Java Interview Questions

Difference between Procedure Oriented and object oriented programming language?

	Procedure Oriented	object oriented
Divided into	In POP program is divided into small parts called functions.	In OOP program is divided into parts called objects.
Importance	In POP importance is not given to data but to function as well as sequence of actions to be done.	In OOP importance is given to the data rather than procedures or functions because it works as a real world.
Approach	POP follows TOP Down approach	OOP follows Bottom UP approach
Access Specifier	POP does not have any access specifier.	OOP has access specifier named public, private, protected etc.
Data Moving	In POP data can move freely from function to function in the system.	In OOP object can move and communicate with each other through member functions.
Expansion	To add new data and function in POP is not so easy	OOP provides an easy way to add new data and function.

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

(i.e. easily remembered as A-PIE).

What is Abstraction?

Abstraction refers to the act of representing essential features without including the background details or explanations.

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.

What is the difference between abstraction and encapsulation?

Abstraction focuses on the outside view of an object (i.e. the interface) Encapsulation (information hiding) prevents clients from seeing it's inside view, where the behavior of the abstraction is implemented.

Abstraction solves the problem in the design side while Encapsulation is the Implementation.

Encapsulation is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs.

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

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.

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).

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.

Note:

From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:

Method overloading

Method overriding through inheritance

Method overriding through the Java interface

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.

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.

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.

Note:

Overloaded methods MUST change the argument list

Overloaded methods CAN change the return type

Overloaded methods CAN change the access modifier

Overloaded methods CAN declare new or broader checked exceptions

A method can be overloaded in the same class or in a subclass

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.
Note:
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
You cannot override a method marked static

What is a class?

class

- A class is a specification or blue print or template of an object.
- Class is a logical construct; an object has physical reality.
- Class is a structure
- Binding the data with its related and corresponding functions.
- Class is the base for encapsulation.
- Class is a user defined data type in java.
- Class will act as the base for encapsulation and implement the concept of encapsulation through objects.
- Any java applications look like collection of classes but whereas c- application looks like collection of functions.
- Class contains variables and methods.

What is an object ?

Object

- Object is nothing but instance (dynamic memory allocation) of a class.
- The dynamic memory allocated at run time for the members [non-static variables] of the class is known as object.
- Object is an encapsulated form of all non-static variables and non-static methods of a particular class.
- The process of creating objects out of class is known as instantiation.

What are the Object Characteristics?

The three key characteristics of Object are

- State
- Behaviour

- Identity

State:

- Instance variables value is called object state.
- An object state will be changed if instance variables value is changed.

Behavior:

- Behaviour of an object is defined by instance methods.
- Behaviour of an object is depends on the messages passed to it.
- So an object behaviour depends on the instance methods.

Identity:

- Identity is the hash code of an object, it is a 32-bit integer number created randomly and assigned to an object by default by JVM.
- Developer can also generate hash code of an object based on the state of that object by overriding hashcode() method of java.lang.Object class.

Then if state is changed, automatically hashcode will be changed.

What Object Contains?

- Object of any class contains only data.
- Apart from the data object of a class would not contains anything else.
- Object of a class would not contain any functionalities or logic.
- Thus object of a class would be representing only data and not represent logic.

What is state of the Object?

The data present inside object of a class at that point of time is known as state of the object.

What is behaviour of the object?

The functionalities associated with the object => Behaviour of the object.
The state of the object changes from time-to-time depending up on the functionalities that are executed on that object but whereas behaviour of the object would not change.

In how many ways can you create an object in Java?

There are four ways of creating objects in Java:

1. Using new operator

Employee obj = new Employee ();
Here, we are creating Employee class object 'obj' using new operator.

2. Using factory methods:

Number Format obj = NumberFormat. getNumberInstance( );
Here, we are creating NumberFormat object using the factory method getNumberInstance( )

3. Using newInstance( ) method. Here we should follow two steps, as:

(a) First, store the class name 'Employee' as a string into an object. For this purpose, factory metod forName( ) of the class 'Class' will be useful:
Class c = Class.forName("Employee");
We should note that there is a class with the name 'Class' in java.lang package.
(b) Next, create another object to the class whose name is in the object c. For this purpose , we need newInstance( ) method of the class 'Class' as:
Employee obj = (Employee)c.newInstance( );

4. By cloning an already available object, we can create another object. Creating exact copy of an existing object is called 'cloning'.
Employee obj1 = new Employee ( );
Employee obj2 = (Employee)obj1.clone( );
Earlier, we created obj2 by cloning the Employee object obj1.clone( ) method of Object class is used to clone object.We should note that there is a class by the name 'Object' in java.lang package.

Define 5 different places to define Object of a class in java?
Object of a class can be defined at 5 different places.
1. Defining object of a class as local to a method or local variable.
2. Defining object of a class as an instance variable of another class.
3. Defining object of a class as the parameter of the method.
4. Defining object of a class as the return data type of the method.
5. Defining object of a class as the static variable or class variable of another class.

What is a Reference variable?
Reference variable is a variable which would be representing the address of the object.
Reference will act as a pointer and handler to the object.
Since reference variable always points an object.
In practice we call the reference variable also as an object.
We can create an object by using new operator.
If we want to call any method inside the class we need object
Syntax: A obj= new A();

What is object graph?

Object graph is a graph showing relationship between different objects in memory.

Explain what can we use primitive data types as objects?
Primitive data types like int can be handled as objects by the use of their respective wrapper classes. For example, Integer is a wrapper class for primitive data type int. We can apply different methods to a wrapper class, just like any other object.

What is mutable object and immutable object?
If an object value is changeable then we can call it as Mutable object. (Ex., StringBuffer, …) If you are not allowed to change the value of an object, it is immutable object. (Ex., String, Integer, Float, …)

How many objects are created in the following piece of code?

```
MyClass c1, c2, c3;
c1 = new MyClass ();
c3 = new MyClass ();
```

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

Explain what 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.

Explain what 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 of the heap.

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
Answer: -
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.

What is memory leak?
A memory leak is where an unreferenced object that will never be used again still hangs around in memory and doesn't get garbage collected.

I want my class to be developed in such a way that no other class (even derived class) can create its objects. Explain what 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.

What are the Object and Class classes used for? Which class should you use to obtain design information about an object?

Differentiate between a Class and an Object?

The Object class is the highest-level class in the Java class hierarchy. The Class class is used to represent the classes and interfaces that are loaded by a Java program. The Class class is used to obtain information about an object's design. A Class is only a definition or prototype of real life object. Whereas an object is an instance or living representation of real life object. Every object belongs to a class and every class contains one or more related objects.

What is the difference between an object and an instance?

An Object May not have a class definition. eg int a[] where a is an array.
An Instance should have a class definition.

eg MyClass my=new MyClass();
my is an instance.

What is the difference between instance, object, reference and a class?

Class: A class is a user defined data type with set of data members & member functions
Object: An Object is an instance of a class
Reference: A reference is just like a pointer pointing to an object
Instance: This represents the values of data members of a class at a particular time

What is a singleton class?

Or

What is singleton pattern?

This design pattern is used by an application to ensure that at any time there is only one instance of a class created. You can achieve this by having the private constructor in the class and having a getter method which returns an object of the class and creates one for the first time if its null.

What is the purpose of the System class?

The purpose of the System class is to provide access to system resources.

What is the purpose of the Runtime class?

The purpose of the Runtime class is to provide access to the Java runtime system.
It returns the runtime information like memory availability.

* Runtime.freeMemory() --> Returns JVM Free Memory
* Runtime.maxMemory() --> Returns the maximum amount of memory that the JVM will attempt to use.
* Runtime.gc() --> It also helps to run the garbage collector

What would happen if you say this = null?

It will come up with Error Message
"The left-hand side of an assignment must be a variable".

What are the differences between method overloading and method overriding?

Overloaded Method	Overridden Method
Arguments Must change	Must not change
Return type Can change	Can't change except for covariant returns
Exceptions Can change	Can reduce or eliminate. Must not throw new or broader checked exceptions
Access Can change	Must not make more restrictive

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.

Is it possible to override the main method?

NO, because main is a static method. A static method can't be overridden in Java.

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.

```
// From subclass
    super.overriddenMethod();
```

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.

Note:

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.

