1. What are the principle concepts of OOPS? There are four principle concepts upon which object oriented design and programming rest. They are:

- Abstraction
- Polymorphism
- Inheritance
- Encapsulation

2. What is Abstraction? Abstraction refers to the act of representing essential features without including the background details or explanations. The process of hiding working style of an object and showing an information of an object in understandable manner. E.g. ATM Machine

3. What is Encapsulation? Encapsulation is a technique used for hiding the properties and behaviors of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.

4. What is the difference between abstraction and encapsulation?

Abstraction	Encapsulation
Abstraction solves the problem in the design level.	Encapsulation solves the problem in the implementation level.
Abstraction is used for hiding the unwanted data and giving relevant data.	Encapsulation means hiding the code and data into single unit to protect the data from outside world.
Abstraction lets you focus on what the object does instead of how it does it.	Encapsulation means hiding the internal details or mechanics of how an object does something.
Abstraction - outer layout , used in terms of design e.g. outer look of a mobile phone , like it has a display screen and keypad button to dial a number.	Encapsulation – inner layout, used in terms of implementation. E.g. inner implementation details of a mobile phone, how keypad button and display screen are connected with each other using circuits.

5. What is Inheritance? Inheritance is the process by which objects of one class acquire the properties of objects of another class. A class that is inherited is called a superclass. The class that does the inheriting is called a subclass. Inheritance is done by using the keyword extends. The two most common reasons to use inheritance are:

- To promote code reuse
- To use polymorphism

6. What is Polymorphism? Polymorphism is briefly described as "one interface, many implementations." Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form.

7. How does Java implement polymorphism? (Inheritance, Overloading and Overriding are used to achieve Polymorphism in java). Polymorphism manifests itself in Java in the form of multiple methods having the same name.

- In some cases, multiple methods have the same name, but different formal argument lists **(overloaded methods).**
- In other cases, multiple methods have the same name, same return type, and same formal argument list **(overridden methods).**

8. Explain the different forms of Polymorphism. There are two types of polymorphism one is **Compile time polymorphism** and the other is **Run time polymorphism**. **Compile time polymorphism** is method overloading. **Runtime time polymorphism** is done using inheritance and interface.

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

9. What is runtime polymorphism or dynamic method dispatch? In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

10. What is Dynamic Binding? Binding refers to the linking of a procedure call to the code to be executed in response to the call. **Dynamic binding (also known as late binding)** means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

11. What is method overloading? Method Overloading means to have two or more methods with same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same semantic operation but differ by argument number or type.

- Overloaded methods MUST change the argument list
- Overloaded methods CAN change the return type and access modifier
- Overloaded methods CAN declare new or broader checked exceptions
- A method can be overloaded in the same class or in a subclass

12. What is method overriding? Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass. The key benefit of overriding is the ability to define behavior that's specific to a particular subclass type.

- The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can't override a method marked public and make it protected).
- You cannot override a method marked final or static

13. What are the difference between method overloading and method overriding?

Method Overloading	Method Overriding
Method overloading is used to increase the readability of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.
Method overloading is performed with in class.	Method overriding occurs in two classes that have IS-A (inheritance) relationship
In case of method overloading, parameter must be different.	In case of method overriding, parameter must be same.
Method overloading is the example of compile time polymorphism.	Method overriding is example of run time polymorphism.
Overloaded method	Overridden method
In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in the method overloading. But you must have to change the parameter.	Return type must be same or covariant in method overriding.

`class OverloadingExample{`
`    static int add(int a, int b){`
`      return a+b;`
`   }`
`    static int add(int a, int b, int c){return a+b+c;}`
`}`
	`class Animal{`
	`    void eat(){System.out.println("eating...");}`
	`}`
	`class Dog extends Animal{`
	`     void eat(){System.out.println("eating bread...");}`
	`}`

14. Can overloaded methods be override too? Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. Compiler will not binding the method calls since it is overloaded, because it might be overridden now or in the future.

15. Is it possible to override the main method? NO, because main is a static method. A static method can't be overridden in Java.

16. How to invoke a superclass version of an Overridden method? To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance, or use the super prefix in the subclass itself. From the point of the view of the subclass, the super prefix provides an explicit reference to the superclass' implementation of the method. **super.overriddenMethod ();**

17. What is super? Super is a keyword which is used to access the method or member variables from the superclass. If a method hides one of the member variables in its superclass, the method can refer to the hidden variable through the use of the super keyword. In the same way, if a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword.

- You can only go back one level.
- In the constructor, if you use super (), it must be the very first code, and you cannot access any this.xxx variables or methods to compute its parameters.

18. How do you prevent a method from being overridden? To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means "this is the final implementation of this method", the end of its inheritance hierarchy.

```
public final void exampleMethod() {  //  Method statements   }
```

19. What is an Interface? An interface is a description of a set of methods that conforming implementing classes must have.

- You can't mark an interface as final.
- Interface variables must be static.
- An Interface cannot extend anything but another interfaces.

20. Can we instantiate an interface? You can't instantiate an interface directly, but you can instantiate a class that implements an interface.

21. Can we create an object for an interface? Yes, it is always necessary to create an object implementation for an interface. Interfaces cannot be instantiated in their own right, so you must write a class that implements the interface and fulfill all the methods defined in it.

22. Do interfaces have member variables? Interfaces may have member variables, but these are implicitly public, static, and final- in other words, interfaces can declare only constants, not instance variables that are available to all implementations and may be used as key references for method arguments for example.

23. What modifiers are allowed for methods in an Interface? public, static , default and abstract modifiers are allowed for methods in interfaces.

24. What is a marker interface? Marker interfaces are those which do not declare any required methods, but signify their compatibility with certain operations. The **java.io.Serializable interface** and **Cloneable** are typical marker interfaces. These do not contain any methods, but classes must implement this interface in order to be serialized and de-serialized.
25. What is an abstract class? Abstract class that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.

- If even a single method is abstract, the whole class must be declared abstract.
- Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.
- You can't mark a class as both abstract and final.

26. Can we instantiate an abstract class? An abstract class can never be instantiated. Its sole purpose is to be extended (subclasses).
27. What are the difference between Interface and Abstract class?

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

28. When should I use abstract classes and when should I use interfaces?

Use Interfaces when…

- You see that something in your design will change frequently.
- If various implementations only share method signatures then it is better to use Interfaces.
- You need some classes to use some methods which you don't want to be included in the class, then you go for the interface, which makes it easy to just implement and make use of the methods defined in the interface.

Use Abstract Class when…

- If various implementations are of the same kind and use common behavior or status then abstract class is better to use.
- When you want to provide a generalized form of abstraction and leave the implementation task with the inheriting subclass.
- Abstract classes are an excellent way to create planned inheritance hierarchies. They're also a good choice for nonleaf classes in class hierarchies.

29. When you declare a method as abstract, can other nonabstract methods access it? Yes, other nonabstract methods can access a method that you declare as abstract.
30. Can there be an abstract class with no abstract methods in it? Yes, there can be an abstract class without abstract methods.
31. What is Constructor?

- A constructor is a special method whose task is to initialize the object of its class.
- It is special because its name is the **same as the class name**.
- They do not have return types, not even **void** and therefore they cannot return values.
- They **cannot be inherited**, though a derived class can call the base class constructor.
- Constructor is invoked whenever an object of its associated class is created.

32. How does the Java default constructor be provided? If a class defined by the code does **not** have any constructor, compiler will automatically provide one no-parameter-constructor (default-constructor) for the class in the byte code. The access modifier (public/private/etc.) of the default constructor is the same as the class itself.

34. What are the difference between Constructors and Methods?

Constructors	Methods
Create an instance of a class	Group Java statements
Cannot be *abstract, final, native, static*, or *synchronized*	Can be *abstract, final, native, static*, or *synchronized*
No return type, not even void	void or a valid return type
Same name as the class (first letter is capitalized by convention) -- usually a noun	Any name except the class. Method names begin with a lowercase letter by convention -- usually the name of an action
Refers to another constructor in the same class. If used, it must be the first line of the constructor	Refers to an instance of the owning class. Cannot be used by static methods.
Calls the constructor of the parent class. If used, must be the first line of the constructor	Calls an overridden method in the parent class
Constructors are not inherited	Methods are inherited

35. How are this () and super () used with constructors?

- **Constructors use *this*** to refer to another constructor in the same class with a different parameter list.
- **Constructors use *super*** to invoke the superclass's constructor. If a constructor uses *super*, it must use it in the first line; otherwise, the compiler will complain.

36. What are the difference between Class Methods and Instance Methods?

Class Methods	Instance Methods
Class methods are methods which are declared as static. The method can be called without creating an instance of the class	Instance methods on the other hand require an instance of the class to exist before they can be called, so an instance of a class needs to be created by using the new keyword. Instance methods operate on specific instances of classes.
Class methods can only operate on class members and not on instance members as class methods are unaware of instance members.	Instance methods of the class can also not be called from within a class method unless they are being called on an instance of that class.
Class methods are methods which are declared as static. The method can be called without creating an instance of the class.	Instance methods are not declared as static.

37. What are Access Specifiers? One of the techniques in object-oriented programming is encapsulation. It concerns the hiding of data in a class and making this class available only through methods. Java allows you to control access to classes, methods, and fields via so-called access specifiers.

39. What are Access Specifiers/Modifiers available in Java? Java offers four access specifiers, listed below in decreasing accessibility:

- **Public**- *public* classes, methods, and fields can be accessed from everywhere.
- **Protected**- *protected* methods and fields can only be accessed **within the same class** to which the methods and fields belong, **within its subclasses**, and **within classes of the same package.**
- **Default(no specifiers)-** If you do not set access to specific level, then such a class, method, or field will be accessible from inside the **same package** to which the **class, method, or field belongs**, but not from outside this package.
- **Private**- *private* methods and fields can only be **accessed within the same class** to which the methods and fields belong. *private* methods and fields are not visible within subclasses and are not inherited by subclasses.

Situation	public	protected	default	private
Accessible to class from same package?	yes	yes	yes	no
Accessible to class from different package?	yes	no, *unless it is a subclass*	no	no

40. What is final modifier? The final modifier keyword makes that the programmer cannot change the value anymore. The actual meaning depends on whether it is applied to a class, a variable, or a method.

- **final Classes**- A final class cannot have subclasses. will cause compilation error
- **final Variables**- A final variable cannot be changed once it is initialized.
- **final Methods**- A final method cannot be overridden by subclasses. will cause compilation error.

41. What are the uses of final method? There are two reasons for marking a method as final:

- Disallowing subclasses to change the meaning of the method.
- Increasing efficiency by allowing the compiler to turn calls to the method into inline Java code.

42. What is static block? Static block which executed exactly once when the class is first loaded into JVM. Before going to the main method the static block will execute.

43. What are static variables? Variables that have only one copy per class are known as static variables. They are not attached to a particular instance of a class but rather belong to a class as a whole. They are declared by using the static keyword as a modifier. **static type varIdentifier;** where, the name of the variable is varIdentifier and its data type is specified by type.
Note: Static variables that are not explicitly initialized in the code are automatically initialized with a default value. The default value depends on the data type of the variables.

44. What is the difference between static and non-static variables? 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.

45. What are static methods? Methods declared with the keyword static as modifier are called static methods or class methods. They are so called because they affect a class as a whole, not a particular instance of the class. Static methods are always invoked without reference to a particular instance of a class.
Note: **The use of a static method suffers from the following restrictions:**

- A static method can only call other static methods.
- A static method must only access static data.
- A static method **cannot** reference to the current object using keywords super or this.

46. What is an Iterator?

- The Iterator interface is used to step through the elements of a Collection.
- Iterators let you process each element of a Collection.
- Iterators are a generic way to go through all the elements of a Collection no matter how it is organized.
- Iterator is an Interface implemented a different way for every Collection.

47. How do you traverse through a collection using its Iterator? To use an iterator to traverse through the contents of a collection, follow these steps:

- Obtain an iterator to the start of the collection by calling the collection ***iterator ()*** method.
- Set up a loop that makes a call to ***hasNext ()***. Have the loop iterate as long as ***hasNext ()*** returns **true**.
- Within the loop, obtain each element by calling **next ()**.

48. How do you remove elements during Iteration? Iterator also has a method ***remove ()*** when remove is called, the current element in the iteration is deleted.

49. What is the difference between Enumeration and Iterator?

Enumeration	Iterator
Enumeration doesn't have a remove() method	Iterator has a remove() method
Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects	Can be abstract, final, native, static, or synchronized
java.util.Enumeration doesn't allows to remove elements from collection during iteration in java	Java.util.Iterator allows to remove elements from collection during iteration by using remove () method in java.
nextElement () Method Returns the next element of this enumeration if this enumeration object has at least one more element to provide. **hasMoreElements ()** returns true if enumeration contains more elements.	nextElement () has been changed to next() in Iterator. hasMoreElements () has been changed to hasNext() in Iterator.
JDK 1.0	JDK 2.0
Enumeration returned by Vector is fail-safe	Iterator returned by Vector are fail-fast

Note: **So Enumeration is used whenever we want to make Collection objects as Read-only.**

50. How is ListIterator? **ListIterator** is just like Iterator, except it allows us to access the collection in either the forward or backward direction and lets us modify an element

51. What is the List interface?

- The List interface provides support for ordered collections of objects.
- Lists may contain duplicate elements.

52. What are the main implementations of the List interface? The main implementations of the List interface are as follows:

- **ArrayList**: Resizable-array implementation of the List interface. The best all-around implementation of the List interface.
- **Vector**: Synchronized resizable-array implementation of the List interface with additional "legacy methods."

- **LinkedList**: Doubly-linked list implementation of the List interface. May provide better performance than the ArrayList implementation if elements are frequently inserted or deleted within the list. Useful for queues and double-ended queues (deques).

53. What are the advantages of ArrayList over arrays? Some of the advantages ArrayList has over arrays are:

- It can grow dynamically
- It provides more powerful insertion and search mechanisms than arrays.

54. Difference between ArrayList and Vector?

ArrayList	Vector
ArrayList is **NOT** synchronized by default.	Vector List is synchronized by default.
ArrayList can use only Iterator to access the elements.	Vector list can use Iterator and Enumeration Interface to access the elements.
The ArrayList increases its array size by 50 percent if it runs out of room.	A Vector defaults to doubling the size of its array if it runs out of room
ArrayList has no default size.	While vector has a default size of 10.
ArrayList was introduced in second version of java i.e. JDK 2.0	Vector was introduced in first version of java i.e. JDK 1.0
Enumeration is fail-fast, means any modification made to ArrayList during iteration using Enumeration will throw ConcurrentModificationException in java.	Enumeration is fail-safe, means any modification made to Vector during iteration using Enumeration don't throw any exception in java.

55. How to obtain Array from an ArrayList? Array can be obtained from an ArrayList using ***toArray ()*** method on ArrayList.

```
List arrayList = new ArrayList ();
arrayList.add (â€¦
ObjectÂ a [] = arrayList.toArray ();
```

56. Why insertion and deletion in ArrayList is slow compared to LinkedList?

- **ArrayList** internally uses and array to store the elements, when that array gets filled by inserting elements a new array of roughly 1.5 times the size of the original array is created and all the data of old array is copied to new array.
- During deletion, all elements present in the array after the deleted elements have to be moved one step back to fill the space created by deletion. Deletion in linked list is fast because it involves only updating the next pointer in the node before the deleted node and updating the previous pointer in the node after the deleted node.

57. Why are Iterators returned by ArrayList called Fail Fast? Because, if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a **ConcurrentModificationException.** Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

58. How do you decide when to use ArrayList and When to use LinkedList? If you need to support random access, without inserting or removing elements from any place other than the end, then ArrayList offers the optimal collection. If, however, you need to frequently add and remove elements from the middle of the list and only access the list elements sequentially, then LinkedList offers the better implementation.

59. What is the Set interface?

- The Set interface provides methods for accessing the elements of a finite mathematical set
- Sets do not allow duplicate elements
- Contains no methods other than those inherited from Collection
- It adds the restriction that duplicate elements are prohibited
- Two Set objects are equal if they contain the same elements

60. What are the main Implementations of the Set interface? The main implementations of the List interface are as follows:

- HashSet
- TreeSet
- LinkedHashSet
- EnumSet

61. What is a HashSet?

- A HashSet is an unsorted, unordered Set.
- It uses the hashcode of the object being inserted.

- Use this class when you want a collection with no duplicates and you don’t care about order when you iterate through it.

62. What is a TreeSet? TreeSet is a Set implementation that keeps the elements in sorted order. The elements are sorted according to the natural order of elements or by the comparator provided at creation time.

63. What is an EnumSet? An EnumSet is a specialized set for use with enum types, all of the elements in the EnumSet type that is specified, explicitly or implicitly, when the set is created.

64. Difference between HashSet and TreeSet?

HashSet	TreeSet
HashSet is under set interface i.e. it does not guarantee for either sorted order or sequence order.	TreeSet is under set i.e. it provides elements in a sorted order (acceding order).
We can add any type of elements to hash set.	We can add only similar types of elements to tree set.

65. What is a Map?

- A map is an object that stores associations between keys and values (key/value pairs).
- Given a key, you can find its value. Both keys and values are objects.
- The keys must be unique, but the values may be duplicated.
- Some maps can accept a null key and null values, others cannot.

66. What are the main Implementations of the Map interface? The main implementations of the List interface are as follows:

- HashMap
- HashTable
- TreeMap
- EnumMap

67. What is a TreeMap? TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. **TreeMap is based on the Red-Black tree data structure.**

68. How do you decide when to use HashMap and when to use TreeMap? For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then TreeMap is your better alternative. Depending upon the size of your collection, it may be faster to add elements to a HashMap, then convert the map to a TreeMap for sorted key traversal.

69. Difference between HashMap and Hashtable?

java.util.HashMap	java.util.Hashtable
Java.util.HashMap is **not synchronized.**	Java.util.Hashtable is **synchronized.**
HashMap is not synchronized, hence its operations are **faster** as compared to Hashtable in java.	Hashtable is synchronized, hence its operations are **slower** as compared to HashMap in java.
HashMap allows to store **one null key** and **many null values** i.e. many keys can have null value in java.	Hashtable does **not allow to store null key or null value**. Any attempt to store null key or value throws runtimeException (NullPointerException) in java.
HashMap was introduced in second version of java i.e. **JDK 2.0**	Hashtable was introduced in first version of java i.e. **JDK 1.0**
In non-multithreading environment it is recommended to use HashMap than using Hashtable in java.	In **java 5 i.e. JDK 1.5**, it is **recommended** to use ConcurrentHashMap than using Hashtable.
HashMap does not extends Dictionary in java.	Hashtable extends Dictionary (which maps non-null keys to values. In a given Dictionary we can look up value corresponding to key) in java.

Note: Only one NULL is allowed as a key in HashMap. HashMap does not allow multiple keys to be NULL. Nevertheless, it can have multiple NULL values.

71. What are the different Collection Views That Maps Provide? Maps Provide Three Collection Views.

- **Key Set** - allow a map's contents to be viewed as a set of keys.
- **Values Collection** - allow a map's contents to be viewed as a set of values.

- **Entry Set** - allow a map's contents to be viewed as a set of key-value mappings.

72. What is a KeySet View? KeySet is a set returned by the ***keySet ()*** method of the Map Interface, It is a set that contains all the keys present in the Map.

73. What is a Values Collection View? Values Collection View is a collection returned by the ***values ()*** method of the Map Interface, It contains all the objects present as values in the map.

74. What is an EntrySet View? Entry Set view is a set that is returned by the ***entrySet ()*** method in the map and contains Objects of type Map. Entry each of which has both Key and Value.

75. How do you sort an ArrayList (or any list) of user-defined objects? Create an implementation of the ***java.lang.Comparable*** **interface** that knows how to order your objects and pass it to ***java.util.Collections.sort*** **(List, Comparator).**

76. What is the Comparable interface? The Comparable interface is used to sort collections and arrays of objects using the Collections.sort () and java.utils.Arrays.sort () methods respectively. The objects of the class implementing the Comparable interface can be ordered. The Comparable interface in the generic form is written as follows:

interface Comparable<T> where T is the name of the type parameter.

All classes implementing the Comparable interface must implement the compareTo () method that has the return type as an integer. The signature of the compareTo () method is as follows: **int i = object1.compareTo (object2)**

If object1 < object2: The value of i returned will be negative.
If object1 > object2: The value of i returned will be positive.
If object1 = object2: The value of i returned will be zero.

77. What are the difference between the Comparable and Comparator interfaces?

Comparable	Comparator
It uses the compareTo () method. int objectOne.compareTo(objectTwo).	It uses the compare () method. int compare(ObjOne, ObjTwo)
It is necessary to modify the class whose instance is going to be sorted.	A separate class can be created in order to sort the instances.
Only one sort sequence can be created.	Many sort sequences can be created.
Comparable is used to compare instances of same class in java.	Comparator can be used to compare instances of same or different classes in java.
It is frequently used by the API classes.	It used by third-party classes to sort instances.
For using Comparable, original Class must implement it.	Class itself can implement Comparator or any other class can implement Comparator. Hence avoiding modification to original class.
Provides sorting only on one criteria, because Comparable can be implemented by original class only in java.	We can use Comparator to sort class on many criteria because class itself or any other class can implement Comparator in java.
java.lang.Comparable.	import java.util.Comparator
Comparable can be implemented by class which need to define a natural ordering for its objects.	Comparator is implemented when one wants a different sorting order and define custom way of comparing two instances.
Collections.sort (list) uses Comparable interface for sorting class.	Collections.sort(list,new ComparatorName()); Uses Comparator interface for sorting class.

78. Is there any difference between creating string with and without new operator? When String is created **without new operator**, it will be created in string pool. When String is created **using new** operator, it will force JVM to create new string in heap (not in string pool).

79. Can we use custom object as key in HashMap? If yes then how? For using object as Key in HashMap, we must **implements equals and hashcode method**. Classes must be made **immutable classes**.

80. How do we override equals and hashcode method, write a code to use Employee (i.e. custom object) as key in HashMap?

```
@Override
  public boolean equals(Object obj){
       if(obj==null)
            return false;
       if(this.getClass()!=obj.getClass())
            return false;
       Employee emp=(Employee)obj;
```

```
        return (emp.id==this.id || emp.id.equals(this.id))
                    && (emp.name==this.name || emp.name.equals(this.name));
    }
    @Override
    public int hashCode(){
        int hash=(this.id==null ? 0: this.id.hashCode() ) +
                (this.name==null ? 0: this.name.hashCode() );
        return hash;
    }
```

90. What classes should i prefer to use a key in HashMap? Use Integer class as key in HashMap

```
public class StringInMap {
  public static void main(String...a){
    //HashMap's key=Integer class  (Integer's api has already overridden hashCode() and equals() method for us )
      Map<Integer, String> hm=new HashMap<Integer, String>();
      hm.put(1, "data");
      hm.put(1, "data OVERRIDDEN");
      System.out.println(hm.get(1));
  }
}
/*OUTPUT :- data OVERRIDDEN  */
```

91. In Integer's API, how Integer class has overridden equals () and hashCode () method

```
public int hashCode() {
      return value;
 }
public boolean equals(Object obj) {
      if (obj instanceof Integer) {
       return value == ((Integer)obj).intValue();
      }
      return false;
 }
```

92. Can overriding of hashcode () method cause any performance issues? Improper implementation of hashCode() can cause performance issues, because in that most of the key-value pairs will be stored on same bucket location and unnecessary time will be consumed while fetching value corresponding to key.

93. What are immutable classes in java? How we can create immutable classes in java? And what are advantages of using immutable classes? Any change made to object of immutable class produces new object.
Example- **String is Immutable class** in java, any changes made to Sting class.
We must follow following steps for creating immutable classes –
1) Final class - Make class final so that it cannot be inherited
2) private member variable:- Making member variables private ensures that fields cannot be accessed outside class.
3) final member variable -> Make member variables final so that once assigned their values cannot be changed
4) Constructor -> Initialize all fields in constructor. Assign all mutable member variable using new keyword.
5) Don't provide **setter methods** in class/ provide only getter methods.
Advantages of using Immutable class:-

- **Thread safe -** Immutable classes are thread safe, they will never create race condition.
- **Key in HashMap -** Immutable classes are can be used as key in Map (HashMap etc.)
- **HashCode is cached -** JVM caches the HashCode of Immutable classes used in application. JVM need not to calculate hashcode again. Hence, performance of application is improved significantly.
- **If Immutable class throws Exception -** If Immutable class throws Exception, they are never left in undesirable state.

94. What are some immutable classes in java? Integer, Double, Long, Short, Byte, and all other Wrapper classes.

95. Difference between List and Set in java?

java.util.List	**java.util.Set**
Java.util.List is ordered collection it **maintain insertion order** in java.	Java.util.Set implementation does not **maintain insertion order**.
List **allows to store duplicate elements** in java.	Set does **not allow to store duplicate elements** in java.
List allows to store **many null keys** in java.	Set implementations allow to add only **one null** in java**.** TreeSet does not allow to add null in java.
List implementations provide get method to get element on specific index in java. ArrayList, Vector, copyOnWriteArrayList and LinkedList provides - **get(int index)**	Set implementations does not provide any such get method to get element on specified index in java

ArrayList, LinkedList, Vector, CopyOnWriteArrayList implementing classes	**HashSet, CopyOnWriteArraySet, LinkedHashSet, TreeSet**, **ConcurrentSkipListSet**, **EnumSet** classes
ListIterator method returns listIterator to iterate over elements in List in java.**listIterator provides** additional methods as compared to iterator like **hasPrevious(), previous(), nextIndex(),previousIndex(), add(E element), set(E element)**	Set does not provide anything like listIterator. It simply return Iterator in java.
List are Resizable-array implementation of the java.util.**List** interface in java.	Set uses **Map** for their implementation. Hence, structure is map based and resizing depends on Map implementation. Example > **HashSet** internally uses **HashMap**.
As **ArrayList** uses array for implementation it is index based structure, hence provides random access to elements. But **LinkedList** is not indexed based structure in java.	Set is not index based structure at all in java.

96. Difference between ArrayList and LinkedList in java?

Property	java.util.ArrayList	java.util.LinkedList
Structure	Java.util.ArrayList is index based structure in java.	A java.util.LinkedList is a node based structure in java.
Resizable	**ArrayList is Resizable-array in java.**	New node is created for storing new element in LinkedList in java.
Initial capacity	Java.util.ArrayList is created with initial capacity of 10 in java.	Linked List's initial capacity is 0 in java.
Ensuring Capacity/ resizing.	Array List's size is **increased by 50%** i.e. after resizing its size become 15 in java.	Linked List's initial capacity is 0, its size grow with addition of each and every element in java.
RandomAccess interface	ArrayList implements RandomAccess	LinkedList does not implement RandomAccess interface in java.
AbstractList and AbstractSequentialList	ArrayList extends AbstractList (abstract class) which provides implementation to List interface to minimize the effort required to implement this interface backed by RandomAccess interface.	LinkedList extends AbstractSequentialList (abstract class), AbstractSequentialList extends AbstractList. In LinkedList, data is accessed sequentially, so for obtaining data at specific index, iteration is done on nodes sequentially in java.
How get (index) method works?	Get method of ArrayList directly gets element on specified index. Hence, offering O (1) complexity in java.	Get method of LinkedList iterates on nodes sequentially to get element on specified index. Hence, offering O (n) complexity in java.
When to use	**Use ArrayList when get operations is more frequent than add and remove operations in java.**	**Use LinkedList when add and remove operations are more frequent than get operations in java.**

97. Difference between HashSet vs LinkedHashSet vs TreeSet in java:-

Property	java.util.HashSet	java.util.LinkedHashSet	java.util.TreeSet
Insertion order	Java.util.HashSet does not maintains insertion order in java.	Java.util.LinkedHashSet maintains insertion order in java.	Java.util.TreeSet is sorted by natural order in java.
Null elements	HashSet allows to store one null in java.	LinkedHashSet allows to store one null in java.	TreeSet does not allows to store any null in java.
Data structure internally used for storing data	For storing elements HashSet internally uses HashMap.	For storing elements LinkedHashSet internally uses LinkedHashMap.	For storing elements TreeSet internally uses TreeMap.
Introduced in which java version	JDK 2.0	JDK 4.0	JDK 2.0
Implements which interface	HashSet implements java.util.Set interface.	LinkedHashSet implements java.util.Set interface.	TreeSet implements java.util.Set java.util.SortedSet Java.util.NavigableSet interface.

98. Difference between java.util.HashMap and java.util.concurrent.ConcurrentHashMap in java:-

java.util.HashMap	java.util.concurrent.ConcurrentHashMap
HashMap is not synchronized.	ConcurrentHashMap is synchronized.
Synchronized HashMap's performance is slower as compared to ConcurrentHashMap.	ConcurrentHashMap performance is faster as compared to HashMap
HashMap allows to store one null key and many null values	ConcurrentHashMap does not allow to store null key or null value.
The iterators returned by the iterator() method of HashMap are fail-fast	Iterators are fail-safe.

