1. Difference between comparator and comparable interface:

Parameter	Comparable	Comparator
Sorting logic	Sorting logic must be in same class whose objects are being sorted. Hence this is called natural ordering of objects (provides single sorting sequence)	Sorting logic need not be in same class. Hence we can write different sorting based on different attributes of objects to be sorted. (provides multiple sorting sequence)E.g. Sorting using id,name etc.
Implement ation	Class whose objects to be sorted must implement this interface	Class whose objects to be sorted do not need to implement this interface. Some other class can implement this interface.
Sorting method	int compareTo(Object o1) This method compares this object with o1 object and returns a integer.	int compare(Object o1,Object o2) This method compares o1 and o2 objects. and returns a integer.
Calling method	Collections.sort(List)	Collections.sort(List, Comparator)
Package	Java.lang.Comparable	Java.util.Comparator

Example:

Comparable

```
public class Country implements Comparable {
    @Override
    public int compareTo(Object arg0) {
        Country country = (Country) arg0;
        return (this.countryId < country.countryId) ? -1
                : (this.countryId > country.countryId) ? 1 : 0;
    }
}
```

Comparator

```
public class CountrySortByIdComparator implements Comparator<Country> {
    @Override
    public int compare(Country country1, Country country2) {
        return (country1.getCountryId() < country2.getCountryId()) ? -1
                : (country1.getCountryId() > country2.getCountryId()) ? 1
                : 0;
    }
}
```

2. Can we override a static method? If we do what will happen?

- No,Static methods can't be overridden as it is part of a class rather than an object.
- If a subclass defines a static method with the same signature as a static method in the superclass, the method in the subclass hides the one in the superclass.
- The version of the hidden static method that gets invoked depends on whether it is invoked from the superclass or the subclass.

3. Example of Java's final classes:

- java.lang.String
- java.lang.StringBuffer
- java.lang.Math
- java.lang.System
- All Wrapper class(Integer, Double, etc)

4. What is the use of final ArrayList?

- This means that you cannot rebind the variable to point to a different collection instance.
- But still can add to ArrayList new elements, remove elements and update it.

5. How to make a collection unmodifiable?

Collections.unmodifiableCollection(list) , Collections.unmodifiableList(list), Collections.unmodifiableSet(list), etc

can be used to make a collection unmodified.

And an attempt to modify the collection will result in an UnsupportedOperationException.

Eg: List<String> list = Collections.unmodifiableList(new ArrayList<String>());

6.Difference between unmodifiable collection and immutable collection.

An unmodifiable collection is often a wrapper around a modifiable collection which other code may still have access to. So while you can't make any changes to it if you only have a reference to the unmodifiable collection, you can't rely on the contents not changing.

An immutable collection guarantees that nothing can change the collection any more. If it wraps a modifiable collection, it makes sure that no other code has access to that modifiable collection. Note that although no code can change which objects the collection contains references to, the objects themselves may still be mutable

7. Why do we use finally block? How can we stop the execution of finally block?

Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc.

The finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).

8. When to use abstract class and when interface?

- Abstract classes are useful in a situation when some general methods should be inherited and specialization behavior should be implemented by subclasses
- Interfaces are useful in a situation when all its properties need to be implemented by subclasses (i.e.)

useful in cases when only the behavior of the method is specified buit not concerned about who implements its behavior.

9. Why are layered architectures so useful?

- allows to distinguish and distribute the responsibilities that your application, your code has to deliver value to the end user.

That is, the user interface does not contain components or elements that handle business logic. Business logic resides in a separate layer. A data access layer does not have anything to do with presenting the data, it deals with the database.

- Easier to understand.
- Easier to write.
- Easier to test.
- Easier to extend.

10. toString() overrides from which class?

The toString() method is overridden from the Object class.

If any code needs some information of an object of a class, then it can get it by using this method.

The toString() method of an object gets invoked automatically, when an object reference is passed in the System.out.println() method.

11. Difference between HTTP and HTTPS:

HTTP	HTTPS
is unsecured	is secured.
uses port 80 for communication	uses port 443 for communication
operates at Application Layer	operates at Transport Layer
No encryption is there	uses encryption

12. How HTTPS works?

For HTTPS connection, a server must have a public key certificate, which embeds key information with a verification of the key owner's identity.