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
}
```

What is an Interface?

An interface is a description of a set of methods that conforming implementing classes must have.

Note:

You can't mark an interface as final.

Interface variables must be static.

An Interface cannot extend anything but another interfaces.

Can we instantiate an interface?

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

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.

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.

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

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.

What is an abstract class?
Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.
Note:
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.

Can we instantiate an abstract class?
An abstract class can never be instantiated. Its sole purpose is to be extended (subclassed).

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.

When you declare a method as abstract, can other non abstract methods access it?
Yes, other non abstract methods can access a method that you declare as abstract.

Can there be an abstract class with no abstract methods in it?
Yes, there can be an abstract class without abstract methods.

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.

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.

Can constructor be inherited?
No, constructor cannot be inherited, though a derived class can call the base class constructor.

What are the differences between Contructors and Methods?

	Constructors	**Methods**
Purpose	Create an instance of a class	Group Java statements
Modifiers	Cannot be abstract, final, native, native, static, or synchronized	Can be abstract, final, static, or synchronized
Return Type	No return type, not even void	void or a valid return type
Name	Same name as the class	Any name except the class.
Inheritance	Constructors are not inherited	Methods are inherited

- 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.

- Super 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

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.

What are the differences between Class Methods and Instance Methods?

- Class Methods
- Instance Methods

a) **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.

b) 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.

c) 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.

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.

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..

What are Access Specifiers 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 specifier)- 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.

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.
final Variables- A final variable cannot be changed once it is initialized.
final Methods- A final method cannot be overridden by subclasses.

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.

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

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.

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.

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.

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.

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 collectionâ€™s 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().

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.

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

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

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

What is the List interface?
The List interface provides support for ordered collections of objects.
Lists may contain duplicate elements.

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).

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.

Difference between ArrayList and Vector ?

	ArrayList	Vector
a)	ArrayList is NOT synchronized by default.	Vector List is synchronized by default.
b)	ArrayList can use only Iterator to access the elements	Vector list can use Iterator and Enumeration Interface to access the elements.
c)	The ArrayList increases its array size by size 50 percent if it runs out of room.	A Vector defaults to doubling the of its array if it runs out of room
d)	ArrayList has no default size.	While vector has a default size-10

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();
```

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. In linked list data is stored in nodes that have reference to the previous node and the next node so adding element is simple as creating the node an updating the next pointer on the last node and the previous pointer on the new node. 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.

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.

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.

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

What are the main Implementations of the Set interface ?

The main implementations of the List interface are as follows:

HashSet

TreeSet

LinkedHashSet

EnumSet

What is a HashSet ?

A HashSet is an unsorted, unordered Set.

It uses the hashcode of the object being inserted (so the more efficient your hashcode() implementation the better access performance you'll get).

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

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.

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.

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
We can add any type of elements to hash set.	We can add only similar types of elements to tree set.

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.

What are the main Implementations of the Map interface ?

The main implementations of the List interface are as follows:

HashMap

HashTable

TreeMap

EnumMap

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.

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.

How does a Hashtable internally maintain the key-value pairs?
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.

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.

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.

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.

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.

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).

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.

What are the differences between the Comparable and Comparator interfaces ?

Comparable	Comparato
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. It is frequently used by the API classes.	Many sort sequences can be created. It used by third-party classes to sort instances.

What is a Thread?

In Java, "thread" means two different things:

- An instance of class java.lang.Thread.
- A thread of execution.

An instance of Thread is just…an object. Like any other object in Java, it has variables and methods, and lives and dies on the heap. But a thread of execution is an individual process (a "lightweight" process) that has its own call stack. In Java, there is one thread per call stack—or, to think of it in reverse, one call stack per thread. Even if you don't create any new threads in your program, threads are back there running.

The main() method, that starts the whole ball rolling, runs in one thread, called (surprisingly) the main thread. If you looked at the main call stack (and you can, any time you get a stack trace from something that happens after main begins, but not within another thread), you'd see that main() is the first method on the stack— the method at the bottom. But as soon as you create a new thread, a new stack materializes and methods called from that thread run in a call stack that's separate from the main() call stack.

What is difference between thread and process?

Differences between threads and processes are:-

1. Threads share the address space of the process that created it; processes have their own address.

2. Threads have direct access to the data segment of its process; processes have their own copy of the data segment of the parent process.

3. Threads can directly communicate with other threads of its process; processes must use inter process communication to communicate with sibling processes.

4. Threads have almost no overhead; processes have considerable overhead.

5. New threads are easily created; new processes require duplication of the parent process.

6. Threads can exercise considerable control over threads of the same process; processes can only exercise control over child processes.

7. Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process; changes to the parent process do not affect child processes.

What are the advantages or usage of threads?

Threads support concurrent operations. For example,

- Multiple requests by a client on a server can be handled as an individual client thread.
- Long computations or high-latency disk and network operations can be handled in the background without disturbing foreground computations or screen updates.

Threads often result in simpler programs.

- In sequential programming, updating multiple displays normally requires a big while-loop that performs small parts of each display update. Unfortunately, this loop basically simulates an operating system scheduler. In Java, each view can be assigned a thread to provide continuous updates.

• Programs that need to respond to user-initiated events can set up service routines to handle the events without having to insert code in the main routine to look for these events.

Threaded applications exploit parallelism.
• A computer with multiple CPUs can literally execute multiple threads on different functional units without having to simulating multi-tasking ("time sharing").
• On some computers, one CPU handles the display while another handles computations or database accesses, thus, providing extremely fast user interface response times.

What is use of synchronized keyword?
synchronized keyword can be applied to static/non-static methods or a block of code. Only one thread at a time can access synchronized methods and if there are multiple threads trying to access the same method then other threads have to wait for the execution of method by one thread. Synchronized keyword provides a lock on the object and thus prevents race condition. E.g.

```
public void synchronized method(){}
public void synchronized staticmethod(){}
public void myMethod(){

        synchronized (this){          // synchronized keyword on block of  code
        }
}
```

What is the difference when the synchronized keyword is applied to a static method or to a non static method?
When a synch non static method is called a lock is obtained on the object. When a synch static method is called a lock is obtained on the class and not on the object. The lock on the object and the lock on the class donâ€™t interfere with each other. It means, a thread accessing a synch non static method, then the other thread can access the synch static method at the same time but canâ€™t access the synch non static method.

What is a volatile keyword?
In general each thread has its own copy of variable, such that one thread is not concerned with the value of same variable in the other thread. But sometime this may not be the case. Consider a scenario in which the count variable is holding the number of times a method is called for a given class irrespective of any thread calling, in this case irrespective of thread access the count has to be increased so the count variable is declared as volatile. The copy of volatile variable is stored in the main memory, so every time a thread access the variable

even for reading purpose the local copy is updated each time from the main memory. The volatile variable also have performance issues.

What is the difference between yield() and sleep()?

yield() allows the current thread to release its lock from the object and scheduler gives the lock of the object to the other thread with same priority.

sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn't release the lock.

What is the difference between wait() and sleep()?

-→ wait() is a method of Object class. sleep() is a method of Object class.

-→ sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn't release the lock. wait() allows thread to release the lock and goes to suspended state. The thread is only active when a notify() or notifAll() method is called for the same object.

What is difference between notify() and notfiyAll()?

notify( ) wakes up the first thread that called wait( ) on the same object.
notifyAll( ) wakes up all the threads that called wait( ) on the same object. The highest priority thread will run first.

What happens if a start method is not invoked and the run method is directly invoked?

If a thread has been instantiated but not started it is said to be in new state. Unless until a start() method is invoked on the instance of the thread, it will not said to be alive. If you do not call a start() method on the newly created thread instance thread is not considered to be alive. If the start() method is not invoked and the run() method is directly called on the Thread instance, the code inside the run() method will not run in a separate new thread but it will start running in the existing thread.

What happens when start() is called?

A new thread of execution with a new call stack starts. The state of thread changes from new to runnable. When the thread gets chance to execute its target run() method starts to run.

If code running is a thread creates a new thread what will be the initial priority of the newly created thread?

When a code running in a thread creates a new thread object , the priority of the new thread is set equal to the priority of the thread which has created it.

What are the daemon threads?
Daemon thread are service provider threads run in the background, these not used to run the application code generally. When all user threads (non-daemon threads) complete their execution, the jvm exit the application whatever may be the state of the daemon threads. Jvm does not wait for the daemon threads to complete their execution if all user threads have completed their execution.

How many locks does an object have?
Each object has only one lock.

Why would you use a synchronized block vs. synchronized method?
Synchronized blocks place locks for shorter periods than synchronized methods.

There are two classes: A and B. The class B need to inform a class A when some important event has happened. What Java technique would you use to implement it?
If these classes are threads I'd consider notify() or notifyAll(). For regular classes you can use the Observer interface.

How do you ensure that N thread can access N resources without deadlock
Key point here is order, if you acquire resources in a particular order and release resources in reverse order you can prevent deadlock.

What is the difference between Composition and Inheritance in OOP?

