

Q: What is Action Class?
A: The Action is part of the controller. The purpose of Action Class is to translate the HttpServletRequest to the business logic. To use the Action, we need to Subclass and overwrite the execute() method. The ActionServlet (commad) passes

the parameterized class to Action Form using the execute() method. There should be no database interactions in the action. The action should receive the request, call business objects (which then handle database, or interface with J2EE, etc) and then determine where to go next. Even better, the business objects could be handed to the action at runtime (IoC style) thus removing any dependencies on the model. The return type of the execute method is ActionForward which is used by the Struts Framework to forward the request to the file as per the value of the returned ActionForward object.

Q: Write code of any Action Class?
A: Here is the code of Action Class that returns the *ActionForward* object.
TestAction.java

```
package roseindia.net;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class TestAction extends Action
{
  public ActionForward execute(
    ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response) throws Exception{
      return mapping.findForward("testAction");
  }
}
```

Q: What is ActionForm?
A: An ActionForm is a JavaBean that extends `org.apache.struts.action.ActionForm`. ActionForm maintains the session state for web application and the ActionForm object is automatically populated on the server side with data entered from a form on the client side.

Q: What is Struts Validator Framework?
A: Struts Framework provides the functionality to validate the form data. It can be use to validate the data on the users browser as well as on the server side. Struts Framework emits the java scripts and it can be used validate the form data on the

client browser. Server side validation of form can be accomplished by sub classing your From Bean with **DynaValidatorForm** class.

The Validator framework was developed by David Winterfeldt as third-party add-on to Struts. Now the Validator framework is a part of Jakarta Commons project and it can be used with or without Struts. The Validator framework comes integrated with the Struts Framework and can be used without doing any extra settings.

Q. Give the Details of XML files used in Validator Framework?
A: The Validator Framework uses two XML configuration files **validator-rules.xml** and **validation.xml**. The **validator-rules.xml** defines the standard validation routines, these are reusable and used in **validation.xml**. to define the form specific validations. The **validation.xml** defines the validations applied to a form bean.

Q. How you will display validation fail errors on jsp page?
A: Following tag displays all the errors:
<html:errors/>

Q. How you will enable front-end validation based on the xml in validation.xml?
A: The <html:javascript> tag to allow front-end validation based on the xml in validation.xml. For example the code: <html:javascript formName="logonForm" dynamicJavascript="true" staticJavascript="true" /> generates the client side java script for the form "logonForm" as defined in the validation.xml file. The <html:javascript> when added in the jsp file generates the client site validation script.

Question: What is RequestProcessor and RequestDispatcher?
Answer: The controller is responsible for intercepting and translating user input into actions to be performed by the model. The controller is responsible for selecting the next view based on user input and the outcome of model operations. The Controller receives the request from the browser, invoke a business operation and coordinating the view to return to the client.

The controller is implemented by a java servlet, this servlet is centralized point of control for the web application. In struts framework the controller responsibilities are implemented by several different components like

The ActionServlet Class
The RequestProcessor Class
The Action Class

The ActionServlet extends the **javax.servlet.http.httpServlet** class. The ActionServlet class is not abstract and therefore can be used as a concrete controller by your application.
The controller is implemented by the ActionServlet class. All incoming requests are mapped to the central controller in the deployment descriptor as follows.

```
<servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
</servlet>
```

All request URIs with the pattern *.do are mapped to this servlet in the deployment descriptor as follows.

```
<servlet-mapping>
<servlet-name>action</servlet-name>
     <url-pattern>*.do</url-pattern>
     <url-pattern>*.do</url-pattern>
```

A request URI that matches this pattern will have the following form.
http://www.my_site_name.com/mycontext/actionName.do

The preceding mapping is called extension mapping, however, you can also specify path mapping where a pattern ends with /* as shown below.

```
<servlet-mapping>
   <servlet-name>action</servlet-name>
   <url-pattern>/do/*</url-pattern>
<url-pattern>*.do</url-pattern>
```

A request URI that matches this pattern will have the following form.
http://www.my_site_name.com/mycontext/do/action_Name
The class **org.apache.struts.action.requestProcessor** process the request from the controller. You can sublass the RequestProcessor with your own version and modify how the request is processed.