JDK 1.2	JDK 1.5
HashMap implements java.util.Map	ConcurrentHashMap implements java.util.Map and java.util.concurrent.ConcurrentMap

98. Difference between java.util.HashMap vs Hashtable vs LinkedHashMap vs TreeMap in java:-

Property	HashMap	Hashtable	LinkedHashMap	TreeMap
Insertion order	HashMap does not maintains insertion order in java.	Hashtable does not maintains insertion order in java.	LinkedHashMap maintains insertion order in java.	TreeMap is sorted by natural order of keys in java.
Performance	HashMap is not synchronized, hence its operations are faster as compared to Hashtable.	Hashtable is synchronized, hence its operations are slower as compared HashMap.	Time and space overhead is there because for maintaining order it internally uses Doubly Linked list.	TreeMap must be used only when we want sorting based on natural order. Otherwise sorting operations cost performance.
Null keys and values	HashMap allows to store one null key and many null values	Hashtable does not allow to store null key or null value.	LinkedHashMap allows to store one null key and many null values	TreeMap does not allow to store null key but allow many null values.
Implements which interface	java.util.Map	java.util.Map	java.util.Map	java.util.Map java.util.SortedMap java.util.NavigableMap
Complexity of put, get and remove methods	O(1)	O(1)	O(1)	O(log(n))
Extends java.util.Dictionary	Doesn't extends Dictionary.	extends Dictionary	Doesn't extends Dictionary.	Doesn't extends Dictionary.
Introduced in which java version?	JDK 2.0	JDK 1.0	JDK 4.0	JDK 2.0
Implementation uses?	HashMap use buckets	Hashtable use buckets	LinkedHashMap uses doubly linked lists	TreeMap uses Red black tree

99. Difference between java.util.HashMap vs IdentityHashMap:-

Property	java.util.HashMap	java.util.IdentityHashMap
Keys comparison object-equality vs reference-equality	HashMap when comparing keys (and values) performs object-equality not reference-equality.	IdentityHashMap when comparing keys (and values) performs reference-equality in place of object-equality.
Initial size	Constructs a new HashMap, Its initial capacity is 16 in java.	Constructs a new IdentityHashMap, with maximum size of 21 in java.
Introduced in which java version	JDK 2.0	JDK 4.0
Program	In a HashMap, two keys k1 and k2 are equal if and only if (k1==null? k2==null: k1.equals (k2)).	In an IdentityHashMap, two keys k1 and k2 are equal if and only if (k1==k2).
Overridden equals () and hashCode () method call?	Overridden equals () and hashCode () method are called when put, get methods are called in HashMap.	Overridden equals () and hashCode () method are not called when put, get methods are called in IdentityHashMap.
Application - can maintain proxy object	HashMap cannot be used to maintain proxy object.	IdentityHashMap can be used to maintain proxy objects.

100. Difference between Collection and Collections in java?

Collection: - java.util.Collection is the root interface in the hierarchy of Java Collection framework. The JDK does not provide any classes which directly implements java.util.Collection interface, but it provides classes such as ArrayList, LinkedList, vector, HashSet, EnumSet, LinkedHashSet, TreeSet, CopyOnWriteArrayList, CopyOnWriteArraySet, ConcurrentSkipListSet which implements more specific sub interfaces like Set and List in java.

Collections: - java.util.Collections is a utility class which consists of static methods that operate on or return Collection in java.

Java.util.Collections provides method like:-

- **reverse** method for reversing List in java.
- **shuffle** method for shuffling elements of List in java.
- **UnmodifiableCollection**, **unmodifiableSet**, **unmodifiableList**, **unmodifiableMap** methods for making List, Set and Map unmodifiable in java.
- **min** method to return smallest element in Collection in java.

- **max** method to return smallest element in Collection.
- **sort** method for sorting List.
- **synchronizedCollection**, **synchronizedSet**, **synchronizedList**, **synchronizedMap** methods for synchronizing List, Set and Map respectively in java.

101. Difference between java.util.Iterator and java.util.ListIterator

java.util.ListIterator	java.util.Iterator
hasPrevious () method returns true if this listIterator has more elements when traversing the list in the reverse direction.	No such method in java.util.Iterator.
previous () returns previous element in iteration (traversing in backward direction). if the iteration has no previous elements than NoSuchElementException is thrown.	No such method in java.util.Iterator.
nextIndex () method returns the index of the element that would be returned by a subsequent call to next() method. If listIterator is at the end of the list than method returns size of list.	No such method in java.util.Iterator.
previousIndex () method returns the index of the element that would be returned by a subsequent call to previous() method. If listIterator is at the start of the list than method returns -1.	No such method in java.util.Iterator.
add (E element) Method inserts the specified element into the list. The element is inserted immediately before the element that would be returned by next	No such method in java.util.Iterator.
set(E element) Method replaces the last element returned by next() or previous() method with the specified element. This call can be made only if neither remove nor add have been called after the last call to next or previous.	No such method in java.util.Iterator.
All the implementations of List interface like **ArrayList, LinkedList, Vector, CopyOnWriteArrayList** classes returns listIterator.	All Implementation classes of **Collection interface's** sub interfaces like **Set and List** return iterator.

102. Difference between java.util.ArrayList and java.util.concurrent.CopyOnWriteArrayList in java

Property	java.util.ArrayList	java.util.concurrent.CopyOnWriteArrayList
synchronization	ArrayList is not synchronized	CopyOnWriteArrayList is synchronized
Iterator and listIterator	Iterator and listIterator returned by ArrayList are Fail-fast, means any structural modification made to ArrayList during iteration using Iterator or listIterator will throw ConcurrentModificationException in java.	Iterator and listIterator returned by CopyOnWriteArrayList are Fail-safe in java.
Enumeration is fail-fast	Enumeration returned by ArrayList is fail-fast, means any structural modification made to ArrayList during iteration using Enumeration will throw ConcurrentModificationException.	Enumeration returned by CopyOnWriteArrayList is fail-safe.
Iterate using enhanced for loop	Iteration done on ArrayList using enhanced for loop is Fail-fast, means any structural modification made to ArrayList during iteration using enhanced for loop will throw ConcurrentModificationException.	Iteration done on CopyOnWriteArrayList using enhanced for loop is Fail-safe.
Performance	ArrayList is not synchronized, hence its operations are faster as compared to CopyOnWriteArrayList.	CopyOnWriteArrayList is synchronized, hence its operations are slower as compared to ArrayList.
AbstractList	ArrayList extends AbstractList (abstract class) which provides implementation to List interface to minimize the effort required to implement this interface backed by RandomAccess interface.	CopyOnWriteArrayList does not extends AbstractList, though CopyOnWriteArrayList also implements RandomAccess interface.
java version	JDK 2.0	JDK 5.0
Package	java.util	java.util.concurrent

103. Difference between java.util.HashSet and java.util.concurrent.CopyOnWriteArraySet in java:-

Property	java.util.HashSet	java.util.concurrent.CopyOnWriteArraySet
synchronization	HashSet is not synchronized	CopyOnWriteArraySet is synchronized
Iterator	Iterator returned by HashSet is Fail-fast, means any structural modification made to HashSet during iteration using Iterator will throw ConcurrentModificationException in java.	Iterator returned by CopyOnWriteArraySet is Fail-safe in java.
Enumeration is fail-fast	Enumeration returned by HashSet is fail-fast, means any structural modification made to HashSet during iteration	Enumeration returned by CopyOnWriteArraySet is fail-safe.

	using Enumeration will throw ConcurrentModificationException.	
Iterate using enhanced for loop	Iteration done on HashSet using enhanced for loop is Fail-fast, means any structural modification made to HashSet during iteration using enhanced for loop will throw ConcurrentModificationException.	Iteration done on CopyOnWriteArraySet using enhanced for loop is Fail-safe.
Performance	HashSet is not synchronized, hence its operations are faster as compared to CopyOnWriteArraySet.	CopyOnWriteArraySet is synchronized, hence its operations are slower as compared to HashSet.
java version	JDK 2.0	JDK 5.0
Package	java.util	java.util.concurrent

104. Difference between java.util.TreeSet and java.util.concurrent.ConcurrentSkipListSet in java:-

Property	java.util.TreeSet	java.util.concurrent.ConcurrentSkipListSet
synchronization	TreeSet is not synchronized	ConcurrentSkipListSet is synchronized
Iterator	Iterator returned by TreeSet is Fail-fast, means any structural modification made to TreeSet during iteration using Iterator will throw ConcurrentModificationException in java.	Iterator returned by ConcurrentSkipListSet is Fail-safe in java.
Enumeration is fail-fast	Iteration done on TreeSet using enhanced for loop is Fail-fast, means any structural modification made to TreeSet during iteration using enhanced for loop will throw ConcurrentModificationException	Iteration done on ConcurrentSkipListSet using enhanced for loop is Fail-safe.
Performance	TreeSet is not synchronized, hence its operations are faster as compared to ConcurrentSkipListSet.	ConcurrentSkipListSet is synchronized, hence its operations are slower as compared to TreeSet.
java version	JDK 2.0	JDK 6.0
Package	java.util	java.util.concurrent

105. Difference between java.util.TreeMap and java.util.concurrent.ConcurrentSkipListMap:-

Property	java.util.TreeMap	java.util.concurrent.ConcurrentSkipListMap
synchronization	TreeMap is not synchronized	ConcurrentSkipListMap is synchronized
Iterator	The iterators returned by the iterator() method of Map's “collection view methods" are fail-fast	The iterators returned by the iterator() method of Map's “collection view methods" are fail-safe
Implements which interface	Map SortedMap NavigableMap	Map SortedMap NavigableMap ConcurrentNavigableMap
Performance	TreeMap is not synchronized, hence its operations are faster as compared to ConcurrentSkipListMap.	ConcurrentSkipListMap is synchronized, hence its operations are slower as compared to TreeMap.
java version	JDK 2.0	JDK 6.0
Package	java.util	java.util.concurrent

106. What do you mean by fail-fast and fast-safe? What is ConcurrentModificationException?

Fail-fast: - Iterator returned by few Collection framework Classes are fail-fast, means any structural modification made to these classes during iteration will throw **ConcurrentModificationException**. Some important classes whose returned iterator is fail-fast:-

- ArrayList
- LinkedList
- vector
- HashSet

Fail-safe:- Iterator returned by few Collection framework Classes are fail-safe, means any structural modification made to these classes during iteration won't throw any Exception. Some important classes whose returned iterator is fail-safe:-

- CopyOnWriteArrayList
- CopyOnWriteArraySet
- ConcurrentSkipListSet

107. Features of finally:-

- Finally block is **always executed** irrespective of exception is thrown or not.
- Finally is **keyword** in java.
- Finally block is optional in java, we may use it or not.

108. Finally block is not executed in following scenarios:-

- Finally is not executed when System.exit is called.

- If in case JVM crashes because of some java.util.Error.

109. Program to show finally block is executed when exception is not thrown.

```
public class ExceptionTest {
   public static void main(String[] args) {
         try{
               int i=10/1;
         }catch(ArithmeticException e){
               System.out.println("ArithmeticException handled in catch block");
         }
         finally{
               System.out.println("finally block executed");
         }
         System.out.println("code after try-catch-finally block");
   }
}
/*OUTPUT
finally block executed
code after try-catch-finally block */
```

110. Program to show finally block is executed when exception is thrown, in this case catch and finally both blocks are executed.

```
public class ExceptionTest {
  public static void main(String[] args) {
        try{
              int i=10/0; //will throw ArithmeticException
        }catch(ArithmeticException e){
              System.out.println("ArithmeticException handled in catch block");
        }
        finally{
              System.out.println("finally block executed");
        }
      System.out.println("code after try-catch-finally block");
  }
}
/*OUTPUT
ArithmeticException handled in catch block
finally block executed
code after try-catch-finally block
*/
```

111. Program to show finally block is executed when exception is thrown and not handled properly, in this case catch blocks does not executes, finally block executes alone.

```
public class ExceptionTest {
  public static void main(String[] args) {
        try{
              int i=10/0; //will throw ArithmeticException
        }catch(IndexOutOfBoundsException e){
              System.out.println("IndexOutOfBoundsException handled in catch block");
        }
        finally{
              System.out.println("finally block executed");
        }
        System.out.println("code after try-catch-finally block");
  }
}
/*OUTPUT
finally block executed
Exception in thread "main" java.lang.ArithmeticException: / by zero
  at finally3.ExceptionTest.main(ExceptionTest.java:7)
*/
```

112. Program to show finally is not executed when System.exit is called.

```
public class ExceptionTest {
  public static void main(String[] args) {
        try{
              System.out.println("in try block");
              System.exit(0);
        }finally{
              System.out.println("finally block executed");
        }
  }
}
/*OUTPUT
in try block
*/
```

113. Difference between checked and unchecked exceptions:-

checked exception	unchecked exception
Checked exceptions are also known as compile Time exceptions.	Unchecked exceptions are also known as runtime exceptions.
Checked exceptions are those which need to be taken care at compile time.	Unchecked exceptions are those which need to be taken care at runtime.
We cannot proceed until we fix compilation issues which are most likely to happen in program, this helps us in avoiding runtime problems up to lot of extent.	Whenever runtime exception occurs execution of program is interrupted, but by handling these kind of exception we avoid such interruptions and end up giving some meaningful message to user.

`class UserException extends Exception {`
`  UserException(String s) {`
`        super(s);`
`  }`
`}`
By extending java.lang.**Exception**, we can create checked exception.
	`class UserException extends RuntimeException {`
	`  UserException(String s) {`
	`        super(s);`
	`  }`
	`}`
	By extending java.lang.**RuntimeException**, we can create unchecked exception.

For propagating checked exceptions method must throw exception by using throws keyword.	Unchecked exceptions are automatically propagated in java.
If superclass method throws/declare checked exception • overridden method of subclass **can** declare/**throw narrower** (subclass of) **checked exception** • overridden method of subclass **cannot** declare/**throw broader** (superclass of) **checked exception** • overridden method of subclass **can** declare/**throw any unchecked /RuntimeException**	If superclass method throws/declare unchecked • overridden method of subclass **can** declare/**throw any unchecked /RuntimeException (superclass or subclass)** • overridden method of subclass **cannot** declare/**throw any checked exception**
The class Exception and all its subclasses that are not also subclasses of RuntimeException are checked exceptions.	The class RuntimeException and all its subclasses are unchecked exceptions. Likewise, The class Error and all its subclasses are unchecked exceptions.
SQLException, IOException, ClassNotFoundException	NullPointerException, ArithmeticException ArrayIndexOutOfBoundsException.

114. What are exception handling keywords in java?

- **try -** Any exception occurring in try block is catched by catch block.
- **catch -** catch block is always followed by try block.
- **finally -**finally block can only exist if try or try-catch block is there, finally block can't be used alone in java.
- **throw -** throw is a keyword in java. Throw keyword allows us to throw checked or unchecked exception.
- **throws -** throws is written in method's definition to indicate that method can throw exception.

115. Difference between Exception and Error in java?

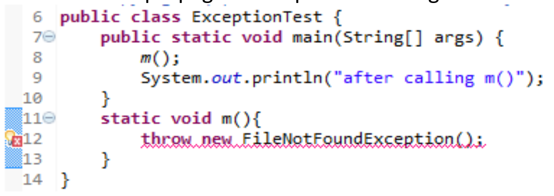
Exception	Error
Exception does not indicate any serious problem.	Error indicates some serious problems that our application should not try to catch.
Exception are divided into checked and unchecked exceptions	Error are not divided further into such classifications.
The class Exception and all its subclasses that are not also subclasses of RuntimeException are checked exceptions. The	Error and its subclasses are regarded as unchecked exceptions

class RuntimeException and all its subclasses are unchecked exceptions.	
checked exceptions:- SQLException, IOException, ClassNotFoundException **unchecked exceptions:-** NullPointerException, ArithmeticException,	VirtualMachineError, IOError, AssertionError, ThreadDeath, OutOfMemoryError, StackOverflowError.
Application must catch the Exception because they does not cause any major threat to application.	Application must not catch the Error because they does cause any major threat to application.

116. Difference between throw and throws in java?

throw	throws
Throw keyword is used to throw an exception explicitly.	Throws keyword is used to declare an exception.

Throw is used inside method.
Example :- static void m(){
throw new FileNotFoundException();
}
	Throws is used in method declaration.
	Example :- static void m() throws FileNotFoundException{
	}

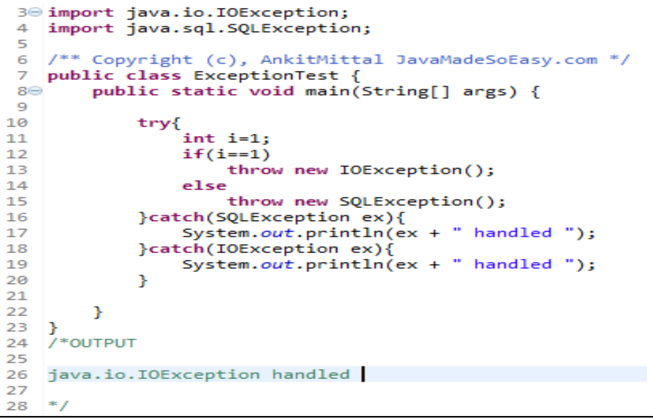
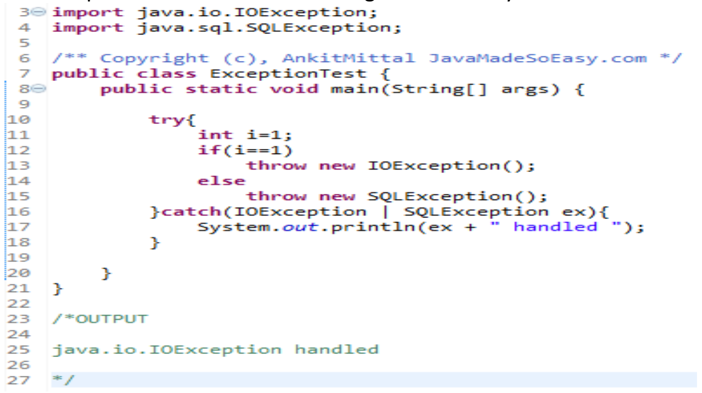
Throw is always followed by instanceof Exception class. Example :- throw new FileNotFoundException()	Throws is always followed by name of Exception class. Example :- throws FileNotFoundException
Throw can be used to throw only one exception at time. **Example :-** throw new FileNotFoundException()	Throws can be used to throw multiple exception at time. **Example :-** throws FileNotFoundException, NullPointerException
Throw cannot propagate exception to calling method. 	Throws can propagate exception to calling method.

```java
public class ExceptionTest {
    public static void main(String[] args) {
        m();
        System.out.println("after calling m()");
    }
    static void m(){
        throw new FileNotFoundException();
    }
}
```

117. Difference between multiple catch block and multi catch syntax:-

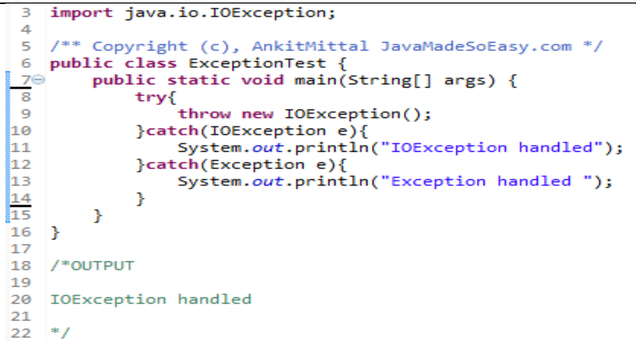
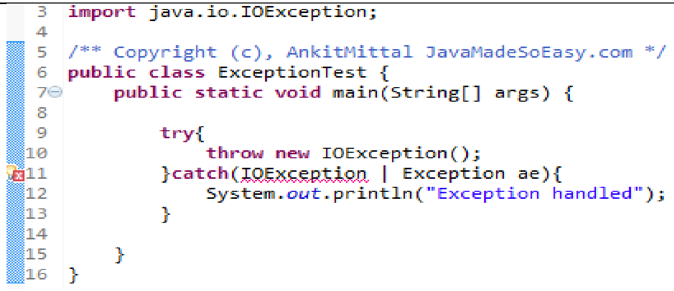
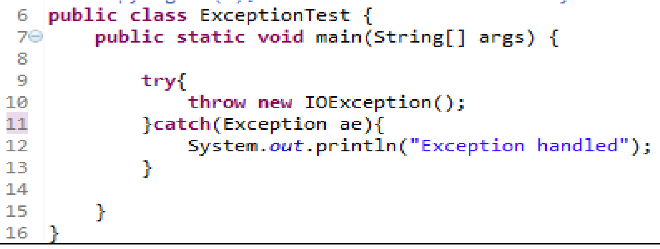
multiple catch block	multi catch syntax
Multiple catch blocks were introduced in prior versions of Java 7 and does not provide any automatic resource management.	Multi catch syntax was introduced in java 7 for improvements in multiple exception handling which helps in automatic resource management.

syntax for writing multiple catch block
try{
}catch(IOException ex1){
} catch(SQLException ex2){
}
	multi catch syntax
	try{
	}catch(IOException | SQLException ex){
	}

For catching IOException and SQLException we need to write two catch block like this 	with the help of multi catch syntax we can catch IOException and SQLException in one catch block using multi catch syntax like this 
When multiple catch blocks are used , first catch block could be subclass of Exception class handled in following catch blocks like this > IOException is subclass of Exception.	If **Multi catch syntax** is used to catch subclass and its superclass than compilation error will be thrown. IOException and Exception in **multi catch syntax** will cause compilation error “The exception **IOException** is already caught by the alternative **Exception**”.

```java
import java.io.IOException;
import java.sql.SQLException;

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class ExceptionTest {
    public static void main(String[] args) {

        try{
            int i=1;
            if(i==1)
                throw new IOException();
            else
                throw new SQLException();
        }catch(SQLException ex){
            System.out.println(ex + " handled ");
        }catch(IOException ex){
            System.out.println(ex + " handled ");
        }

    }
}
/*OUTPUT

java.io.IOException handled

*/
```

```java
import java.io.IOException;
import java.sql.SQLException;

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class ExceptionTest {
    public static void main(String[] args) {

        try{
            int i=1;
            if(i==1)
                throw new IOException();
            else
                throw new SQLException();
        }catch(IOException | SQLException ex){
            System.out.println(ex + " handled ");
        }

    }
}

/*OUTPUT

java.io.IOException handled

*/
```

 `import java.io.IOException;`
 	
`/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */`
`public class ExceptionTest {`
`    public static void main(String[] args) {`
`        try{`
`            throw new IOException();`
`        }catch(IOException e){`
`            System.out.println("IOException handled");`
`        }catch(Exception e){`
`            System.out.println("Exception handled ");`
`        }`
`    }`
`}`
`/*OUTPUT`
 	
`IOException handled`
 	
`*/`
	 `import java.io.IOException;`
	
	`/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */`
	`public class ExceptionTest {`
	`    public static void main(String[] args) {`
	
	`        try{`
	`            throw new IOException();`
	`        }catch(IOException | Exception ae){`
	`            System.out.println("Exception handled");`
	`        }`
	
	`    }`
	`}`
	Solution We must use only Exception to catch its subclass like this
	 `public class ExceptionTest {`
	`    public static void main(String[] args) {`
	
	`        try{`
	`            throw new IOException();`
	`        }catch(Exception ae){`
	`            System.out.println("Exception handled");`
	`        }`
	
	`    }`
	`}`

Does not provide such features.	Features of multi catch syntax :- • Has improved way of catching multiple exceptions. • This syntax does not looks clumsy. • Reduces developer efforts of writing multiple catch blocks. • Allows us to catch more than one exception in one catch block. • Helps in automatic resource management.

118. Difference between Final, Finally and Finalize:-

Final	Finally	Finalize
Final can be applied to variable, method and class in java.	Finally is a block.	Finalize is a method.
Final is a keyword in java.	Finally Is a keyword in java.	Finalize is not a keyword in java.

Final memberVariable of class must be initialized at time of declaration, once initialized final memberVariable cannot be assigned a new value.
Final method cannot be overridden, any attempt to do so will cause compilation error.
Final class cannot be extended, any attempt to do so will cause compilation error.
	try or try-catch block can be followed by finally block
	try-finally block,
	try{
	}finally{
	}
	
	Try-catch-finally block.
	try{
	}catch(Exception e){
	}finally{
	}
		Finalize method is called before garbage collection by JVM, Finalize method is called for any cleanup action that may be required before garbage collection.
		@Override
		protected void finalize() throws Throwable {
		try {
		
		System.*out*.println("in finalize() method, "
		+
		"doing cleanup activity");
		
		} **catch** (Throwable throwable) {
		throw throwable;
		}
		}

	finally block is not executed in following scenarios :- **Finally is not executed when System.exit is called.** **If in case JVM crashes because of some java.util.Error.**	finalize() method is defined in java.lang.Object
	Finally block can only exist if try or try-catch block is there, finally block can't be used alone in java.	We can force early garbage collection in java by using following methods :- **System.*gc*(); Runtime.*getRuntime*().gc();**

		System.*runFinalization*(); **Runtime.*getRuntime*().runFinalization();**
	Finally is always executed irrespective of exception thrown.	If any uncaught exception is thrown inside finalize method - **exception is ignored,** **thread is terminated and** **Object is discarded.**
	Currently executing thread calls finally method.	JVM does not guarantee which daemon thread will invoke the finalize method for an object.

120. What is cloning in java? Cloning is done for copying the object, cloning can be done using shallow or deep copy. Clone is a protected method - clone method can't be called outside class without inheritance. Clone is native method, if not overridden its implementation is provided by JVM. It returns Object - Means explicitly cast is needed to convert it to original object. Class must implement marker interface java.lang.Cloneable. If class doesn't implement Cloneable than calling clone method on its object will throw CloneNotSupportedException. **By default clone method do shallow copy.**

```
protected native Object clone() throws CloneNotSupportedException;
```

Shallow copy- If we implement Cloneable interface, we must override clone method and call super.clone () from it, invoking super.clone () will do shallow copy.
Deep copy - We need to provide custom implementation of clone method for deep copying. When the copied object contains some other object its references are copied recursively in deep copy.

121. How garbage collector knows that the object is not in use and needs to be removed? Garbage collector reclaims objects that are no longer being used, clears their memory, and keeps the memory available for future allocations. This is done via bookkeeping the references to the objects. Any unreferenced object is a garbage and will be collected.

122. Can Java thread object invoke start method twice?

```
public class MyExmpCode extends Thread{
        public void run(){
                System.out.println("Run");
        }
        public static void main(String a[]){
                Thread t1 = new Thread(new MyExmpCode());
                t1.start();
                t1.start();
        }
}       No, it throws IllegalThreadStateException
```

123. Give the list of Java Object class methods.

- **clone () -** Creates and returns a copy of this object.
- **equals () -** Indicates whether some other object is "equal to" this one.
- **finalize () -** Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
- **getClass () -** Returns the runtime class of an object.
- **hashCode () -** Returns a hash code value for the object.
- **notify () -** Wakes up a single thread that is waiting on this object's monitor.
- **notifyAll () -** Wakes up all threads that are waiting on this object's monitor.
- **toString () -** Returns a string representation of the object.
- **wait () -** Causes current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.

124. Can we call servlet destroy () from service ()? As you know, destroy () is part of servlet life cycle methods, it is used to kill the servlet instance. Servlet Engine is used to call destroy (). In case, if you call destroy method from service (), it just execute the code written in the destroy (), but it won't kill the servlet instance. Destroy () will be called before killing the servlet instance by servlet engine.

125. Can we override static method? We cannot override static methods. Static methods are belongs to class, not belongs to object. Inheritance will not be applicable for class members

126. Can you list serialization methods? Serialization interface does not have any methods. It is a marker interface. It just tells that your class can be Serializable.

127. What is the difference between super () and this ()? Super () is used to call super class constructor, whereas this () used to call constructors in the same class, means to call parameterized constructors.

128. How to prevent a method from being overridden? By specifying final keyword to the method you can avoid overriding in a subclass. Similarly one can use final at class level to prevent creating subclasses.

129. Can we create abstract classes without any abstract methods? Yes, we can create abstract classes without any abstract methods.

130. How to destroy the session in servlets? By calling invalidate () method on session object, we can destroy the session.

131. What is transient variable? Transient variables cannot be serialized. During serialization process,
Transient variable states will not be serialized. State of the value will be always defaulted after deserialization.

132. In case, there is a return at the end of try block, will execute finally block? Yes, the finally block will be executed even after writing return statement at the end of try block. It returns after executing finally block.

133. What is default value of a Boolean? False

134. What is daemon thread? Daemon thread is a low priority thread. It runs intermittently in the back ground, and takes care of the garbage collection operation for the java runtime system. By calling setDaemon () method is used to create a daemon thread.

135. Does each thread in java uses separate stack? In Java every thread maintains its own separate stack. It is
Called Runtime Stack but they share the same memory.

136. Find out below switch statement output.

```
public static void main(String a[]){
        int price = 6;
        switch (price) {
                case 2: System.out.println("It is: 2");
                    ☑   default: System.out.println("It is: default");
                    ☑   case 5: System.out.println("It is: 5");
                    ☑   case 9: System.out.println("It is: 9");
        }
}
```

137. Does System.exit () in try block executes code in finally block?

```
try{
                System.out.println("I am in try block");
                System.exit(1);
        } catch(Exception ex){
                ex.printStackTrace();
        } finally {
                System.out.println ("I am in finally block!!!");
        }
```

It will not execute finally block. The program will be terminated after System.exit () statement.

139. In java, are true and false keywords? true, false, and null might seem like keywords, but they are actually
literals. You cannot use them as identifiers in your programs.

