JAVA INTERVIEW QUESTIONS AND ANSWERS

BOOSTING YOUR JAVA CAREER

STATHIS MANEAS

Java Code Geeks
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

Java Interview Questions

Contents

Preface

In this guide we will discuss about different types of questions that can be used in a Java interview, in order for the employer to test your skills in Java and object-oriented programming in general.

In the following sections we will discuss about object-oriented programming and its characteristics, general questions regarding Java and its functionality, collections in Java, garbage collectors, exception handling, Java applets, Swing, JDBC, Remote Method Invocation (RMI), Servlets and JSP.

About the Author

Sotirios-Efstathios (Stathis) Maneas is a postgraduate student at the Department of Informatics and Telecommunications of The National and Kapodistrian University of Athens. His main interests include distributed systems, web crawling, model checking, operating systems, programming languages and web applications.

Chapter 1

Object Oriented Programming (OOP)

Java is a computer programming language that is concurrent, class-based and object-oriented. The advantages of object oriented software development are shown below:

- Modular development of code, which leads to easy maintenance and modification.
- Reusability of code.
- Improved reliability and flexibility of code.
- Increased understanding of code.

Object-oriented programming contains many significant features, such as **encapsulation**, **inheritance**, **polymorphism** and **abstraction**. We analyze each feature separately in the following sections.

1.1 Encapsulation

Encapsulation provides objects with the ability to hide their internal characteristics and behavior. Each object provides a number of methods, which can be accessed by other objects and change its internal data. In Java, there are three access modifiers: public, private and protected. Each modifier imposes different access rights to other classes, either in the same or in external packages. Some of the advantages of using encapsulation are listed below:

- The internal state of every objected is protected by hiding its attributes.
- It increases usability and maintenance of code, because the behavior of an object can be independently changed or extended.
- It improves modularity by preventing objects to interact with each other, in an undesired way.

You can refer to our tutorial here for more details and examples on encapsulation.

1.2 Polymorphism

Polymorphism is the ability of programming languages to present the same interface for differing underlying data types. A polymorphic type is a type whose operations can also be applied to values of some other type.

1.3 Inheritance

Inheritance provides an object with the ability to acquire the fields and methods of another class, called base class. Inheritance provides re-usability of code and can be used to add additional features to an existing class, without modifying it.

1.4 Abstraction

Abstraction is the process of separating ideas from specific instances and thus, develop classes in terms of their own functionality, instead of their implementation details. Java supports the creation and existence of abstract classes that expose interfaces, without including the actual implementation of all methods. The abstraction technique aims to separate the implementation details of a class from its behavior.

1.5 Differences between Abstraction and Encapsulation

Abstraction and encapsulation are complementary concepts. On the one hand, abstraction focuses on the behavior of an object. On the other hand, encapsulation focuses on the implementation of an object's behavior. Encapsulation is usually achieved by hiding information about the internal state of an object and thus, can be seen as a strategy used in order to provide abstraction.

Chapter 2

General Questions about Java

2.1 What is JVM ? Why is Java called the Platform Independent Programming Language?

A Java virtual machine (JVM) is a process virtual machine that can execute Java bytecode. Each Java source file is compiled into a bytecode file, which is executed by the JVM. Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

2.2 What is the Difference between JDK and JRE ?

The Java Runtime Environment (JRE) is basically the Java Virtual Machine (JVM) where your Java programs are being executed. It also includes browser plugins for applet execution. The Java Development Kit (JDK) is the full featured Software Development Kit for Java, including the JRE, the compilers and tools (like JavaDoc, and Java Debugger), in order for a user to develop, compile and execute Java applications.

2.3 What does the “static” keyword mean ? Can you override private or static method in Java ?

The static keyword denotes that a member variable or method can be accessed, without requiring an instantiation of the class to which it belongs. A user cannot override static methods in Java, because method overriding is based upon dynamic binding at runtime and static methods are statically binded at compile time. A static method is not associated with any instance of a class so the concept is not applicable.

2.4 Can you access non static variable in static context ?

A static variable in Java belongs to its class and its value remains the same for all its instances. A static variable is initialized when the class is loaded by the JVM. If your code tries to access a non-static variable, without any instance, the compiler will complain, because those variables are not created yet and they are not associated with any instance.

2.5 What are the Data Types supported by Java ? What is Autoboxing and Unboxing ?

The eight primitive data types supported by the Java programming language are:

- byte
- short
- int
- long
- float
- double
- boolean
- char

`Autoboxing` is the automatic conversion made by the Java compiler between the primitive types and their corresponding object wrapper classes. For example, the compiler converts an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this operation is called `unboxing`.

2.6 What is Function Overriding and Overloading in Java ?

Method overloading in Java occurs when two or more methods in the same class have the exact same name, but different parameters. On the other hand, method overriding is defined as the case when a child class redefines the same method as a parent class. Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides.