When using an https connection,

- the server responds to the initial connection by offering a list of encryption methods it supports.
- In response, the client selects a connection method, and the client & server exchange certificates to authenticate their identities.
- After this is done, both parties exchange the encrypted information after ensuring that both are using the same key (ie) Server and Client now share a secret which they can use to encrypt messages going back and forth.

13. How are Java objects stored in memory?

There are two kinds of memory used in Java.

Stack memory: stores primitive types and the addresses of objects.

Heap memory:The object values are stored in heap memory.

An object reference on the stack is only an address that refers to the place in heap memory where that object is kept.

14. What's the difference between equals() and ==?

In Java, when the "==" operator is used to compare 2 objects, it checks to see if the objects refer to the same place in memory.

The "==" operator compares the objects' location(s) in memory.

The equals method is defined in the Object class, which is actually meant to compare the contents of 2 objects, and not their location in memory.

by default equals() will behave the same as the "==" operator and compare object locations. But, when overriding the equals() method, you should compare the values of the object instead.

15. how equals() method works internally?

The Object which all other Java objects extend has an equals method which allows you to check to see if two objects are equal, which means you call equals on any object you want.

That is, for any reference values x and y, this method returns true if and only if x and y refer to the same object (x==y has the value true).

When you compare two instances using ==, you are actually comparing their memory addresses to see if they are references to the same object.

16. Can we define static variables inside a static method? If yes, whether that is a global variable?

You can not declare variable as static inside a method.
Inside method all variables are local variables that has no existence outside this method that's why they can't be static.

17.Hashing function rule:

Hash function should return the same hash code each and every time, when function is applied on same or equal objects.

In other words, two equal objects must produce same hash code consistently.

18. Default implementation of hashCode() function:

A hashcode is a number generated from any object. This is what allows objects to be stored/retrieved quickly in a Hashtable.

All objects in java inherit a default implementation of hashCode() function defined in Object class.

This function produce hash code by typically converting the internal address of the object into an integer, thus producing different hash codes for all different objects.

19. How HasMap will work internally?

in HashMap to store key value pair it has an inner class Entry which implements Map.Entry

here key is marked as final and along with it two other fields are there: hash and next

instances of Entry class are stored in an array which is transient.

Put():

- key object is checked for null. If key is null, value is stored in table[0] position. Because hash code for null is always 0.
- a hash value is calculated using key's hash code by calling its hashCode() method. This hash value is used to calculate index in array for storing Entry object.
- Now indexFor(hash, table.length) function is called to calculate exact index position for storing the Entry object.
- two unequal objects can have same hash code value and in that case objects will be stored in same array location [called bucket].
- Entry class had an attribute "next". This attribute always points to next object in chain. This is exactly the behavior of LinkedList.
- If there is already an object sitting on calculated index, its next attribute is checked. If it is null, and current Entry object becomes next node in LinkedList. If next variable is not null, procedure is followed until next is evaluated as null.

Get():

the way key uniqueness is determined in put() method , same logic is applied in get() method also.

The moment HashMap identify exact match for the key object passed as argument, it simply returns the value object stored in current Entry object.

20. Can we add a duplicate key in hashmap with different value? How will it work internally?

We can add duplicate key with different value in hashmap. But the old value of the duplicate key will be replaced by the new value.

Also in hashMap null values can be inserted.

21. Can we have multiple 'sturts.xml' configuration file? How to specify it?

Yes. We can have more than one struts.xml configuration files.

It is specified using <include> tag.

22. Define JSTL:

- The Java Server Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
- JSTL has support for
  - Core Tags
  - Formatting tags
  - SQL tags
  - XML tags

- JSTL Functions
- Specified using taglib in jsp.

23. Singleton Class:

```
Class MyClass{
        private static MyClass obj = null;
        private MyClass(){ .. }
        public static MyClass getInstance() {
                if(obj == null) {
                        obj = new MyClass();
                }
                return obj;
        }
}
```

24. The 5 main differences betwen HashMap and Hashtable

HashMap and Hashtable both implement java.util.Map.