One of the good question to check the candidate's object-oriented programming skills. There are several differences between Composition and Inheritance in Java, some of them are following:

1. The Composition is more flexible because you can change the implementation at runtime by calling setXXX() method, but Inheritance cannot be changed i.e. you cannot ask a class to implement another class at runtime.

2. Composition builds HAS-A relationship while Inheritance builds IS-A relationship e.g. A Room HAS A Fan, but Mango IS-A Fruit.

3. The parent-child relationship is best represented using Inheritance but If you just want to use the services of another class use Composition.

What will happen if you call return statement or System.exit on try or catch block ? will finally block execute?
finally block will execute even if you put return statement in try block or catch block but finally block won't run if you call System.exit form try or catch.

If a method throws NullPointerException in super class, can we override it with a method which throws RuntimeException?

you can very well throw super class of RuntimeException in overridden method but you can not do same if its checked Exception.

How do you handle error condition while writing stored procedure or accessing stored procedure from java?

stored procedure should return error code if some operation fails but if stored procedure itself fail than catching SQLException is only choice.

Does it matter in what order catch statements for FileNotFoundException and IOExceptipon are written?

Yes, it does. The FileNotFoundException is inherited from the IOException. Exception's subclasses have to be caught first.

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.

What is Inheritance ?

Inheritance is one of the main features of any object oriented programming. Using inheritance concept, a class (Sub Class) can inherit properties of another class (Super Class). So it is used for reuse features.

Main advantage of using Inheritance ?

Using this feature you can reuse features, less memory wastage, less time required to develop and application and improve performance of application.

Types of Inheritance supported by Java ?

Java support only four types of inheritance in java.

- Single Inheritance
- Multi level Inheritance
- Hybrid Inheritance
- Hierarchical Inheritance

Why use Inheritance ?

- For Code Re usability.
- For class Re usability.
- For Method Overriding.

What is multilevel inheritance ?

Getting the properties from one class object to another class object level wise with different priorities.

What is Multiple Inheritance ?

The concept of Getting the properties from multiple class objects to sub class object with same priorities is known as multiple inheritance.

Why Java Doesn't Support multiple Inheritance ?

Due to ambiguity problem.

What happens if super class and sub class having same field name ?

Super class field will be hidden in the sub class. You can access hidden super class field in sub class using super keyword.

Can we reduce the visibility of the inherited or overridden method ?

Ans. No.

Which of the following is tightly bound ? Inheritance or Composition ?

Ans. Inheritance.

Does a class inherit the constructor of its super class ?

Ans. No

How Inheritance can be implemented in java?

Inheritance can be implemented in JAVA using below two keywords:

1.extends

2.implements

extends is used for developing inheritance between two classes and two interfaces.

implements keyword is used to developed inheritance between interface and class.

What is the difference between Inheritance and Encapsulation?

Inheritance is an object oriented concept which creates a parent-child relationship. It is one of the ways to reuse the code written for parent class but it also forms the basis of Polymorphism. On the other hand, Encapsulation is an object oriented concept which is used to hide the internal details of a class e.g. HashMap encapsulate how to store elements and how to calculate hash values.

What is the difference between Inheritance and Abstraction?

Abstraction is an object oriented concept which is used to simply things by abstracting details. It helps in the designing system. On the other hand, Inheritance allows code reuse. You can reuse the functionality you have already coded by using Inheritance.

What is the difference between Polymorphism and Inheritance?

Both Polymorphism and Inheritance goes hand on hand, they help each other to achieve their goal. Polymorphism allows flexibility, you can choose which code to run at runtime by overriding.

What is the difference between Composition and Inheritance in OOP?

One of the good question to check the candidate's object-oriented programming skills. There are several differences between Composition and Inheritance in Java, some of them are following:

1. The Composition is more flexible because you can change the implementation at runtime by calling setXXX() method, but Inheritance cannot be changed i.e. you cannot ask a class to implement another class at runtime.

2. Composition builds HAS-A relationship while Inheritance builds IS-A relationship e.g. A Room HAS A Fan, but Mango IS-A Fruit.

3. The parent-child relationship is best represented using Inheritance but If you just want to use the services of another class use Composition.

Can we override static method in Java?

No, you cannot override a static method in Java because it's resolved at compile time. In order for overriding to work, a method should be virtual and resolved at runtime because objects are only available at runtime. This is one of the tricky Java questions, where interviewer tries to confuse you. A programmer is never sure about whether they can override or overload a static method in Java.

Can we overload a static method in Java?

Yes, you cannot overload a static method in Java. Overloading has nothing to do with runtime but the signature of each method must be different. In Java, to change the method signature, you must change either number of arguments, type of arguments or order of arguments.

Can we override a private method in Java?

No, you cannot override a private method in Java because the private method is not inherited by the subclass in Java, which is essential for overriding. In fact, a private method is not visible to anyone outside the class and, more importantly, a call to the

private method is resolved at compile time by using Type information as opposed to runtime by using the actual object.

What is method hiding in Java?

Since the static method cannot be overridden in Java, but if you declare the same static method in subclass then that would hide the method from the superclass. It means, if you call that method from subclass then the one in the subclass will be invoked but if you call the same method from superclass then the one in superclass will be invoked. This is known as method hiding in Java.

Can a class implement more than one interface in Java?

Yes, A class can implement more than one interface in Java e.g. A class can be both Comparable and Serializable at the same time. This is why the interface should be the best use for defining Type as described in Effective Java. This feature allows one class to play a polymorphic role in the program.

Can a class extends more than one class in Java?

No, a class can only extend just one more class in Java. Though Every class also, by default extend the java.lang.Object class in Java.

Can an interface extends more than one interface in Java?

Yes, unlike classes, an interface can extend more than one interface in Java. There are several example of this behavior in JDK itself e.g. java.util.List interface extends both Collection and Iterable interface to tell that it is a Collection as well as it allows iteration via Iterator.

What will happen if a class extends two interfaces and they both have a method with same name and signature?

In this case, a conflict will arise because the compiler will not able to link a method call due to ambiguity. You will get a compile time error in Java.

Can we pass an object of a subclass to a method expecting an object of the super class?

Yes, you can pass that because subclass and superclass are related to each other by Inheritance which provides IS-A property. I mean Banana is a Fruit so you can pass banana if somebody expect fruit. Now there are scenario, where you can't do e.g. when subclass violates the Liskov Substitution principle i.e. you cannot pass a plastic banana to someone expecting fruit :-), The eat() function will throw exception.

What is the Liskov substitution principle?

The Liskov substitution principle is one of the five object-oriented design principles, collectively know as SOLID principles. This design principle is L of SOLID acronym. The Liskov substitution principle states that in an object oriented program if a function or method is expecting an object of base class then it should work fine with a derived class object as well. If it cannot function properly with derived class object then the derived class is violating the Liskov Substitution principle.
For example, if a method is expecting a List you can also pass ArrayList or LinkedList and it should work just fine because ArrayList and LinkedList both follow Liskov Substitution Principle, but the java.sql.Date which is a subclass of java.util.Date in Java violates Liskov Substitution Principle because you cannot pass an object of java.sql.Date class to a method which is expecting an object of java.util.Date, Why? because all time-related method will throw java.lang.UnsupportedOperationException.

Here is another example of violating The Liskov Substitution Principle, Square is a special type of Rectangle whose adjacent sides are equal but making Square extending Rectangle violates LSP principle.

How to call a method of a subclass, if you are holding an object of the subclass in a reference variable of type superclass?

You can call a method of the subclass by first casting the object hold by reference variable of superclass into the subclass. Once you hold the object in subclass reference type, you can call methods from the subclass.

What are the Rules in defining a constructor?

- Constructor name should be same as class name.
- It should not contain return type.
- It should not contain Non Access Modifiers: final ,static, abstract, synchronized
- In it logic return statement with value is not allowed.
- It can have all four accessibility modifiers: private , public, protected, default
- It can have parameters
- It can have throws clause: we can throw exception from constructor.
- It can have logic, as part of logic it can have all java legal statement except return statement with value.
- We can not place return in constructor.

Can we define a method with same name of class?
Yes, it is allowed to define a method with same class name. No compile time error and no runtime error is raised, but it is not recommended as per coding standards.

If we place return type in constructor prototype will it leads to Error?
No, because compiler and JVM considers it as a method.

How compiler and JVM can differentiate constructor and method definitions of both have same class name?
By using return type , if there is a return type it is considered as a method else it is considered as constructor.

How compiler and JVM can differentiate constructor and method invocations of both have same class name?
By using new keyword, if new keyword is used in calling then constructor is executed else method is executed.