140. What are the different session tracking methods?

- **Cookies:** You can use HTTP cookies to store information. Cookies will be stored at browser side.
- **URL rewriting:** With this method, the information is carried through URL as request parameters. In general added parameter will be sessionid, userid.
- **HttpSession:** Using HttpSession, we can store information at server side. HttpSession provides methods to handle session related information.
- **Hidden form fields:** By using hidden form fields we can insert information in the webpages and these information will be sent to the server. These fields are not visible directly to the user, but can be viewed using view source option from the browsers. The hidden form fields are as given below:
    - <input type='hidden' name='siteName' value='java2novice'/>

141. What are the types of ResultSet? The type of a ResultSet object determines the level of its functionality in
Two areas: the ways in which the cursor can be manipulated, and **how concurrent changes made to the underlying data source are reflected by the ResultSet object.** The sensitivity of a ResultSet object is determined by one of three different ResultSet types: **The default ResultSet type is TYPE_FORWARD_ONLY.**

TYPE_FORWARD_ONLY: The result set cannot be scrolled; its cursor moves forward only, from before the first row to after the last row. The rows contained in the result set depend on how the underlying database generates the results. That is, it contains the rows that satisfy the query at either the time the query is executed or as the rows are retrieved.

TYPE_SCROLL_INSENSITIVE: The result can be scrolled; its cursor can move both forward and backward relative to the current position, and it can move to an absolute position. The result set is insensitive to changes made to the underlying data source while it is open. It contains the rows that satisfy the query at either the time the query is executed or as the rows are retrieved.

TYPE_SCROLL_SENSITIVE: The result can be scrolled; its cursor can move both forward and backward relative to the current position, and it can move to an absolute position. The result set reflects changes made to the underlying data source while the result set remains open.

142. What is difference between wait and sleep methods in java?

sleep (): It is a static method on Thread class. It makes the current thread into the "Not Runnable" state for specified amount of time. During this time, the thread keeps the lock (monitors) it has acquired.

wait (): It is a method on Object class. It makes the current thread into the "Not Runnable" state. Wait is called on an object, not a thread. Before calling wait () method, the object should be synchronized, means the object should be inside synchronized block. The call to wait () releases the acquired lock.

143. What is servlet context? The servlet context is an interface which helps to communicate with other servlets. It contains information about the Web application and container. It is kind of application environment. Using the context, a servlet can obtain URL references to resources, and store attributes that other servlets in the context can use.

144. What happens if one of the members in a class does not implement Serializable interface? When you try to serialize an object which implements Serializable interface, incase if the object includes a reference of a non Serializable object then NotSerializableException will be thrown.

145. What is race condition? A race condition is a situation in which two or more threads or processes are reading or writing some shared data, and the final result depends on the timing of how the threads are scheduled. Race conditions can lead to unpredictable results and subtle program bugs. A thread can prevent this from happening by locking an object. When an object is locked by one thread and another thread tries to call a synchronized method on the same object, the second thread will block until the object is unlocked.

146. How to get current time in milli seconds? System.currentTimeMillis () returns the current time in milliseconds.
It is a static method, returns long type.

147. How can you convert Map to List?

```
public static void main(String a[]){
        Map<String, String> wordMap = new HashMap<String, String>();
        Set<Entry<String, String>> set = wordMap.entrySet();
        List<Entry<String, String>> list = new ArrayList<Entry<String, String>>(set);
}
```

148. What is strictfp keyword? By using strictfp keyword, we can ensure that floating point operations take place precisely.

149. What is System.out in Java? Here out is an instance of PrintStream. It is a static member variable in System class. This is called standard output stream, connected to console.

150. What is difference between ServletOutputStream and PrintWriter?

ServletOutputStream: ServletResponse.getOutputStream () returns a ServletOutputStream suitable for writing binary data in the response. The servlet container does not encode the binary data, it sends the raw data as it is.

PrintWriter: ServletResponse.getWriter () returns PrintWriter object which sends character text to the client. The PrintWriter uses the character encoding returned by getCharacterEncoding (). If the response's character encoding has not been specified then it does default character encoding.

151. What is java static import? By using static imports, we can import the static members from a class rather than the classes from a given package. For example, Thread class has static sleep method, below example gives an idea:

```
import static java.lang.Thread;
public class MyStaticImportTest {
        public static void main(String[] a) {
                try{
                        sleep(100);
                } catch(Exception ex){

                }
        }
}
```

152. When to use String and StringBuffer? We know that String is immutable object. We cannot change the value
Of a String object once it is initiated. If we try to change the value of the existing String object then it creates new object rather than changing the value of the existing object. So in case, we are going to do more modifications on String, then use StringBuffer. StringBuffer updates the existing objects value, rather creating new object.

154. What is wrapper class in java? Everything in java is an object, except primitives. Primitives are int, short, long, boolean, etc. Since they are not objects, they cannot return as objects, and collection of objects. To support this, java provides wrapper classes to move primitives to objects. Some of the wrapper classes are Integer, Long, Boolean, etc.

155. Is Iterator a Class? Iterator is an interface. It is not a class. It is used to iterate through each and every element in a list. Iterator is implemented Iterator design pattern.

156. What is java classpath? The classpath is an environment variable. It is used to let the compiler know where the class files are available for import.
157. Can a class in java be private? We cannot declare top level class as private. Java allows only public and default modifier for top level classes in java. Inner classes can be private.
159. What is the initial state of a thread when it is started? When the thread is created and started, initially it will be in the ready state.
160. What is the super class for Exception and Error? The super class or base class for Exception and Error is Throwable.
161. What is Class.forName ()? Class.forName () loads the class into the ClassLoader.
162. Can interface be final? No. We cannot instantiate interfaces, so in order to make interfaces useful we must create subclasses. The final keyword makes a class unable to be extended.
163. What is default value of a local variables? The local variables are not initialized to any default values. We should not use local variables without initialization. Even the java compiler throws error.
164. What is local class in java? In java, local classes can be defined in a block as in a method body or local block.
165. Can we initialize uninitialized final variable? Yes. We can initialize blank final variable in constructor, only in constructor. The condition here is the final variable should be non-static.
166. Can we declare abstract method as final? No, we cannot declare abstract method as final. We have to
Proved implementation to abstract methods in subclasses.
167. Can we have finally block without catch block? Yes, we can have finally block without catch block.
168. What is pass by value and pass by reference?
Pass by value: Passing a copy of the value, not the original reference.
Pass by reference: Passing the address of the object, so that you can access the original object.
169. Can we declare main method as private? Yes, we can declare main method as private. It compiles without
any errors, but in runtime, it says main method is not public.
170. What is the difference between preemptive scheduling and time slicing?
Preemptive scheduling: The highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence.
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.
171. Can non-static member classes (Local classes) have static members? No, non-static member classes cannot have static members. Because, an instance of a non-static member class or local class must be created in the context of an instance of the enclosing class. You can declare constants, means static final variables.
172. What are the environment variables do we need to set to run Java? We need to set two environment variables those are PATH and CLASSPATH.
173. Can you serialize static fields of a class? Since static fields are not part of object state, they are part of class, serialization ignores the static fields.
174. What is the difference between declaring a variable and defining a variable? When variable declaration we just mention the type of the variable and its name, it does not have any reference to live object. But defining means combination of declaration and initialization. The examples are as given below:
Declaration: List list;
Defining: List list = new ArrayList ();
175. Where can we use serialization? Whenever an object has to send over the network, those objects should be serialized. Also if the state of an object is to be saved, objects need to be serialized.
176. What modifiers are allowed for methods in an Interface? Only public and abstract modifiers are allowed for methods in an interfaces.
177. What is the purpose of Runtime and System class? **The purpose of the Runtime class** is to provide access to the Java runtime system. The runtime information like memory availability, invoking the garbage collector, etc. **The purpose of the System class** is to provide access to system resources. It contains accessibility to standard input, standard output, error output streams, current time in millis, terminating the application, etc.
178. Which one is faster? ArrayList or Vector? Why? ArrayList is faster than Vector. The reason is synchronization. Vector is synchronized. As we know synchronization reduces the performance.
179. What is the difference between static synchronized and synchronized methods? **Static synchronized** methods synchronize on the class object. If one thread is executing a static synchronized method, all other threads trying to execute any static synchronized methods will be blocked. **Non-static synchronized methods** synchronize on this ie the instance of the class. If one thread is executing a synchronized method, all other threads trying to execute any synchronized methods will be blocked.
180. What is the order of catch blocks when catching more than one exception? When you are handling multiple catch blocks, make sure that you are specifying exception sub classes first, then followed by exception super classes. Otherwise we will get compile time error.

181. What is the difference between the prefix and postfix forms of the increment (++) operator? **The prefix form** first performs the increment operation and then returns the value of the increment operation. **The postfix form** first returns the current value of the expression and then performs the increment operation on that value. For example:
int count=1; System.out.println (++count); displays 2.
int count=1; System.out.println(count++); displays 1.
182. What is hashCode? The hashcode of a Java Object is simply a number, it is 32-bit signed int that allows an object to be managed by a hash-based data structure. We know that hash code is a unique id number allocated to an object by JVM. If two objects are equals then these two objects should return same hash code. So we have to implement hashcode () method of a class in such way that if two objects are equals, ie compared by equal () method of that class, then those two objects must return same hash code. If you are overriding hashCode you need to override equals method also.
183. What are the restrictions when overriding a method? Overriding methods must have the same name, parameter list, and same return type. i.e., they must have the exact signature of the method we are going to override, including return type. The overriding method cannot be less visible than the method it overrides. i.e., a public method cannot be override to private. The overriding method may not throw any exceptions that may not be thrown by the overridden method.
184. What is the use of assert keyword? Java assertion feature allows developer to put assert statements in Java source code to help unit testing and debugging. Assert keyword validates certain expressions. It replaces the if block effectively and throws an AssertionError on failure.
185. What is adapter class? An adapter class provides the default implementation of all methods in an event listener interface. Adapter classes are very useful when you want to process only few of the events that are handled by a particular event listener interface. You can define a new class by extending one of the adapter classes and implement only those events relevant to you.
186. What is difference between break, continue and return statements? **The break statement** results in the termination of the loop, it will come out of the loop and stops further iterations. **The continue statement** stops the current execution of the iteration and proceeds to the next iteration. **The return statement** takes you out of the method. It stops executing the method and returns from the method execution.
187. What is the difference between while and do-while statements? **The while statement** verifies the condition before entering into the loop to see whether the next loop iteration should occur or not. **The do-while statement** executes the first iteration without checking the condition, it verifies the condition after finishing each iteration. The do-while statement will always execute the body of a loop at least once.
188. When does the compiler provides the default constructor? The compiler provides a default constructor if no other constructors are available in the class. In case the class contains parameterized constructors, compiler doesn't provide the default constructor.
189. What are the advantages of java package? Java packages helps to resolve naming conflicts when different packages have classes with the same names. This also helps you organize files within your project. For example, java.io package do something related to I/O and java.net package do something to do with network and so on. If we tend to put all .java files into a single package, as the project gets bigger, then it would become a nightmare to manage all your files.
190. What is dynamic class loading? Dynamic loading is a technique for programmatically invoking the functions of a class loader at run time. Let us look at how to load classes dynamically by using **Class.forName (String className);** method, it is a static method. The above static method returns the class object associated with the class name. The string className can be supplied dynamically at run time. Once the class is dynamically loaded the **class.newInstance ()** method returns an instance of the loaded class. It is just like creating a class object with no arguments. A **ClassNotFoundException** is thrown when an application tries to load in a class through its class name, but no definition for the class with the specified name could be found.
191. What happens if you do not provide a constructor? Java does not actually require an explicit constructor in the class description. If you do not include a constructor, the Java compiler will create a default constructor in the byte code with an empty argument.
192. Difference between shallow cloning and deep cloning of objects?

Shallow Copy	Deep Copy
Cloned Object and original object are not 100% disjoint.	Cloned Object and original object are 100% disjoint.
Any changes made to cloned object will be reflected in original object or vice versa.	Any changes made to cloned object will not be reflected in original object or vice versa.
Default version of clone method creates the shallow copy of an object.	To create the deep copy of an object, you have to override clone method.
Shallow copy is preferred if an object has only primitive fields.	Deep copy is preferred if an object has references to other objects as fields.

Shallow copy is fast and also less expensive.	Deep copy is slow and very expensive.

193. Can we have interfaces with no defined methods in java? The interfaces with no defined methods act like markers. They just tell the compiler that the objects of the classes implementing the interfaces with no defined methods need to be treated differently. Marker interfaces are also known as “tag” interfaces.

194. What is the difference between “==” and equals () method? **The == (double equals)** returns true, if the variable reference points to the same object in memory. This is called “shallow comparison”. **The equals() method** calls the user implemented equals() method, which compares the object attribute values. **The equals() method** provides “deep comparison” by checking if two objects are logically equal as opposed to the shallow comparison provided by the operator ==. **If equals() method** does not exist in a user supplied class then the inherited Object class's equals() method will be called which evaluates if the references point to the same object in memory. In this case, the object.equals () works just like the "==" operator.

196. Can we have private constructor in java? Private constructor is used if you do not want other classes to instantiate the object. Private constructors are used in singleton design pattern, factory method design pattern.

197. Why do we need generics in java?

1) Stronger type checks at compile time: A Java compiler applies strong type checking to generic code and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find.

2) Elimination of casts: If you use generics, then explicit type casting is not required.

3) Enabling programmers to implement generic algorithms: By using generics, programmers can implement generic algorithms that work on collections of different types, can be customized, and are type safe and easier to read.

198. What is the difference between a product and a project?

A project is an endeavor with a clear definition of what needs to be delivered and the date when it needs to be delivered. Actually, it may seem that a product is a project, but it is not. Because, there is no clear definition of what needs to be delivered and there is no clear definition of the date when it needs to be delivered.

The product backlog is a collection of all possible ideas and additions to a product. The stories in the product backlog have no particular priority or schedule, although you can sort and organize them in hierarchies in the breakdown view by drag & drop. Thus, if you wish to prioritize your stories but don’t want to create projects or iterations, just use the breakdown view!

199. How does substring () method works on a string? String in java is a sequence of characters. String is more like a utility class which works on that character sequence. This character sequence is maintained as an array called value [], for example: - **private final char value [];** String internally defines two private variables called offset and count to manage the char array. The declarations can be as shown below:

/ the offset is the first index of the storage that is used. */ private final int offset;**

/ the count is the number of characters in the String. */ private final int count;**

Every time we create a substring from any string object, substring () method assigns the new values of offset and count variables. The internal char array is unchanged. This is a possible source of memory leak if substring () method is used without care.

200. What is the difference between a Java Library and a framework?

A library is a collection of class definitions and its implementations. The main benefits of creating library is simply code reuse. A simple example is one of the other developer written code for sending emails. If you think it is a generic module. Most of the places this code can be reusable. If we can make it a library (jar), we can include this library in our code, and call those methods. The classes and methods normally define specific operations in a domain specific area.

In framework, all the control flow is already defined, and there is a bunch of predefined places that you should fill out with your code. We use framework to developed applications. A framework defines a skeleton where the application defines its own features to fill out the skeleton. In this way, your code will be called by the framework when appropriately. The benefit is that developers do not need to worry about if a design is good or not, but just about implementing domain specific functions.

201. What is difference between Lambda Expression and Anonymous class? **The key difference between Anonymous class and Lambda expression** is the usage of 'this' keyword. In the anonymous classes, ‘this’ keyword resolves to anonymous class itself, whereas for lambda expression ‘this’ keyword resolves to enclosing class where lambda expression is written. **Another difference between lambda expression and anonymous class** is in the way these two are compiled. Java compiler compiles lambda expressions and convert them into private method of the class. It uses invoke dynamic instruction that was added in Java 7 to bind this method dynamically.

201. What is HTTP basic authentication? In the context of a HTTP transaction, basic access authentication is a method for an HTTP user agent to provide a user name and password when making a request.

HTTP Basic authentication implementation is the simplest technique for enforcing access controls to web resources because it doesn't require cookies, session identifier and login pages. Rather, HTTP Basic authentication uses static, standard HTTP headers which means that no handshakes have to be done in anticipation.

When the user agent wants to send the server authentication credentials it may use the Authorization header. The **Authorization header is constructed as follows:**

1) Username and password are combined into a string "username: password"
2) The resulting string is then encoded using Base64 encoding
3) The authorization method and a space i.e. "Basic" is then put before the encoded string.
For example, if the user agent uses 'Aladdin' as the username and 'open sesame' as the password then the header is formed as follows: **Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==**
202. What is functional interface in java? Functional Interface is an interface with just one abstract method declared in it. **Runnable interface is an example of a Functional Interface**. It has only run () method declared in it.
Lambda expression works on functional interfaces to replace anonymous classes.**@FunctionalInterface** is a new annotation added in Java 8 to indicate that an interface declaration is intended to be a functional interface as defined by the Java Language Specification. **@FunctionalInterface** can be used for compiler level errors when the interface you have annotated is not a valid Functional Interface.
203. What is the difference between HTTP methods GET and POST? When you use **GET method**, the data will be sent to the server as a query parameters. These are appended to the URL as a key value pair. In the below URL, you can see how data is passed to the server as key value pair. These values will be visible at the address bar. URL character length is limited, so you cannot use it if you are sending large data. GET is recommended to use for querying information from server, kind of search operations. GET requests should never be used when dealing with sensitive data.
http://java2novice.com/history?name=madhu&language=java
POST method sends data as part of HTTP message body, data sent to the server, will not be visible to the user. POST requests cannot be cached. It does not have any character length restrictions. POST is recommended to submit data to be processed to a specified resource.
204. What is difference between CountDownLatch and CyclicBarrier in Java? Both CyclicBarrier and CountDownLatch are used to implement a scenario where one Thread waits for one or more Thread to complete their job before starts processing. **The differences are:**
1) CyclicBarrier is reusable, CountDownLatch is not.
2) Both CyclicBarrier and CountDownLatch wait for fixed number of threads.
3) CountDownLatch is advance able but CyclicBarrier is not.
205. Can Enum extend any class in Java? Enum cannot extend any class in java, the reason is by default, Enum extends abstract base class java.lang.Enum. Since java does not support multiple inheritance for classes, Enum cannot extend another class.
206. Can Enum implements any interface in Java? Yes, Enum can implement any interface in Java. Since enum is a type, similar to any other class and interface, it can implement any interface in java. This gives lot of flexibility in some cases to use Enum to implement some other behavior.
207. Can we have constructor in abstract class? You cannot instantiate abstract class in java. In order to use abstract class in Java, You need to extend it and provide a concrete class. Abstract class is commonly used to define base class for a type hierarchy with default implementation, which is applicable to all child classes. Now based on these details, can we have constructor in abstract class? The answer is YES, we can have. You can either explicitly provide constructor to abstract class or if you don't, compiler will add default constructor of no argument in abstract class. This is true for all classes and it's also applies on abstract class.
208. What is MVC pattern? MVC is a design pattern called Model-View-Controller. It decouples data access logic from business logic.
Model: The Model contains the core of the application's functionality. It encapsulates the state of the application. Sometimes the only functionality it contains is state. It knows nothing about the view or controller.
View: The view provides the presentation of the model. It is the look and feel of the application. The view can access the model getters, but it has no knowledge of the setters. In addition, it knows nothing about the controller. The view should be notified when changes to the model occur.
Controller: The controller reacts to the user input. It creates and sets the model and helps to identify which view should be part of response.
209. What is the difference between Servlet and Filter? **A filter is an object** that can transform the header and content (or both) of a request or response. Filters differ from web components in that filters usually do not themselves create a response. Instead, a filter provides functionality that can be "attached" to any kind of web resource. Consequently, a filter should not have any dependencies on a web resource for which it is acting as a filter; this way it can be composed with more than one type of web resource.
The main tasks that a filter can perform are as follows:
1) Query the request and act accordingly.
2) Block the request-and-response pair from passing any further.
3) Modify the request headers and data. You do this by providing a customized version of the request.
4) Modify the response headers and data. You do this by providing a customized version of the response.
5) Interact with external resources.

Servlet is used for performing the action which needs to be taken for particular request like user login, get the response based on user role, interacts with database for getting the data, business logic execution, etc.

210. What is the difference between application server and web server? **Web Server** is designed to serve HTTP Content. **Application Server** can also serve HTTP Content but is not limited to just HTTP. It can be provided other protocol support such as RMI/RPC Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content. **Most of the application servers** have Web Server as integral part of them that means App Server can do whatever Web Server is capable of. Additionally Application Server have components and features to support Application level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services etc. **As web servers** are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while service a page request, static contents such as images/Static html is served by web server that interprets the request. Using some kind of filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server **Example of such configuration is Apache HTTP Server and BEA WebLogic Server. Apache HTTP Server is Web Server and BEA WebLogic is Application Server.**

211. What is the difference between JPA and Hibernate? JPA is just a specification which needs concrete implementation. The default implementation provided by oracle is "Eclipselink" now. Toplink is donated by Oracle to Eclipse foundation to merge with eclipselink. Using Eclipselink, one can be sure that the code is portable to any implementation if need arises. Hibernate is also a full JPA implementation + MORE. Hibernate is super set of JPA with some extra Hibernate specific functionality. So application developed in Hibernate may not be compatible when switched to other implementation. Still hibernate is choice of majority of developers as JPA implementation and widely used. Another JPA implementation is OpenJPA, which is an extension of Kodo implementation.

212. What is difference between the Value Object and JDO? **Java Data Objects (JDO)** is really a technology of persistence used to keep objects of Java in databases which provides the advantage to its developers by manipulating all details at applications level and helps developers to concentrate on coding that is not database specific. **The Value Objects represents** an abstracted design blueprint, which provides a generic holder of data known as Data transfer Object which can hold data so that it can be transferred between the customer and databases. JDO provides the advantage of persisting data, while the Value Object is in charge of keeping data provisionally during the period of data transfer only. In other words, the Value Object does not provide persistence.

214. What is Dependency Injection? When using Dependency Injection, objects are given their dependencies at run time rather than compile time (car manufacturing time). So that we can now change the Wheel whenever we want. This concept says that you do not create your objects but describe how they should be created. You don't directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container is then responsible for hooking it all up.

215. What are the different types of dependency injections in spring? Spring supports 2 types of dependency injection, they are:

1) Constructor-based dependency injection: It is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.

2) Setter-based dependency injection: It is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

216. What is BeanFactory in spring? A BeanFactory is like a factory class that contains a collection of beans. The BeanFactory holds Bean Definitions of multiple beans within itself and then instantiates the bean whenever asked for by clients. The BeanFactory is the actual container which instantiates, configures, and manages a number of beans. These beans typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the BeanFactory. BeanFactory also takes part in the life cycle of a bean, making calls to custom initialization and destruction methods.

218. How to make a bean as singleton in spring? Beans defined in spring framework are singleton beans. There is an attribute in bean tag called ‘singleton’. If it is specified as true then bean becomes singleton and if it sets to false then the bean becomes a prototype bean, it is non-singleton class. By default this property is set to true. So, all the beans in spring framework are by default singleton beans.

220. What are different types of spring auto-wiring modes? Autowiring is used to build relationships between the collaborating beans. Spring container can automatically resolve collaborators for beans. In spring framework, you can wire beans automatically with auto-wiring feature. To enable it, just define the “autowire” attribute in <bean>. In spring, 5 Auto-wiring modes are supported.

No: Default, no auto wiring, set it manually via “ref” attribute

ByName: Auto wiring by property name. If the name of a bean is same as the name of other bean property, auto wire it.

ByType: Auto wiring by property data type. If data type of a bean is compatible with the data type of other bean property, auto wire it.

Constructor: byType mode in constructor argument.

Autodetect: If a default constructor is found, use “autowired by constructor”; otherwise, use “autowire by type”.

221. What is the difference between ORM, JPA and Hibernate?

ORM: Object Relational Mapping is concept/process of converting the data from Object oriented language to relational DB and vice versa. For example in java it’s done with the help of reflection and jdbc.

Hibernate: It’s the implementation of above concept.

JPA: It’s the one step above ORM. Its high level API and specification so that different ORM tools can implement so that it provides the flexibility to developer to change the implementation from one ORM to another (for example if application uses the JPA api and implementation is hibernate. In future it can switch to IBatis if required. But on the other if application directly lock the implementation with Hibernate without JPA platform, switching is going to be herculean task)

222. What is stream pipelining in Java 8? Stream pipelining is the concept of chaining operations together. This is done by splitting the operations that can happen on a stream into two categories. They are **"intermediate operations"** and **"terminal operations".** Each intermediate operation returns an instance of Stream itself when it runs, an arbitrary number of intermediate operations can, therefore, be set up to process data forming a processing pipeline. There must then be a terminal operation which returns a final value and terminates the pipeline.

223. What is interface default method in java 8? All you need to do is add "default" keyword in front of the implementation method with in the interface.

224. Why do we need to implement a method within the interface? Let's say you have an interface which has multiple methods, and multiple classes are implementing this interface. One of the method implementation can be common across the class, we can make that method as a default method, so that the implementation is common for all classes.

225. How to work with existing interfaces? Second scenario where you have already existing application, for a new requirements we have to add a method to the existing interface. If we add new method then we need to implement it throughout the implementation classes. By using the Java 8 default method we can add a default implementation of that method which resolves the problem.

226. When working with multiple inheritance: If we have two interfaces, one with default method and another with just method signature (normal way of defining method in the interfaces).

228. What is @SpringBootApplication annotation in spring boot project? @SpringBootApplication annotation was introduced in Spring Boot 1.2.0 version. This annotation is equivalent to declaring these 3 annotations.

1) @Configuration
2) @EnableAutoConfiguration
3) @ComponentScan

The following are the parameters accepted in the @SpringBootApplication annotation:

- **exclude**: Exclude the list of classes from the auto configuration.
- **excludeNames**: Exclude the list of fully qualified class names from the auto configuration. This parameter added since spring boot 1.3.0.
- **scanBasePackageClasses**: Provide the list of classes that has to be applied for the @ComponentScan.
- **scanBasePackages** Provide the list of packages that has to be applied for the @ComponentScan. This parameter added since spring boot 1.3.0.

229. What Embedded Containers Does Spring Boot Support? Spring boot is a Java based framework that supports application services. It runs as a standalone jar with an embedded servlet container or as a WAR file inside a container. The spring boot framework supports three different types of embedded servlet containers: Tomcat (default), Jetty and Undertow.

230. What are the advantages and disadvantages of Spring Boot?

Spring Boot Advantages

1. Simplified & version conflict free dependency management through the starter POMs.
2. We can quickly setup and run standalone, web applications and micro services at very less time.
3. You can just assemble the jar artifact which comes with an embedded Tomcat, Jetty or Undertow application server and you are ready to go.
4. Spring Boot provides HTTP endpoints to access application internals like detailed metrics, application inner working, health status, etc.
5. No XML based configurations at all. Very much simplified properties. The beans are initialized, configured and wired automatically.
6. The Spring Initializer provides a project generator to make you productive with the certain technology stack from the beginning. You can create a skeleton project with web, data access (relational and NoSQL datastores), cloud, or messaging support.

Spring Boot Disadvantages

1. Spring boot may unnecessarily increase the deployment binary size with unused dependencies.
2. If you are a control freak, I doubt Spring Boot would fit your needs.
3. Spring Boot sticks well with micro services. The Spring Boot artifacts can be deployed directly into Docker containers. In a large and monolithic based applications, I would not encourage you to use Spring Boot.

231. What is Spring Boot Actuator? Spring Boot Actuator is a sub-project of Spring Boot. It adds several production grade services to your application with little effort on your part. Actuators enable production-ready features to a Spring Boot application, without having to actually implement these things yourself. The Spring Boot Actuator is mainly used to get the internals of running application like health, metrics, info, dump, environment, etc. which is similar to your production environment monitoring setup.
Here are some of the most common Spring Boot Actuator endpoints:

1. **/health**: It shows application health information.
2. **/info**: It displays arbitrary application info.
3. **/metrics**: It gives all metrics related information for the current application.
4. **/trace**: It displays trace information for last few HTTP requests.

232. What is Spring Boot Initializer? The Spring Initializer is ultimately a web application that can generate a Spring Boot project structure for you. It doesn't generate any application code, but it will give you a basic project structure and either a Maven or a Gradle build specification to build your code with. All you need to do is write the application code. Spring Initializer can be used several ways, including:

1. A web-based interface.
2. Via Spring Tool Suite.
3. Using the Spring Boot CLI.

233. How to reload Spring Boot Application without restarting server? Include below dependency in your pom.xml file. By default, any entry on the classpath that points to a folder will be monitored for changes. With this dependency any changes you save, the embedded tomcat will restart. This helps developers to improve the productivity.