2.7 What is a Constructor, Constructor Overloading in Java and Copy-Constructor

A constructor gets invoked when a new object is created. Every class has a constructor. In case the programmer does not provide a constructor for a class, the Java compiler (Javac) creates a default constructor for that class. The constructor overloading is similar to method overloading in Java. Different constructors can be created for a single class. Each constructor must have its own unique parameter list. Finally, Java does support copy constructors like C++, but the difference lies in the fact that Java doesn't create a default copy constructor if you don't write your own.

2.8 Does Java support multiple inheritance ?

No, Java does not support multiple inheritance. Each class is able to extend only on one class, but is able to implement more than one interfaces.

2.9 What is the difference between an Interface and an Abstract class ?

Java provides and supports the creation both of abstract classes and interfaces. Both implementations share some common characteristics, but they differ in the following features:

- All methods in an interface are implicitly abstract. On the other hand, an abstract class may contain both abstract and non-abstract methods.

- A class may implement a number of Interfaces, but can extend only one abstract class.
- In order for a class to implement an interface, it must implement all its declared methods. However, a class may not implement all declared methods of an abstract class. Though, in this case, the sub-class must also be declared as abstract.
- Abstract classes can implement interfaces without even providing the implementation of interface methods.
- Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.
- Members of a Java interface are public by default. A member of an abstract class can either be private, protected or public.
- An interface is absolutely abstract and cannot be instantiated. An abstract class also cannot be instantiated, but can be invoked if it contains a main method.

Also check out the Abstract class and Interface differences for JDK 8.

2.10 What are pass by reference and pass by value ?

When an object is passed by value, this means that a copy of the object is passed. Thus, even if changes are made to that object, it doesn't affect the original value. When an object is passed by reference, this means that the actual object is not passed, rather a reference of the object is passed. Thus, any changes made by the external method, are also reflected in all places.

Chapter 3

Java Threads

3.1 What is the difference between processes and threads ?

A process is an execution of a program, while a Thread is a single execution sequence within a process. A process can contain multiple threads. A Thread is sometimes called a lightweight process.

3.2 Explain different ways of creating a thread. Which one would you prefer and why ?

There are three ways that can be used in order for a Thread to be created:

- A class may extend the Thread class.
- A class may implement the Runnable interface.
- An application can use the Executor framework, in order to create a thread pool.

The Runnable interface is preferred, as it does not require an object to inherit the Thread class. In case your application design requires multiple inheritance, only interfaces can help you. Also, the thread pool is very efficient and can be implemented and used very easily.

3.3 Explain the available thread states in a high-level.

During its execution, a thread can reside in one of the following states:

- **Runnable**: A thread becomes ready to run, but does not necessarily start running immediately.
- **Running**: The processor is actively executing the thread code.
- **Waiting**: A thread is in a blocked state waiting for some external processing to finish.
- **Sleeping**: The thread is forced to sleep.
- **Blocked on I/O**: Waiting for an I/O operation to complete.
- **Blocked on Synchronization**: Waiting to acquire a lock.
- **Dead**: The thread has finished its execution.

3.4 What is the difference between a synchronized method and a synchronized block ?

In Java programming, each object has a lock. A thread can acquire the lock for an object by using the synchronized keyword. The synchronized keyword can be applied in a method level (coarse grained lock) or block level of code (fine grained lock).

3.5 How does thread synchronization occurs inside a monitor ? What levels of synchronization can you apply ?

The JVM uses locks in conjunction with monitors. A monitor is basically a guardian that watches over a sequence of synchronized code and ensuring that only one thread at a time executes a synchronized piece of code. Each monitor is associated with an object reference. The thread is not allowed to execute the code until it obtains the lock.

3.6 What's a deadlock ?

A condition that occurs when two processes are waiting for each other to complete, before proceeding. The result is that both processes wait endlessly.

3.7 How do you ensure that N threads can access N resources without deadlock ?

A very simple way to avoid deadlock while using N threads is to impose an ordering on the locks and force each thread to follow that ordering. Thus, if all threads lock and unlock the mutexes in the same order, no deadlocks can arise.

Chapter 4

Java Collections

4.1 What are the basic interfaces of Java Collections Framework ?

Java Collections Framework provides a well designed set of interfaces and classes that support operations on a collections of objects. The most basic interfaces that reside in the Java Collections Framework are:

- Collection, which represents a group of objects known as its elements.
- Set, which is a collection that cannot contain duplicate elements.
- List, which is an ordered collection and can contain duplicate elements.
- Map, which is an object that maps keys to values and cannot contain duplicate keys.

4.2 Why Collection doesn't extend Cloneable and Serializable interfaces ?

The Collection interface specifies groups of objects known as elements. Each concrete implementation of a Collection can choose its own way of how to maintain and order its elements. Some collections allow duplicate keys, while some other collections don't. The semantics and the implications of either cloning or serialization come into play when dealing with actual implementations. Thus, the concrete implementations of collections should decide how they can be cloned or serialized.