Why return type is not allowed for constructor?
As there is a possibility to define a method with same class name , return type is not allowed to constructor to differentiate constructor block from method block.

Why constructor name is same as class name?
Every class object is created using the same new keyword , so it must have information about the class to which it must create object . For this reason constructor name should be same as class name.

Can we declare constructor as private?

- Yes we can declare constructor as private.
- All four access modifiers are allowed to constructor.
- We should declare constructor as private for not to allow user to create object from outside of our class.
- Basically we will declare private constructor in Singleton design pattern.

Do we have Copy Constructor in Java?
Like C++, Java also supports copy constructor. But, unlike C++, Java **doesn't create a default copy constructor** if you don't write your own.
To copy the values of one object into another in java, you can use:

- Constructor
- Assigning the values of one object into another
- clone() method of Object class

What is Constructor Chaining ?

Constructor Chaining is a technique of calling another constructor from one constructor. this() is used to call same class constructor where as super() is used to call super class constructor.

```
// Java program to illustrate Constructor Chaining
// within same class Using this() keyword
class Temp
{
   // default constructor 1
   // default constructor will call another constructor
   // using this keyword from same class
   Temp()
   {
      // calls constructor 2
      this(5);
      System.out.println("The Default constructor");
   }

   // parameterized constructor 2
   Temp(int x)
   {
      // calls constructor 3
      this(5, 15);
      System.out.println(x);
   }

   // parameterized constructor 3
   Temp(int x, int y)
   {
      System.out.println(x * y);
   }

   public static void main(String args[])
   {
      // invokes default constructor first
      new Temp();
   }
}
```

Can we call sub class constructor from super class constructor?
No. There is no way in java to call sub class constructor from a super class constructor.

What happens if you keep a return type for a constructor?
Ideally, Constructor must not have a return type. By definition, if a method has a return type, it's not a constructor. It will be treated as a normal method. But compiler gives a warning saying that method has a constructor name. Example:

```
class GfG
{
    int GfG()
    {
        return 0;    // Warning for the return type
    }

}
```

What is No-arg constructor?
Constructor without arguments is called no-arg constructor. Default constructor in java is always a no-arg constructor.

```

class GfG
{
    public GfG()
    {
        //No-arg constructor
    }

}
```

How a no – argument constructor is different from default Constructor?
If a class contains no constructor declarations, then a default constructor with no formal parameters and no throws clause is implicitly declared.

If the class being declared is the primordial class Object, then the default constructor has an empty body. Otherwise, the default constructor simply invokes the superclass constructor with no arguments.

What are private constructors and where are they used?
Like any method we can provide access specifier to the constructor. If it's made private, then it can only be accessed inside the class.
The major scenarios where we use private constructor:

- Internal Constructor chaining
- Singleton class design pattern

When do we need Constructor Overloading?
Sometimes there is a need of initializing an object in different ways. This can be done using constructor overloading. Different constructors can do different work by implementing different line of codes and are called based on the type and no of parameters passed. According to the situation , a constructor is called with specific number of parameters among overloaded constructors.

Do we have destructors in Java?
No, Because Java is a garbage collected language you cannot predict when (or even if) an object will be destroyed. Hence there is no direct equivalent of a destructor.

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.

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.

Which is superclass of Exception?

"Throwable", the parent class of all exception related classes.

What are the advantages of using exception handling?

Exception handling provides the following advantages over "traditional" error management techniques:

- Separating Error Handling Code from "Regular" Code.
- Propagating Errors up the Call Stack.
- Grouping Error Types and Error Differentiation.

What are the types of Exceptions in Java

There are two types of exceptions in Java, unchecked exceptions and checked exceptions.

- **Checked exceptions**: A checked exception is some subclass of Exception (or Exception itself), excluding class RuntimeException and its subclasses. Each method must either handle all checked exceptions by supplying a catch clause or list each unhandled checked exception as a thrown exception.
- **Unchecked exceptions:** All Exceptions that extend the RuntimeException class are unchecked exceptions. Class Error and its subclasses also are unchecked.

Why Errors are Not Checked?

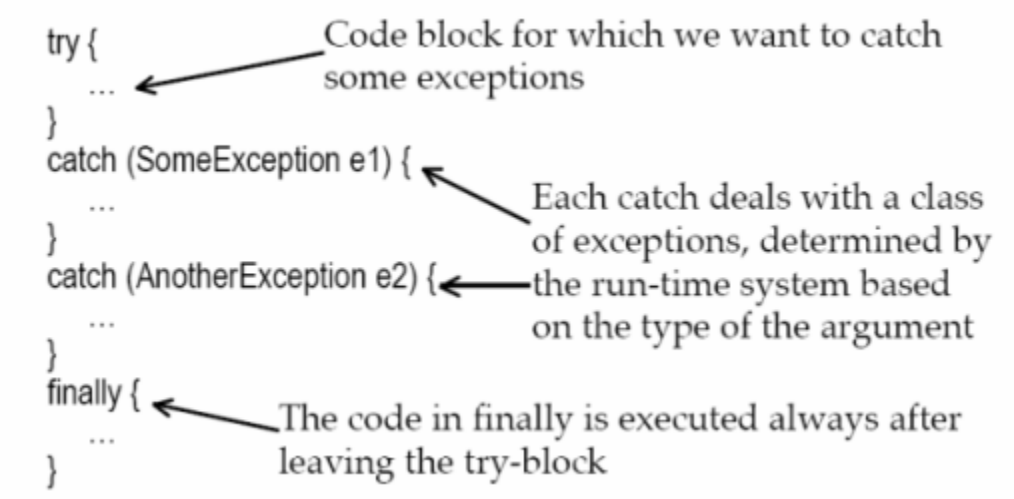
A unchecked exception classes which are the *error* classes (Error and its subclasses) are exempted from compile-time checking because they can occur at many points in the program and recovery from them is difficult or impossible. A program declaring such exceptions would be pointlessly.

Why Runtime Exceptions are Not Checked?

The *runtime exception* classes (RuntimeException and its subclasses) are exempted from compile-time checking because, in the judgment of the designers of the Java programming language, having to declare such exceptions would not aid significantly in establishing the correctness of programs. Many of the operations and constructs of the Java programming language can result in runtime exceptions. The information available to a compiler, and the level of analysis the compiler performs, are usually not sufficient to establish that such run-time exceptions cannot occur, even though this may be obvious to the programmer. Requiring such exception classes to be declared would simply be an irritation to programmers.

Explain the significance of try-catch blocks?

Whenever the exception occurs in Java, we need a way to tell the JVM what code to execute. To do this, we use the try and catch keywords. The try is used to define a block of code in which exceptions may occur. One or more catch clauses match a specific exception to a block of code that handles it.



What is the use of finally block?

The finally block encloses code that is always executed at some point after the try block, whether an exception was thrown or not. This is right place to close files, release your network sockets, connections, and perform any other cleanup your code requires.

Note: If the try block executes with no exceptions, the finally block is executed immediately after the try block completes. It there was an exception thrown, the finally block executes immediately after the proper catch block completes

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.

Can we have the try block without catch block?

Yes, we can have the try block without catch block, but finally block should follow the try block.

Note: It is not valid to use a try clause without either a catch clause or a finally clause.

What is the difference throw and throws?

throws: Used in a method's signature if a method is capable of causing an exception that it does not handle, so that callers of the method can guard themselves against that exception. If a method is declared as throwing a particular class of exceptions, then any other method that calls it must either have a try-catch clause to handle that exception or must be declared to throw that exception (or its superclass) itself.

A method that does not handle an exception it throws has to announce this:

```
public void myfunc(int arg) throws MyException {
    ...
 }
```

throw: Used to trigger an exception. The exception will be caught by the nearest try-catch clause that can catch that type of exception. The flow of execution stops immediately after the throw statement; any subsequent statements are not executed.

To throw an user-defined exception within a block, we use the throw command:

```
throw new MyException("I always wanted to throw an exception!");
```

How to create custom exceptions?

A. By extending the Exception class or one of its subclasses.

Example:

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

What are the different ways to handle exceptions?

There are two ways to handle exceptions:

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

What will happen if you call return statement or System.exit on try or catch block ? will finally block execute?

finally block will execute even if you put return statement in try block or catch block but finally block won't run if you call System.exit form try or catch.

If a method throws NullPointerException in super class, can we override it with a method which throws RuntimeException?

you can very well throw super class of RuntimeException in overridden method but you can not do same if its checked Exception.

How do you handle error condition while writing stored procedure or accessing stored procedure from java?

stored procedure should return error code if some operation fails but if stored procedure itself fail than catching SQLException is only choice.