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-devtools</artifactId>
   <optional>true</optional>
</dependency>
```

234. What are the key components of Spring Boot framework? Spring Boot Framework has mainly four major components.
Spring Boot Starters: The main responsibility of Spring Boot Starter is to combine a group of common or related dependencies into single dependencies. Spring Boot starters can help to reduce the number of manually added dependencies just by adding one dependency. So instead of manually specifying the dependencies just add one starter. Examples are spring-boot-starter-web, spring-boot-starter-test, spring-boot-starter-data-jpa, etc.
Spring Boot AutoConfigurator: One of the common complaint with Spring is, we need to make lot of XML based configurations. Spring Boot AutoConfigurator will simplify all these XML based configurations. It also reduces the number of annotations.
Spring Boot CLI: Spring Boot CLI (Command Line Interface) is a Spring Boot software to run and test Spring Boot applications from command prompt. When we run Spring Boot applications using CLI, then it internally uses Spring Boot Starter and Spring Boot AutoConfigurate components to resolve all dependencies and execute the application.
Spring Boot Actuator: Spring Boot Actuator is a sub-project of Spring Boot. It adds several production grade services to your application with little effort on your part. Actuators enable production-ready features to a Spring Boot application, without having to actually implement these things yourself. The Spring Boot Actuator is mainly used to get the internals of running application like health, metrics, info, dump, environment, etc. which is similar to your production environment monitoring setup.
235. What are the different types of bean scope in Spring framework? In the spring bean configurations, bean attribute called 'scope' defines what kind of object has to created and returned. If no bean scope is specified in bean configuration file, then it will be by default 'singleton'. There are 5 types of bean scopes available, they are:
1) singleton: Returns a single bean instance per Spring IoC container.
2) prototype: Returns a new bean instance each time when requested.
3) request: Returns a single instance for every HTTP request call.
4) session: Returns a single instance for every HTTP session.
5) global session: global session scope is equal as session scope on portlet-based web applications.
236. What are the standard Spring build-in events? Spring core framework provides application level event firing and event listening which is based on the standard Observer design pattern. There are built-in application events available or we can create our own custom events in spring. Here is the list of standard built-in Spring events:

1. **ContextRefreshedEvent:** Event fired when an ApplicationContext gets initialized or refreshed (refreshed via context.refresh () call).
2. **ContextStartedEvent:** This event is published when the ApplicationContext is started. Event fired when context.start () method is called.
3. **ContextStoppedEvent:** This event is published when the ApplicationContext is stopped. Event fired when context.stop () method is called.

4. **ContextClosedEvent:** This event is published when the ApplicationContext is closed. Event fired when context.close () method is called. A closed context reaches its end of life; it cannot be refreshed or restarted.
5. **RequestHandledEvent:** This event can only be used in spring MVC environment. It is called just after an HTTP request is completed.

237. What is Spring IoC container? IOC (Inversion of Control pattern) is also known as dependency injection. IOC directs the programmers to depict how to create objects instead of actually creating them. But in this design pattern, this control has been given to assembler and assembler will instantiate required class if needed. As the name implies Inversion of control means now we have inverted the control of creating the object from our own using new operator to container or framework. Now it's the responsibility of container to create object as required. The IOC container is responsible to instantiate, configure and assemble the objects. The IOC container gets informations from the XML file and works accordingly. The main tasks performed by IOC container are:

1. To instantiate the application class.
2. To configure the object.
3. To assemble the dependencies between the objects.

There are two types of IOC containers. They are:

1. BeanFactory
2. ApplicationContext

239. Difference between constructor injection and setter injection in Spring.
In **setter injection** strategy, we trust the Inversion of control (IOC) container that it will first create the bean first but will do the injection right before using the bean using the setter methods. And the injection is done according to your configuration. If you somehow misses to specify any beans to inject in the configuration, the injection will not be done for those beans and your dependent bean will not function accordingly when it will be in use!
But in **constructor injection** strategy, container imposes (or must impose) to provide the dependencies properly while constructing the bean. This was addressed as "container-agnostic manner", as we are required to provide dependencies while creating the bean, thus making the visibility of dependency, independent of any IOC container.

240. What are the various ways to obtain Streams in Java-8? In Java-8, Streams can be obtained in a number of ways. Some examples are:

1. From a Collection via the stream () and parallelStream () methods.
2. From an array via Arrays.stream (Object []).
3. From static factory methods on the stream classes, such as Stream.of (Object []), IntStream.range (int, int) or Stream.iterate (Object, UnaryOperator).
4. The lines of a file can be obtained from BufferedReader.lines ().
5. Streams of file paths can be obtained from methods in Files.
6. Streams of random numbers can be obtained from Random.ints ().
7. Numerous other stream-bearing methods in the JDK, including BitSet.stream (), Pattern.splitAsStream (java.lang.CharSequence), and JarFile.stream ().

241. List Java-8 Streams intermediate operations. Java 8 Stream intermediate operations return another Stream which allows you to call multiple operations in a form of a query. Stream intermediate operations do not get executed until a terminal operation is invoked. All Intermediate operations are lazy, so they're not executed until a result of a processing is actually needed. Traversal of the Stream does not begin until the terminal operation of the pipeline is executed. Here is the list of all Stream intermediate operations:

1. filter()
2. map()
3. flatMap()
4. distinct()
5. sorted()
6. peek()
7. limit()
8. skip()

242. List Java-8 Streams terminal operations. Java-8 Stream terminal operations produces a non-stream, result such as primitive value, a collection or no value at all. Terminal operations are typically preceded by intermediate operations which return another Stream which allows operations to be connected in a form of a query.Here is the list of all Stream terminal operations:

1. toArray()
2. collect()
3. count()
4. reduce()
5. forEach()

6. forEachOrdered()
7. min()
8. max()
9. anyMatch()
10. allMatch()
11. noneMatch()
12. findAny()
13. findFirst()

243. Can we reuse Java-8 Streams? A stream should be operated on (invoking an intermediate or terminal stream operation) only once. A stream implementation may throw IllegalStateException if it detects that the stream is being reused. So the answer is no, streams are not meant to be reused.

244. What is the difference between Closure and Lambda in Java 8? Lambdas are a language construct (anonymous functions), closures are an implementation technique to implement first-class functions (whether anonymous or not).

A ***lambda*** is just an anonymous function - a function defined with no name. In some languages, they are equivalent to named functions. In fact, the function definition is re-written as binding a lambda to a variable internally. In other languages, like Python, there are some (rather needless) distinctions between them, but they behave the same way otherwise.

A ***closure*** is a function that is evaluated in its own environment, which has one or more bound variables that can be accessed when the function is called. They come from the functional programming world, where there are a number of concepts in play. Closures are like lambda functions, but smarter in the sense that they have the ability to interact with variables from the outside environment of where the closure is defined.

245. Does Java 8 Lambda supports recursive call? In general, Lambda implementations are mostly anonymous functions. In recursion, a method calls itself. Since anonymous function doesn't have a name, it cannot be called by itself. That means an anonymous Lambda cannot be called by itself. But if we have a Lambda function declaration as a member variable or class variable, Java 8 supports recursion with Lambda functions. Java 8 does not support Lambda function declaration with local variable.

246. Can Java 8 default methods override equals, hashCode and toString? The methods declared in java.lang.Object class cannot be override in Java 8 default methods. It is forbidden to define default methods in interfaces for methods in java.lang.Object. Default interface methods can be overwritten in classes implementing the interface and the class implementation of the method has a higher precedence than the interface implementation, even if the method is implemented in a superclass. Since all classes inherit from java.lang.Object, the methods in java.lang.Object would have precedence over the default method in the interface and be invoked instead.

247. Hibernate Eager vs Lazy Fetch Type? The relationships are defined through joins in database. Hibernate represents joins in the form of associations like One-to-One, One-to-Many and Many-to-One. It is required to define Fetch Type when you use any of these associations. Fetch Type decides on whether or not to load all the data belongs to associations as soon as you fetch data from parent table. Fetch type supports two types of loading: Lazy and Eager. By default, Fetch type would be Lazy.

FetchType.LAZY: It fetches the child entities lazily, that is, at the time of fetching parent entity it just fetches proxy (created by cglib or any other utility) of the child entities and when you access any property of child entity then it is actually fetched by hibernate.

FetchType.EAGER: it fetches the child entities along with parent. Lazy initialization improves performance by avoiding unnecessary computation and reduce memory requirements. Eager initialization takes more memory consumption and processing speed is slow. Having said that, depends on the situation either one of these initialization can be used.

248. What is javac? It produces the java byte code from *.java file. It is the intermediate representation of your source code that contains instructions.

249. What is class? Class is nothing but a template that describes the data and behavior associated with instances of that class

250. Different Data types in Java.

1. byte – 8 bit (are esp. useful when working with a stream of data from a network or a file).
2. short – 16 bit
3. char – 16 bit Unicode
4. int – 32 bit (whole number)
5. float – 32 bit (real number)
6. long – 64 bit (Single precision)
7. double – 64 bit (double precision)

251. What is Unicode? Java uses Unicode to represent the characters. Unicode defines a fully international character set that can represent all of the characters found in human languages.

252. What are Literals? A literal is a value that may be assigned to a primitive or string variable or passed as an argument to a method.

253. Dynamic Initialization? Java allows variables to be initialized dynamically, using any expression valid at the time the variable is declared.

254. What is Type casting in Java? To create a conversion between two incompatible types, we must use a cast. There are two types of casting in java: automatic casting (done automatically) and explicit casting (done by programmer).
255. How can I put all my classes and resources into one file and run it? Use a JAR file. Put all the files in a JAR, then run the app like this: **Java -jar [-options] jarfile [args...]**
256. java.lang.* get imported by default. For using String and Exception classes, you don't need to explicitly import this package. The major classes inside this package are

1. Object class
2. Data type wrapper classes
3. Math class
4. String class
5. System and Runtime classes
6. Thread classes
7. Exception classes
8. Process classes
9. Class classes

258. What are Constructors? Constructors are used for creating an instance of a class, they are invoked when an instance of class gets created. Constructor name and class name should be same and it doesn't have a return type.
260. Can a constructor call another constructor? Yes. A constructor can call another constructor of same class using this keyword. For e.g. this () calls the default constructor. Note: this () must be the first statement in the calling constructor.
261. Can a constructor call the constructor of parent class? Yes. In fact it happens by default. A child class constructor always calls the parent class constructor. However we can still call it using super keyword. For e.g. super () can be used for calling super class default constructor. Note: super () must be the first statement in a constructor.
262. What is static variable in java? Static variables are also known as class level variables. A static variable is same for all the objects of that particular class in which it is declared.
265. What is the difference between import java.util.Date and java.util.*? The star form (java.util.*) includes all the classes of that package and that may increase the compilation time – especially if you import several packages. However it doesn't have any effect run-time performance.
266. What are two different ways to call garbage collector? System.gc () OR Runtime.getRuntime ().gc ().
267. What is an exception? Exceptions are abnormal conditions that arise during execution of the program. It may occur due to wrong user input or wrong logic written by programmer.
268. Can static block throw exception? Yes, A static block can throw exceptions. It has its own limitations: It can throw only Runtime exception (Unchecked exceptions), In order to throw checked exceptions you can use a try-catch block inside it.
269. ClassNotFoundException vs NoClassDefFoundError?
1) ClassNotFoundException occurs when loader could not find the required class in class path.
2) NoClassDefFoundError occurs when class is loaded in classpath, but one or more of the class which are required by other class, are removed or failed to load by compiler.
270. What is a Java Bean? A JavaBean is a Java class that follows some simple conventions including conventions on the names of certain methods to get and set state called Introspection. Because it follows conventions, it can easily be processed by a software tool that connects Beans together at runtime. JavaBeans are reusable software components.
271. What is Multithreading? It is a process of executing two or more part of a program simultaneously. Each of these parts is known as threads. In short the process of executing multiple threads simultaneously is known as multithreading.
272. What is the main purpose of having multithread environment? Maximizing CPU usage and reducing CPU idle time
273. What are the main differences between Process and thread? Explain in brief.
1) One process can have multiple threads. A thread is a smaller part of a process.
2) Every process has its own memory space, executable code and a unique process identifier (PID) while every thread has its own stack in Java but it uses process main memory and shares it with other threads.
3) Threads of same process can communicate with each other using keyword like wait and notify etc. This process is known as inter process communication.
274. Explain yield and sleep?
yield () – It causes the currently executing thread object to temporarily pause and allow other threads to execute.
sleep () – It causes the current thread to suspend execution for a specified period. When a thread goes into sleep state it doesn't release the lock.
275. What is the difference between sleep () and wait ()?
sleep () – It causes the current thread to suspend execution for a specified period. When a thread goes into sleep state it doesn't release the lock
wait () – It causes current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

277. What does join ( ) method do? if you use join() ,it makes sure that as soon as a thread calls join, the current thread(yes, currently running thread) will not execute unless the thread you have called join is finished.

278. Preemptive scheduling vs. time slicing?

1) The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized.
2) Time slicing means task executes for a defined slice/ period of time and then enter in the pool of ready state. The scheduler then determines which task execute next based on priority or other factor.

279. Can we call run () method of a Thread class? Yes, we can call run () method of a Thread class but then it will behave like a normal method. To actually execute it in a Thread, you should call Thread.start () method to start it.

280. What is Starvation? Starvation describes a situation where a thread is unable to gain regular access to shared resources and is unable to make progress. This happens when shared resources are made unavailable for long periods by "greedy" threads. For example, suppose an object provides a synchronized method that often takes a long time to return. If one thread invokes this method frequently, other threads that also need frequent synchronized access to the same object will often be blocked.

281. What is deadlock? Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

282. What is Serialization and de-serialization? Serialization is a process of converting an object and its attributes to the stream of bytes. De-serialization is recreating the object from stream of bytes; it is just a reverse process of serialization. To know more about serialization with example program, refer this article.

283. Do we need to implement any method of Serializable interface to make an object Serializable? No. In order to make an object Serializable we just need to implement the interface Serializable. We don't need to implement any methods.

285. A string class is immutable or mutable? String class is immutable that's the reason once its object gets created, it cannot be changed further.

286. Difference between StringBuffer and StringBuilder class?

1) StringBuffer is thread-safe but StringBuilder is not thread safe.
2) StringBuilder is faster than StringBuffer.
3) StringBuffer is synchronized whereas StringBuilder is not synchronized.

287. What is toString () method in Java? The toString () method returns the string representation of any object.

288. Difference between Iterator and ListIterator in java

1) Iterator is used for traversing List and Set both. We can use ListIterator to traverse List only, we cannot traverse Set using ListIterator.
2) We can traverse in only forward direction using Iterator. Using ListIterator, we can traverse a List in both the directions (forward and backward).
3) We cannot obtain indexes while using Iterator. We can obtain indexes at any point of time while traversing a list using ListIterator. The methods nextIndex () and previousIndex () are used for this purpose.
4) We cannot add element to collection while traversing it using Iterator, it throws ConcurrentModificationException when you try to do it. We can add element at any point of time while traversing a list using ListIterator.
5) We cannot replace the existing element value when using Iterator. By using set (E e) method of ListIterator we can replace the last element returned by next () or previous () methods.
6)Methods of Iterator: hasNext() , next() ,remove()
Methods of ListIterator: add(E e) , hasNext() ,hasPrevious(), next(),nextIndex(),previous(),previousIndex(),remove(), set(E e)

289. What is the difference between an Inner Class and a Sub-Class? An **Inner class** is a class which is nested within another class. An Inner class has access rights for the class which is nesting it and it can access all variables and methods defined in the outer class. **A sub-class** is a class which inherits from another class called super class. Sub-class can access all public and protected methods and fields of its super class.

290. What is a singleton class? Give a practical example of its usage. A singleton class in java can have only one instance and hence all its methods and variables belong to just one instance. Singleton class concept is useful for the situations when there is a need to limit the number of objects for a class. The best example of singleton usage scenario is when there is a limit of having only one connection to a database due to some driver limitations or because of any licensing issues.

291. What is ternary operator? Give an example. Ternary operator, also called conditional operator is used to decide which value to assign to a variable based on a Boolean value evaluation. It's denoted as?

```
public class conditionTest {
    public static void main(String args[]) {
         String status;
          int rank = 3;
          status = (rank == 1) ? "Done" : "Pending";
          System.out.println(status);
     }
}
```

292. How garbage collection is done in Java? In java, when an object is not referenced any more, garbage collection takes place and the object is destroyed automatically. For automatic garbage collection java calls either System.gc () method or Runtime.gc () method.

293. In multi-threading how can we ensure that a resource isn't used by multiple threads simultaneously?
In multi-threading, access to the resources which are shared among multiple threads can be controlled by using the concept of synchronization. Using synchronized keyword, we can ensure that only one thread can use shared resource at a time and others can get control of the resource only once it has become free from the other one using it.

294. How can we make copy of a java object? We can use the concept of cloning to create copy of an object. Using clone, we create copies with the actual state of an object. Clone () is a method of Cloneable interface and hence, Cloneable interface needs to be implemented for making object copies

295. Describe different states of a thread. A thread in Java can be in either of the following states:

- Ready: When a thread is created, it's in Ready state.
- Running: A thread currently being executed is in running state.
- Waiting: A thread waiting for another thread to free certain resources is in waiting state.
- Dead: A thread which has gone dead after execution is in dead state.

296. Can we have static methods in an Interface? Static methods can't be overridden in any class while any methods in an interface are by default abstract and are supposed to be implemented in the classes being implementing the interface. So it makes no sense to have static methods in an interface in Java.

297. In Java thread programming, which method is a must implementation for all threads? Run () is a method of Runnable interface that must be implemented by all threads.

298. I want to control database connections in my program and want that only one thread should be able to make database connection at a time. How can I implement this logic? This can be implemented by use of the concept of synchronization. Database related code can be placed in a method which has synchronized keyword so that only one thread can access it at a time.

299. How can an exception be thrown manually by a programmer? In order to throw an exception in a block of code manually, throw keyword is used. Then this exception is caught and handled in the catch block.

```
public void topMethod(){
    try{
        excMethod();
    }catch(ManualException e){ }
}
public void excMethod{
    String name=null;
    if(name == null){
        throw (new ManualException("Exception thrown manually ");
    }
}
```

300. I want my class to be developed in such a way that no other class (even derived class) can create its objects. How can I do so? If we declare the constructor of a class as private, it will not be accessible by any other class and hence, no other class will be able to instantiate it and formation of its object will be limited to itself only.

301. How objects are stored in Java? In java, each object when created gets a memory space from a heap. When an object is destroyed by a garbage collector, the space allocated to it from the heap is re-allocated to the heap and becomes available for any new objects.

302. How can we find the actual size of an object on the heap? In java, there is no way to find out the exact size of an object on the heap.

303. Which of the following classes will have more memory allocated?
Class A: Three methods, four variables, no object
Class B: Five methods, three variables, no object
Memory isn't allocated before creation of objects. Since for both classes, there are no objects created so no memory is allocated on heap for any class.

304. What happens if an exception is not handled in a program? If an exception is not handled in a program using try catch blocks, program gets aborted and no statement executes after the statement which caused exception throwing.

305. I have multiple constructors defined in a class. Is it possible to call a constructor from another constructor's body? If a class has multiple constructors, it's possible to call one constructor from the body of another one using this ().

306. What's meant by anonymous class? An anonymous class is a class defined without any name in a single line of code using new keyword.

```
public java.util.Enumeration testMethod() {
    return new java.util.Enumeration()  {
        @Override
        public boolean hasMoreElements() {
            // TODO Auto-generated method stub
            return false;
         }
        @Override
        public Object nextElement() {
            // TODO Auto-generated method stub
            return null;
        }
}
```

307. If an application has multiple classes in it, is it okay to have a main method in more than one class? If there is main method in more than one classes in a java application, it won't cause any issue as entry point for any application will be a specific class and code will start from the main method of that particular class only.

308. I want to persist data of objects for later use. What's the best approach to do so? The best way to persist data for future use is to use the concept of serialization.

309. String and StringBuffer both represent String objects. Can we compare String and StringBuffer in Java? Although String and StringBuffer both represent String objects, we can't compare them with each other and if we try to compare them, we get an error.

310. Which API is provided by Java for operations on set of objects? Java provides a Collection API which provides many useful methods which can be applied on a set of objects. Some of the important classes provided by Collection API include ArrayList, HashMap, TreeSet and TreeMap.

311. Can we cast any other type to Boolean Type with type casting? No, we can neither cast any other primitive type to Boolean data type nor can cast Boolean data type to any other primitive data type.

312. Can we use different return types for methods when overridden? The basic requirement of method overriding in Java is that the overridden method should have same name, and parameters. But a method can be overridden with a different return type as long as the new return type extends the original.

```
Class B extends A{
    A method(int x){
       //original method
    }
    B method(int x){
      //overridden method
    }
}
```

313. What's the order of call of constructors in inheritance? In case of inheritance, when a new object of a derived class is created, first the constructor of the super class is invoked and then the constructor of the derived class is invoked.

314. Differentiate between a constructor and a method? Can we mark constructors final? A constructor constructs the value, by providing data for the object. It is a special type of method that is used to initialize the object. The constructor 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. A method can be invoked using the dot operator and has its own name, and a return type. **No, declaring the constructor as final is not possible.**

315. Describe synchronization with respect to multi-threading. Synchronization is the method to control the access of multiple threads to shared resources, with respect to multi-threading. One thread can modify a shared variable when not in synchronization even when another thread is in the process of using or updating the same shared variable. This can lead to significant errors.

317. What if the main () method is declared as private? When the main () method is declared as private, the program compiles but during runtime it shows "main () method not public." message.

318. What happens when the static modifier is removed from the signature of the main () method? When the static modifier is removed from the signature of the main () method, the Program compiles but at runtime throws an error "NoSuchMethodError".

319. What is the first argument of the String array in main () method? The string array in main () has no element, the String array is empty.

320. What do you mean by Platform Independence? Platform Independence provides feasibility to run and compile the program in one platform and execute the program on any other platform.

321. What is a numeric promotion? Numeric promotions of a numeric operator are used for the conversion of the operands into a common type. In order to perform calculation easily, numeric promotion, conversion is performed.

1. It is the conversion of a smaller numeric type to a larger numeric type so that integer and floating-point operations can be performed over it.
2. Here byte, char, and short values are converted to int values.
3. The int values are converted to long values, and the long and float values are converted to double values.

322. What is false sharing in the context of multi-threading? On multi-core systems, false sharing is one of the well-known performance issues. Here each process has its local cache. When threads on different processor, modify variables false sharing occurs, that resides on the same cache line. As the thread may access different global variables completely, false sharing can be hard to detect.

323. What are the methods used to implement for the key Object in HashMap? Equals and hash code methods are to be implemented In order to use any object as Key in Hash Map, in Java.

324. What is an immutable object? Java classes whose objects cannot be modified once they are created are known as Immutable classes. Any modification of Immutable object results formation of the new object. E.g. String, Integer, and other wrapper class.

325. Differentiate JAR and WAR files

JAR files

1. Full form of JAR files is Java Archive Files.
2. Aggregating many files into one is allowed in JAR files
3. The JAR is usually used to hold Java classes in a library.

WAR files

1. Full form of WAR files is Web Archive Files.
2. XML, Java classes, and JavaServer pages are stored in WAR
3. Mainly used for Web Application purposes.

326. What is a JIT compiler? Just-In-Time (JIT) compiler is used to improve the performance. JIT compiles parts of the bytecode that has similar functionality which in turn reduces the amount of time needed for compilation. The term "compiler" here refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

327. What is the multi-catch block in Java? Multi-catch block makes the code shorter and cleaner when every catch block has similar code. We can catch multiple exceptions in a single catch block using this feature.

328. What is an exception? An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.

329. What is error? An Error indicates that a non-recoverable condition has occurred that should not be caught. Error, a subclass of Throwable, is intended for drastic problems, such as OutOfMemoryError, which would be reported by the JVM itself.

330. What are the advantages of using exception handling? Exception handling provides the following advantages over "traditional" error management techniques:-

1. Separating Error Handling Code from "Regular" Code.
2. Propagating Errors Up the Call Stack.
3. Grouping Error Types and Error Differentiation.

331. Why Errors are Not Checked? An unchecked exception classes which are the error classes (Error and its subclasses) are exempted from compile-time checking because they can occur at many points in the program and recovery from them is difficult or impossible. A program declaring such exceptions would be pointlessly.

332. what if there is a break or return statement in try block followed by finally block? If there is a return statement in the try block, the finally block executes right after the return statement encountered, and before the return executes.

333. How to create custom exceptions? By extending the Exception class or one of its subclasses.