Once the controller receives a client request, it delegates the handling of the request to a helper class. This helper knows how to execute the business operation associated with the requested action. In the Struts framework this helper class is descended of org.apache.struts.action.Action class. It acts as a bridge between a client-side user action and business operation. The Action class decouples the client request from the business model. This decoupling allows for more than one-to-one mapping between the user request and an action. The Action class also can perform other functions such as authorization, logging before invoking business operation. the Struts Action class contains several methods, but most important method is the execute() method.

```
public ActionForward execute(ActionMapping mapping,
          ActionForm form, HttpServletRequest request, HttpServletResponse
response)    throws Exception;
```

The execute() method is called by the controller when a request is received from a client. The controller creates an instance of the Action class if one doesn't already exist. The strut framework will create only a single instance of each Action class in your application.

Action are mapped in the struts configuration file and this configuration is loaded into memory at startup and made available to the framework at runtime. Each Action element is represented in memory by an instance of the org.apache.struts.action.ActionMapping class . The ActionMapping object contains a path attribute that is matched against a portion of the URI of the incoming request.

```
<action>
      path= "/somerequest"
      type="com.somepackage.someAction"
      scope="request"
      name="someForm"
      validate="true"
      input="somejsp.jsp"
  <forward name="Success" path="/action/xys" redirect="true"/>
  <forward name="Failure" path="/somejsp.jsp" redirect="true"/>
</action>
```

Once this is done the controller should determine which view to return to the client. The execute method signature in Action class has a return type org.apache.struts.action.ActionForward class. The ActionForward class represents a destination to which the controller may send control once an action has completed. Instead of specifying an actual JSP page in the code, you can declaratively associate as action forward through out the application. The action forward are specified in the configuration file.

```
<action>
      path= "/somerequest"
      type="com.somepackage.someAction"
      scope="request"
      name="someForm"
      validate="true"
      input="somejsp.jsp"
   <forward name="Success" path="/action/xys" redirect="true"/>
   <forward name="Failure" path="/somejsp.jsp" redirect="true"/>
</action>
```

The action forward mappings also can be specified in a global section, independent of any specific action mapping.

```
<global-forwards>
   <forward name="Success" path="/action/somejsp.jsp" />
   <forward name="Failure" path="/someotherjsp.jsp" />
</global-forwards>
```

public interface RequestDispatcher

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or given by a particular name. This interface is intended to wrap servlets, but a servlet container can create RequestDispatcher objects to wrap any type of resource.

getRequestDispatcher

public RequestDispatcher getRequestDispatcher(java.lang.String path)

Returns a RequestDispatcher object that acts as a wrapper for the resource located at the given path. A RequestDispatcher object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.
The pathname must begin with a "/" and is interpreted as relative to the current context root. Use getContext to obtain a RequestDispatcher for resources in

foreign contexts. This method returns null if the ServletContext cannot return a RequestDispatcher.

Parameters:
  path - a String specifying the pathname to the resource
Returns:
  a RequestDispatcher object that acts as a wrapper for the resource at the specified path
See Also:
  RequestDispatcher, getContext(java.lang.String)

getNamedDispatcher

public RequestDispatcher getNamedDispatcher(java.lang.String name)

Returns a RequestDispatcher object that acts as a wrapper for the named servlet. Servlets (and JSP pages also) may be given names via server administration or via a web application deployment descriptor. A servlet instance can determine its name using ServletConfig.getServletName().
This method returns null if the ServletContext cannot return a RequestDispatcher for any reason.

Parameters:
  name - a String specifying the name of a servlet to wrap
Returns:
  a RequestDispatcher object that acts as a wrapper for the named servlet
See Also:
  RequestDispatcher, getContext(java.lang.String), ServletConfig.getServletName()

Question: Why cant we overide create method in StatelessSessionBean?
Answer: From the EJB Spec : - A Session bean's home interface defines one or morecreate(...) methods. Each create method must be named create and must match one of the ejbCreate methods defined in the enterprise Bean class. The return type of a create method must be the enterprise Bean's remote interface type. The home interface of a stateless session bean must have one create method that takes no arguments.