Does it matter in what order catch statements for FileNotFoundException and IOExceptipon are written?

Yes, it does. The FileNotFoundException is inherited from the IOException. Exception's subclasses have to be caught first.

What can go wrong if you replace && with & in the following code:

```
String a=null;
if (a!=null && a.length()>10) {...}
```

A single ampersand here would lead to a NullPointerException.

Can we have static method inside an abstract class?

Theoretically yes, but in practice we should avoid it. As static method can't be overridden in a subclass and also it is to be used directly with ClassName without instantiation. This defeats the purpose of the abstract class.

What is abstract method in Java?

An abstract method is a method without body. You just declare method, without defining it and use abstract keyword in method declaration. All method declared inside Java Interface are by default abstract. Here is an example of abstract method in Java

```
public void abstract printVersion();
```

Now, In order to implement this method, you need to extend abstract class and override this method.

Can abstract class contains main method in Java ?

Yes, abstract class can contain main method, it just another static method and you can execute Abstract class with main method, until you don't create any instance.

What is default class modifier?

- A class is called a Default Class is when there is no access modifier specified on a class.
- Default classes are visible inside the same package only.
- Default access is also called Package access.

5) What are the different method access modifiers?

Let's discuss about access modifiers in order of increasing access.

private

a. Private variables and methods can be accessed only in the class they are declared.

b. Private variables and methods from SuperClass are NOT available in SubClass.

default or package

a. Default variables and methods can be accessed in the same package Classes.

b. Default variables and methods from SuperClass are available only to SubClasses in same package.

protected

a. Protected variables and methods can be accessed in the same package Classes.

b. Protected variables and methods from SuperClass are available to SubClass in any package

public

a. Public variables and methods can be accessed from every other Java classes.

b. Public variables and methods from SuperClass are all available directly in the SubClass

What is the use of a final modifier on a class?

Final class cannot be extended. Example of Final class in Java is the String class. Final is used very rarely as it prevents re-use of the class.Consider the class below which is declared as final.

What is the use of a final modifier on a method?

Final methods cannot be overridden. Consider the class FinalMemberModifiersExample with method finalMethod which is declared as final.

```
public class FinalMemberModifiersExample {

   final void finalMethod(){

   }

}
```

What is a Final variable?

Once initialized, the value of a final variable cannot be changed.

```
final int finalValue = 5;

//finalValue = 10; //COMPILER ERROR
```

What is a final argument?

Final arguments value cannot be modified. Consider the example below:

```
void testMethod(final int finalArgument){

   //final argument cannot be modified

   //Below line, uncommented, causes compilation Error

   //finalArgument = 5;//COMPILER ERROR

}
```

What happens when a variable is marked as volatile?

Volatile can only be applied to instance variables.

A volatile variable is one whose value is always written to and read from "main memory". Each thread has its own cache in Java. The volatile variable will not be stored on a Thread cache.

What is a Static Variable?

Static variables and methods are class level variables and methods. There is only one copy of the static variable for the entire Class. Each instance of the Class (object) will NOT have a unique copy of a static variable. Let's start with a real world example of a Class with static variable and methods.

Static Variable/Method – Example

count variable in Cricketer class is static. The method to get the count value getCount() is also a static method.

Can we use a field or a method declared without access modifiers outside the package.?

No, we can't use a field or a method with no-access (default) specifiers outside the package in which their class is defined.

Can a method or a class be final and abstract at the same time.?

No, it is not possible. A class or a method can not be final and abstract at the same time. final and abstract are totally opposite in nature. final class or final method must not be modified further where as abstract class or abstract method must be modified further.

Can we declare a class as private.?

We can't declare an outer class as private. But, we can declare an inner class (class as a member of another class) as private.

Can we declare an abstract method as private also.?

No, abstract methods can not be private. They must be public or protected or default so that they can be modified further.

Can we declare a class as protected.?

We can't declare an outer class as protected. But, we can declare an inner class (class as a member of another class) as protected.

A class can not be declared with synchronized keyword. Then, why we call classes like Vector, StringBuffer are synchronized classes.?

Any classes which have only synchronized methods and blocks are treated as synchronized classes. Classes like Vector, StringBuffer have only synchronized methods. That's why they are called as synchronized classes.

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.

Why do we need generics in java?
Code that uses generics has many benefits over non-generic code:

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

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

A library is a collection of class definitions and its implementations. The main benifits 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 develope 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.

Can Enum extend any class in Java?

Enum can not 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 can not extend another class.

Can we have constructor in abstract class?

You can not 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 its also applies on abstract class.

Explain Abstract Class.

Abstract class contains either all abstract functions or a combination of abstract/concrete functions, or all abstract functions. A class extending the abstract class, need not to

implement all the functions of the abstract class. An abstract class can have any type of variables.

What is the difference between abstract class and interface?

a) Abstract class is a class which contain one or more abstract methods, which has to be implemented by sub classes. An abstract class can contain no abstract methods also i.e. abstract class may contain concrete methods. A Java Interface can contain only method declarations and public static final constants and doesn't contain their implementation. The classes which implement the Interface must provide the method definition for all the methods present.

b) Abstract class definition begins with the keyword "abstract" keyword followed by Class definition. An Interface definition begins with the keyword "interface".

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

d) All variables in an Interface are by default – public static final while an abstract class can have instance variables.

e) An interface is also used in situations when a class needs to extend an other class apart from the abstract class. In such situations its not possible to have multiple inheritance of classes. An interface on the other hand can be used when it is required to implement one or more interfaces. Abstract class does not support Multiple Inheritance whereas an Interface supports multiple Inheritance.

f) An Interface can only have public members whereas an abstract class can contain private as well as protected members.

g) A class implementing an interface must implement all of the methods defined in the interface, while a class extending an abstract class need not implement any of the methods defined in the abstract class.

h) The problem with an interface is, if you want to add a new feature (method) in its contract, then you MUST implement those method in all of the classes which implement that interface. However, in the case of an abstract class, the method can be simply implemented in the abstract class and the same can be called by its subclass

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

j) Interfaces are often used to describe the peripheral abilities of a class, and not its central identity, E.g. an Automobile class might implement the Recyclable interface, which could apply to many otherwise totally unrelated objects.

Note: There is no difference between a fully abstract class (all methods declared as abstract and all fields are public static final) and an interface.

Note: If the various objects are all of-a-kind, and share a common state and behavior, then tend towards a common base class. If all they share is a set of method signatures, then tend towards an interface.

Similarities:

Neither Abstract classes nor Interface can be instantiated.

Can abstract class be instantiated?

No an abstract class can't be instantiated. We can only instantiate a subclass of the abstract class. Condition is that sub class should also not be abstract.

What is 'Abstract Interface'?

All interfaces are abstract interfaces. All methods are abstract (method signatures) in an interface.

Can abstract class have constructors?

Unlike Interfaces, Abstract class can have a constructor. To access the constructor, we have to first extend the abstract class. Then the subclass can be instantiated and the constructor will be called (if not overridden).

Abstract Class implementing an Interface. What would be the behaviour?

In such a case, abstract class need not implement all methods of the interface as it is abstract.

What happens if we define abstract class as final?

It's not permitted in Java. A final class can't be extended. That defeats the very purpose of abstract class. Abstract and Final are in fact opposites of each other.

What methods can be there inside abstract class?

Abstract class can have both abstract and concrete methods. Not compulsory to have all methods abstract.

Is String a keyword in java?

No. String is not a keyword in java. String is a final class in java.lang package which is used to represent the set of characters in java.

Is String a primitive type or derived type?

String is a derived type.

In how many ways you can create string objects in java?

There are two ways to create string objects in java. One is using new operator and another one is using string literals. The objects created using new operator are stored in the heap memory and objects created using string literals are stored in string constant pool.

```
String s1 = new String("abc"); //Creating string object using new operator

String s2 = "abc";       //Creating string object using string literal
```

What is string constant pool?

String objects are most used data objects in Java. Hence, java has a special arrangement to store the string objects. String Constant Pool is one such arrangement. String Constant Pool is the memory space in heap memory specially allocated to store the string objects created using string literals. In String Constant Pool, there will be no two string objects having the same content.

Whenever you create a string object using string literal, JVM first checks the content of the object to be created. If there exist an object in the string constant pool with the same content, then it returns the reference of that object. It doesn't create a new object. If the content is different from the existing objects then only it creates new object.

What is special about string objects as compared to objects of other derived types?

One special thing about string objects is that you can create string objects without using new operator i.e using string literals. This is not possible with other derived types (except wrapper classes). One more special thing about strings is that you can concatenate two string objects using '+'. This is the relaxation java gives to string objects as they will be used most of the time while coding. And also java provides string constant pool to store the string objects.