```
class MyException extends Exception {
  public MyException() {
    super();
  }
   public MyException(String s) {
    super(s);
  }
 }
```

334. What are the different ways to handle exceptions? There are two ways to handle exceptions:

1. Wrapping the desired code in a try block followed by a catch block to catch the exceptions.
2. List the desired exceptions in the throws clause of the method and let the caller of the method handle those exceptions.

335. What is busy spin, and why should you use it? Busy spin is known as one of the techniques to wait for events without freeing CPU. This is often done to avoid losing data in CPU cache, which could get lost if the thread is paused and resumed in some other core. This is only valuable if you need to wait for a short amount of time, e.g. in microseconds or nanoseconds.

336. LMAX Disruptor frameworks, a high-performance inter-thread messaging library has a BusySpinWait Strategy, which is centered on this model and uses a busy spin loop for EventProcessors waiting on the barrier.
337. How do you take thread dump in Java? In Windows, you can press **Ctrl + Break.** This will instruct JVM to print thread dump in standard out, and it may go to console or log file depending on your application configuration.
338. Describe what a thread-local variable is in Java? Thread-local variables are variables restricted to a thread. It is like thread's own copy which is not shared between a multitudes of threads. Java offers a ThreadLocal class to upkeep thread-local variables. This is one of the many ways to guarantee thread-safety. However, it is important to be mindful while using a thread local variable in a controlled environment, e.g. with web servers where worker thread outlives any application variable. Any thread local variable which is not taken away once its work is done can hypothetically cause a memory leak in Java application.
339. What's the difference between Callable and Runnable? Both of these are interfaces used to carry out task to be executed by a thread. Callable can return a value, while Runnable cannot. Callable can throw a checked exception, while Runnable cannot. Runnable has been around since Java 1.0, while Callable was introduced as part of Java 1.5.
340. What do the Thread.class methods run () and start () do? Thread.run () will execute in the calling thread, i.e. a new thread is not created, the code is executed synchronously. Thread.start () will execute the same code, but in a new asynchronous thread.
341. Is it possible to cast an int value into a byte variable? What would happen if the value of int is larger than byte? Yes, it is possible but int is 32 bit long in Java, while byte is 8 bit long in Java. Therefore when you can cast an int to byte higher, 24 bits are gone and a byte can only hold a value between -128 to 128.
342. Which class contains method: Cloneable or Object? Java.lang.Cloneable is marker interface and does not contain at all any method. Clone method is well-defined in the object class. Remember that clone () is a native method
343. Is ++ operator thread-safe in Java? ++ is not thread-safe in Java because it involves multiple commands such as reading a value, implicating it, and then storing it back into memory. This can be overlapped between multiple threads.
344. Is it possible to store a double value in a long variable without casting? No, it is not possible to store a double value into a long variable without casting since the range of double is more, meaning you would need to type cast.
345. Which one will take more memory: an int or Integer? An integer object will take more memory as it stores metadata overhead about the object. An int is primitive, therefore it takes less space.
346. What is constructor chaining in Java? Constructor chaining in Java is when you call one constructor from another. This generally occurs when you have multiple, overloaded constructor in the class.
347. What is the size of int in 64-bit JVM? The size of an int variable is constant in Java, it is always 32-bit regardless of platform. This means the size of primitive int is identical in both 32-bit and 64-bit Java Virtual Machine.
348. What is the size of an int variable in 32-bit and 64-bit JVM? The size of int is identical in both 32-bit and 64-bit JVM, and it is always 32-bits or 4 bytes.
349. How does WeakHashMap work? WeakHashMap operates like a normal HashMap but uses WeakReference for keys. Meaning if the key object does not devise any reference then both key/value mapping will become appropriate for garbage collection.
350. How do you identify if JVM is 32-bit or 64-bit from Java Program? This can be identified by checking some system properties such as **sun.arch.data.model or os.arch.**
351. Explain Java Heap Space: - When a Java process has started using Java command, memory is distributed to it. Part of this memory is used to build heap space, which is used to assign memory to objects every time they are formed in the program.
352. How do you locate memory usage from a Java program? How much of the percent is used? You can use memory related methods from java.lang.Runtime class to get the free memory, total memory and maximum heap memory in Java. From using these methods, you can find out how much percent of the heap is used and how much heap space is outstanding.
Runtime.freeMemory () return the amount of free memory in bytes, **Runtime.totalMemory ()** returns the total memory in bytes **and Runtime.maxMemory ()** returns the maximum memory in bytes.
353. What is the difference between Stack and Heap in Java?

1. Heap memory is used by all the parts of the application whereas stack memory is used only by one thread of execution.
2. Whenever an object is created, it's always stored in the Heap space and stack memory contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space.
3. Memory management in stack is done in LIFO manner whereas it's more complex in Heap memory because it's used globally.

354. What is a.hashCode () used for? How is it related to a.equals (b)? HashCode () method returns an int hash value corresponding to an object. It is used in hash-based collection classes. It is very closely related to equals () method. Two objects which are identical to each other using equals () method needs to have the same hash code.
355. What happens when a finally block has a return statement? The returned value will override any value returned by the corresponding try block.
356. What is the difference between poll () and remove () method? Both poll () and remove () take out the object from the Queue but if poll () fails, then it returns null. However, if remove () fails, it throws exception.

357. Is it possible for two unequal objects to have the same hashcode? Yes, two unequal objects can have the same hashcode. This is why collision can occur in HashMap. The equal hashcode contract only says that two equal objects must have the identical hashcode, but there is no indication to say anything about the unequal object.
358. How do you test static method? PowerMock library can be used to test static methods in Java.
359. What is the difference between @Before and @BeforeClass annotation? The code marker **@Before** is executed before each test, while **@BeforeClass** runs once before the entire test fixture. If your text class has ten tests, **@Before** code will be executed ten times, but **@BeforeClass** will be executed only once. In general, you would use **@BeforeClass** when numerous tests are required to share the same computationally expensive setup code. Starting a database connection falls into this category. You are able to move code from **@BeforeClass into @Before,** but your test run may be delayed**. BeforeClass** is run as static initializer, therefore it will run before the class instance of your test fixture is created.
360. What's the difference between unit, integration and functional testing? A unit tests a small isolated piece of code such as a method. It doesn't interact with any other systems such as a database or another application**. An integration test** tests how your code plays with other systems, such as a database, a cache or a third-party application. **A functional test** is the testing of the complete functionality of an application. This may involve the use of an automated tool to carry out more complex user interactions with your system. It tests certain flows/use cases of your application.
361. What is the difference between state-based unit testing and interaction-based unit testing? State-based unit testing tests that the resulting state of a piece of code under test is as expected. **Interaction-based testing** tests that the piece of code under tests followed a certain flow or invoked certain methods as expected.
362. Explain Liskov Substitution Principle. According to the Liskov Substitution Principle, Subtypes must be appropriate for super type i.e. methods or functions which use super class type must be able to work with object of subclass with no issues. LSP is closely related to Single Responsibility Principle and Interface Segregation Principle. If a class has more functionality, then subclass might not upkeep some of the functionality and does violate LSP.
363. What is Adapter pattern and when would you use it? Adapter pattern provides interface conversion. For example, if your client is using some interface but you have something else, you can write an adapter to bridge them together.
364. What is the difference between Adapter and Decorator pattern? The adapter pattern is used to bridge the gap in the middle of two interfaces, but Decorator pattern is used to add an extra level of indirection to support distributed, controlled or intelligent access.
365. What is Template method pattern? Template pattern provides an outline of an algorithm and lets you configure or customize its steps. For example, you are able to view a sorting algorithm as a template to sort object. It describes steps for sorting but lets you arrange how to associate them using Comparable or something comparable in another language. This pattern uses double dispatch to supplement another level of indirection.
366. What is the difference between nester static class and top-level class? A public top-level class must have the same name as the name of the source file – there is no obligation for nested static class. **A nester static class** is at all times inside a top-level class and you need to use the name of the top-level class to refer nested static class. For example, HashMap.Entry is a nester static class, whereby HashMap is a top-level class and Entry is a nested static class.
367. Is it possible to write a regular expression to check if String is a number? A numeric String is only able to contain digits i.e. 0-9 and +/- sign. By using this information, you can write following regular expression to check if given String is number or not.
368. What is the difference between Serializable and Externalizable in Java? Serializable interface is used to make Java classes Serializable so that they can be transmitted over the network or their state can be kept on disk. However, it influences default serialization built-in JVM, which is pricey, fragile, and unsecured. Externalizable lets you fully control the Serialization process, identify a customer binary format and enhance security measure.
369. What is the difference between DOM and SAX parser in Java? DOM parser loads the whole XML into memory to create a tree-based DOM model. This helps it quickly locate nodes and make a change in the structure of XML. SAX parser is an event based parser and does not load the whole XML into memory. For this reason, DOM is quicker than SAX but it needs more memory and is not fitting to parse large XML files.
370. What are the important features of Java 5 release?

1. Generics
2. Enhanced for Loop
3. Autoboxing/Unboxing
4. Typesafe Enums
5. Varargs
6. Static Import
7. Metadata (Annotations)
8. Instrumentation

371. Explain 5 features introduced in JDK 1.6.

1. Scripting Language Support
2. JDBC 4.0 API
3. Java Compiler API

4. Pluggable Annotations
5. Native PKI, Java GSS, Kerberos and LDAP support.
6. Integrated Web Services.
7. Lot more enhancements.

372. Explain 5 features introduced in JDK 1.7.

1. Strings in switch Statement
2. Type Inference for Generic Instance Creation
3. Multiple Exception Handling
4. Support for Dynamic Languages
5. Try with Resources
6. Java nio Package
7. Binary Literals, underscore in literals
8. Diamond Syntax
9. Automatic null Handling

373. Explain 5 features introduced in JDK 1.8.

1. Lambda Expressions
2. Pipelines and Streams
3. Date and Time API
4. Default Methods
5. Type Annotations
6. Nashhorn JavaScript Engine
7. Concurrent Accumulators
8. Parallel operations
9. PermGen Error Removed
10. TLS SNI

374. What are the important features of Java 9 release?

1. Java 9 REPL (JShell)
2. Java 9 Module System
3. Factory Methods for Immutable List, Set, Map and Map.Entry
4. Private methods in Interfaces
5. Reactive Streams
6. GC (Garbage Collector) Improvements

375. What are the important features of Java 10 release? Java 10 is mostly a maintenance release

1. Local-Variable Type Inference
2. Enhance java.util.Locale and related APIs to implement additional Unicode extensions of BCP 47 language tags.
3. Enable the HotSpot VM to allocate the Java object heap on an alternative memory device, such as an NV-DIMM, specified by the user.
4. Provide a default set of root Certification Authority (CA) certificates in the JDK.

376. What is the difference between Object Oriented Programming and Object Based Programming? Object oriented programming supports all the usual OOP features such as inheritance and polymorphism. It also has no built in objects. **Object based programming** does not support inheritance or polymorphism and does have some built in objects.

378. What is Java Annotations? Java Annotations provide information about the code and they have no direct effect on the code they annotate. Annotation is metadata about the program embedded in the program itself. It can be parsed by the annotation parsing tool or by compiler. We can also specify annotation availability to either compile time only or till runtime also. Java Built-in annotations are @Override, @Deprecated and @SuppressWarnings.

379. What is the use of System class? Java System Class is one of the core classes. One of the easiest way to log information for debugging is System.out.print () method. System class is final so that we can't subclass and override its behavior through inheritance. System class doesn't provide any public constructors, so we can't instantiate this class and that's why all of its methods are static.

380. What is instanceof keyword? We can use instanceof keyword to check if an object belongs to a class or not. We should avoid it's usage as much as possible.

```
public static void main(String args[]){
        Object str = new String("abc");
        if(str instanceof String){
                System.out.println("String value:"+str);
        }
        if(str instanceof Integer){
                System.out.println("Integer value:"+str);
```

```
            }
}
```
381. What is J2EE? J2EE means Java 2 Enterprise Edition. The functionality of J2EE is developing multitier web-based applications. The J2EE platform is consists of a set of services, application programming interfaces (APIs), and protocols.

382. What are the four components of J2EE application?

1. Application client's components.
2. Servlet and JSP technology are web components.
3. Business components (JavaBeans).
4. Resource adapter components

383. What are types of J2EE clients?

1. Applets
2. Application clients
3. Java Web Start-enabled clients, by Java Web Start technology.
4. Wireless clients, based on MIDP technology.

384. What is considered as a web component? Java Servlet and Java Server Pages technology components are web components. Servlets are Java programming language that dynamically receives requests and makes responses. JSP pages execute as servlets but allow a more natural approach to creating static content.

385. What is JSF? JavaServer Faces (JSF) is a user interface (UI) designing framework for Java web applications. JSF provides a set of reusable UI components, a standard for web applications. JSF is based on MVC design pattern. It automatically saves the form data to the server and populates the form date when display on the client side.

386. What is Hibernate? Hibernate is an open source object-relational mapping and query service. In hibernate we can write HQL instead of SQL which save developers to spend more time on writing the native SQL. Hibernate has a more powerful association, inheritance, polymorphism, composition, and collections. It is a beautiful approach for persisting into the database using the Java objects. Hibernate also allows you to express queries using Java-based criteria.

387. What is the limitation of hibernate?

1. Slower in executing the queries than queries are used directly.
2. Only query language support for composite keys.
3. No shared references to value types.

388. What are the advantages of hibernate?

1. Hibernate is portable i mean database independent, Vendor independence.
2. Standard ORM also supports JPA
3. Mapping of the Domain object to the relational database.
4. Hibernate is better than plain JDBC.
5. JPA provider in JPA based applications.

389. What is ORM? ORM stands for Object-Relational mapping. The objects in a Java class which is mapped into the tables of a relational database using the metadata that describes the mapping between the objects and the database. It works by transforming the data from one representation to another.

390. Difference between save and saveorupdate?

Save () – This method in Hibernate is used to stores an object in the database. It inserts an entry if the record doesn't exist, otherwise not.

Saveorupdate () -This method in hibernate is used for updating the object using identifier. If the identifier is missing this method calls save (). If the identifier exists, it will call update method.

391. Difference between load and get method? Load () can't find the object from cache or database, an exception is thrown, and the **load ()** method never returns null. **Get ()** method returns null if the object can't be found. **Load ()** method may return a proxy instead of a real persistent instance **get ()** never returns a proxy.

392. How to invoke a stored procedure in hibernate? { ? = call thisISTheProcedure ()}

393. What are the benefits of ORM?

1. Productivity
2. Maintainability
3. Performance
4. Vendor independence

394. What are the Core interfaces of Hibernate framework?

1. Session Interface
2. SessionFactory Interface
3. Configuration Interface
4. Transaction Interface
5. Query and Criteria Interface

395. What is the file extension used for hibernate mapping file? File should be like this: **filename.hbm.xml**

396. What is the file name of hibernate configuration file? File should be like this: **hibernate.cfg.xml**
397. How Hibernate is database independent explain? Only changing the full property full database can be replaced.
<property name="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</property> and
<property name="hibernate.connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
398. How to add Hibernate mapping file in hibernate configuration file?
By <mapping resource=" filename.hbm.xml"/>
399. Define connection pooling? Connection pooling is a mechanism reuse the connection which contains the number of already created object connection. So whenever it is necessary for an object, this mechanism is used to get objects without creating it.
400. What is the Hibernate proxy? An object proxy is just a way to avoid retrieving an object until you need it.
401. What do you create a SessionFactory?
Configuration cfg = new Configuration ();
cfg.addResource ("dir/hibernate.hbm.xml");
cfg.setProperties (System.getProperties ());
SessionFactory sessions = cfg.buildSessionFactory ();
402. What is HQL? HQL stands for Hibernate Query Language. Hibernate allows to the user to express queries in its portable SQL extension, and this is called as HQL. It also allows the user to express in native SQL.
403. What are the Collection types in Hibernate? Set, List, Array, Map, Bag are collection type in Hibernate.
404. What is a thin client? A thin client is a program interface to the application that does not have any operations like the query of databases, execute complex business rules, or connect to legacy applications.
405. Differentiate between .ear, .jar and .war files.
.jar files: These files are with the .jar extension. The .jar files contain the libraries, resources and accessories files like property files.
.war files: These files are with the .war extension. The .war file contains JSP, HTML, JavaScript and other files necessary for the development of web applications.
.ear files: The .ear file contains the EJB modules of the application.
406. What is the JSP tag? In JSP tags can be divided into four different types.

1. Directives
2. Declarations
3. Scriplets
4. Expressions

407. How to access web.xml init parameters from JSP page? For example, if you have:
<context-param> <param-name>Id</param-name> <param-value>this is the value</param-value></context-param>
You can access this parameter: - **Id: <h: outputText value="# {initParam ['Id']}"/>**
408. What are JSP Directives?
1. page Directives <%@page language="java" %>
2. include Directives: <%@ include file="/header.jsp" %>
3. taglib Directives <%@ taglib uri="tlds/taglib.tld" prefix="html" %>
408. What is the EAR file? An EAR file is a JAR file with an .ear extension. A J2EE application with all of its modules is delivered in an EAR file.
409. What will happen when you compile and run the following code?

```
public class MyClass {
  public static void main(String args[]){
     int array[]=new int[]{1,2,3};
     System.out.println(array [1]);
  }
}
```

Answer: Compiled and shows output : 2

410. What is action mapping? In action mapping, we specify action class for particular URL i.e. path and different target view i.e. forwards on to which request response will be forwarded. The ActionMapping represents the information that the ActionServlet knows about the mapping of a particular request to an instance of a particular Action class. The mapping is passed to the execute () method of the Action class, enabling access to this information directly.
411. What is the MVC? MVC stands Model-View-Controller.
Model: Model in many applications represent the internal state of the system as a set of one or more JavaBeans.
View: The View is most often constructed using JavaServer Pages (JSP) technology.
Controller: The Controller is focused on receiving requests from the client and producing the next phase of the user interface to an appropriate View component. The primary component of the Controller in the framework is a servlet of class ActionServlet. This servlet is configured by defining a set of ActionMapping.

412. What are different modules in spring? There are seven core modules in spring

1. The Core container module
2. O/R mapping module (Object/Relational)
3. DAO module
4. Application context module
5. Aspect Oriented Programming
6. Web module
7. MVC module

413. What is Bean Factory, have you used XMLBean factory? XmlBeanFactory is one of the implementation of bean Factory **org.springframework.beans.factory.xml.** XmlBeanFactory is used to create bean instance defined in our xml file. BeanFactory factory = new XmlBeanFactory (new FileInputStream ("beans.xml"));
ClassPathResource resource = new ClassPathResource ("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory (resource);

414. What is spring? Spring is a lightweight open source framework for the development of enterprise application that resolves the complexity of enterprise application development is also providing a cohesive framework for J2EE application development which is primarily based on IOC (inversion of control) or DI (dependency injection) design pattern. Spring is set to be a framework which helps Java programmer for development of code and it provides IOC container, Dependency Injector, MVC flow and many other APIs for the java programmer.

415. What is the functionality of ActionServlet and RequestProcessor?

1. Receiving the HttpServletRequest
2. Populating JavaBean from the request parameters
3. Displaying response on the web page Issues
4. Content type issues handling
5. Provide extension points

416. ActionServlet, RequestProcessor, and Action classes are the components of Controller

417. What is default scope in Spring? Singleton.

418. What are advantages of Spring usage?

1. Pojo based programming enables reuse component.
2. Improve productivity and subsequently reduce development cost.
3. Dependency Injection can be used to improve testability.
4. Spring required enterprise services without a need for the expensive application server.
5. It reduces coupling in code and improves maintainability.

419. What are the Benefits of Spring Framework?

1. Extensive usage of Components
2. Reusability
3. Decoupling
4. Reduces coding effort by using pattern implementations such as singleton, factory, service locator etc.
5. Removal of leaking connections
6. Declarative transaction management
7. Easy to integrate with third party tools and technologies.

420. Lifecycle interfaces in spring?

1) InitializingBean

```
<bean id="expInitBean" init-method="init"/>
public class ExpBean {
   public void init() {
       // do some initialization code
   }
 } OR
<bean id=" expInitBean "/>
public class ExpBean implements InitializingBean {
   public void afterPropertiesSet() {
      // do some initialization code
   }
 }
```

2) DisposableBean

```
<bean id="expInitBean" destroy-method="cleanup"/>
public class ExpBean {
  public void cleanup() {
    // do some destruction code (like releasing pooled connections)
  }
 } OR
<bean id="expInitBean"/>
public class ExpBean implements DisposableBean {
  public void destroy() {
    // do some destruction code (like releasing pooled connections)
  }
 }
```

421. How to Create Object without using the keyword "new" in java? Without new, the Factory methods are used to create objects for a class. For example **Calendar c=Calender.getInstance ();**
Here Calendar is a class, and the method getInstance () is a Factory method which can create an object for Calendar class.
422. What is a servlet? Servlets is a server-side component that provides a powerful mechanism for developing server side programs. Servlets is a server, as well as platform-independent and Servlets, are designed for various protocols. Most commonly used HTTP protocols. Servlets use the classes in the java packages **javax.servlet, javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse, javax.servlet.http.HttpSession.** All servlets must implement the Servlet interface, which defines life-cycle methods.
423. Servlet is pure java object or not? Yes, pure java object.
424. What must be implemented by all Servlets? The Servlet Interface must be implemented by all servlets

425. What are the phases of the servlet life cycle? The life cycle of a servlet consists of the following phases:

1. Servlet class loading
2. Servlet instantiation
3. the init method
4. Request handling (call the service method)
5. Removal from service (call the destroy method)

426. What are Advices in spring? It is the execution of an aspect. Advice is like making your application learn a new trick. They are usually introduced at joinpoints.
427. Name the types of transaction management that are supported by spring? Transaction management supported by spring are:

1. Declarative transaction management.
2. Programmatic transaction management.

428. Is Singleton beans are thread safe in Spring Framework? No, singleton beans are not thread-safe in spring framework.
429. What is Bean Factory? Bean Factory is core of the spring framework and, it is a Lightweight container which loads bean definitions and manages your beans. Beans are configured using XML file and manage singleton defined bean. It is also responsible for life cycle methods and injects dependencies. It also removes adhoc singletons and factories.
430. Define Bean Wiring? Bean wiring is the creation of associations between application components that are between the beans in a particular spring container.
431. What is called Spring MVC? A Spring MVC is a single shared controller instance and it is used to handle request type controllers, interceptors which run in the IOC container. It also allows multiple Dispatcher Servlets which can share application context interface but not class based interface.
432. Why Spring framework is needed? Spring framework is needed because it is –

1. Very Light Weight Container
2. Framework
3. IOC
4. AOP

433. Explain the RowCallbackHandler in spring? The RowCallbackHandler is called for each row in ResultSet and is used to read values from the ResultSet.
434. Define Application context module? This is a very important module and supplies various necessary services like EJB integration, remoting, JNDI access and scheduling. It transforms spring into a framework. It also broadens the idea of BeanFactory by application of lifecycle events, providing support for internationalization messages and validation.
435. Write about AOP module? AOP module is utilized for creating aspects for spring applications. It also enables support for metadata programming in spring.
436. What is a BeanFactory Interface? Bean factory interface is used to provide configuration framework for object creation and basic functionality around object management.
437. State the differences between ApplicationContext and BeanFactory in spring?

1. ApplicationContext allows more than one config files to exist while BeanFactory only permits one.
2. ApplicationContext can print events to beans registered as listeners. This feature is not supported by BeanFactory.

3. ApplicationContext also provides support for application of lifecycle events, internationalization messages and validation and also provides services like EJB integration, remoting, JNDI access and scheduling. These features too are not supported by Bean Factory.

440. How to start using spring? Following steps needs to be done to start with the spring:

1. Download Spring and its dependent file from spring's site.
2. Create application context xml to define beans and its dependencies
3. Integrate application context xml with web.xml
4. Deploy and Run the application

441. What are the methods of bean life cycle? There are two important methods of Bean life cycle:

1. **Setup –** called when bean is loaded into container
2. **Teardown –** called when bean is unloaded into container

442. What are the different types of events of Listeners? Following are the different types of events of listeners:

1. **ContextClosedEvent –** This event is called when the context is closed.
2. **ContextRefreshedEvent –** This event is called when context is initialized or refreshed
3. **RequestHandledEvent –** This event is called when the web context handles request

443. Differentiate between singleton and prototype bean? Singleton means only one bean is defined per object instance while **Prototype** means one definition to more than one object instances in spring.

444. Write about Core container module? Core container module is responsible for the basic functionality of the spring framework. The whole spring framework is built with this module as a base.

445. What is AOP module? This AOP module is used for spring enabled application. Support has been provided AOP alliance to ensure the interoperability between spring and other AOP frameworks. It instructs spring to add annotations to the source code and tell how to apply aspects.

446. What is AOP Alliance? AOP alliance is an open-source project which is aimed at promoting adoption of AOP. The AOP alliance's goal is to define a common set of components and interfaces so as to improve interoperability among different AOP implementations.

447. What is called spring configuration file? Spring configuration file is an XML file and it contains class information. It also describes how these classes are configured and interact with each other.

448. What are different types of Autowire? There are four different types of Auto wire:

1. byName
2. byType
3. constructor
4. autodetect

449. When are declarative and programmatic transaction management used? When only a small amount of transactional operations is there, it is advised to use **Programmatic transaction management.** But if there is a big amount of transactional operations to be taken care of, **declarative transaction management is preferred.**

451. What is an Aspect? Aspect is also called as logging which is required throughout the application. Logging or aspect is a cross cutting functionality in an application using AOP.

452. What is a Joinpoint? The point where an aspect can be introduced in the application is known as a joinpoint. This point could be a field being modified, a method being called or even an exception being thrown. At these points, the new aspect's code can be added to introduce a new behavior to the application. Aspect code can be inserted at this point into normal flow of application to change the current behavior.

453. What is called an Advice? Advice will tell application on new behavior and it is the implementation of an aspect. It is inserted into an application at the joinpoint. Advice is the implementation of an aspect. It is something like telling your application of a new behavior. Generally, the advice is inserted into an application at joinpoints.

454. What is a Pointcut? Pointcut is used to allow where the advice can be applied.

455. What is weaving? Weaving is used to create new proxy object by applying aspects to target object.

456. In what points, can weaving be applied? Following are the points where weaving can be applied:

1. Compile Time
2. Class load Time
3. Runtime

457. What are the different types of AutoProxying? Following are the different types of AutoProxying:

1. BeanNameAutoProxyCreator
2. DefaultAdvisorAutoProxyCreator
3. Metadata autoproxying

458. How can beans be made singleton or prototype? The bean tag has an attribute called 'singleton'. The bean is singleton if its value is 'TRUE', otherwise the bean is a prototype.

459. What classes are used to Control the database connection? Following are the classes that are used to control database connection:

1. Data Source Utils
2. SmartData Source
3. AbstractData Source
4. SingleConnection DataSource
5. DriverManager DataSource
6. TransactionAware DataSourceProxy
7. DataSource TransactionManager

460. Describe about DAO in spring framework? DAO is used to provide integration of Java database connectivity and Object relational mapping objects. DAO is spring framework provides connection for JDBC, hibernate, JDO, JPA, Common client interface and Oracle.

461. What is Autoproxying? Autoproxying is used to create proxy automatically for the spring users.

462. What is Metadata Autoproxying? Metadata Autoproxying can be performed inspiring which can be driven by metadata. This is determined by source level attributes and keeps metadata inside the source code. This maintains metadata in one place and mainly used for declarative transaction support.

463. What is 'Throws advice' in spring? 'Throws Advice' define the behavior when an exception occurs. It is an interface and it has no methods which need to be implemented. A class that implements this interface should have method with this signature:

1. Void samplethrow (Throw table t)
2. Void samplethrow(Method m, Object[] o, Object target, Throw tablet)

464. What are the various editors used in spring work? The various custom editors provided by the Spring Framework are:

1. PropertyEditor
2. URLEditor
3. ClassEditor
4. CustomDateEditor
5. FileEditor
6. LocaleEditor
7. StringArrayPropertyEditor
8. StringTrimmerEditor

465. How is Hibernate accessed using the spring framework? Hibernate can be accessed in the following two ways:

1. By IOC with a Callback and HibernateTemplate.
2. By applying an AOP Interceptor and broadening the HibernateDaoSupport.

466. What are the various Channels supported by Spring 2.0?

1. Pollable Channel
2. Subscribable Channel
3. PublishSubscribe Channel
4. Queue Channel
5. Priority Channel
6. Rendezvous Channel
7. Direct Channel
8. Executor Channel
9. Scoped Channel

467. Why is declarative transaction management preferred in spring? Declarative transaction management has minimum impact on the application code and, therefore, is an idealistic lightweight container.

468. What are the different scopes of spring bean? Scopes of spring bean are Singleton, prototype, request, session and global session.

470. Write the benefits of using IOC? The major benefits of dependency injection or IOC are that it reduces the amount of coding required for the application. This allows the testing of the application to be done quickly and easily as no JNDI lookup mechanism or singletons are required. IOC containers also support lazy loading and eager installation of services.

471. What is Inner bean? What is the drawback of inner bean? If a bean element is directly embedded in a property tag while wiring beans, then the bean is called Inner Bean. Its drawback is that it cannot be reprocessed.

472. What are the types of Advice? There are five types of Advice:

1. **Before advice:** Advice that is executed prior to a joinpoint is called the 'before advice'.
2. **After returning advice:** Advice that is executed after the normal completion of a joinpoint is called the 'after returning advice'.
3. **After throwing advice:** Advice that is executed only if a method exits abnormally by throwing an exception, is called the 'after throwing advice'.
4. **After (finally) advice:** Advice that is executed irrespective of how a joinpoint exits is called 'after finally advice'.
5. **Around advice:** Advice that borders a joinpoint, for example, a method invocation, is called an 'around advice'. This can be used to perform special activities before and after the invocation of method.

473. What is called PreparedStatementCreator? PreparedStatementCreator is one of the most commonly used interfaces for writing data to the database. CreatePreparedStatement () is a method that can be used to create and return PreparedStatement from the Connection argument, and exception handling is automatically taken care of. When this interface is implemented, a different interface SqlProvider can also be implemented which has a method called getSql (). This method is useful for providing sql strings to the JdbcTemplate. It does not handle SQLExceptions.

474. What is SQLProvider? SQLProvider has only one method called getSql()and it is implemented using PreparedStatementCreator implementers. It is mainly used for debugging.

475. Write about BatchPreparedStatementSetter? BatchPreparedStatementSetter is used to update more than a single row in one go, they can use BatchPreparedStatementSetter. This interface provides two methods they are

1. setValues(PreparedStatement ps, int i) throws SOL exception
2. int getBatchSize

476. What is the better method of using JDBC in spring? If JDBC is used with the template class called JdbcTemplate, it gives a better performance.

477. What exceptions do the DAO classes, use in spring throw? In spring DAO classes only throws SQLException.

478. Explain the advantages of using DAO module? The database code can be kept clean and simple by using the DAO module. This helps in preventing problems that rise because of poor handling of closures of database resources. Also, the DAO module utilizes the AOP module to enable objects in the spring application to use transaction management services.

479. Name the significant ApplicationContext implementations used in the spring framework?

1. ClassPathXmlApplicationContext
2. FileSystemXmlApplicationContext
3. XmlWebApplicationContext

480. How is a bean added to a spring application?