HashMap	Hashtable
HashMap is non-synchronized and cannot be shared between multiple threads without proper synchronization.	Hashtable is synchronized, which means Hashtable is thread-safe and can be shared between multiple threads
HashMap allows null values as key	Hashtable doesn't allow nulls
Iterator in the HashMap is a fail-fast iterator	enumerator in the Hashtable is not
If HashMap is only used by one thread, it is faster than Hashtable.	synchronization Hashtable is much slower than HashMap if used in Single threaded environment.
HashMap does not guarantee that the order of the map will remain constant over time.	Hashtable guarantees order and is of natural order

25. return statement in try, catch and finally block. Will the return in finally be executed?

- If a return system is encountered in either try or catch block, java will try to execute the code in finally block and then it returns the value specified in the return statement.
- If in case a return system is encountered in either try or catch block, and while executing the finally block, if again it finds a return statement in finally then that is executed and hence it returns the value specified in finally block. (This scenario throws a warning stating that finally is not complete normally)

26. Can we invoke static method using null reference?

Yes. We can invoke a static method using null reference.

```
TestClass test = null;
test.staticMethod();
```

Reason: When accessing a static member through an object reference expression, only the declared type of the reference matters. This means that:

- It doesn't matter if the reference is actually null, since no instance is required
- If the reference is not null, it doesn't matter what the runtime type of the object is

27. Benefits of making a class immutable

- are simple to construct, test, and use
- are automatically thread-safe and have no synchronization issues
- do not need a copy constructor
- do not need an implementation of clone
- allow hashCode to use lazy initialization, and to cache its return value
- if an immutable object throws an exception, it's never left in an undesirable or indeterminate state

28. How to create an immutable class?

- Don't provide "setter" methods — methods that modify fields or objects referred to by fields.
- Make the constructor private and provide a Factory method to store object creation logic in single place
- Make all fields final and private
- Don't allow subclasses to override methods. declare the class as final.
- Identify the mutable variables and return new objects with copied content for all mutable objects. Immutable variables can be returned safely without extra effort.

29. What is a pointer and does Java support pointers?

Pointer is a reference handle to a memory location. Improper handling of pointers leads to memory leaks and reliability issues hence Java doesn't support the usage of pointers.

30. Is Java a pure object oriented language?

Java uses primitive data types and hence is not a pure object oriented language.

31. How to define a constant variable in Java?

The variable should be declared as static and final. So only one copy of the variable exists for all instances of the class and the value can't be changed also.

```
static final int MAX_LENGTH = 50;
```

32. main() method features:

- main()method is called by the JVM even before the instantiation of the class hence it is declared as static.
- main() method be overloaded. You can have any number of main()methods with different method signature and implementation in the class.

- Main()method doesn't return anything hence declared void.
- main()method accepts an array of String object as argument.
- main() method can be declared final. Any inheriting class will not be able to have it's own default main()method.

33. Can we execute a program without main() method?

Yes, one of the way is static block but in previous version of JDK not in JDK 1.7.

34. I want to print "Hello" even before main() is executed. How will you achieve that?

Print the statement inside a static block of code. Static blocks get executed when the class gets loaded into the memory and even before the creation of an object. Hence it will be executed before the main()method. And it will be executed only once.

35. What is not allowed to do with Generics?

- You can't have static field of type
- You can not create an instance of T
- Generics are not compatible with primitives in declarations
- You can't create Generic exception class

36. How Generics works in Java ? What is type erasure ?

Generics is implemented using Type erasure, compiler erases all type related information during compile time and no type related information is available during runtime.

For example List<String> is represented by only List at runtime.

Generic type is translated to Raw type by compiler during runtime.

List <T extends String>
⇓
List <String>

37. Which Collections class uses LRU cache mechanism?

LRU – Least Recently Used.

LinkedHashMap provides a method called removeEldestEntry() which is called by put() and putAll() and can be used to instruct to remove eldest entry.