Question: Is struts threadsafe?Give an example?
Answer: Struts is not only thread-safe but thread-dependant. The response to a request is handled by a light-weight Action object, rather than an individual servlet. Struts instantiates each Action class once, and allows other requests to be threaded through the original object. This core strategy conserves resources and provides the best possible throughput. A properly-designed application will exploit this further by routing related operations through a single Action.

Question: Can we Serialize static variable?
Answer: Serialization is the process of converting a set of object instances that contain references to each other into a linear stream of bytes, which can then be sent through a socket, stored to a file, or simply manipulated as a stream of data. Serialization is the mechanism used by RMI to pass objects between JVMs, either as arguments in a method invocation from a client to a server or as return values from a method invocation. In the first section of this book, There are three exceptions in which serialization doesnot necessarily read and write to the stream. These are
1. Serialization ignores static fields, because they are not part of any particular object's state.
2. Base class fields are only handled if the base class itself is serializable.
3. Transient fields. There are four basic things you must do when you are making a class serializable. They are:

1. Implement the Serializable interface.
2. Make sure that instance-level, locally defined state is serialized properly.
3. Make sure that superclass state is serialized properly.
4. Override equals( )and hashCode( ).
   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 ....
   (Source: http://www.oreilly.com/catalog/javarmi/chapter/ch10.html)

Question: What are the uses of tiles-def.xml file, resourcebundle.properties file, validation.xml file?
Answer: tiles-def.xml is is an xml file used to configure tiles with the struts application. You can define the layout / header / footer / body content for your

View. See more at http://www.roseindia.net/struts/using-tiles-defs-xml.shtml.

The **resourcebundle.properties** file is used to configure the message (error/ other messages) for the struts applications.

The file validation.xml is used to declare sets of validations that should be applied to Form Beans. Fpr more information please visithttp://www.roseindia.net/struts/address_struts_validator.shtml.

Question: What is the difference between perform() and execute() methods?
Answer: Perform method is the method which was deprecated in the Struts Version 1.1. In Struts 1.x, Action.perform() is the method called by the ActionServlet. This is typically where your business logic resides, or at least the flow control to your JavaBeans and EJBs that handle your business logic. As we already mentioned, to support declarative exception handling, the method signature changed in perform. Now execute just throws Exception. Action.perform() is now deprecated; however, the Struts v1.1 ActionServlet is smart enough to know whether or not it should call perform or execute in the Action, depending on which one is available.

Question: What are the various Struts tag libraries?
Answer: Struts is very rich framework and it provides very good and user friendly way to develop web application forms. Struts provide many tag libraries to ease the development of web applications. These tag libraries are:
* Bean tag library - Tags for accessing JavaBeans and their properties.
* HTML tag library - Tags to output standard HTML, including forms, text boxes, checkboxes, radio buttons etc..
* Logic tag library - Tags for generating conditional output, iteration capabilities and flow management
* Tiles or Template tag library - For the application using tiles
* Nested tag library - For using the nested beans in the application

Question: What do you understand by DispatchAction?
Answer: DispatchAction is an action that comes with Struts 1.1 or later, that lets you combine Struts actions into one class, each with their own method. The org.apache.struts.action.DispatchAction class allows multiple operation to mapped to the different functions in the same Action class.

For example:
A package might include separate RegCreate, RegSave, and RegDelete Actions, which just perform different operations on the same RegBean object. Since all of these operations are usually handled by the same JSP page, it would be handy to also have them handled by the same Struts Action.

A very simple way to do this is to have the submit button modify a field in the form which indicates which operation to perform.

```
<html:hidden property="dispatch" value="error"/>
<SCRIPT>function set(target)
{document.forms[0].dispatch.value=target;}</SCRIPT>
<html:submit onclick="set('save');">SAVE</html:submit>
<html:submit onclick="set('create');">SAVE AS NEW</html:submitl>
<html:submit onclick="set('delete);">DELETE</html:submit>
```

Then, in the Action you can setup different methods to handle the different operations, and branch to one or the other depending on which value is passed in the dispatch field.

```
String dispatch = myForm.getDispatch();
if ("create".equals(dispatch)) { ...
if ("save".equals(dispatch)) { ...
```

The Struts Dispatch Action [org.apache.struts.actions] is designed to do exactly the same thing, but without messy branching logic. The base perform method will check a dispatch field for you, and invoke the indicated method. The only catch is that the dispatch methods must use the same signature as perform. This is a very modest requirement, since in practice you usually end up doing that anyway.