What do you mean by mutable and immutable objects?

Immutable objects are like constants. You can't modify them once they are created. They are final in nature. Whereas mutable objects are concerned, you can perform modifications to them.

Which is the final class in these three classes – String, StringBuffer and StringBuilder?

All three are final. (Interviewer will ask this type of questions to confuse you)

What is the difference between String, StringBuffer and StringBuilder?

Why StringBuffer and StringBuilder classes are introduced in java when there already exist String class to represent the set of characters?

The objects of String class are immutable in nature. i.e you can't modify them once they are created. If you try to modify them, a new object will be created with modified content. This may cause memory and performance issues if you are performing lots of string modifications in your code. To overcome these issues, StingBuffer and StringBuilder classes are introduced in java.

How many objects will be created in the following code and where they will be stored in the memory?

```
String s1 = "abc";
 String s2 = "abc";
```

Only one object will be created and this object will be stored in the string constant pool.

How do you create mutable string objects?

Using StringBuffer and StringBuilder classes. These classes provide mutable string objects.

Which one will you prefer among "==" and equals() method to compare two string objects?

I prefer *equals()* method because it compares two string objects based on their content. That provides more logical comparison of two string objects. If you use *"=="* operator, it

checks only references of two objects are equal or not. It may not be suitable in all situations. So, rather stick to *equals()* method to compare two string objects.

Which class will you recommend among String, StringBuffer and StringBuilder classes if I want mutable and thread safe objects?

StringBuffer**How do you convert given string to char array?**

Using *toCharArray()* method.

How many objects will be created in the following code and where they will be stored?

```
String s1 = new String("abc");
 String s2 = "abc";
```

Here, two string objects will be created. Object created using new operator(s1) will be stored in the heap memory. The object created using string literal(s2) is stored in the string constant pool.

Where exactly string constant pool is located in the memory?

Inside the heap memory. JVM reserves some part of the heap memory to store string objects created using string literals.

I am performing lots of string concatenation and string modification in my code. which class among string, StringBuffer and StringBuilder improves the performance of my code. Remember I also want thread safe code?

StringBuffer class gives better performance in this scenario. As *String* class is immutable, if you use this class, a new object will be created after every string concatenation or string modification. This will lower the performance of the code. You can use *StringBuilder* also, but it is not thread safe. So, *StringBuffer* will be optimal choice here.

What is string intern?

String object in the string constant pool is called as *String Intern*. You can create an exact copy of heap memory string object in string constant pool. This process of creating an exact copy of heap memory string object in the string constant pool is called interning. *intern()* method is used for interning.

What is the main difference between Java strings and C, C++ strings?

In C and C++, strings are terminated with null character. But in java, strings are not terminated with null character. Strings are treated as objects in java.

How many objects will be created in the following code and where they will be stored?

```
String s1 = new String("abc");
 String s2 = new String("abc");
```

Two objects will be created and they will be stored in the heap memory.

Can we call String class methods using string literals?

Yes, we can call String class methods using string literals. Here are some examples,

```
"abc".charAt(0)
 "abc".compareTo("abc")
 "abc".indexOf('c')
```

do you have any idea why strings have been made immutable in java?

a) Immutable strings increase *security*. As they can't be modified once they are created, so we can use them to store sensitive data like username, password etc.

b) Immutable strings are *thread safe*. So, we can use them in a multi threaded code without synchronization.

c) String objects are used in class loading. If strings are mutable, it is possible that wrong class is being loaded as mutable objects are modifiable.

What do you think about string constant pool? Why they have provided this pool as we can store string objects in the heap memory itself?

String constant pool increases the reusability of existing string objects. When you are creating a string object using string literal, JVM first checks string constant pool. If that object is available, it returns reference of that object rather creating a new object. This will

also speed up your application as only reference is returned and also saves the memory as no two objects with same content are created.

What is the similarity and difference between String and StringBuffer class?

The main similarity between *String* and *StringBuffer* class is that both are thread safe. The main difference between them is that *String* objects are immutable where as *StringBuffer* objects are mutable.

What is the similarity and difference between StringBuffer and StringBuilder class?

The main similarity between *StringBuffer* and *StringBuilder* class is that both produces mutable string objects. The main difference between them is that *StringBuffer* class is thread safe where as *StringBuilder* class is not thread safe.

How can you create an immutable class in java?

Here are the steps to create immutable class: Declare the class as final, we can not extend the final class.

public final class MyTestImmutable { ... }

Declare all fields as final. Final fields can not be changed once its assigned.

private final int salary;

Do not provide any method which can change the state of the object, for example the setter methods which changes the values of the instance variables.

The “this” reference is not allowed to escape during construction from the immutable class and the immutable class should have exclusive access to fields that contain references to mutable objects like arrays, collections and mutable classes like Date etc by:

Declaring the mutable references as private. Not returning or exposing the mutable references to the caller.

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.

When to use String and StringBuffer?

We know that String is immutable object. We can not 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 incase, we are going to do more modificatios on String, then use StringBuffer. StringBuffer updates the existing objects value, rather creating new object.

How to remove non-ascii characters from a string?

Some times we need to handle text data, wherein we have to handle only ascii characters. Below example shows how to remove non-ascii characters from the given string by using regular expression.

How to remove html tags from a String?

In case if a string contains html tags, then below example helps to trim the html tags from the string. The example uses regular expression to trim the html tags from the string.

What is a Thread?

In Java, "thread" means two different things:

- An instance of class java.lang.Thread.
- A thread of execution.

An instance of Thread is just…an object. Like any other object in Java, it has variables and methods, and lives and dies on the heap. But a thread of execution is an individual process (a "lightweight" process) that has its own call stack. In Java, there is one thread per call stack—or, to think of it in reverse, one call stack per thread. Even if you don't create any new threads in your program, threads are back there running.

The main() method, that starts the whole ball rolling, runs in one thread, called (surprisingly) the main thread. If you looked at the main call stack (and you can, any time you get a stack trace from something that happens after main begins, but not within

another thread), you'd see that main() is the first method on the stack— the method at the bottom. But as soon as you create a new thread, a new stack materializes and methods called from that thread run in a call stack that's separate from the main() call stack.

What is difference between thread and process?

Differences between threads and processes are:-

1. Threads share the address space of the process that created it; processes have their own address.

2. Threads have direct access to the data segment of its process; processes have their own copy of the data segment of the parent process.

3. Threads can directly communicate with other threads of its process; processes must use inter process communication to communicate with sibling processes.

4. Threads have almost no overhead; processes have considerable overhead.

5. New threads are easily created; new processes require duplication of the parent process.

6. Threads can exercise considerable control over threads of the same process; processes can only exercise control over child processes.

7. Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process; changes to the parent process do not affect child processes.

What are the advantages or usage of threads?

Threads support concurrent operations. For example,

• Multiple requests by a client on a server can be handled as an individual client thread.

• Long computations or high-latency disk and network operations can be handled in the background without disturbing foreground computations or screen updates.

Threads often result in simpler programs.

• In sequential programming, updating multiple displays normally requires a big while-loop that performs small parts of each display update. Unfortunately, this loop basically simulates an operating system scheduler. In Java, each view can be assigned a thread to provide continuous updates.

• Programs that need to respond to user-initiated events can set up service routines to handle the events without having to insert code in the main routine to look for these events.

Threaded applications exploit parallelism.

• A computer with multiple CPUs can literally execute multiple threads on different functional units without having to simulating multi-tasking ("time sharing").

• On some computers, one CPU handles the display while another handles computations or database accesses, thus, providing extremely fast user interface response times.

What is use of synchronized keyword?

synchronized keyword can be applied to static/non-static methods or a block of code. Only one thread at a time can access synchronized methods and if there are multiple threads trying to access the same method then other threads have to wait for the execution of method by one thread. Synchronized keyword provides a lock on the object and thus prevents race condition. E.g.

```
public void synchronized method(){}
public void synchronized staticmethod(){}
public void myMethod(){

        synchronized (this){          // synchronized keyword on block of  code
        }
}
```

What is the difference when the synchronized keyword is applied to a static method or to a non static method?

When a synch non static method is called a lock is obtained on the object. When a synch static method is called a lock is obtained on the class and not on the object. The lock on the object and the lock on the class donâ€™t interfere with each other. It means, a thread accessing a synch non static method, then the other thread can access the synch static method at the same time but canâ€™t access the synch non static method.

What is a volatile keyword?