We have to override the removeEldestEntry() of LinkedHashMap.

```
Map<String,String> lhm = new LinkedHashMap(MAX_ENTRIES + 1, .75F, false) {
        protected boolean removeEldestEntry(Map<String,String> eldest) {
                return size() > MAX_ENTRIES;
        }
};
```

Parameters:

initialCapacity - the initial capacity

loadFactor - the load factor

accessOrder - the ordering mode - true for access-order, false for insertion-order

38. Can we use Generics with Array?

Array doesn't support Generics and that's why usually prefer List over Array because List can provide compile time type-safety over Array.

39. Difference between List<?> and List<Object> in Java?

- List<?> is List of unknown type while List<Object> is essentially List of any Type.
- You can assign List<String>, List<Integer> to List<?> but you can not assign List<String> to List<Object>.

40. What is blank or uninitialized final variable?

- A final variable that is not initialized at the time of declaration is known as blank final variable.
- If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful.
- It can be initialized only in constructor.

41. Can we declare a constructor final?

No, because constructor is never inherited.

42. Is final method inherited?

Yes, final method is inherited but you cannot override it.

43. Is volatile keyword enough?

If two threads are both reading and writing to a shared variable, then using the volatile keyword for that is not enough. You need to use synchronization in that case to guarantee that the reading and writing of the variable is atomic.

But in case one thread reads and writes the value of a volatile variable, and other threads only read the variable, then the reading threads are guaranteed to see the latest value written to the volatile variable. Without making the variable volatile, this would not be guaranteed.

44. What is downcasting?

When Subclass type refers to the object of Parent class, it is known as downcasting.

Class A { } Class B ext A{
B b = new A();

45. Uses of java package

- Package is a way to organize files in java when a project consists of multiple modules.
- It also helps resolve naming conflicts.
- Package's access level also allows you to protect data from being used by the non-authorized classes.

46. What are the ways to refer to a class that is present in different package?

There are 3 different ways:

- Using fully qualified name (But this is not a good practice.) - class MyDate extends java.util.Date
- import the only class you want to use - import java.util.Date;
- import all the classes from the particular package - import java.util.*;

47. Where not to use Assertion?

- Assertion should not be used to check arguments in the public methods because it should result in appropriate run-time exception e.g. IllegalArgumentException, NullPointerException etc.
- Do not use assertion, if you don't want any error in any situation.

48. What is the purpose of instance initializer block?

Instance Initializer block is used to initialize the instance data member at run-time each time when object of the class is created.

Note: The java compiler copies the code of instance initializer block in every constructor.

49. Rules for instance initializer block :

- The instance initializer block is created when instance of the class is created.
- The instance initializer block is invoked after the parent class constructor is invoked (i.e. after super() constructor call).
- The instance initializer block comes in the order in which they appear.

50. What The Java equivalent of const keyword?

Variable type	Constant object/variable contents
Primitive	final variable/final array reference
Object	Use an immutable object or create a subclass that forces immutability.

51. Is there a goto statement in Java? If not what is its Java equivalent?

- The Java keyword list specifies the goto keyword, but it is marked as "not used".
- You could use a labeled break statement for its equivalent.

52. What are the access specifiers?

There are four access specifiers Java supports,

- public
- protected
- default (not specified at all)
- private

53. What are the access modifiers?

- Static
- final
- abstract
- transient
- synchronized

54. What is the difference between String str = "SOME" and String str = new String("SOME")?

- String str="SOME" uses the String pool
- String str="SOME" is better for most purposes, because it implicitly instantiates and initializes a String object with value "SomeValue".
- String str = new String("SOME") always create a new object on the heap.
- new String("SOME") is rarely used except to create an independent copy of an existing string variable. String b = new String(a)

55. Difference between checked and unchecked exceptions

Checked Exceptions	Unchecked Exceptions
the checked exceptions are checked at compile-time	unchecked exceptions are checked at runtime
the program gives a compilation error	the program won't give a compilation error
Checked exceptions should either be catched or should be thrown since these exceptions will not propagate in call stack	Unchecked exceptions need not be declared since these exceptions propagate in call stack
Checked exceptions are subclasses of Exceptions	All Unchecked exceptions are sub classes of RuntimeException class.