To convert an Action that was switching on a dispatch field to a DispatchAction, you simply need to create methods like this

```
public ActionForward create(
    ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response)
  throws IOException, ServletException { ...

public ActionForward save(
    ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response)
  throws IOException, ServletException { ...
```

Cool. But do you have to use a property named dispatch? No, you don't. The only other step is to specify the name of of the dispatch property as the "parameter" property of the action-mapping. So a mapping for our example might look like this:

```
<action
  path="/reg/dispatch"
  type="app.reg.RegDispatch"
  name="regForm"
  scope="request"
  validate="true"
  parameter="dispatch"/>
```

If you wanted to use the property "o" instead, as in o=create, you would change the mapping to

```
<action
  path="/reg/dispatch"
  type="app.reg.RegDispatch"
  name="regForm"
  scope="request"
  validate="true"
  parameter="o"/>
```

Again, very cool. But why use a JavaScript button in the first place? Why not use several buttons named "dispatch" and use a different value for each?

You can, but the value of the button is also its label. This means if the page designers want to label the button something different, they have to coordinate the Action programmer. Localization becomes virtually impossible.
(Source: http://husted.com/struts/tips/002.html).

Question: How Struts relates to J2EE?
Answer: Struts framework is built on J2EE technologies (JSP, Servlet, Taglibs), but it is itself not part of the J2EE standard.

Question: What is Struts actions and action mappings?
Answer: A Struts action is an instance of a subclass of an Action class, which implements a portion of a Web application and whose perform or execute method returns a forward.

An action can perform tasks such as validating a user name and password.

An action mapping is a configuration file entry that, in general, associates an action name with an action. An action mapping can contain a reference to a form bean that the action can use, and can additionally define a list of local forwards that is visible only to this action.

An action servlet is a servlet that is started by the servlet container of a Web server to process a request that invokes an action. The servlet receives a forward from the action and asks the servlet container to pass the request to the forward's URL. An action servlet must be an instance of an org.apache.struts.action.ActionServlet class or of a subclass of that class. An action servlet is the primary component of the controller.

Question: Can I setup Apache Struts to use multiple configuration files?
Answer: Yes Struts can use multiple configuration files. Here is the configuration example:

```
<servlet>
     <servlet-name>banking</servlet-name>
     <servlet-class>org.apache.struts.action.ActionServlet
     </servlet-class>
     <init-param>
     <param-name>config</param-name>
     <param-value>/WEB-INF/struts-config.xml,
          /WEB-INF/struts-authentication.xml,
          /WEB-INF/struts-help.xml
       </param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
```

Question: What are the disadvantages of Struts?
Answer: Struts is very robust framework and is being used extensively in the industry. But there are some disadvantages of the Struts:
a) High Learning Curve
Struts requires lot of efforts to learn and master it. For any small project less experience developers could spend more time on learning the Struts.

b) Harder to learn
Struts are harder to learn, benchmark and optimize.

Question: What is Struts Flow?
Answer: Struts Flow is a port of Cocoon's Control Flow to Struts to allow complex workflow, like multi-form wizards, to be easily implemented using continuations-capable JavaScript. It provides the ability to describe the order of Web pages that have to be sent to the client, at any given point in time in an application. The code is based on a proof-of-concept Dave Johnson put together to show how the Control Flow could be extracted from Cocoon. (Ref: http://struts.sourceforge.net/struts-flow/index.html)

Question: What are the difference between <bean:message> and <bean:write>?
Answer: <bean:message>: This tag is used to output locale-specific text (from the

properties files) from a MessageResources bundle.

<bean:write>: This tag is used to output property values from a bean. <bean:write> is a commonly used tag which enables the programmers to easily present the data.