In general each thread has its own copy of variable, such that one thread is not concerned with the value of same variable in the other thread. But sometime this may not be the case. Consider a scenario in which the count variable is holding the number of times a method is called for a given class irrespective of any thread calling, in this case irrespective of thread access the count has to be increased so the count variable is declared as volatile. The copy of volatile variable is stored in the main memory, so every time a thread access the variable even for reading purpose the local copy is updated each time from the main memory.

The volatile variable also have performance issues.

What is the difference between yield() and sleep()?

yield() allows the current thread to release its lock from the object and scheduler gives the lock of the object to the other thread with same priority.

sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn't release the lock.

What is the difference between wait() and sleep()?

-→ wait() is a method of Object class. sleep() is a method of Object class.

-→ sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn't release the lock. wait() allows thread to release the lock and goes to suspended state. The thread is only active when a notify() or notifAll() method is called for the same object.

What is difference between notify() and notfiyAll()?

notify( ) wakes up the first thread that called wait( ) on the same object.
notifyAll( ) wakes up all the threads that called wait( ) on the same object. The highest priority thread will run first.

What happens if a start method is not invoked and the run method is directly invoked?

If a thread has been instantiated but not started it is said to be in new state. Unless until a start() method is invoked on the instance of the thread, it will not said to be alive. If you do not call a start() method on the newly created thread instance thread is not considered to be alive. If the start() method is not invoked and the run() method is directly called on the Thread instance, the code inside the run() method will not run in a separate new thread but it will start running in the existing thread.

What happens when start() is called?

A new thread of execution with a new call stack starts. The state of thread changes from new to runnable. When the thread gets chance to execute its target run() method starts to run.

If code running is a thread creates a new thread what will be the initial priority of the newly created thread?

When a code running in a thread creates a new thread object , the priority of the new thread is set equal to the priority of the thread which has created it.

What are the daemon threads?

Daemon thread are service provider threads run in the background, these not used to run the application code generally. When all user threads (non-daemon threads) complete their

execution, the jvm exit the application whatever may be the state of the daemon threads. Jvm does not wait for the daemon threads to complete their execution if all user threads have completed their execution.

What all constructors are present in the Thread class?
Thread()
Thread(Runnable target)
Thread(Runnable target, String name)
Thread(String name)

Can the variables or classes be Synchronized?
Only methods & blocks can be synchronized.

How many locks does an object have?
Each object has only one lock.

Why would you use a synchronized block vs. synchronized method?
Synchronized blocks place locks for shorter periods than synchronized methods.

There are two classes: A and B. The class B need to inform a class A when some important event has happened. What Java technique would you use to implement it?
If these classes are threads I'd consider notify() or notifyAll(). For regular classes you can use the Observer interface.

How do you ensure that N thread can access N resources without deadlock
Key point here is order, if you acquire resources in a particular order and release resources in reverse order you can prevent deadlock.

What is static in java?
Static is a keyword in java.
One of the Important keyword in java.
Clear understanding of static keyword is required to build projects.
We have static variables, static methods , static blocks.
Static means class level.

Why we use static keyword in java?
Static keyword is mainly used for memory management.
Static variables get memory when class loading itself.
Static variables can be used to point common property all objects.

What is static variable in java?

Variables declared with static keyword is known as static variables.
Static variables gets memory on class loading.
Static variables are class level.
If we change any static variable value using a particular object then its value changed for all objects means it is common to every object of that class.
We can not declare local variables as static it leads to compile time error "illegal start of expression".
Because being static variable it must get memory at the time of class loading, which is not possible to provide memory to local variable at the time of class loading.

What is static method in java with example

Method which is having static in its method definition is known as static method.
JVM will not call these static methods automatically. Developer needs to call these static methods from main method or static block or variable initialization.
Only Main method will be called by JVM automatically.
We can call these static methods by using class name itself no need to create object.

What is static block in java?

Static blocks are the blocks which will have braces and with static keyword.
These static blocks will be called when JVM loads the class.
Static blocks are the blocks with static keyword.
Static blocks wont have any name in its prototype.
Static blocks are class level.
Static block will be executed only once.
No return statements.
No arguments.
No this or super keywords supported.

What is the need of static block?

Static blocks will be executed at the time of class loading.
So if you want any logic that needs to be executed at the time of class loading that logic need to place inside the static block so that it will be executed at the time of class loading.

Why main method is static in java?

To execute main method without creating object then the main method should be static so that JVM will call main method by using class name itself.

Can we write static public void main(String [] args)?

Yes we can define main method like static public void main(String[] args){}

Order of modifiers we can change.

Can we call super class static methods from sub class?
Yes we can call super class static methods from sub class.

What are the different types of memory used by JVM ?
Ans. Class , Heap , Stack , Register , Native Method Stack.

What are various types of Class loaders used by JVM ?
Ans. Bootstrap - Loads JDK internal classes, java.* packages.
Extensions - Loads jar files from JDK extensions directory - usually lib/ext directory of the JRE
System - Loads classes from system classpath.

What is PermGen or Permanent Generation ?
Ans. The memory pool containing all the reflective data of the java virtual machine itself, such as class and method objects. With Java VMs that use class data sharing, this generation is divided into read-only and read-write areas. The Permanent generation contains metadata required by the JVM to describe the classes and methods used in the application. The permanent generation is populated by the JVM at runtime based on classes in use by the application. In addition, Java SE library classes and methods may be stored here.

How does volatile affect code optimization by compiler?
Ans. Volatile is an instruction that the variables can be accessed by multiple threads and hence shouldn't be cached. As volatile variables are never cached and hence their retrieval cannot be optimized.

What things should be kept in mind while creating your own exceptions in Java?
Ans. All exceptions must be a child of Throwable.
If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class.
You want to write a runtime exception, you need to extend the RuntimeException class.

Q1 What is Collection ? What is a Collections Framework ? What are the benefits of Java Collections Framework ?

Collection : A collection (also called as container) is an object that groups multiple elements into a single unit.

Collections Framework : Collections framework provides unified architecture for manipulating and representing collections.

Benefits of Collections Framework :

1. Improves program quality and speed
2. Increases the chances of reusability of software
3. Decreases programming effort.

What is the root interface in collection hierarchy ?

Root interface in collection hierarchy is **Collection interface .** Few interviewers may argue that Collection interface extends **Iterable interface**. So iterable should be the root interface. But you should reply iterable interface present in java.lang package not in java.util package .It is clearly mentioned in Oracle Collection docs , that Collection interface is a member of the Java Collections framework. For Iterable interface Oracle doc , iterable interface is not mentioned as a part of the Java Collections framework .So if the question includes collection hierarchy , then you should answer the question as Collection interface (which is found in java.util package).

What is the difference between Collection and Collections ?

Collection is an interface while Collections is a java class , both are present in java.util package and part of java collections framework.

Which collection classes are synchronized or thread-safe ?

Stack, Properties, Vector and Hashtable can be used in multi threaded environment because they are synchronized classes (or thread-safe).

What is the difference between List and Set ?

Set contain only unique elements while List can contain duplicate elements.
Set is unordered while List is ordered . List maintains the order in which the objects are added .

What is the difference between Map and Set ?

Map object has unique keys each containing some value, while Set contain only unique values.

What are the classes implementing List and Set interface ?

Class implementing List interface : ArrayList , Vector , LinkedList ,

Class implementing Set interface : HashSet , TreeSet

Which design pattern followed by Iterator ?

It follows iterator design pattern. Iterator design pattern provides us to navigate through the collection of objects by using a common interface without letting us know about the underlying implementation.

Enumeration is an example of Iterator design pattern.

Which methods you need to override to use any object as key in HashMap ?

To use any object as key in HashMap , it needs to implement equals() and hashCode() method .

What is the difference between Queue and Stack ?

Queue is a data structure which is based on FIFO (first in first out) property . An example of Queue in real world is buying movie tickets in the multiplex or cinema theaters.

Stack is a data structure which is based on LIFO (last in first out) property . An example of Stack in real world is insertion or removal of CD from the CD case.

How to reverse the List in Collections ?

There is a built in reverse method in Collections class . reverse(List list) accepts list as parameter.
Collections.reverse(listobject);

How to convert the array of strings into the list ?

Arrays class of java.util package contains the method asList() which accepts the array as parameter.
So,

String[] wordArray = {"Love Yourself" , "Alive is Awesome" , "Be in present"};
List wordList = Arrays.asList(wordArray);

What is the difference between ArrayList and Vector ?
It is one of the frequently asked collection interview question , the main differences are Vector is synchronized while ArrayList is not . Vector is slow while ArrayList is fast . Every time when needed, Vector increases the capacity twice of its initial size while ArrayList increases its ArraySize by 50%.