56. In what order are the exceptions specified in catch statements?

- Always catch the most specific exception first and then the most generic one.
- If not all the exceptions will be catched in the First Catch block because Exception is superclass of all the exceptions if it is declared like below:

```
}catch (Exception e) {
        System.out.println("Err: Exception Occurred");

}catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Err: Array Out of Bound");
}
```

57. How are checked exceptions thrown? What exception can be thrown instead of checked exception?

- Checked exceptions are thrown using either throw or throws keyword.
- Exception class can be thrown instead of checked exceptions or customized exceptions can also be thrown

58. Is it better to catch a single exception or catch detailed multiple exceptions?
Whenever we need to handle exceptions which is specific to certain exception, we can't catch them as general exception.
Eg: Instead of catching general SQLException we should catch ConstraintViolationException or PrimaryKeyViolation which would be more useful to give meaningful messages.

59. Can we able to modify catch block's parameter?

- Yes. We can modify if it is catching only a single exception.
- If a catch block handles more than one exception type, then the catch parameter is implicitly final.

60. Which is the least exception among IOException and SQLException?
Both exceptions are direct sub classes of Exception class.

61. What will happen if an exception is occurs in finally block?

- If it is a run-time exception then it will propagate in call stack.
- If it is a compile time exception then we need to either catch it or throw it.

62. Difference between List and Set

List	Set
List is an ordered collection it maintains the insertion order	Set is an unordered collection except LinkedHashSet(insertion order)
List allows duplicate elements	Set doesn't allow duplicate elements
List allows any number of null values	Set can have only a single null value except TreeSet(doesn't allow null values)
ListIterator can be used to traverse a List in both the directions	Set uses Iterator

63. Difference between ArrayList and LinkedList

ArrayList	LinkedList
internally uses dynamic array	internally uses doubly linked list
Manipulation with ArrayList is slow because it internally uses array. If any element is removed from the array, all the bits are shifted in memory.	Manipulation with LinkedList is faster than ArrayList because it uses doubly linked list so no bit shifting is required in memory.
ArrayList maintains only indexes and element data	LinkedList maintains element data and two pointers for neighbor nodes hence the memory consumption is high
ArrayList class can act as a list	LinkedList class can act as a list and queue
Random Access list	Sequential access list

64. What is the difference between Java 1.6 and 1.7?

- String in Switch Expression
- Underscores Between Digits in Numeric Literals (int i= 23_43_654;)
- Integral Types as Binary Literals (int i= 0b00_01_00;)
- Handling multiple exceptions in a single catch block
- Try-with-resources Statement
- Automatic Type Inference in Generic object instantiation (diamond <>)
- Earlier to JDK 1.7, to print static blocks no main() method is required. But from JDK 1.7, if no main() exists, static blocks will not be executed.

65. What do you mean by Java is platform independent?

- In java, when we execute the source code it generates the .class file comprising the bytecodes.
- Bytecodes are easily interpreted by JVM which is available with every type of OS we install.

66. What is the purpose of Home.class? [.class?]

Home.class will return the representation of the Home class as a Class object.
In most cases, this expression is used when one is using reflection, and needs a way to refer to the class itself rather than an instance of the class.

67. Difference between Java heap memory and stack

Java Heap Memory:

- Heap memory is used by java runtime to allocate memory to Objects and JRE classes.
- Whenever we create any object, it's always created in the Heap space.
- Garbage Collection runs on the heap memory to free the memory used by objects that doesn't have any reference.
- Any object created in the heap space has global access and can be referenced from anywhere of the application.

Java Stack Memory:

- Java Stack memory is used for execution of a thread.
- Stack memory is always referenced in LIFO (Last-In-First-Out) order.
- Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method.
- As soon as method ends, the block becomes unused and become available for next method.

Stack memory size is very less compared to Heap memory.

68. How is out of memory error occurring? What should we do to avoid it?

- The virtual machine on which the Java program is running on may have a limit to the amount of memory it will give access to.
- The system on which your program is running may have run out of physical and virtual memory.
- Your application may just be consuming too much memory.

Possible Solutions:

- In the first case, you may be able to modify the maximum heap size of the virtual machine .
- In the second or third you'll need to redesign your application.
- Can call garbage collector to clean but not sure whether it will happen.