Question: What is LookupDispatchAction?
Answer: An abstract Action that dispatches to the subclass mapped execute method. This is useful in cases where an HTML form has multiple submit buttons with the same name. The button name is specified by the parameter property of the corresponding ActionMapping.
(Ref.http://struts.apache.org/1.2.7/api/org/apache/struts/actions/LookupDispatchAction.html).

Question: What are the components of Struts?
Answer: Struts is based on the MVC design pattern. Struts components can be categories into **Model**, **View** and **Controller**.
Model: Components like business logic / business processes and data are the part of Model.
View: JSP, HTML etc. are part of View
Controller: Action Servlet of Struts is part of Controller components which works as front controller to handle all the requests.

Question: What are Tag Libraries provided with Struts?
Answer: Struts provides a number of tag libraries that helps to create view components easily. These tag libraries are:
a) Bean Tags: Bean Tags are used to access the beans and their properties.
b) HTML Tags: HTML Tags provides tags for creating the view components like forms, buttons, etc..
c) Logic Tags: Logic Tags provides presentation logics that eliminate the need for scriptlets.
d) Nested Tags: Nested Tags helps to work with the nested context.

Question: What are the core classes of the Struts Framework?
Answer: Core classes of Struts Framework are ActionForm, Action, ActionMapping, ActionForward, ActionServlet etc.

Question: What are difference between ActionErrors and ActionMessage?
Answer: ActionMessage: A class that encapsulates messages. Messages can be either global or they are specific to a particular bean property.
Each individual message is described by an ActionMessage object, which contains a message key (to be looked up in an appropriate message resources database), and up to four placeholder arguments used for parametric substitution in the resulting message.

ActionErrors: A class that encapsulates the error messages being reported by the validate() method of an ActionForm. Validation errors are either global to the entire ActionForm bean they are associated with, or they are specific to a particular bean property (and, therefore, a particular input field on the corresponding form).

Question: How you will handle exceptions in Struts?
Answer: In Struts you can handle the exceptions in two ways:
a) Declarative Exception Handling: You can either define global exception handling tags in your struts-config.xml or define the exception handling tags within <action>..</action> tag.
Example:

```
<exception

      key="database.error.duplicate"

      path="/UserExists.jsp"

      type="mybank.account.DuplicateUserException"/>
```

b) Programmatic Exception Handling: Here you can use try{}catch{} block to handle the exception.

Question: What do you understand by JSP Actions?
Answer: JSP actions are XML tags that direct the server to use existing components or control the behavior of the JSP engine. JSP Actions consist of a typical (XML-based) prefix of "jsp" followed by a colon, followed by the action name followed by one or more attribute parameters.

There are six JSP Actions:

```
<jsp:include/>
<jsp:forward/>
<jsp:plugin/>
<jsp:usebean/>
<jsp:setProperty/>
<jsp:getProperty/>
```

Question: What is the difference between `<jsp:include page = ... >` and `<%@ include file = ... >`?.
Answer: Both the tag includes the information from one page in another. The differences are as follows:
`<jsp:include page = ... >`: This is like a function call from one jsp to another jsp. It is executed (the included page is executed and the

generated html content is included in the content of calling jsp) each time the client page is accessed by the client. This approach is useful to for modularizing the web application. If the included file changed then the new content will be included in the output.

<%@ include file = ... >: In this case the content of the included file is textually embedded in the page that have <%@ include file=".."> directive. In this case in the included file changes, the changed content will not included in the output. This approach is used when the code from one jsp file required to include in multiple jsp files.

Question: What is the difference between `<jsp:forward page = ... >` and response.sendRedirect(url),?.
Answer: The <jsp:forward> element forwards the request object containing the client request information from one JSP file to another file. The target file can be an HTML file, another JSP file, or a servlet, as long as it is in the same application context as the forwarding JSP file.
sendRedirect sends HTTP temporary redirect response to the browser, and browser creates a new request to go the redirected page. The response.sendRedirect kills the session variables.