```
<? xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN">
<beans>
<bean id="foo"/>
<bean id="bar"/>
</beans>
```

The bean tag has an ID attribute which stores the bean name and a class attributes which specifies the full class name.

481. What are ORM integration modules? Object/relational mapping (ORM) tool is supported by spring over straight JDBC by implementing the ORM module. Spring can join various important ORM frameworks, including JDO, iBatis SQL Maps and Hibernate.

482. What is the web module? The web module enables the creation of a web application without XML. The web.xml file needs to be configured for using the web module.

483. What is DataAccessException? DataAccessException is a RuntimeException. It is an Unchecked Exception. The user cannot be forced to handle these kinds of exceptions.

484. What is XMLBeanFactory? Spring includes several applications of Bean factory. Out of these, **org.springframework.beans.factory.xml.XmlBeanFactory** is a very important one. It loads the beans on the basis of the definitions stored in an XML file. For the creation of an XmlBeanFactory, a **java.io.InputStream** is passed to the constructor. The InputStream provides the XML to the factory. For example, for retrieval of the bean, the getBean () method is called by passing the name of the desired bean.

MyBean helloBean = (MyBean) factory.getBean ("helloBean");

485. Name the Exception class which is connected to the exceptions thrown by the applications? It is the DataAccessException given by org.springframework.dao.DataAccessException

486. What are the important beans lifecycle methods? All in all, two bean lifecycle methods are there. The first method is the setup method which is called during the loading of the bean into the container. The second is when the bean is unloaded from the container, and this method is called the teardown.

487. How can the default lifecycle methods of beans be nullified? The tag, bean, has two useful attributes which can be used to define special initialization and destruction methods. For Example, two new methods forSetup and forTeardown can be added to the Foo class in the following way:

```
<beans>
<bean id="bar" init-method="forSetup" destroy="forTeardown"/>
</beans>
```

488. What is a Target? A target is the class that is advised. This class can either be a class to which we want to add a special behavior to or a third party class. The target class is free to center on its major concern using the AOP concepts, regardless of any advice that is being applied.

489. Explain the term Proxy? The term proxy refers to an object which is produced the application of an advice to the target object.

490. What is cross cutting concern and concern in spring AOP? Cross cutting concern: It is a concern which is applicable throughout the application and it affects the entire application. E.g. Security, logging and data transfer are the concerns which are needed in almost every module of an application.
Concern: Concern is a behavior that we want to have in a module of an application. Issues in which we are interested defines our concern.
491. What's Hibernate? Hibernate is a popular framework of Java which allows an efficient Object Relational mapping using configuration files in XML format. After java objects mapping to database tables, database is used and handled using Java objects without writing complex database queries.
492. What is ORM? ORM (Object Relational Mapping) is the fundamental concept of Hibernate framework which maps database tables with Java Objects and then provides various API's to perform different types of operations on the data tables.
493. What's the usage of Configuration Interface in hibernate? Configuration interface of hibernate framework is used to configure hibernate. It's also used to bootstrap hibernate. Mapping documents of hibernate are located using this interface.
494. How can we use new custom interfaces to enhance functionality of built-in interfaces of hibernate? We can use extension interfaces in order to add any required functionality which isn't supported by built-in interfaces.
495. Should all the mapping files of hibernate have .hbm.xml extension to work properly? No, having .hbm.xml extension is a convention and not a requirement for hibernate mapping file names. We can have any extension for these mapping files.
496. How do we create session factory in hibernate? To create a session factory in hibernate, an object of configuration is created first which refers to the path of configuration file and then for that configuration, session factory is created as given in the example below:
Configuration config = new Configuration ();
config.addResource (myinstance/configuration.hbm.xml);
config.setProperties (System.getProperties ());
SessionFactory sessions = config.buildSessionFactory ();
497. What are POJOs and what's their significance? POJOs (Plain Old Java Objects) are java beans with proper getter and setter methods for each and every properties. Use of POJOs instead of simple java classes results in an efficient and well-constructed code.
498. What's HQL? HQL is the query language used in Hibernate which is an extension of SQL. HQL is very efficient, simple and flexible query language to do various type of operations on relational database without writing complex database queries.
499. How can we invoke stored procedures in hibernate? In hibernate we can execute stored procedures using code as below:

```
<sql-query name="getStudents" callable="true">
<return alias="st" class="Student">
<return-property name="std_id" column="STD_ID"/>
<return-property name="s_name" column="STD_NAME"/>
<return-property name="s_dept" column="STD_DEPARTMENT"/>
{ ? = call selectStudents() }
</return>
</sql-query>
```

500. What is criteria API? Criteria is a simple yet powerful API of hibernate which is used to retrieve entities through criteria object composition.
501. What are the benefits of using Hibernate template? Following are some key benefits of using Hibernate template:
a. Session closing is automated.
b. Interaction with hibernate session is simplified.
c. Exception handling is automated.
502. How can we see hibernate generated SQL on console? We need to add following in hibernate configuration file to enable viewing SQL on the console for debugging purposes:

```
<property name="show_sql">true</property>
```

503. What are the two types of collections in hibernate? Following are the two types of collections in hibernate:
a. Sorted Collection
b. Order Collection
504. What the benefits are of hibernate over JDBC?
a. Hibernate can be used seamlessly with any type of database as its database independent while in case of JDBC, developer has to write database specific queries.
b. Using hibernate, developer doesn't need to be an expert of writing complex queries as HQL simplifies query writing process while in case of JDBC, its job of developer to write and tune queries.
c. In case of hibernate, there is no need to create connection pools as hibernate does all connection handling automatically while in case of JDBC, connection pools need to be created.
505. How can we get hibernate statistics? We can get hibernate statistics using getStatistics () method of SessionFactory class as shown below: **SessionFactory.getStatistics ()**
506. What is transient instance state in Hibernate? If an instance is not associated with any persistent context and also, it has never been associated with any persistent context, then it's said to be in transient state.

507. How can we reduce database write action times in Hibernate? Hibernate provides dirty checking feature which can be used to reduce database write times. Dirty checking feature of hibernate updates only those fields which require a change while keeps others unchanged.
508. What's the usage of callback interfaces in hibernate? Callback interfaces of hibernate are useful in receiving event notifications from objects. For example, when an object is loaded or deleted, an event is generated and notification is sent using callback interfaces.
509. When an instance goes in detached state in hibernate? When an instance was earlier associated with some persistent context (e.g. a table) and is no longer associated, it's called to be in detached state.
510. What the four ORM levels are in hibernate? Following are the four ORM levels in hibernate:
a. Pure Relational
b. Light Object Mapping
c. Medium Object Mapping
d. Full Object Mapping
511. What's transaction management in hibernate? How it works? Transaction management is the process of managing a set of statements or commands. In hibernate; transaction management is done by transaction interface as shown in below code:

```
Session s = null;
Transaction tr = null;
try {
   s = sessionFactory.openSession();
   tr = s.beginTransaction();
   doTheAction(s);
   tr.commit();
} catch (RuntimeException exc) {
   tr.rollback();
} finally {
   s.close();
}
```

512. What the two methods are of hibernate configuration? We can use any of the following two methods of hibernate configuration:
a. XML based configuration (using hibernate.cfg.xml file)
b. Programmatic configuration (Using code logic)
513. What is the default cache service of hibernate? Hibernate supports multiple cache services like EHCache, OSCache, SWARMCache and TreeCache and default cache service of hibernate is EHCache.
514. What are the two mapping associations used in hibernate? In hibernate; we have following two types of mapping associations between entities:
a. One-to-One Association
b. Many-to-Many Association
515. What's the usage of Hibernate QBC API? Hibernate Query By Criteria (QBC) API is used to create queries by manipulation of criteria objects at runtime.
516. In how many ways, objects can be fetched from database in hibernate? Hibernate provides following four ways to fetch objects from database:
a. Using HQL
b. Using identifier
c. Using Criteria API
d. Using Standard SQL
517. How primary key is created by using hibernate? Database primary key is specified in the configuration file hbm.xml. Generator can also be used to specify how primary key is being created in the database.
In the below example, deptId acts as primary key:

```
<id name="deptId" type="string" >
<column name="columnId" length="30"/>
<generator/>
</id>
```

518. How can we reattach any detached objects in Hibernate? Objects which have been detached and are no longer associated with any persistent entities can be reattached by calling session.merge () method of session class.
519. What are different ways to disable hibernate second level cache? Hibernate second level cache can be disabled using any of the following ways:
a. By setting use_second_level_cache as false.
b. By using CACHEMODE.IGNORE
c. Using cache provider as org.hibernate.cache.NoCacheProvider

520. What is ORM metadata? All the mapping between classes and tables, properties and columns, Java types and SQL types etc. is defined in ORM metadata.
521. Which one is the default transaction factory in hibernate? With hibernate 3.2, default transaction factory is JDBCTransactionFactory.
522. What's the role of JMX in hibernate? Java Applications and components are managed in hibernate by a standard API called JMX API. JMX provides tools for development of efficient and robust distributed, web based solutions.
523. How can we bind hibernate session factory to JNDI? Hibernate session factory can be bound to JNDI by making configuration changes in hibernate.cfg file.
524. In how many ways objects can be identified in Hibernate? Object identification can be done in hibernate in following three ways:
a. Using Object Identity: Using == operator.
b. Using Object Equality: Using equals() method.
c. Using database identity: Relational database objects can be identified if they represent same row.
525. What different fetching strategies are of hibernate? Following fetching strategies are available in hibernate:
1. Join Fetching
2. Batch Fetching
3. Select Fetching
4. Sub-select Fetching
526. How mapping of java objects is done with database tables? To map java objects with database tables, we need to have Java beans properties names same as column names of a database table. Then mapping is provided in hbm.xml file as given below:

```
<hibernate-mapping>
<class name="Student"  table="tbl_student">
<property  column="studentname" length="255"
name="studentName" not-null="true"  type="java.lang.String"/>
<property  column="studentDisciplne" length="255"
name="studentDiscipline" not-null="true"  type="java.lang.String"/>
</class>
</hibernate-mapping>
```

527. What are derived properties in hibernate? Derived properties are those properties which are not mapped to any columns of a database table. Such properties are calculated at runtime by evaluation of any expressions.
528. What is meant by a Named SQL Query in hibernate and how it's used? Named SQL queries are those queries which are defined in mapping file and are called as required anywhere. For example, we can write a SQL query in our XML mapping file as follows:

```
<sql-query name = "studentdetails">
<return alias="std"/>
SELECT std.STUDENT_ID AS {std.STUDENT_ID},
std.STUDENT_DISCIPLINE AS {std.discipline},

FROM Student std WHERE std.NAME LIKE :name
</sql-query>
```

Then this query can be called as follows:

```
List students = session.getNamedQuery (studentdetails) .setString(TomBrady, name).setMaxResults(50).list();
```

528. What's the use of version property in hibernate? Version property is used in hibernate to know whether an object is in transient state or in detached state.
529. What is attribute oriented programming? In Attribute oriented programming, a developer can add Meta data (attributes) in the java source code to add more significance in the code. For Java (hibernate), attribute oriented programming is enabled by an engine called XDoclet.
530. What's the use of session.lock () in hibernate? session.lock () method of session class is used to reattach an object which has been detached earlier. This method of reattaching doesn't check for any data synchronization in database while reattaching the object and hence may lead to lack of synchronization in data.
531. Does hibernate support polymorphism? Yes, hibernate fully supports polymorphism. Polymorphism queries and polymorphism associations are supported in all mapping strategies of hibernate.
532. What the three inheritance models are of hibernate? Hibernate has following three inheritance models:
a. Tables per Concrete Class
b. Table per class hierarchy
c. Table per sub-class
533. How can we map the classes as immutable? If we don't want an application to update or delete objects of a class in hibernate, we can make the class as immutable by setting mutable=false
534. What's general hibernate flow using RDBMS? General hibernate flow involving RDBMS is as follows:
a. Load configuration file and create object of configuration class.

b. Using configuration object, create sessionFactory object.
c. From sessionFactory, get one session.
d. Create HQL query.
e. Execute HQL query and get the results. Results will be in the form of a list.

535. What is Light Object Mapping? Light Object Mapping is one of the levels of ORM quality in which all entities are represented as classes and they are mapped manually.

536. What's difference between managed associations and hibernate associations? Managed associations relate to container management persistence and are bi-directional while hibernate associations are unidirectional.

1) Explain what is Maven? How does it work? Maven is a project management tool. It provides the developer a complete build lifecycle framework. On executing Maven commands, it will look for POM file in Maven; it will run the command on the resources described in the POM.

2) List out what are the aspects does Maven Manages? Maven handles following activities of a developer
• Build
• Documentation
• Reporting
• Dependencies
• SCMs
• Releases
• Distribution
• Mailing list

3) Mention the three build lifecycle of Maven?
• Clean: Cleans up artifacts that are created by prior builds
• Default (build): Used to create the application
• Site: For the project generates site documentation

4) Explain what is POM? In Maven, POM (Project Object Model) is the fundamental unit of work. It is an XML file which holds the information about the project and configuration details used to build a project by Maven.

5) Explain what is Maven artifact? Usually an artifact is a JAR file which gets arrayed to a Maven repository. One or more artifacts a maven build produces such as compiled JAR and a sources JAR.
Each artifact includes a group ID, an artifact ID and a version string.

6) Explain what is Maven Repository? What are their types? A Maven repository is a location where all the project jars, library jars, plugins or any other particular project related artifacts are stored and can be easily used by Maven. Their types are **local, central and remote**

7) Why Maven Plugins are used? Maven plugins are used to
• Create a jar file
• Create war file
• Compile code files
• Unit testing of code
• Documenting projects
• Reporting

8) List out the dependency scope in Maven? The various dependency scope used in Maven are:
• Compile: It is the default scope, and it indicates what dependency is available in the classpath of the project
• Provided: It indicates that the dependency is provided by JDK or web server or container at runtime
• Runtime: This tells that the dependency is not needed for compilation but is required during execution
• Test: It says dependency is available only for the test compilation and execution phases
• System: It indicates you have to provide the system path
• Import: This indicates that the identified or specified POM should be replaced with the dependencies in that POM's section

9) Mention how profiles are specified in Maven? Profiles are specified in Maven by using a subset of the elements existing in the POM itself.

10) Explain how you can exclude dependency? By using the exclusion element, dependency can be excluded

11) Mention the difference between Apache Ant and Maven?
• Ant is a toolbox – Maven is a framework
• Ant does not have formal conventions like project directory structure – Maven has conventions
• Ant is procedural; you have to tell to compile, copy and compress – Maven is declarative (information on what to make & how to build)
• Ant does not have lifecycle; you have to add sequence of tasks manually – Maven has a lifecycle
• Ant scripts are not reusable – Maven plugins are reusable

12) In Maven what are the two setting files called and what are their location? In Maven, the setting files are called settings.xml, and the two setting files are located at

• Maven installation directory: $M2_Home/conf/settings.xml
• User’s home directory: ${ user.home }/ .m2 / settings.xml
13) List out what are the build phases in Maven? Build phases in Maven are
• Validate
• Compile
• Test
• Package
• Install
• Deploy
14) List out the build, source and test source directory for POM in Maven?
• Build = Target
• Source = src/main/java
• Test = src/main/test
15) Where do you find the class files when you compile a Maven project? You will find the class files ${basedir}/target/classes/.
16) Explain what would the “jar: jar” goal do? jar: jar will not recompile sources; it will imply just create a JAR from the target/classes directory considering that everything else has been done
17) List out what are the Maven’s order of inheritance? The maven’s order of inheritance is
• Parent Pom
• Project Pom
• Settings
• CLI parameters
18) For POM what are the minimum required elements? The minimum required elements for POM are project root, modelVersion, groupID, artifactID and version
19) Explain how you can produce execution debug output or error messages? To produce execution debug output you could call Maven with X parameter or e parameter
20) Explain how to run test classes in Maven? To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named “test.”
1. Explain JSP and tell its uses. JSP stands for Java Server Pages. It is a presentation layer technology independent of platform. It comes with SUN’s J2EE platforms. They are like HTML pages but with Java code pieces embedded in them. They are saved with a .jsp extension. They are compiled using JSP compiler in the background and generate a Servlet from the page.
2. What is the requirement of a tag library? A collection of custom tags is called a Tag Library. Recurring tasks are handled more easily and reused across multiple applications to increase productivity. They are used by Web Application designers who focus on presentation rather than accessing database or other services. Some popular libraries are String tag library and Apache display tag library.
3. Explain JSP Technology. JSP is a standard extension of Java and is defined on top of Servlet extensions. Its goal is to simplify management and creation of dynamic web pages. It is platform-independent, secure, and it makes use of Java as a server side scripting language.
4. Explain implicit objects in JSP. Objects created by web container and contain information regarding a particular request, application or page are called Implicit Objects. They are:
1) response
2) exception
3) application
4) request
5) session
6) page
7) out
8) config
9) pageContext
5. How can multiple submits due to refresh button clicks be prevented? Using a Post/Redirect/Get or a PRG pattern, this problem can be solved.
1. A form filled by the user is submitted to the server using POST or GET method. The state in the database and business model are updated.
2. A redirect response is used to reply by the servlet for a view page.
3. A view is loaded by the browser using the GET command and no user data is sent. This is safe from multiple submits as it is a separate JSP page.
6. Is JSP technology extensible? Yes, JSP is easily extensible by use and modification of tags, or custom actions, encapsulated in tag libraries.

7. Differentiate between response.sendRedirect (url) and <jsp: forward page = …>.
<jsp.forward> element forwards the request object from 1 JSP file to another. Target file can be HTML, servlet or another JSP file, but it should be in the same application context as forwarding JSP file.
sendRedirect send HTTP temporary redirect response to the browser. The browser then creates a new request for the redirected page. It kills the session variables.
8. Can a subsequent request be accessed with one's servlet code, if a request attribute is already sent in his JSP? The request goes out of scope, thus, it cannot be accessed. However, if a request attribute is set in one's servlet, then it can be accessed in his JSP.A JSP is a server side component and the page in translated to a Java servlet, and then executed. Only HTML code is given as output.
9. How to include static files in a JSP page? Static pages are always included using JSP include directive. This way the inclusion is performed in the translation phase once. Note that a relative URL must be supplied for file attribute. Although static resources may be included, it is not preferred as each request requires inclusion.
10. Why is it that JComponents have add () and remove () methods but Component doesn't? JComponent is a subclass of Container. It contains other Components and JComponents.
11. How can a thread safe JSP page be implemented? It can be done by having them implemented by the SingleThreadModel Interface. Add <%@page isThreadSafe="false" %> directive in the JSP page.
12. How can the output of JSP or servlet page be prevented from being cached by the browser?
Using appropriate HTTP header attributes to prevent the dynamic content output by a JSP page from being cached by the browser.
13. How to restrict page errors display in a JSP page? By setting up an "ErrorPage" attribute of PAGE directory to the name of the error page in the JSP page, and then in the error jsp page set "isErrorpage="TRUE", Errors can be stopped from getting displayed.
14. What are JSP Actions? They are XML tags, which direct the server to using existing components or control behavior of JSP Engine. They consist of a typical prefix of "jsp:" and action name.
<jsp: include/>
<jsp: getProperty/>
<jsp: forward/>
<jsp: setProperty/>
<jsp: usebean/>
<jsp: plugin/>
15. Differentiate between <jsp: include page=…> and <%@include file=…>. Both these tags include information from 1 page to another. The first tag acts as a function call between two Jsp's. It is executed each time client page is accessed by the client. It is useful to modularize the web application. New content is included in the output. The second tag content of file is textually embedded having similar directive. The changed content is not included in the output. It is helpful when code from one jsp is required by several jsp's.
16. Can constructor be used instead of init (), to initialize servlet? Yes, it is possible. But it is not preferred because init () was developed because earlier Java versions could not invoke constructors with arguments dynamically. So they could not assign a servletConfig. Today, however, servlet containers still call only no-arg constructor. So there is no access to servletContext or servletConfig.
17. Explain lifecycle methods.
1) jspInit (): The container calls this to initialize servlet instance. It is called only once for the servlet instance and preceded every other method.
2) _jspService (): The container calls this for each request and passes it on to the objects.
3) jspDestroy (): It is called by the container just before destruction of the instance.
18. Explain JSP Output comments? They are comments that can be viewed in HTML Source File.
19. Define Expression: - Expression tag is used to insert Java values directly in the output. Its syntax is <%=expression %> It contains a scripting language expression that is evaluated, then converted to a string, and then inserted where the expression comes in JSP file.
20. Define Composition. Composition has a stronger relationship with the object than Aggregation.
21. Define JSP Scriptlet. It a JSP tag that encloses Java code in JSP pages. Their syntax is <% %>. Code written in Scriptlet executes every time the program is run.
22. How can information from one JSP be passed to another JSP? The tag <Jsp:param> allows us to pass information between multiple Jsp's.
23. Explain the uses of <jsp: usebean> tag.

```
<jsp:usebean>
 id="beanInstName"
 scope= "page | application"
 class="ABC.class"  type="ABC.class"
```

</jsp: usebean>
This tag creates an instance of java bean. It firstly tries to find if bean instance already exist and assign stores a reference in the variable. Type is also specified; otherwise it instantiates from the specified class storing a reference in the new variable.
24. Explain handling of runtime exceptions. Errorpage attribute is used to uncatch the run-time exceptions forwarded automatically to an error processing page. It redirects the browser to JSP page error.jsp if any uncaught exception is faces during request handling. It is an error processing page.
25. Why does _jspService () start with a '_' but other lifecycle methods do not? Whatever content made in a jsp page goes inside the _jspService () method by the container. If it is override, the compiler gives an error, but the other 2 lifecycles can be easily override. So '_' shows that we cannot override this method.
26. Explain the various scope values for <jsp: usebean> tag. <jsp: usebean> tag is used to use any java object in the jsp page. Some scope values are:
1) application
2) request
3) page
4) session
27. Show the 2 types of comments in JSP. The 2 types are:

1. **<%–JSP Comment–%>**
2. **<!–HTML comment–>**

28. Can Static method be Override? We can declare static methods with same signature in subclass, but it is not considered overriding as there won't be any run-time polymorphism. Hence the answer is 'No'.
29. Explain JSP directives. JSP directives are messages to JSP Engine. They serve as a message from page to container and control the processing of the entire page. They can set global values like class declaration. They do not produce output and are enclosed in < %@….%>
30. Explain page Directives. Page Directives inform the JSP Engine about headers and facilities that the page receives from the environment. It is found at the top of all JSP pages. Its syntax is <%@ page attribute="value">
31. Show attributes of page directives.
1) Session : It shows if a session data is available to the page.
2) Import : it shows packages that are imported.
3) isELIgnored : It shows whether EL expressions are ignored when JSP translates into a servlet.
4) contentType : it allows the user to specify the content type of page.
32. What is Include directive? They include directive statically inserts the contents of a resource into the current JSP. It helps in the reuse of code without duplication. And includes contents of the file at translation time. Its syntax is as follows <%@ include file="Filename" %>.
33. What are standard actions in JSP? They affect overall runtime behavior of a page and response sent to the client. They are used to include a file at request time, to instantiate a JavaBean or find one. They are also used to generate a browser-specific code or forward a request to a new page.
34. Explain the jsp: setProperty action. It is used to give values to properties of beans that have been referenced beforehand.
<jsp: useBean id="ABC"…/> … <jsp: setProperty name="ABC" property="myProperty"…
jsp:setproperty is executed even if a new bean is instantiated or existing bean is found.
By adding </jsp.useBean> at the end of the code, the condition for execution is inverted i.e. It is not executed if existing object was found and only if a new object was instantiated.
35. Define Static Block. It is used to start the static data member. It is executed before classloading.
36. Explain jsp: plugin action. This action helps in insertion of a specific object in the browser or embed the element needed to specify the running of applet using Java plugin.
37. Explain client and server side validation. JavaScript is used for the client-side validation. It takes place within the browser. JavaScript is used to submit the form data if validation is successful. Validation errors require no extra network trip because form cannot be submitted. Validation is also carried out in the server after submission. If validation fails, extra network trip is required to resend the form to the client.
38. What is Translation Phase? JSP engine translates and compiles a JSP file to a servlet. This servlet moves to the execution phase where requests and responses are handled. They are compiled for the first time they are accessed unless manually compiled ahead of time. The manual or explicit compilation is useful for long and convoluted programs.
39. Perform a Browser Redirection from a JSP Page. <% response.sendRedirect (URL); %>
Or we can change the location of the HTTP header attribute as follows:
<% response.setStatus (HttpServletResponse.SC_MOVED_PERMANENTLY); response.setHeader (URL); %>
40. Give uses of Object Cloning. Object cloning is used to create an exact copy of an object by typing the same code or using various other techniques.
41. How to forward a request to another source. <jsp: forward page="/Page2.jsp" />

42. How can Automatic creation of session be prevented in a JSP page? JSP page automatically create sessions for requests. By typing the following, it can be avoided. **<%@ page session="false" %>**
43. How can you avoid Scriptlet code in JSP? JavaBeans or Custom Tags can be used instead of Scriptlet code.
44. Explain the jspDestroy () method. Whenever a JSP page is about to be destroyed, the container invokes the jspDestroy () method from the javax.servlet.jsp.JspPage interface. Servlets destroy methods are similar to it. It can be easily overridden to perform cleanup, like when closing a database connection.
45. Explain the <jsp: param> action. It is an action used with include or forward standard actions. It helps in passing the parameter names and values to a resource.
46. Explain static method. A static method is of the class and not the object of a class. It can be invoked without instance of a class. Static members can also access the static data and change its value.
47. How to disable scripting? Scripting can be easily disabled by setting scripting-invalid element of the deployment descriptor to true. It is a sub-element of property group. It can be false as well.
48. Define JSP Declaration. JSP Declaration are tags used in declaring variables. They are enclosed in < %!%> Tag. They are used in declaring functions and variables.

```
<%@page contentType="text/html" %>
 <html><body>
 <%!
   int a=0;
   private int getCount(){
       a++;
     return a;
 }
 %>
 <p>Values of a are:</p>
 <p><%=getCount()%></p>
 </body> </html>
```

49. How can HTML Output be prevented from being cached?