69. Can we add final to an overridden method? If so what will happen?

- Yes we can add final to an overridden method.
- But when a class extends this subclass then it cannot override subclass's final method.

70. Difference between extending Thread class and implementing Runnable interface.

Thread class	Runnable interface
By extending Thread, each of your threads has a unique object associated with it	Implementing Runnable, many threads can share the same object instance.
FirstThread firstThread = new FirstThread();	Runner r = new Runner(); Thread t1 = new Thread(r, "Thread A"); Thread t2 = new Thread(r, "Thread B");
A class that extends Thread class is exactly a thread.	A class that implements Runnable is not a thread and just a class.
When we extend the Thread class we can't inherit from any other class and in that case implementing the Runnable interface is more appropriate.	The Runnable interface should be used if you are only planning to override the run() method and no other Thread methods. (if extending this class the programmer should not modify the behavior of the class)

71. Difference between asynchronous and synchronous threads.

- When you execute something synchronously, you wait for it to finish before moving on to another task.
- When you execute something asynchronously, you can move on to another task before it finishes.
- Objects pass messages to each other, and the receipt of some message causes an appropriate message-handler -- a Java method -- to be executed.
- Most of these messages are synchronous: their handlers don't return until they're finished doing what they do.
- Other messages are asynchronous: the handler returns immediately, before the requested operation completes. Meanwhile, work is going on in the background to satisfy the original request.

72. What are programmatic and declarative exception handling?

Usually there are two ways in which you can catch the exceptions:

Programmatic Exception Handling :

- In this approach the exceptions are caught using normal java language try/catch block.
- In this approach the flow of control is also maintained by the programs.
- The main drawback of the approach is the developer has to write the code for the flow of the application.

Declarative Exception Handling :

- Declarative Exception Handling is the way of handling Exceptions with the help of xml files so there is no need to write exception-handling code in the application.
- In this approach the exceptions are defined in the struts-config file or web.xml and in case of the exception occurs the control is automatically passed to the appropriate error page.
- The biggest benefit of Declarative Exception Handling is if there is requirement to change the exception handling mechanism, changes can be made to the xml file, without recompilation of java code.

73. What is ConcurrentHashMap?

- ConcurrentHashMap is thread safe without synchronizing the whole map.
- It does not lock the Map while you are reading from it.
- It does not lock the entire Map when writing to it. It only locks the part of the Map that is being written to, internally.
- It doesn't throw a ConcurrentModificationException if one thread tries to modify it while another is iterating over it.

74. What is the purpose of RandomAccess interface?

- Marker interface used by List implementations to indicate that they support fast (generally constant time) random access.
- The primary purpose of this interface is to allow generic algorithms to alter their behavior to provide good performance when applied to either random or sequential access lists.

75. How Java enabled High Performance?
Java uses Just-In-Time compiler to enable high performance.
Just-In-Time compiler is a program that turns Java bytecode into instructions that can be sent directly to the processor.

76. What do you mean by Object?
Object is a runtime entity and it's state is stored in fields and behavior is shown via methods. Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

77. Why is String class considered immutable?
- Requirement of String Pool
- Caching Hashcode
- Secured and thread-safe

78. What is JAR file?
JAR files is Java Archive File and it aggregates many files into one. It holds Java classes in a library. JAR files are built on ZIP file format and have .jar file extension.

79. What is a WAR file?
This is Web Archive File and used to store XML, java classes, and JavaServer pages. which is used to distribute a collection of JavaServer Pages, Java Servlets, Java classes, XML files, static Web pages etc.

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

81. What are Wrapper classes?
- a wrapper class wraps (encloses) around a data type and gives it an object appearance.
- Since Java is object oriented and wherever the data type is required as an object, this object can be used.
- Wrapper classes include methods to unwrap the object and give back the data type.
- All of the primitive wrapper classes in Java are immutable.

82. What is Composition in Java?
- Composition is the design technique to implement has-a relationship in classes.
- Java composition is achieved by using instance variables that refers to other objects.
- In the composition approach, the subclass becomes the "front-end class," and the superclass becomes the "back-end class.