Question: Identify the advantages of JSP over Servlet.

a) Embedding of Java code in HTML pages
b) Platform independence
c) Creation of database-driven Web applications
d) Server-side programming capabilities

Answer :- Embedding of Java code in HTML pages

Write the following code for a JSP page:
```
<%@ page language = "java" %>

<HTML>
<HEAD><TITLE>RESULT PAGE</TITLE></HEAD>
<BODY>
<%

PrintWriter print = request.getWriter();
print.println("Welcome");

%>
```

</BODY>
</HTML>
Suppose you access this JSP file, Find out your answer.
a) A blank page will be displayed.
b) A page with the text Welcome is displayed
c) An exception will be thrown because the implicit out object is not used
d) An exception will be thrown because PrintWriter can be used in servlets only

Answer :- A page with the text Welcome is displayed

Question: What are implicit Objects available to the JSP Page?
Answer: Implicit objects are the objects available to the JSP page. These objects are created by Web container and contain information related to a particular request, page, or application. The JSP implicit objects are:

Variable	Class	Description
application	javax.servlet.ServletContext	The context for the JSP page's servlet and any Web components contained in the same application.
config	javax.servlet.ServletConfig	Initialization information for the JSP page's servlet.
exception	java.lang.Throwable	Accessible only from an error page.
out	javax.servlet.jsp.JspWriter	The output stream.
page	java.lang.Object	The instance of the JSP page's servlet processing the current request. Not typically used by JSP page authors.
pageContext	javax.servlet.jsp.PageContext	The context for the JSP page. Provides a single API to manage the various scoped attributes.
request	Subtype of javax.servlet.ServletRequest	The request triggering the execution of the JSP page.
response	Subtype of javax.servlet.ServletResponse	The response to be returned to the client. Not typically used by JSP page authors.
session	javax.servlet.http.HttpSession	The session object for the client.

Question: What are all the different scope values for the <jsp:useBean> tag?
Answer:<jsp:useBean> tag is used to use any java object in the jsp page. Here are the scope values for <jsp:useBean> tag:
a) page
b) request
c) session and
d) application

Question: What is JSP Output Comments?
Answer: JSP Output Comments are the comments that can be viewed in the HTML source file.
Example:
<!-- This file displays the user login screen -->
and
<!-- This page was loaded on
<%= (new java.util.Date()).toLocaleString() %> -->

Question: What is expression in JSP?
Answer: Expression tag is used to insert Java values directly into the output. Syntax for the Expression tag is:
<%= expression %>
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. The following expression tag displays time on the output:
<%=new java.util.Date()%>

Question: What types of comments are available in the JSP?
Answer: There are two types of comments are allowed in the JSP. These are *hidden* and *output* comments. A hidden comments does not appear in the generated output in the html, while output comments appear in the generated output.
Example of hidden comment:
<%-- This is hidden comment --%>
Example of output comment:
<!-- This is output comment -->

Question: What is JSP declaration?
Answer: JSP Declaratives are the JSP tag used to declare variables. Declaratives are enclosed in the <%! %> tag and ends in semi-colon. You declare variables and functions in the declaration tag and can use anywhere in the JSP. Here is the example of declaratives:

```
<%@page contentType="text/html" %>

<html>

<body>

<%!
int cnt=0;
private int getCount(){
//increment cnt and return the value
cnt++;
return cnt;
}
%>
<p>Values of Cnt are:</p>
<p><%=getCount()%></p>
</body>
</html>
```

Question: What is JSP Scriptlet?
Answer: JSP Scriptlet is jsp tag which is used to enclose java code in the JSP pages. Scriptlets begins with **<%** tag and ends with **%>** tag. Java code written inside scriptlet executes every time the JSP is invoked.
Example:

```
 <%
 //java codes
  String userName=null;
  userName=request.getParameter("userName");
  %>
```

Question: What are the life-cycle methods of JSP?
Answer: Life-cycle methods of the JSP are:
a) **jspInit()**: The container calls the jspInit() to initialize the servlet instance. It is called before any other method, and is called only once for a servlet instance.
b)**_jspService():** The container calls the _jspservice() for each request and it passes the request and the response objects. _jspService() method cann't be overridden.
c) **jspDestroy():** The container calls this when its instance is about to destroyed.
The jspInit() and jspDestroy() methods can be overridden within a JSP page.