What is the difference between HashMap and Hashtable ?
It is one of the most popular collections interview question for java developer . Make sure you go through this once before appearing for the interview .
Main differences between HashMap and Hashtable are :

a. HashMap allows one null key and any number of null values while Hashtable does not allow null keys and null values.
b. HashMap is not synchronized or thread-safe while Hashtable is synchronized or thread-safe .

What is the difference between peek(),poll() and remove() method of the Queue interface ?
Both poll() and remove() method is used to remove head object of the Queue. The main difference lies when the Queue is empty().
If Queue is empty then poll() method will return null . While in similar case , remove() method will throw NoSuchElementException .
peek() method retrieves but does not remove the head of the Queue. If queue is empty then peek() method also returns null.

What is the difference between Iterator and ListIterator.
Using Iterator we can traverse the list of objects in forward direction . But ListIterator can traverse the collection in both directions that is forward as well as backward.

What is the difference between Array and ArrayList in Java ?
This question checks whether student understand the concept of static and dynamic array.
Some main differences between Array and ArrayList are :
a. Array is static in size while ArrayList is dynamic in size.
b. Array can contain primitive data types while ArrayList can not contain primitive data types.

What is the difference between HashSet and TreeSet ?
Main differences between HashSet and TreeSet are :
a. HashSet maintains the inserted elements in random order while TreeSet maintains elements in the sorted order
b. HashSet can store null object while TreeSet can not store null object.

What is the difference between HashMap and ConcurrentHashMap ?
a. HashMap is not synchronized while ConcurrentHashMap is synchronized.
b. HashMap can have one null key and any number of null values while ConcurrentHashMap does not allow null keys and null values .

How HashMap works in Java ?
HashMap works on the principle of Hashing.

What is the difference between LinkedList and ArrayList in Java ?
Main differences between LinkedList and ArrayList are :
a. LinkedList is the doubly linked list implementation of list interface , while , ArrayList is the resizable array implementation of list interface.
b. LinkedList can be traversed in the reverse direction using descendingIterator() method provided by the Java Api developers , while , we need to implement our own method to traverse ArrayList in the reverse direction .

Why Map interface does not extend the Collection interface in Java Collections Framework ?
One liner answer : **Map interface is not compatible with the Collection interface.**
Explanation : Since Map requires key as well as value , for example , if we want to add key-value pair then we will use put(Object key , Object value) . So there are two parameters required to add element to the HashMap object . In Collection interface add(Object o) has only one parameter.
The other reasons are Map supports valueSet , keySet as well as other appropriate methods which have just different views from the Collection interface.

When to use ArrayList and when to use LinkedList in application?
ArrayList has constant time search operation O(1) .Hence, ArrayList is preferred when there are more get() or search operation .
Insertion , Deletion operations take constant time O(1) for LinkedList. Hence, LinkedList is preferred when there are more insertions or deletions involved in the application.

Write the code for iterating the list in different ways in java ?
There are two ways to iterate over the list in java :
a. using Iterator
b. using for-each loop

How HashSet works internally in java ?
HashSet internally uses HashMap to maintain the uniqueness of elements. We have already discussed in detail hashset internal working in java.

What is CopyOnWriteArrayList ? How it is different from ArrayList in Java?

CopyOnWriteArrayList is a thread safe variant of ArrayList in which all mutative operations like add , set are implemented by creating a fresh copy of the underlying array. It guaranteed not to throw ConcurrentModificationException. It permits all elements including null. It is introduced in jdk 1.5 .

What is BlockingQueue in Java Collections Framework?
BlockingQueue implements the java.util.Queue interface . BlockingQueue supports operations that wait for the queue to become non-empty when retrieving an element , and wait for space to become available in the queue when storing an element . It does not accept null elements.
Blocking queues are primarily designed for the producer-consumer problems.
BlockingQueue implementations are thread-safe and can also be used in inter-thread communications.
This concurrent Collection class was added in jdk 1.5

How TreeMap works in Java ?
TreeMap internally uses Red-Black tree to sort the elements in natural order.

What is the difference between Fail- fast iterator and Fail-safe iterator ?
Main differences between Fail-fast and Fail-safe iterators are :
a. Fail-fast throw ConcurrentModificationException while Fail-safe does not.
b. Fail-fast does not clone the original collection list of objects while Fail-safe creates a copy of the original collection list of objects.

How do you use a custom object as key in Collection classes like HashMap ?
If one is using the custom object as key then one needs to override equals() and hashCode() method and one also need to fulfill the contract.
If you want to store the custom object in the SortedCollections like SortedMap then one needs to make sure that equals() method is consistent to the compareTo() method. If inconsistent , then collection will not follow their contracts ,that is , Sets may allow duplicate elements.

What is hash-collision in Hashtable ? How it was handled in Java?
In Hashtable , if two different keys have the same hash value then it lead to hash -collision. A bucket of type linkedlist used to hold the different keys of same hash value.

Explain the importance of hashCode() and equals() method ? Explain the contract also ?
HashMap object uses Key object hashCode() method and equals() method to find out the index to put the key-value pair. If we want to get value from the HashMap same both methods are used . Somehow, if both methods are not implemented correctly , it will result in two keys producing the same hashCode() and equals() output. The problem will arise that HashMap will treat both output same instead of different and overwrite the most recent key-value pair with the previous key-value pair.
Similarly all the collection classes that does not allow the duplicate values use hashCode() and equals() method to find the duplicate elements.So it is very important to implement them correctly.

Contract of hashCode() and equals() method
a. If object1.equals(object2) , then object1.hashCode() == object2.hashCode() should always be true.

b. If object1.hashCode() == object2.hashCode() is true does not guarantee object1.equals(object2)

What are concurrentCollectionClasses?
In jdk1.5 , Java Api developers had introduced new package called java.util.concurrent that have thread-safe collection classes as they allow collections to be modified while iterating . The iterator is fail-safe that is it will not throw ConcurrentModificationException.
Some examples of concurrentCollectionClasses are :
a. CopyOnWriteArrayList
b. ConcurrentHashMap

How do you convert a given Collection to SynchronizedCollection ?
One line code : Collections.synchronizedCollection(Collection collectionObj) will convert a given collection to synchronized collection.

What is IdentityHashMap ?
IdentityHashMap
IdentityHashMap is a class present in java.util package. It implements the Map interface with a hash table , using reference equality instead of object equality when comparing keys and values.In other words , in IdentityHashMap two keys k1 and k2 are considered equal if only if (k1==k2).
IdentityHashMap is not synchronized. Iterators returned by the iterator() method are fail-fast , hence , will throw ConcurrentModificationException.

What is WeakHashMap ?
WeakHashMap :
WeakHashMap is a class present in java.util package similar to IdentityHashMap. It is a Hashtable based implementation of Map interface with weak keys. An entry in WeakHashMap will automatically be removed when its key is no longer in ordinary use. More precisely the presence of a mapping for a given key will not prevent the key from being discarded by the garbage collector. It permits null keys and null values.
Like most collection classes this class is not synchronized.A synchronized WeakHashMap may be constructed using the Collections.synchronizedMap() method. Iterators returned by the iterator() method are fail-fast , hence, will throw ConcurrentModificationException.

How will you make Collections readOnly ?
We can make the Collection readOnly by using the following lines code:
General : Collections.unmodifiableCollection(Collection c)
Collections.unmodifiableMap(Map m)
Collections.unmodifiableList(List l)
Collections.unmodifiableSet(Set s)

What is UnsupportedOperationException?

This exception is thrown to indicate that the requested operation is not supported.
Example of UnsupportedOperationException:
In other words, if you call add() or remove() method on the readOnly collection . We know readOnly collection can not be modified . Hence , UnsupportedOperationException will be thrown.

Suppose there is an Employee class. We add Employee class objects to the ArrayList. Mention the steps need to be taken , if I want to sort the objects in ArrayList using the employeeId attribute present in Employee class.

a. Implement the Comparable interface for the Employee class and now to compare the objects by employeeId we will override the emp1.compareTo(emp2)
b. We will now call Collections class sort method and pass the list as argument , that is ,
Collections.sort(empList)

If you want to add more java collections interview questions and answers or in case you have any doubts related to the Java Collections framework , then please mention in the comments.

What do you need to do to use a custom object as a key in Collection classes like Map or Set?

The answer is: If you are using any custom object in Map as key, you need to override equals() and hashCode() method, and make sure they follow their contract. On the other hand if you are storing a custom object in Sorted Collection e.g. SortedSet or SortedMap, you also need to make sure that your equals() method is consistent to compareTo()method, otherwise that collection will not follow there contacts e.g. Set may allow duplicates.