```
<% response.setHeader("Cache-Control", "no=store");
     response.setDateHeader("Expires", 0);
 %>
```

50. How is JSP better than Servlet technology? JSP is a technology on the server's side to make content generation simple. They are document centric, whereas servlets are programs. A Java server page can contain fragments of Java program, which execute and instantiate Java classes. However, they occur inside HTML template file. It provides the framework for development of a Web Application.
1) Explain what is REST and RESTFUL? REST represents REpresentational State Transfer; it is a relatively new aspect of writing web API. RESTFUL is referred for web services written by applying REST architectural concept are called Restful services, it focuses on system resources and how state of resource should be transported over HTTP protocol to different clients written in different language. In RESTFUL web service HTTP methods like GET, POST, PUT and DELETE can be used to perform CRUD operations.
2) Explain the architectural style for creating web API? The architectural style for creating web api are

1. HTTP for client server communication
2. XML/JSON as formatting language
3. Simple URI as the address for the services
4. Stateless communication

3) Mention what tools are required to test your web API? SOAPUI tool for SOAP WS and Firefox "poster" plugin for RESTFUL services.
4) Mention what are the HTTP methods supported by REST? HTTP methods supported by REST are:

1. GET: It requests a resource at the request URL. It should not contain a request body as it will be discarded. Maybe it can be cached locally or on the server.
2. POST: It submits information to the service for processing; it should typically return the modified or new resource
3. PUT: At the request URL it update the resource
4. DELETE: At the request URL it removes the resource
5. OPTIONS: It indicates which techniques are supported
6. HEAD: About the request URL it returns meta information

5) Mention whether you can use GET request instead of PUT to create a resource? No, you are not supposed to use POST or GET. GET operations should only have view rights

6) Mention what are resources in a REST architecture? Resources are identified by logical URLs; it is the key element of a RESTful design. Unlike, SOAP web services in REST, you view the product data as a resource and this resource should contain all the required information.

7) Mention what is the difference between AJAX and REST?

AJAX	REST
• In Ajax, the request are sent to the server by using XMLHttpRequest objects. The response is used by the JavaScript code to dynamically alter the current page • Ajax is a set of technology; it is a technique of dynamically updating parts of UI without having to reload the page • Ajax eliminates the interaction between the customer and server asynchronously • REST requires the interaction between the customer and server	• REST have a URL structure and a request/response pattern the revolve around the use of resources • REST is a type of software architecture and a method for users to request data or information from servers • REST requires the interaction between the customer and server

7) Mention some key characteristics of REST? Some key characteristics of REST includes

1. REST is stateless, therefore the SERVER has no state (or session data)
2. With a well-applied REST API, the server could be restarted between two calls as every data is passed to the server
3. Web service mostly uses POST method to make operations, whereas REST uses GET to access resources

8) Mention what are the different application integration styles? The different integration styles include

1. Shared database
2. Batch file transfer
3. Invoking remote procedure (RPC)
4. Swapping asynchronous messages over a message oriented middle-ware (MOM)

9) Explain how JAXB related to RESTful web API? JAXB stands for java arch for XML binding.

10) Mention what is the difference between PUT and POST? "PUT" puts a file or resource at a particular URI and exactly at that URI. If there is already a file or resource at that URI, PUT changes that file or resource. If there is no resource or file there, PUT makes one POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource PUT is idempotent meaning, invoking it any number of times will not have an impact on resources. However, POST is not idempotent, meaning if you invoke POST multiple times it keeps creating more resources

11) Mention which markup language can be used in restful web api? JSON and XML are the two markup language that can be used in restful web api

12) Mention what is the difference between RPC or document style web services? How you determine to which one to choose? In document style web services, we can transport an XML message as part of SOAP request which is not possible in RPC style web service. Document style web service is most appropriate in some application where XML message behaves as document and content of that document can alter and intention of web service does not rely on the content of XML message.

13) Mention what is JAX-WS and JAX-RS? Both JAX-WS and JAX-RS are libraries (APIs) for doing communication in various ways in Java. JAX-WS is a library that can be used to do SOAP communication in JAVA, and JAX-RS lets you do the REST communication in JAVA.

14) List out the tools or API for developing or testing web api? Testing tools for web services for REST APIs includes

1. Spring REST web service using MVC
2. Jersey API
3. CFX
4. Axis
5. Restlet,

15) Mention what is the difference between SOAP and REST?

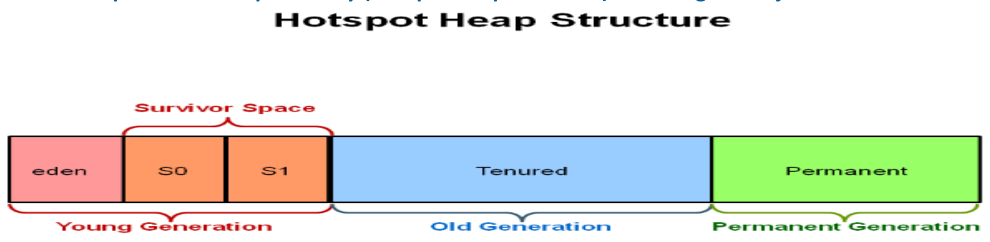
SOAP	REST

• SOAP is a protocol through which two computer communicates by sharing XML document	• Rest is a service architecture and design for network-based software architectures
• SOAP permits only XML	• REST supports many different data formats
• SOAP based reads cannot be cached	• REST reads can be cached
• SOAP is like custom desktop application, closely connected to the server	• A REST client is more like a browser; it knows how to standardized methods and an application has to fit inside it
• SOAP is slower than REST	• REST is faster than SOAP
• It runs on HTTP but envelopes the message	• It uses the HTTP headers to hold meta information

Question 1. Where are objects created in memory? On stack or heap? All Java objects are always created on **heap** in java.
Question 2. What is Garbage Collection process in java? GC (Garbage collection) is the process by which JVM cleans objects (unused objects) from heap to reclaim heap space in java.
Question 3. What is Automatic Garbage Collection in JVM heap memory in java? Automatic garbage collection is the process of

- Identifying objects which are in use in java heap memory and
- Which objects are not in use in java heap memory and
- Deleting the unused objects in java heap memory.

Question 4. How to identify objects which are in use in JVM heap memory in java? Objects in use (or referenced objects) are those objects which are still needed by java program, some part of java program is still pointing to that object.
Question 5. Which objects are not in use in JVM heap memory in java? Objects not in use (or **unreferenced objects**) are those objects which are not needed by java program, no part of java program is pointing to that object.
So, these unused objects can be cleaned in GC (garbage collection) process and memory used by an unreferenced object can be reclaimed.
Question 6. Explain JVM Heap memory (Hotspot heap structure) with diagram in java?



Question 7. What is Throughput in GC (garbage collection) in java? Throughput focuses on maximizing the amount of work by an application in a specific period of time. Examples of how throughput might be measured include

- The number of transactions completed in a given time.
- The number of jobs that a batch program can complete in an hour.
- The number of database queries that can be completed in an hour.

Question 8. What are pauses in GC (garbage collection) in java? Pauses is applications pauses i.e. when application doesn't gives any response because of garbage collection (GC).
Question 9. What is Young Generation in JVM Heap memory in java? New objects are allocated in Young generation. **Young** Generation consists of >
1a) **Eden**,
1b) **S0 (Survivor** space 0**)**
1c) **S1 (Survivor** space 1**)**
Question 10. What is Old Generation in JVM Heap memory in java? The **Old Generation (tenured generation)** is used for storing the long surviving aged objects. Major garbage collection occurs in Old Generation. When the old generation fills up, this causes a **major garbage collection**. Objects are cleaned up from old generation.
Question 11. At what time (or what age) objects are moved from young to old generation in JVM heap?
There is some threshold set for young generation object and when that age is met, the object gets moved to the old generation.
Question 12. What is Permanent Generation in JVM Heap memory in java? Permanent generation (Permgen) Space contains metadata required by JVM to describe the classes and methods used in the application. The permanent generation is included in a full garbage collection in java. The permanent generation space is populated at runtime by JVM based on classes in use in the application. The permanent generation space also contains Java SE library classes and methods in java.JVM garbage collects those classes when classes are no longer required and space may be needed for other classes in java.

Question 13. What is Minor garbage collection in JVM Heap memory in java? Minor garbage collection occurs in Young Generation. When the young generation fills up, this causes a minor garbage collection. **All the non-daemon threads running in application are stopped during minor garbage collections. Daemon threads performs minor garbage collection**.
Question 14. What is Stop the World Event? Minor garbage collections are called **Stop the World** events.
Question 15. What is Major garbage collection in JVM Heap memory in java? Major garbage collection occurs in Old Generation. The Old Generation is used for storing the long surviving aged objects. When the old generation fills up, this causes a major garbage collection. Objects are cleaned up from old generation. Major collection is much slower than minor garbage collection in jvm heap because it involves all live objects.
Question 16. Major garbage collection are Stop the World Event in java? Major garbage collections are also called Stop the World events. All the non-daemon threads running in application are stopped during major garbage collections. Daemon threads performs major garbage collection.
Question 17. Major garbage collections in responsive applications in java? Major garbage collections should be minimized for responsive applications because applications must not be stopped for long.
Question 18. Optimizing Major garbage collections in responsive applications in java? **Selection of appropriate garbage collector for the old generation** space affects the length of the "Stop the World" event for a major garbage collection.
Question 19. What is Full garbage collection in JVM Heap memory in java? Full garbage collection occurs in permanent generation in java. Permanent generation Space contains metadata required by JVM to describe the classes and methods used in the application. The permanent generation space is populated at runtime by JVM based on classes in use in the application. JVM garbage collects those classes when classes are no longer required and space may be needed for other classes in java.
Question 20. Mention some of the most important VM (JVM) PARAMETERS you have used for Young Generation in JVM Heap memory?
-Xmn: -Xmn sets the size of young generation.
  Example: - java -Xmn512m MyJavaProgram
-XX:NewRatio: NewRatio controls the size of young generation.
  Example -XX:NewRatio=3 means that the ratio between the young and old/tenured generation is 1:3.
-XX:NewSize :- NewSize is minimum size of young generation which is allocated at initialization of JVM.
-XX:MaxNewSize :- MaxNewSize is the maximum size of young generation that JVM can use.
-XX:SurvivorRatio : (for survivor space) Survivor Ratio can be used to tune the size of the survivor spaces, but this is often not as important for performance.
  Example: - -XX:SurvivorRatio=6 sets the ratio between each survivor space and Eden to be 1:6.
Question 21. Mention some of the most important VM (JVM) PARAMETERS you have used for Old Generation (tenured) in JVM Heap memory? -XX:NewRatio : NewRatio controls the size of young and old generation.
  Example of using -XX:NewRatio, -XX:NewRatio=3 means that the ratio between the young and old/tenured generation is 1:3.
Question 22. Mention some of the most important VM (JVM) PARAMETERS you have used for Permanent Generation?
-XX:PermSize: It's is initial value of Permanent Space which is allocated at startup of JVM.
  Example java -XX:PermSize=512m MyJavaProgram
It will set initial value of Permanent Space as 512 megabytes to JVM
-XX:MaxPermSize: It's maximum value of Permanent Space that JVM can allot up to.
  Example java -XX:MaxPermSize=512m MyJavaProgram
It will set maximum value of Permanent Space as 512 megabytes to JVM
Question 23. What are different Garbage collectors available in JVM?

- Serial collector / Serial GC (Garbage collector) in java
- Throughput GC (Garbage collector) or Parallel collector in java
- Concurrent Mark Sweep (CMS) collector / concurrent low pause garbage collector
- G1 garbage collector / Garbage first collector
- PS Scavenge
- PS MarkSweep
- ParNew collector

Question 24. What is Serial collector / Serial GC (Garbage collector) in java? Features of Serial GC (Garbage collector) in java

- Serial collector is also called Serial GC (Garbage collector) in java.
- Serial GC (Garbage collector) is rarely used in java.
- Serial GC is designed for the single threaded environments in java.
- In Serial GC (Garbage collector), both minor and major garbage collections are done serially by one thread in java.
- Serial GC uses a mark-compact collection method. The serial garbage collector is the default for client style machines in Java SE 5 and 6.

Question 25. When to Use the Serial GC (garbage Collector) in java?

- The Serial GC is the garbage collector of choice for most applications that do not have low pause time requirements and run on client-style machines.
- Serial garbage collector is also popular in environments where a high number of JVMs are run on the same machine.

Vm (JVM) option for enabling serial GC (garbage Collector) in java: - -XX:+UseSerialGC
Example of Passing Serial GC in Command Line for starting jar:-
java -Xms256m -Xms512m -XX:+UseSerialGC -jar d:\MyJar.jar

Question 26. What is Throughput GC (Garbage collector) or Parallel collector in java?
- Throughput garbage collector is the default garbage collector for JVM in java.
- Throughput garbage collector uses multiple threads to execute a minor collection and so reduces the serial execution time of the application in java.

Question 27. When to Use the Throughput GC (garbage Collector) in java? The Throughput garbage collector should be used when application can afford low pauses in java. And application is running on host with multiple CPU's in java. **-XX:+UseParallelGC**

Question 28. What is Concurrent Mark Sweep (CMS) Collector / concurrent low pause collector in java?
- Concurrent Mark Sweep (CMS) collector collects the old/tenured generation in java.
- Concurrent Mark Sweep (CMS) Collector minimize the pauses by doing most of the garbage collection work concurrently with the application threads in java.
- **Concurrent Mark Sweep (CMS) Collector on live objects >** Concurrent Mark Sweep (CMS) Collector does not copy or compact the live objects. A garbage collection is done without moving the live objects. If fragmentation becomes a problem, allocate a larger heap in java.

Question 29. When to Use the Concurrent Mark Sweep GC (garbage Collector) in java? -   XX:+UseConcMarkSweepGC
- Concurrent Low Pause Collector should be used if your applications that require low garbage collection pause times in java.
- Concurrent Low Pause Collector should be used when your application can afford to share processor resources with the garbage collector while the application is running in java.
- Concurrent Low Pause Collector is beneficial to applications which have a relatively large set of long-lived data (a large tenured generation) and run on machines with two or more processors in java.
- **Heap Structure for CMS garbage Collector: - young** generation, **old** generation, and **permanent** generation of a fixed memory size.

Question 30. What is G1 Garbage Collector (or Garbage First) in java? -XX:+UseG1GC
- G1 garbage collector was introduced in Java 7
- G1 garbage collector - default garbage collector in Java 9
- G1 garbage collector was designed to replace CMS collector (Concurrent Mark-Sweep garbage Collector).
- G1 garbage collector is parallel, concurrent, and incrementally compacting low-pause garbage collector in java.
- G1 garbage collector has much better layout from the other garbage collectors like serial, throughput and CMS garbage collectors in java.
- G1 (Garbage First) collector compacts sufficiently to completely avoid the use of fine-grained free lists for allocation, and instead relies on regions.
- G1 (Garbage First) collector allows customizations by allowing users to specify pause times.
- G1 Garbage Collector (or Garbage First) limits GC pause times and maximizes throughput.

Question 31. G1 (Garbage First) collector functioning: - The heap is split/partitioned into many fixed sized regions (Eden, survivor, old generation regions), but there is not a fixed size for them. This provides greater flexibility in memory usage.

Question 32. When to use G1 garbage collector? G1 must be used when applications that require large heaps with limited GC latency. Application that require heaps around 5-6GB or larger and pause time required below 0.5 seconds.

Question 33. When to switch from CMS (or old garbage collectors) to G1 garbage collector?
- Full GC durations are too long or too frequent.
- The rate of object allocation or promotion varies significantly.
- Long garbage collection (longer than 0.5 to 1 second)

Question 34. What are Difference Serial and Throughput GC (garbage Collector)?
- Serial collector uses one thread to execute garbage collection. Throughput collector uses multiple threads to execute garbage collection.
- Serial GC is the garbage collector of choice for applications that do not have low pause time requirements and run on client-style machines. Throughput GC is the garbage collector of choice for applications that have low pause time requirements.

Question 35. Which methods is called for garbage collection in java? GC (garbage collector) calls finalize method for garbage collection. Finalize method is called only once by garbage collector for an object in java.
Finalize method is in **java.lang.Object class**.

```
public class RunGarbageCollectorExample {
  public static void main(String[] args) {
        System.out.println("About to call garbage collection - using System.gc() method");
        System.gc();
        System.out.println("garbage collection - using System.gc() method called");
  }
}
/*OUTPUT
About to call garbage collection - using System.gc() method
garbage collection - using System.gc() method called
*/
```

Question 36. Can we force early garbage collection in java? Yes, by using following methods:-

- System.gc();
- Runtime.getRuntime().gc();
- System.runFinalization();
- Runtime.getRuntime().runFinalization();

Question 37. Is it good practice to call System.gc () in Java? No

- First of all calling System.gc () does not guarantee that it will immediately start performing garbage collection.
- Even calling System.gc () may not do anything. Call to perform garbage collection may be ignored completely.
- JVM is different for different platforms because java is platform independent language, so you never know about which garbage collector your jvm will run i.e. what algorithm does it follow to perform garbage collection.

Question 38. Is garbage collection guaranteed when we call finalize () methods? Calling finalize methods does not guarantee that it will immediately start performing garbage collection.

Question 39. Which thread performs garbage collection in java? Daemon threads are low priority threads which runs intermittently in background for doing garbage collection (GC) in java.

Question 40. Tell us something about ParNew collector in java? ParNew collector is the young generation collector. It is the parallel copy collector, it uses multiple threads in parallel. Vm parameter for enabling ParNew collector is **-X:+UseParNewGC**.

Question 41. Do you know about PS Scavenge and PS MarkSweep in java?

PS Scavenge

- PS Scavenge is the Young generation collectors
- It is the parallel scavenge collector.
- PS Scavenge uses multiple threads in parallel for garbage collection.
- Vm parameter for enabling PS Scavenge **-XX:+UseParallelGC**

PS MarkSweep

- PS MarkSweep is the old generation collector.
- PS MarkSweep is the parallel scavenge mark sweep collector.
- It uses the multiple threads in parallel for garbage collection.
- Vm parameter for enabling PS MarkSweep **-XX:+UseParallelOldGC**

Question 42. How garbage collection is done using Marking and deletion in java?

1) **Marking: -** Marking is a process in which GC (garbage collector) identifies which parts of memory (occupied by objects) are in use and which are not.
2) **Deletion :-**

Step 2a: Normal Deletion: - Normal deletion removes all the unreferenced objects and leaves referenced objects and pointers to free space.

Step 2b: Deletion with Compacting: - Deletion with Compacting is done to improve the performance than normal deleting. Deletion with Compacting removes all the unreferenced objects and compacts the remaining referenced objects by moving all the referenced objects together.

Question 43. Can you make objects eligible for garbage collection in java? Object which is set explicitly set to **null** becomes **eligible for GC** (garbage collection) in java.

Example 1 >

String s=”abc”; //s is currently not eligible for GC (garbage collection) in java.
s = null;   //Now, s is currently eligible for GC (garbage collection) in java.

Example 2 >

List list =new ArrayList (); //list is currently not eligible for GC (garbage collection).
list = null;   //Now, list is currently eligible for gc (garbage collection).

Question 44. Does variables declared inside block becomes eligible for GC (garbage collection) when we exit that block in java? Yes

```
class MyClass {
  public static void main(String[] args) {
        boolean var = false;
        if (var) { // begin block 1
              int x = 1; // x is declared inside block
              //code inside block...
        } // end block 1 //And now x is eligible for gc (garbage collection)
        else { // begin block 2
              int y = 1;
              //code inside block...
        } // end block 2 //And now y is eligible for gc (garbage collection)
  }
}
```

Question 45. What is monitoring and analyzing the garbage collection in java? Monitoring and analyzing garbage collection in java can be used to get following information:-

- Understanding the garbage collection process.
- Understanding how JVM is currently working.
- What type of garbage collection algorithms are used
- Improving garbage collection performance,
- Analyzing Java heap dumps,
- Monitoring live Java applications,
- Analyze profiling data,
- Detecting Memory leak for classes and arrays,
- Detecting Thread deadlock,
- Detecting abnormal thread termination,
- Detecting OutOfMemoryError problems,
- Finding System and process CPU utilization thresholds,
- Find Heap usage thresholds.
- Find time taken in Garbage collection and Finalizer queue length.

Question 46. How to monitor and analyze the garbage collection in java? We can use different tools to generate the garbage collection information and then analyze it.

- **VisualVM -** It helps us in analyzing threads performance, thread states, CPU consumed by threads etc.
- **JSTACK** - Java stack traces
- **-verbose:gc VM argument**
- **Jstat - Java Virtual Machine Statistics Monitoring Tool: -** "The jstat command displays performance statistics for an instrumented Java HotSpot VM. The target JVM is identified by its virtual machine identifier, or vmid option."
- **JHAT - Java Heap Analysis Tool.**
- **jconsole** - jconsole option can be used to obtain a heap dump.
- **hprof**
- **eclipse plugin**
- **HPjmeter**
- **JFR (Java Flight Recorder)** can be used for detecting memory leak.

Question 47. What is memory leak in java? Consequences of memory leak? Memory leak happens when number of objects (these objects are not needed) created becomes large and time spent in garbage collection increases. Ultimately application becomes very slow, non-responsive and ends up **throwing OutOfMemoryError**. **"Memory leaks ends up throwing OutOfMemoryError but OutOfMemoryError doesn't means memory leak in java". Consequences of memory leak:-**

- Application becomes very slow.
- Time spent in garbage collection increases.

Question 48. Can you please explain some scenarios where you have faced memory leak, OR scenarios where memory leak could happen in java?

1) Static variables/ fields are not garbage collected and can cause memory leak in java
2) Thread Local Variables can cause memory leak in java
3) Memory leak while using Autoboxing and unboxing in java
4) Avoid memory leak using WeakHashMap in java
5) Using custom key in map without Overriding equals() and hashcode() method can cause memory leak
6) Close JDBC Statement, PreparedStatement, CallableStatement , ResultSet and Connections in java to avoid memory leaks
7) Memory leaks can also be caused by native methods in java

8) Memory leak in web applications in java

Question 49. Comment on relation between OutOfMemoryError and garbage collection in java? OutOfMemoryError may be thrown when an excessive amount of time is being by jvm in performing garbage collection and very little memory is being freed. A long lived application might be unintentionally holding references to objects and this prevents the objects from being garbage collected. Holding of objects for a long time is also a kind of memory leak in java.

Question 50. Discuss OutOfMemoryError: GC Overhead limit exceeded in java? OutOfMemoryError: GC Overhead limit exceeded - indicates that the garbage collector is running all the time and Java program is making very slow progress. After a GC (garbage collection), if the garbage collector is spending more than 98% of its time in doing garbage collection and if less than 2% of the java heap memory space is reclaimed, then OutOfMemoryError - GC Overhead limit exceeded - is thrown in java. This OutOfMemoryError is generally thrown because all the live objects are not getting garbage collected properly and java heap space is not available for new objects.

Question 51. What you know about OutOfMemoryError: permgen in java? Generally when we are facing java.lang.OutOfMemoryError - Java permgen space, then we need to change permgen size of tomcat or eclipse or JVM wherever you are facing this error.

Question 52. How to Solve OutOfMemoryError: unable to create new native Thread by passing appropriate jvm parameter?

Solution 1:- Try to increase the the -Xss value so that new threads gets enough stack space.

Solution 2:- you could also increase the heap size available using -Xms and -Xmx options and then try to increase and set appropriate -Xss value.

Example of using –Xss: - java -Xss512m MyJavaProgram

It will set the default stack size of JVM to 512 megabytes.

Question 53. How to Solve OutOfMemoryError: Java heap space by passing appropriate jvm parameter? OutOfMemoryError: Java heap space - is thrown whenever there is insufficient space to allocate an object in the Java heap. Increase the heap size using -Xms and -Xmx jvm parameters as a solution to this issue.

Question 54. Does Exception in thread threadName - java.lang.OutOfMemoryError - Java heap space indicates memory leak? Increase the heap size using -Xms and -Xmx jvm parameters as a solution to this issue.

Question 55. How to set appropriate heap size in eclipse in java? We can make changes in eclipse.ini file.

Where we can configure

- -Xms (minimum heap size which is allocated at initialization of JVM),
- -Xmx (maximum heap size that JVM can use.)
- -XX:MaxPermSize: It's maximum value of Permanent Space that JVM can allot up to.

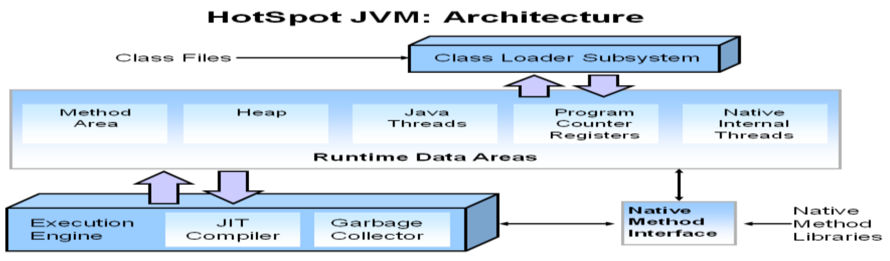
Question 56. What is default garbage collectors for Java 7 / Java 8?

- For server class machine - parallel collector.
- For client class machine - serial collector.

Question 57. What is default garbage collectors for Java 9? G1 garbage collector

Question 1. What is JVM in java? JVM stands for Java virtual machine. JVM is the virtual machine on which java code executes. JVM is responsible for converting byte code into machine specific code.

Question 2. Discuss HotSpot JVM (Java Virtual Machine) Architecture in short? JVM (Java Virtual Machine) consists of **Class Loader Subsystem**, **Runtime Data Areas** and **Execution Engine**.



1) Class Loader Subsystem: - Classloader is used to load class files.

2) Runtime Data Areas:-

2.1) Method Area: - Method area stores data for each and every class like fields, constant pool, method's data and information. Method area is also called class area.

2.2) Heap: - Heap is place where all objects are stored in JVM. Heap even contains arrays because arrays are objects.

2.3) Java Threads (Java thread Stacks):- Whenever new method is called new stack frame is created and it is pushed on top of that thread's stack.

2.4) Program counter registers (PC Registers):- the address of instructions currently and next address being executed.

2.5) Native internal Threads (Native thread stack):- Native internal threads area contains all the informations related to native platform.

3) Execution Engine of JVM:-

3.1) JIT(Just In Time) compiler:- JIT compiler compiles bytecodes to machine code at run time and improves the performance of Java applications.

3.2) Garbage Collector: - Garbage Collector Garbage collection is the process by which JVM clears objects (unused objects) from heap to reclaim heap space.

4) Native method libraries of JVM: - Native method interface is an interface that connects JVM with the native method libraries for executing native methods.

5) JNI, What is Java Native Interface (JNI)? : - Programmers uses the JNI (Java Native Interface) to write the Java native methods when an application cannot be written purely in Java.

Question 3. What is Class Loader Subsystem of JVM? What is its functioning and purpose? Classloader is a subsystem of JVM. Classloader is used to load class files. Classloader verifies the class file using byte code verifier. Class file will only be loaded if it is valid.

Question 4. How stack frames are created when thread calls new method? As we know each and every thread has its own stack. Whenever new method is called new stack frame is created and it is pushed on top of that thread's stack.

Question 5. What does thread stack contains? The stack contain:-

- All the local variables,
- All the parameters,
- All the return address.

Question 6. Does stack stores/contains object OR what stack doesn't contains? Stack never stores object, but it stores object reference.

Question 7. JIT Compiler internal working? JIT compilation does require processor time and memory usage. When the JVM first starts up, lots of methods are called. Compiling all of these methods might can affect startup time significantly, though program ultimately may achieve good performance. Methods are not compiled when they are called first time. For each and every method JVM maintains a call count, which is incremented every time the method is called. The methods are interpreted by JVM until call count not exceeds JIT compilation threshold (The JIT compilation threshold improves performance and helps the JVM to start quickly. The threshold has been selected carefully by java developers to obtain an optimal performances. Balance between startup times and long term performance is maintained). Therefore, very frequently used methods are compiled as soon as JVM has started, and less used methods are compiled later.

Question 8. What are most important/key HotSpot JVM components related to performance? Heap, JIT (Just in Time) Compiler and Garbage collector.

- **Heap and Garbage collector for tuning JVM's performance: -** All the objects are stored in heap. Garbage collector manages the heap at JVM initialization.
- **JIT (Just in Time) Compiler for tuning JVM's performance: -** JIT compiler compiles bytecodes to machine code at run time and improves the performance of Java applications.

Question 9. How JIT improves performance of Most frequently used methods? After a method is compiled, its call count is reset to zero and subsequent calls to the method increment it call count. When the call count of a method reaches a JIT recompilation threshold, the JIT compiler compiles method second time, applying more optimizations as compared to optimizations applied in previous compilation. This process is repeated until the maximum optimization level is reached. Most frequently used methods are always optimized to maximize the performance benefits of using the JIT compiler.

Question 10. What is Difference between JDK, JRE and JVM?

JDK	**JRE**	**JVM**
Java Development Kit	Java Runtime environment	java virtual machine
JDK is required for java development.	JRE provides environment for running/executing programs.	JVM is the virtual machine on which java code executes. JVM is responsible for converting byte code into machine specific code.
We need JDK in your system for: - developing, compiling and Running Java programs.	You need to have JRE in your system for: - Running Java programs.	-

JDK contains- JDK = JRE + JVM	**JRE contains-** JRE = JVM + class libraries (rt.jar) + other libraries (if any).	

Question 11. What are -XX: MinHeapFreeRatio and -XX: MaxHeapFreeRatio vm parameters used for? JVM can grows or shrinks the heap to keep the proportion of free space to live objects within a specific range.

Question 12. What is significance of -XX:+AggressiveHeap VM parameter in java? -XX:+AggressiveHeap is used for Garbage Collection Tuning setting.

Overriding equals and hashcode method

Bucket is ArrayList of Entry.

Entry is LinkedList which contains information about key, value and next.

Entry.next points to next Entry in LinkedList.

Question1. Why do we need to override equals and hashcode method? It's important to override equals () and hashCode () method of class if we are want to use class as key in HashMap. If we don't override equals () and hashCode () method we will be able to put object, but we won't be able to retrieve object.

Question2. Why to override hashcode method? It helps in finding bucket location, where entry (with key-value pair) will be stored .Entry (of type LinkedList) is stored in bucket (ArrayList).

If, hashCode() method is overridden properly, we will find bucket location using hashCode() method, we will obtain Entry on that bucket location, then iterate over each and every Entry (by calling Entry.next) and check whether new and existing keys are equal or not. If keys are equal replace key-value in Entry with new one, else call Entry.next. But, now the question comes how to check whether two keys are equal or not. So, it's time to implement equals () method.

If, hashcode method is not overridden for same key every time hashCode () method is called it might produce different hashcode, there might happen **2 cases** i.e. when **put and get method** are called.

Case 1: when put () method is called- There might be possibility that same Entry (with key-value pair) will get stored at multiple locations in bucket. ***Conclusion>*** key- value pair may get stored multiple times in HashMap.

Case 2: when get () method is called- As there is possibility that hashCode () method might return different hashcode & rather than searching on bucket location where Entry (with key) exists we might be searching for key on some other bucket location. ***Conclusion>*** key existed in HashMap, but still we were not able to locate the bucket location in which it was stored.

Question3. Why to override equals method? Once we have located bucket location in which our Entry (with key-value pair) will be stored, Equals method helps us in finding whether new and existing keys are equal or not.

If we equals method is not overridden - though we will be able to find out correct bucket location if hashCode () method is overridden correctly, but still if equals method is not overridden, there might happen **2 cases** i.e. when **put and get method** are called.

Case 1: when put () method is called-we might end up storing new Entry (with new key-value pair) multiple times on same bucket location (because of absence of equals method, we don't have any way of comparing key's), In this case, even if keys are equal, we will keep on calling Entry.next until we reach last Entry on that bucket location and ultimately we will end up storing new Entry (with new key) again in same bucket location. ***Conclusion>*** key- value pair stored multiple times in HashMap.

Case 2 : when get() method is called- we won't be able to compare two keys (new key with existing **Entry**.key) and we will call **Entry**.next and again we won't be able to compare two keys and ultimately when Entry.next is null - we will return **false**.
Conclusion> key existed in HashMap, but still we were not able to retrieve it.

Question4. How do we override equals and hashcode method, write a code to use Employee as key in HashMap?

```
@Override
  public boolean equals(Object obj){
       if(obj==null)
            return false;
       if(this.getClass()!=obj.getClass())
            return false;
       Employee emp=(Employee)obj;
       return (emp.id==this.id || emp.id.equals(this.id))
                     && (emp.name==this.name || emp.name.equals(this.name));
  }
  @Override
  public int hashCode(){
       int hash=(this.id==null ? 0: this.id.hashCode() ) +
                (this.name==null ? 0: this.name.hashCode() );
       return hash;
  }
```

Question5. What classes should i prefer to use a key in HashMap? We should prefer String, Integer, Long, Double, Float, Short and any other wrapper class. Reason behind using them as a key is that they override equals () and hashCode () method, we need not to write any explicit code for overriding equals () and hashCode () method.

```
import java.util.HashMap;
import java.util.Map;
public class StringInMap {
  public static void main(String...a){
      //HashMap's key=Integer class  (Integer's api has already overridden hashCode() and equals() method for us )
       Map<Integer, String> hm=new HashMap<Integer, String>();
       hm.put(1, "data");
       hm.put(1, "data OVERRIDDEN");
       System.out.println(hm.get(1));
  }
}
/*OUTPUT
data OVERRIDDEN
*/
```

Let's check in Integer's API, how Integer class has overridden equals () and hashCode () method:

```
public int hashCode() {
     return value;
 }
public boolean equals(Object obj) {
     if (obj instanceof Integer) {
      return value == ((Integer)obj).intValue();
     }
     return false;
 }
```

Question6. If two objects have same hashcode, are they always equal? **No**, it's not necessary that object's having same hashcode are always equal. Because same hashcode means object are stored on same bucket location, as key/object in bucket is stored in Entry(Linked List)**,** key**/**object's might be stored on Entry.next

Question7. If two objects equals () method return true, do objects always have same hashcode? **Yes**, two objects can return true only if they are stored on same bucket location. First, hashCode () method must have returned same hashcode for both objects, than on that bucket location's Entry key.equals () is called, which returns true to confirm objects/keys are equal. So, if object's equals return true, they always have same hashcode.

Question8. Can two unequal objects have same hashcode? **Yes**, two unequal objects can have same hashcode.

Question9. What is difference between using instanceOf operator and getClass () in equals method? If we use instanceOf it will return true for comparing current class with its subclass as well, but getClass() will return true only if exactly same class is compared. Comparison with any subclass will return false.

Question10. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
  public Employee(String name) { // constructor
       this.name = name;
  }
  //no hashCode() method
  //no equals() method
  @Override
  public String toString() {
       return "Employee[ name=" + name + "] ";
  }
}
public class Program1 {
  public static void main(String...a){
       HashMap<Employee, String> hm=new HashMap<Employee, String>();
       hm.put(new Employee("a"), "emp1");
       hm.put(new Employee("b"), "emp2");
       hm.put(new Employee("a"), "emp1 OVERRIDDEN");
       System.out.println("HashMap's data> "+hm);
```

```
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=a] =emp1, Employee[ name=b] =emp2}
HashMap's size> 3
null
*/
```

Buckets= as hashCode () method is not there, hashcode generated for 3 objects will be different and we will end up using **3** buckets.
Size= as equals () method is not there, size will be **3**.
Get () =we won't be able to get object.