Eg: **a Person has a Job.**

```
public class Person {
        private Job job;
         public long getSalary() {
                return job.getSalary();
        }
}
```

83. When to use composition over inheritance in java?
- When you reuse code from the superclass, rather than override methods and redefine another polymorphic behavior, then you should use composition instead of inheritance.
- Inheritance should be used only when a subclass is-a superclass and composition for has-a relationship.
- No need to change the inherited class whenever there is an additional method in the superclass.

84. Difference between inheritance and composition

- It is easier to change the interface of class while using composition than inheritance.
- Creation of objects is delayed in composition since only when needed the back-end-class is created whereas in inheritance when the subclass is created the superclass is also created.
- It is easier to add new subclasses (inheritance) than it is to add new front-end classes (composition), because by extending we inherit all the behaviors of superclass automatically.
- The explicit method-invocation approach of composition will often have a performance cost as compared to inheritance's single invocation of an inherited superclass method implementation.

85. Which Java operator is right associative?

The = operator is right associative.

86. What will happen if static modifier is removed from the signature of the main method?

Program throws "NoSuchMethodError" error at runtime

87. What is dot operator?

- The dot operator(.) is used to access the instance variables and methods of class objects.
- It is also used to access sub-packages from a package.

88. Can you declare an interface method static?

- No, because methods of an interface is abstract by default, and static and abstract keywords can't be used together.
- Also interface methods cannot be final, because its implementation should be provided by another class.

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

- One can import the same package or same class multiple times. Neither compiler nor JVM complains about it.
- The JVM will internally load the class only once no matter how many times you import the same class.

90. What is shallow coping and deep coping?

Shallow: A new object is created that has an exact copy of the values in the original object. If any of the fields of the object are references to other objects, just the reference addresses are copied i.e., only the memory address is copied.
Achieved by using clone methods.
Deep: A deep copy copies all fields, and makes copies of dynamically allocated memory pointed to by the fields. A deep copy occurs when an object is copied along with the objects to which it refers.
Achieved by using serialization and also by coping all properties explicitly.

58. What is the relation between value stack and OGNL?

- Value stack is a place in memory where all the data related to action and action itself is stored.
- OGNL is a means through which data in the vlue stack is manipulated and mapped to the properties in bean class.

59. What is an Action class?

- Action class are POJO classes and contains the business data which is used by the view layer with the help of getters and setters.
- It also decides which result to be rendered for the response.

60. Which filter is used as front controller in struts2?

FilterDispatcher was used as front controller in the earlier versions of struts.
StrutsPrepareAndExecuteFilter is the recent one which is used.

61. What is the differece between Push MVC and Pull MVC?

- In Push MVC the data model is controlled and given to the view layer by controllers by putting them in the scoped variables like request or session. Eg: Spring MVC and Struts1
- in Put MCV the data model is constructed in the controllers and are kept in a common place (ie. In actions) which then gets rendered by the view layer. Generally data are stored in Value Stack. Eg: Struts2

62. Why filter is designated as front controller in struts2?

- Servlets needs to provide right value in the <load-on-startup> to initialize config files when the container is loaded, and in the absence of which it gets initialized only when the first request hits.
- The filters in web.xml gets initialized automatically as the container starts.
- Also usage filters introduced interceptors and also provides UI Themes

63. What is DIP?

DIP – Dependency Inversion Principle states:

- High level modules should not depend on low level modules and should depend on abstraction.
- Abstraction should not depend on details and only details should depend on abstraction.

64. What are the steps involved in dB operation in hibernate.

- Define POJO
- Define repository interface and implementation classes (DAO)
- Define service interface ans implementation classes that make use of DAO classes.
- Use an IoC container like Spring to wire hibernate classes and DAO and services.

65. Difference between servlet sendRedirect and forward

- **Redirect request** is used to redirect to resources to different servers or domains.
- **Forward request** is used to forward to resources available within the server from where the call is made.

For example, database update and data display can be separated by redirect. Do the PaymentProcess and then redirect to displayPaymentInfo. If the client refreshes the browser only the displayPaymentInfo will be done again and PyamenProcess will not be repeated. But if you use forward in this scenario, both PaymentProcess and displayPaymentInfo will be re-executed sequentially, which may result in incosistent data.