Question11. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
  public Employee(String name) { // constructor
        this.name = name;
  }
  @Override
  public int hashCode(){
        return (this.name==null ? 0: this.name.hashCode() );
  }
  @Override
  public boolean equals(Object obj){
        Employee emp=(Employee)obj;
        return (emp.name==this.name || emp.name.equals(this.name));
  }
  @Override
  public String toString() {
        return "Employee[ name=" + name + "] ";
  }
}
public class Program2 {
  public static void main(String...a){
        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");
        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=b] =emp2, Employee[ name=a] =emp1 OVERRIDDEN}
HashMap's size> 2
emp1 OVERRIDDEN
*/
```

Buckets= as hashCode () method is overridden perfectly, **2** bucket locations will be used.
Size= as equals () method is there, size will be **2**, value corresponding to Employee with id=1 and name='sam' was **employee1 data** & was overridden by value **employee1 data OVERRIDDEN**
Get () =we will be able to get object.

Question12. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
  public Employee(String name) { // constructor
        this.name = name;
  }
  @Override
  public int hashCode(){
       return 1;
  }
  @Override
  public boolean equals(Object obj){
       return true;
  }
  @Override
  public String toString() {
       return "Employee[ name=" + name + "] ";
  }
}
public class Program3 {
  public static void main(String...a){
       HashMap<Employee, String> hm=new HashMap<Employee, String>();
       hm.put(new Employee("a"), "emp1");
       hm.put(new Employee("b"), "emp2");
       hm.put(new Employee("a"), "emp1 OVERRIDDEN");
       System.out.println("HashMap's data> "+hm);
       System.out.println("HashMap's size> "+hm.size());
       System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=a] =emp1 OVERRIDDEN}
HashMap's size> 1
emp1 OVERRIDDEN
*/
```

Buckets= as hashCode () method returns 1, only **1** bucket location will be used.
Size= As equals() method always returns true, size will be **1**, all three employees will be stored on same bucket location in one Entry (new Entry will keep on overriding previous Entry). We will always get last stored key-value pair only.
Get () =we will be able to get object.

Question13. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
  public Employee(String name) { // constructor
       this.name = name;
  }
  @Override
  public int hashCode(){
       return 1;
  }
  //no equals() method
  @Override
  public String toString() {
       return "Employee[ name=" + name + "] ";
  }
}
public class Program4 {
  public static void main(String...a){
       HashMap<Employee, String> hm=new HashMap<Employee, String>();
       hm.put(new Employee("a"), "emp1");
       hm.put(new Employee("b"), "emp2");
       hm.put(new Employee("a"), "emp1 OVERRIDDEN");
       System.out.println("HashMap's data> "+hm);
       System.out.println("HashMap's size> "+hm.size());
```

```
        System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=b] =emp2, Employee[ name=a] =emp1}
HashMap's size> 3
null
*/
```

Buckets= as hashCode () method returns 1, only **1** bucket location will be used.
Size= as equals () method doesn't exist, size will be **3**, all three employees will be stored on same bucket location but in different Entry.
Get () =we won't be able to get object.

Question14. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
  public Employee(String name) { // constructor
        this.name = name;
  }
   //no hashCode() method
  @Override
  public boolean equals(Object obj){
        return true;
  }
  @Override
  public String toString() {
        return "Employee[ name=" + name + "] ";
  }
}
public class Program5 {
  public static void main(String...a){
        HashMap<Employee, String> hm=new HashMap<Employee, String>();
        hm.put(new Employee("a"), "emp1");
        hm.put(new Employee("b"), "emp2");
        hm.put(new Employee("a"), "emp1 OVERRIDDEN");
        System.out.println("HashMap's data> "+hm);
        System.out.println("HashMap's size> "+hm.size());
        System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=b] =emp2, Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=a] =emp1}
HashMap's size> 3
null
*/
```

Buckets= as hashCode () method is not there, hashcode generated for 3 objects will be different and we will end up using **3** buckets.
Size= though equals () method is their (but because of hashCode () method's absence) **which always returns true**, we won't be able to locate correct bucket location for calling equals () method, so, size will be **3**.
Get () =we won't be able to get object.

Question15. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
  public Employee(String name) { // constructor
        this.name = name;
  }
  @Override
  public int hashCode(){
        return (this.name==null ? 0: this.name.hashCode() );
  }
  //no equals() method
  @Override
  public String toString() {
       return "Employee[ name=" + name + "] ";
  }
}
public class Program6 {
  public static void main(String...a){
       HashMap<Employee, String> hm=new HashMap<Employee, String>();
       hm.put(new Employee("a"), "emp1");
       hm.put(new Employee("b"), "emp2");
       hm.put(new Employee("a"), "emp1 OVERRIDDEN");
       System.out.println("HashMap's data> "+hm);
       System.out.println("HashMap's size> "+hm.size());
       System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=b] =emp2, Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=a] =emp1}
HashMap's size> 3
null
*/
```

Buckets= as hashCode () method is overridden perfectly, **2** bucket locations will be used.
Size= as equals () method is not there, size will be **3**,
Get () =we won't be able to get object.

Question16. How many buckets will be there and what will be size of HashMap?

```
class Employee {
  private String name;
    public Employee(String name) { // constructor
       this.name = name;
  }
  //no hashCode() method
  @Override
  public boolean equals(Object obj){
       Employee emp=(Employee)obj;
       return (emp.name==this.name || emp.name.equals(this.name));
  }
  @Override
  public String toString() {
       return "Employee[ name=" + name + "] ";
  }
}
public class Program7 {
  public static void main(String...a){
       HashMap<Employee, String> hm=new HashMap<Employee, String>();
       hm.put(new Employee("a"), "emp1");
       hm.put(new Employee("b"), "emp2");
       hm.put(new Employee("a"), "emp1 OVERRIDDEN");
       System.out.println("HashMap's data> "+hm);
       System.out.println("HashMap's size> "+hm.size());
       System.out.println(hm.get(new Employee("a")));
  }
}
/*OUTPUT
HashMap's data> {Employee[ name=a] =emp1, Employee[ name=a] =emp1 OVERRIDDEN, Employee[ name=b] =emp2}
HashMap's size> 3
null
```

```
 */
```
Buckets= as hashCode () method is not there, hashcode generated for 3 objects will be different and we will end up using **3** buckets.
Size= though equals () method is their (but because of hashCode () method's absence), we won't be able to locate correct bucket location for calling equals () method, so, size will be **3**.
Get () =we won't be able to get object.
1. What is the most important feature of Java? Java is a platform independent language.
2. What do you mean by platform independence? Platform independence means that we can write and compile the java code in one platform (e.g. Windows) and can execute the class in any other supported platform e.g. (Linux,Solaris,etc).
3. What is a JVM? JVM is Java Virtual Machine which is a run time environment for the compiled java class files.
4. Are JVM's platform independent? JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.
5. What is the difference between a JDK and a JVM? JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.
6. 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.
7. What is the base class of all classes? Java.lang.Object
8. Does Java support multiple inheritance? Java doesn't support multiple inheritance.
9. Is Java a pure object oriented language? Java uses primitive data types and hence is not a pure object oriented language.
10. Are arrays primitive data types? In Java, Arrays are objects.
11. What is difference between Path and Classpath? Path and Classpath are operating system level environment variables. Path is used define where the system can find the executables (.exe) files and classpath is used to specify the location .class files.
12. What are local variables? Local variables are those which are declared within a block of code like methods. Local variables should be initialized before accessing them.
13. What are instance variables? Instance variables are those which are defined at the class level. Instance variables need not be initialized before using them as they are automatically initialized to their default values.
14. 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; is an example for constant.
15. Should a main () method be compulsorily declared in all java classes? No not required. Main () method should be defined only if the source class is a java application.
16. What is the return type of the main () method? Main () method doesn't return anything hence declared void.
17. Why is the main () method declared static? Main () method is called by the JVM even before the instantiation of the class hence it is declared as static.
18. What is the argument of main () method? Main () method accepts an array of String object as argument.
19. Can a main () method be overloaded? Yes. You can have any number of main () methods with different method signature and implementation in the class.
20. Can a main () method be declared final? Yes. Any inheriting class will not be able to have its own default main () method.
21. Does the order of public and static declaration matter in main () method? No. It doesn't matter but void should always come before main ().
22. Can a source file contain more than one class declaration? Yes a single source file can contain any number of Class declarations but only one of the class can be declared as public.
23. What is a package? Package is a collection of related classes and interfaces. Package declaration should be first statement in a java class.
24. Which package is imported by default? Java.lang package is imported by default even without a package declaration.
25. Can a class declared as private be accessed outside its package? Not possible.
26. Can a class be declared as protected? The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in an interface cannot be declared protected.
27. What is the access scope of a protected method? A protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.
28. What is the purpose of declaring a variable as final? A final variable's value can't be changed. Final variables should be initialized before using them.
29. What is the impact of declaring a method as final? A method declared as final can't be overridden. A sub-class can't have the same method signature with a different implementation.
30. I don't want my class to be inherited by any other class. What should i do? You should declared your class as final. But you can't define your class as final, if it is an abstract class. A class declared as final can't be extended by any other class.

31. Can you give few examples of final classes defined in Java API? Java.lang.String, java.lang.Math are final classes.

32. Can a class be declared as static? We cannot declare top level class as static, but only inner class can be declared static.

```
public class Test{
   static class InnerClass   {
      public static void InnerMethod()   {
       System.out.println ("Static Inner Class!");
      }
   }
   public static void main(String args[])   {
     Test.InnerClass.InnerMethod();
   }
}
//output: Static Inner Class!
```

33. When will you define a method as static? When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.

34. What are the restriction imposed on a static method or a static block of code? A static method should not refer to instance variables without creating an instance and cannot use "this" operator to refer the instance.

35. 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.

36. What is the importance of static variable? Static variables are class level variables where all objects of the class refer to the same variable. If one object changes the value then the change gets reflected in all the objects.

37. Can we declare a static variable inside a method? Static variables are class level variables and they can't be declared inside a method. If declared, the class will not compile.

39. Can an abstract class be declared final? Not possible. An abstract class without being inherited is of no use and hence will result in compile time error.

40. What is use of a abstract variable? Variables can't be declared as abstract. Only classes and methods can be declared as abstract.

41. Can you create an object of an abstract class? Not possible. Abstract classes can't be instantiated.

42. Can an abstract class be defined without any abstract methods? Yes it's possible. This is basically to avoid instance creation of the class.

43. Class C implements Interface I containing method m1 and m2 declarations. Class C has provided implementation for method m2. Can i create an object of Class C? No not possible. Class C should provide implementation for all the methods in the Interface I. Since Class C didn't provide implementation for m1 method, it has to be declared as abstract. Abstract classes can't be instantiated.

44. Can a method inside a Interface be declared as final? No not possible. Doing so will result in compilation error. Public and abstract are the only applicable modifiers for method declaration in an interface.

45. Can an Interface implement another Interface? Interfaces doesn't provide implementation hence an interface cannot implement another interface.

46. Can an Interface extend another Interface? Yes an Interface can inherit another Interface, for that matter an Interface can extend more than one Interface.

47. Can a Class extend more than one Class? Not possible. A Class can extend only one class but can implement any number of Interfaces.

48. Why is an Interface be able to extend more than one Interface but a Class can't extend more than one Class? Basically Java doesn't allow multiple inheritance, so a Class is restricted to extend only one class. But an Interface is a pure abstraction model and doesn't have inheritance hierarchy like classes (do remember that the base class of all classes is Object). So an Interface is allowed to extend more than one Interface.

49. Can an Interface be final? Not possible. Doing so will result in compilation error.

50. Can a class be defined inside an Interface? Yes it's possible.

51. Can an Interface be defined inside a class? Yes it's possible.

53. Which object oriented Concept is achieved by using overloading and overriding? Polymorphism.

54. Why does Java not support operator overloading? Operator overloading makes the code very difficult to read and maintain. To maintain code simplicity, Java doesn't support operator overloading.

55. Can we define private and protected modifiers for variables in interfaces? No.

56. What is Externalizable? 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)

57. What modifiers are allowed for methods in an Interface? Only public and abstract modifiers are allowed for methods in interfaces.
58. What is a local, member and a class variable? Variables declared within a method are "local" variables. Variables declared within the class i.e. not within any methods are "member" variables (global variables).Variables declared within the class i.e. not within any methods and are defined as "static" are class variables.
59. What is an abstract method? An abstract method is a method whose implementation is deferred to a subclass.
60. What value does read () return when it has reached the end of a file? The read () method returns -1 when it has reached the end of a file.
61. Can a Byte object be cast to a double value? No, an object cannot be cast to a primitive value.
62. What is the difference between a static and a non-static inner class? A non-static inner class may have object instances that are associated with instances of the class's outer class. A static inner class does not have any object instances.
63. What is an object's lock and which objects have locks? An object's lock is a mechanism that is used by multiple threads to obtain synchronized access to the object. A thread may execute a synchronized method of an object only after it has acquired the object's lock. All objects and classes have locks. A class's lock is acquired on the class's Class object.
64. What is the % operator? It is referred to as the module or remainder operator. It returns the remainder of dividing the first operand by the second operand.
65. When can an object reference be cast to an interface reference? An object reference be cast to an interface reference when the object implements the referenced interface.
66. Which class is extended by all other classes? The Object class is extended by all other classes.
67. Which non-Unicode letter characters may be used as the first character of an identifier? The non-Unicode letter characters $ and _ may appear as the first character of an identifier
68. What restrictions are placed on method overloading? Two methods may not have the same name and argument list but different return types.
69. What is casting? There are two types of casting, casting between primitive numeric types and casting between object references. Casting between numeric types is used to convert larger values, such as double values, to smaller values, such as byte values. Casting between object references is used to refer to an object by a compatible class, interface, or array type reference.
70. What is the return type of a program's main () method? Void.
71. If a variable is declared as private, where may the variable be accessed? A private variable may only be accessed within the class in which it is declared.
72. What do you understand by private, protected and public? These are accessibility modifiers. Private is the most restrictive, while public is the least restrictive. There is no real difference between protected and the default type (also known as package protected) within the context of the same package, however the protected keyword allows visibility to a derived class in a different package.
73. What is Downcasting? Downcasting is the casting from a general to a more specific type, i.e. casting down the hierarchy
74. What modifiers may be used with an inner class that is a member of an outer class? A (non-local) inner class may be declared as public, protected, private, static, final, or abstract.
75. How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters? Unicode requires 16 bits and ASCII require 7 bits although the ASCII character set uses only 7 bits, it is usually represented as 8 bits.UTF-8 represents characters using 8, 16, and 18 bit patterns.UTF-16 uses 16-bit and larger bit patterns.
76. What restrictions are placed on the location of a package statement within a source code file? A package statement must appear as the first line in a source code file (excluding blank lines and comments).
77. What is a native method? A native method is a method that is implemented in a language other than Java.
78. What are order of precedence and associativity, and how are they used? Order of precedence determines the order in which operators are evaluated in expressions. Associated determines whether an expression is evaluated left-to-right or right-to-left.
79. Can an anonymous class be declared as implementing an interface and extending a class? An anonymous class may implement an interface or extend a superclass, but may not be declared to do both.
80. What is the range of the char type? The range of the char type is 0 to 2^{16} - 1 (i.e. 0 to 65535.)
81. What is the range of the short type? The range of the short type is - (2^{15}) to 2^{15} - 1. (I.e. -32,768 to 32,767)
82. Why isn't there operator overloading? Because C++ has proven by example that operator overloading makes code almost impossible to maintain.
83. What does it mean that a method or field is "static"? 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.
84. Is null a keyword? The null value is not a keyword.

85. Which characters may be used as the second character of an identifier, but not as the first character of an identifier? The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier.
86. Is the ternary operator written x: y? z or x? y: z ?It is written x ? y: z.
87. How is rounding performed under integer division? The fractional part of the result is truncated. This is known as rounding toward zero.
88. If a class is declared without any access modifiers, where may the class be accessed? A class that is declared without any access modifiers is said to have package access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.
89. Does a class inherit the constructors of its superclass? A class does not inherit constructors from any of its superclass's.
90. Name the eight primitive Java types. The eight primitive types are byte, char, short, int, long, float, double, and boolean.
91. What restrictions are placed on the values of each case of a switch statement? During compilation, the values of each case of a switch statement must evaluate to a value that can be promoted to an int value.
92. What is the difference between a while statement and a do while statement? A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do while statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do while statement will always execute the body of a loop at least once.
93. What modifiers can be used with a local inner class? A local inner class may be final or abstract.
94. When does the compiler supply a default constructor for a class? The compiler supplies a default constructor for a class if no other constructors are provided.
95. If a method is declared as protected, where may the method be accessed? A protected method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.
96. What are the legal operands of the instanceof operator? The left operand is an object reference or null value and the right operand is a class, interface, or array type.
98. What happens when you add a double value to a String? The result is a String object.
99. What is the difference between inner class and nested class? When a class is defined within a scope od another class, then it becomes inner class. If the access modifier of the inner class is static, then it becomes nested class.
101. What is numeric promotion? Numeric promotion is the conversion of a smaller numeric type to a larger numeric type, so that integer and floating-point operations may take place. In numerical promotion, byte, char, and short values are converted to int values. The int values are also converted to long values, if necessary. The long and float values are converted to double values, as required.
102. What is the difference between a public and a non-public class? A public class may be accessed outside of its package. A non-public class may not be accessed outside of its package.
103. To what value is a variable of the boolean type automatically initialized? The default value of the boolean type is false.
104. What is the difference between the prefix and postfix forms of the ++ operator? The prefix form performs the increment operation and returns the value of the increment operation. The postfix form returns the current value all of the expression and then performs the increment operation on that value.
105. What restrictions are placed on method overriding? Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides. The overriding method may not throw any exceptions that may not be thrown by the overridden method.
106. What is a Java package and how is it used? A Java package is a naming context for classes and interfaces. A package is used to create a separate name space for groups of classes and interfaces. Packages are also used to organize related classes and interfaces into a single API unit and to control accessibility to these classes and interfaces.
107. What modifiers may be used with a top-level class? A top-level class may be public, abstract, or final.
108. What is the difference between an if statement and a switch statement? The if statement is used to select among two alternatives. It uses a boolean expression to decide which alternative should be executed. The switch statement is used to select among multiple alternatives. It uses an int expression to determine which alternative should be executed.
109.What are the practical benefits, if any, of importing a specific class rather than an entire package (e.g. import java.net.* versus import java.net.Socket)?It makes no difference in the generated class files since only the classes that are actually used are referenced by the generated class file. There is another practical benefit to importing single classes, and this arises when two (or more) packages have classes with the same name. Take java.util.Timer and javax.swing.Timer, for example. If I import java.util.* and javax.swing.* and then try to use "Timer", I get an error while compiling (the class name is ambiguous between both packages). Let's say what you really wanted was the javax.swing.Timer class, and the only classes you plan on using in java.util are Collection and HashMap. In this case, some people will prefer to import java.util.Collection and import java.util.HashMap instead of importing java.util.*. This will now allow them to use Timer, Collection, HashMap, and other javax.swing classes without using fully qualified class names in.
110.Can a method be overloaded based on different return type but same argument type ?No, because the methods can be called without using their return type in which case there is ambiguity for the compiler.
111. What happens to a static variable that is defined within a method of a class? Can't do it. You'll get a compilation error.

112.How many static initializers can you have ?As many as you want, but the static initializers and class variable initializers are executed in textual order and may not refer to class variables declared in the class whose declarations appear textually after the use, even though these class variables are in scope.
113. What is the difference between method overriding and overloading? Overriding is a method with the same name and arguments as in a parent, whereas overloading is the same method name but different arguments
114. What is constructor chaining and how is it achieved in Java? A child object constructor always first needs to construct its parent (which in turn calls its parent constructor.). In Java it is done via an implicit call to the no-args constructor as the first statement.
115. What is the difference between the Boolean & operator and the && operator? If an expression involving the Boolean & operator is evaluated, both operands are evaluated. Then the & operator is applied to the operand. When an expression involving the && operator is evaluated, the first operand is evaluated. If the first operand returns a value of true then the second operand is evaluated. The && operator is then applied to the first and second operands. If the first operand evaluates to false, the evaluation of the second operand is skipped.
116. Which Java operator is right associative? The = operator is right associative.
117. Can a double value be cast to a byte? Yes, a double value can be cast to a byte.
118. What is the difference between a break statement and a continue statement? A break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continue statement is used to end the current loop iteration and return control to the loop statement.
119. Can a for statement loop indefinitely? Yes, a for statement can loop indefinitely. For example, consider the following: for (;;);
120. To what value is a variable of the String type automatically initialized? The default value of a String type is null.
121. What is the difference between a field variable and a local variable? A field variable is a variable that is declared as a member of a class. A local variable is a variable that is declared local to a method.
123. What does it mean that a class or member is final? A final class cannot be inherited. A final method cannot be overridden in a subclass. A final field cannot be changed after its initialized, and it must include an initializer statement where it's declared.
124. What does it mean that a method or class is abstract? An abstract class cannot be instantiated. Abstract methods may only be included in abstract classes. However, an abstract class is not required to have any abstract methods, though most of them do. Each subclass of an abstract class must override the abstract methods of its superclass's or it also should be declared abstract.
126. How does Java handle integer overflows and underflows? It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.
127. What is the difference between the >> and >>> operators? The >> operator carries the sign bit when shifting right. The >>> zero-fills bits that have been shifted out.
128. Is sizeof a keyword? The sizeof operator is not a keyword.
129. Which four options describe the correct default values for array elements of the types indicated?

- ☑ int -> 0
- ☑ Dog -> null
- ☑ char -> '\u0000'
- ☑ float -> 0.0f

130. Which one of these lists contains only Java programming language keywords?

- ☑ goto, instanceof, native, finally, default, throws

131. Which will legally declare, construct, and initialize an array?

- ☑ int myList [] = {4, 3, 7};

132. Which three are legal array declarations?

- ☑ int [] myScores [];
- ☑ char [] myChars;
- ☑ Dog myDogs [];

133. Which three piece of codes are equivalent to line 3?

```
public interface Foo {
   int k = 4; /* Line 3 */
}
```

- ☑ final int k = 4;
- ☑ public int k = 4;
- ☑ static int k = 4;

134. Which one of the following will declare an array and initialize it with five numbers?

- ☑ int [] a = {23,22,21,20,19};

135. Which three are valid declarations of a char?

- ☑ char c1 = 064770;

- ☑ char c3 = 0xbeef;
- ☑ char c6 = '\uface';

136. Which is the valid declarations within an interface definition?

- ☑ public double methoda();

137. Which one is a valid declaration of a boolean?

- ☑ boolean b3 = false;

138. Which three are valid declarations of a float?

- ☑ float f1 = -343;
- ☑ float f3 = 0x12345;
- ☑ float f6 = 2.81F;

139. Which is a valid declarations of a String?

- ☑ String s1 = null;

140. What is the numerical range of a char?

- ☑ 0 to 65535

141. Which two of the following statements, inserted independently, could legally be inserted into missing section of this code?

```
class Ticker extends Component {
   public static void main (String [] args)    {
      Ticker t = new Ticker();
      /* Missing Statements? */
   }
}
        1.    boolean test = (t instanceof Ticker);
        2.    boolean test = (t instanceof Component);
```

142. Which of the following are legal lines of code?

- ☑ int w = (int)888.8;
- ☑ byte x = (byte)1000L;
- ☑ long y = (byte)100;
- ☑ byte z = (byte)100L;

143. Which two statements are equivalent?

- ☑ 16>>2
- ☑ 16>>>2

144. Which two statements are equivalent?

- ☑ 3*4
- ☑ 3<<2

145. Which three statements are true?

```
class CompareReference {
   public static void main(String [] args)    {
      float f = 42.0f;
      float [] f1 = new float[2];
      float [] f2 = new float[2];
      float [] f3 = f1;
      long x = 42;
      f1[0] = 42.0f;
   }
}
```

- ☑ f1 == f3
- ☑ x == f1[0]
- ☑ f == f1[0]

146. Which two are equal?

- ☑ (8 >> 2) << 4
- ☑ 128 >>> 2

147. You want subclasses in any package to have access to members of a superclass. Which is the most restrictive access that accomplishes this objective?

- ☑ Protected
- **private** makes a member accessible only from within its own class
- **protected** makes a member accessible only to classes in the same package or subclass of the class

- **Default** access is very similar to protected (make sure you spot the difference) default access makes a member accessible only to classes in the same package.
- **public** means that all other classes regardless of the package that they belong to, can access the member
- **Final** makes it impossible to extend a class, when applied to a method it prevents a method from being overridden in a subclass, when applied to a variable it makes it impossible to reinitialize a variable once it has been initialized.
- **Abstract** declares a method that has not been implemented.
- **Transient** indicates that a variable is not part of the persistent state of an object.
- **Volatile** indicates that a thread must reconcile its working copy of the field with the master copy every time it accesses the variable.

148. Which of the following code fragments inserted, will allow to compile?

```
public class Outer {
   public void someOuterMethod()    {
      //Line 5
   }
   public class Inner { }
   public static void main(String[] argv)    {
      Outer ot = new Outer();
      //Line 10
   }
}
```

☑ new Inner(); //At line 5

149. Which two code fragments will compile?

```
interface Base {
   boolean m1 ();
   byte m2(short s);
}
```

☑ abstract class Class2 implements Base {}
☑ abstract class Class2 implements Base{public boolean m1(){ return (7 > 4); }}

150. Which three form part of correct array declarations?

☑ public int a [ ]
☑ static int [ ] a
☑ public final int [ ] a

151. What is the prototype of the default constructor? public class Test { }

☑ public Test( )

152. What is the most restrictive access modifier that will allow members of one class to have access to members of another class in the same package? You want a class to have access to members of another class in the same package. Which is the most restrictive access that accomplishes this objective? Default access

153. Which of the following is/are legal method declarations?

☑ protected abstract void m1();
☑ static final void m1(){}
☑ synchronized public final void m1() {}
☑ private native void m1();

154. Which cause a compiler error?

☑ int [ ][ ] scores = {2,7,6}, {9,3,45};

155. Which three are valid method signatures in an interface?

☑ public float getVol(float x);
☑ public void main(String [] args);
☑ boolean setFlag(Boolean [] test);

156. What is the widest valid returnType for methodA in line 3? Double

```
public class ReturnIt {
   returnType methodA(byte x, double y) /* Line 3 */    {
      return (long)x / y * 2;
   }
}
```

157. Which is valid in a class that extends class A? public int method1(int a, int b) {return 0; }

```
class A {
   protected int method1(int a, int b) {
      return 0;
   }
}
```

158. Which one creates an instance of an array? int[ ] ia = new int[15];

159. Which two of the following are legal declarations for nonnested classes and interfaces?

- ☑ final public class Test {}
- ☑ abstract public class Test {}

160. Which of the following class level (nonlocal) variable declarations will not compile? private synchronized int e;

161. Which two cause a compiler error?

- ☑ float[ ] f = new float(3);
- ☑ float f2[ ] = new float[ ];

162. Given a method in a protected class, what access modifier do you use to restrict access to that method to only the other members of the same class? private

163. Which is a valid declaration within an interface? public static short stop = 23;

164. Which two code fragments inserted at end of the program, will allow to compile?

```
interface DoMath {
   double getArea(int rad);
}
interface MathPlus {
   double getVol(int b, int h);
}
/* Missing Statements? */
interface AllMath extends DoMath { float getAvg(int h, int l); }
abstract class AllMath implements DoMath, MathPlus { public double getArea(int rad) { return rad * rad * 3.14; } }
```

165. Which two statements are true for any concrete class implementing the java.lang.Runnable interface?

- ☑ The class must contain a method called run () from which all code for that thread will be initiated.
- ☑ The mandatory method must be public, with a return type of void, must be called run (), and cannot take any arguments.

166. Which two statements, added independently at beginning of the program, allow the code to compile?

```
/* Missing statements? */
public class NewTreeSet extends java.util.TreeSet{
   public static void main(String [] args)    {
      java.util.TreeSet t = new java.util.TreeSet();
      t.clear();
   }
   public void clear()     {
      TreeMap m = new TreeMap();
      m.clear();
   }
}
import java.util.*;
import java.util.TreeMap;
```

167. Which three statements are true?

- ☑ The default constructor has the same access as its class.
- ☑ The default constructor invokes the no-arg constructor of the superclass.
- ☑ The compiler creates a default constructor only when there are no other constructors for the class.

168. Which statement is true?

```
package testpkg.p1;
public class ParentUtil {
   public int x = 420;
   protected int doStuff() { return x; }
}
package testpkg.p2;
import testpkg.p1.ParentUtil;
public class ChildUtil extends ParentUtil {
   public static void main(String [] args) {
      new ChildUtil().callStuff();
   }
   void callStuff()   {
      System.out.print("this " + this.doStuff() ); /* Line 18 */
      ParentUtil p = new ParentUtil();
      System.out.print(" parent " + p.doStuff() ); /* Line 20 */
   }
}
```
If line 20 is removed, the code will compile and run.

169. If a is false and b is true then the output is "ELSE"

```
public void foo( boolean a, boolean b){
    if( a )     {
        System.out.println("A"); /* Line 5 */
    }
    else if(a && b) /* Line 7 */  {
        System.out.println( "A && B");
    }
    else /* Line 11 */  {
        if ( !b ) {
            System.out.println( "notB") ;
        }
        else {
            System.out.println( "ELSE" ) ;
        }
    }
}
```

170. Which two are acceptable types for x? Byte and char

```
switch(x){
    default:
        System.out.println("Hello");
}
```

171. Which statement is true? Compilation fails.

```
public void test(int x) {
    int odd = 1;
    if(odd) /* Line 4 */{
        System.out.println("odd");
    }
    else {
        System.out.println("even");
    }
}
```

172. Which statement is true? There is a syntax error on line 6.

```
public class While {
    public void loop()  {
        int x= 0;
        while ( 1 ) /* Line 6 */ {
            System.out.print("x plus one is " + (x + 1)); /* Line 8 */
        }
    }
}
```