4.3 What is an Iterator ?

The Iterator interface provides a number of methods that are able to iterate over any Collection. Each Java Collection contains the iterator method that returns an Iterator instance. Iterators are capable of removing elements from the underlying collection during the iteration.

4.4 What differences exist between Iterator and ListIterator ?

The differences of these elements are listed below:

- An Iterator can be used to traverse the Set and List collections, while the ListIterator can be used to iterate only over Lists.
- The Iterator can traverse a collection only in forward direction, while the ListIterator can traverse a List in both directions.
- The ListIterator implements the Iterator interface and contains extra functionality, such as adding an element, replacing an element, getting the index position for previous and next elements, etc.

4.5 What is difference between fail-fast and fail-safe ?

The Iterator's fail-safe property works with the clone of the underlying collection and thus, it is not affected by any modification in the collection. All the collection classes in java.util package are fail-fast, while the collection classes in java.util.concurrent are fail-safe. Fail-fast iterators throw a `ConcurrentModificationException`, while fail-safe iterator never throws such an exception.

4.6 How HashMap works in Java ?

A HashMap in Java stores key-value pairs. The HashMap requires a hash function and uses hashCode and equals methods, in order to put and retrieve elements to and from the collection respectively. When the put method is invoked, the HashMap calculates the hash value of the key and stores the pair in the appropriate index inside the collection. If the key exists, its value is updated with the new value. Some important characteristics of a HashMap are its capacity, its load factor and the threshold resizing.

4.7 What is the importance of hashCode() and equals() methods ?

In Java, a HashMap uses the hashCode and equals methods to determine the index of the key-value pair and to detect duplicates. More specifically, the hashCode method is used in order to determine where the specified key will be stored. Since different keys may produce the same hash value, the equals method is used, in order to determine whether the specified key actually exists in the collection or not. Therefore, the implementation of both methods is crucial to the accuracy and efficiency of the HashMap.

4.8 What differences exist between HashMap and Hashtable ?

Both the HashMap and Hashtable classes implement the Map interface and thus, have very similar characteristics. However, they differ in the following features:

- A HashMap allows the existence of null keys and values, while a Hashtable doesn't allow neither null keys, nor null values.
- A Hashtable is synchronized, while a HashMap is not. Thus, HashMap is preferred in single-threaded environments, while a Hashtable is suitable for multi-threaded environments.
- A HashMap provides its set of keys and a Java application can iterate over them. Thus, a HashMap is fail-fast. On the other hand, a Hashtable provides an Enumeration of its keys.
- The Hashtable class is considered to be a legacy class.

4.9 What is difference between Array and ArrayList ? When will you use Array over ArrayList ?

The Array and ArrayList classes differ on the following features:

- Arrays can contain primitive or objects, while an ArrayList can contain only objects.
- Arrays have fixed size, while an ArrayList is dynamic.
- An ArrayList provides more methods and features, such as addAll, removeAll, iterator, etc.
- For a list of primitive data types, the collections use autoboxing to reduce the coding effort. However, this approach makes them slower when working on fixed size primitive data types.

4.10 What is difference between ArrayList and LinkedList ?

Both the ArrayList and LinkedList classes implement the List interface, but they differ on the following features:

- An ArrayList is an index based data structure backed by an Array. It provides random access to its elements with a performance equal to O(1). On the other hand, a LinkedList stores its data as list of elements and every element is linked to its previous and next element. In this case, the search operation for an element has execution time equal to O(n).
- The Insertion, addition and removal operations of an element are faster in a LinkedList compared to an ArrayList, because there is no need of resizing an array or updating the index when an element is added in some arbitrary position inside the collection.
- A LinkedList consumes more memory than an ArrayList, because every node in a LinkedList stores two references, one for its previous element and one for its next element.

Check also our article ArrayList vs. LinkedList.

4.11 What is Comparable and Comparator interface ? List their differences.

Java provides the Comparable interface, which contains only one method, called compareTo. This method compares two objects, in order to impose an order between them. Specifically, it returns a negative integer, zero, or a positive integer to indicate that the input object is less than, equal or greater than the existing object. Java provides the Comparator interface, which contains two methods, called compare and equals. The first method compares its two input arguments and imposes an order between them. It returns a negative integer, zero, or a positive integer to indicate that the first argument is less than, equal to, or greater than the second. The second method requires an object as a parameter and aims to decide whether the input object is equal to the comparator. The method returns true, only if the specified object is also a comparator and it imposes the same ordering as the comparator.

4.12 What is Java Priority Queue ?

The PriorityQueue is an unbounded queue, based on a priority heap and its elements are ordered in their natural order. At the time of its creation, we can provide a Comparator that is responsible for ordering the elements of the PriorityQueue. A PriorityQueue doesn't allow null values, those objects that doesn't provide natural ordering, or those objects that don't have any comparator associated with them. Finally, the Java PriorityQueue is not thread-safe and it requires O(log(n)) time for its enqueing and dequeing operations.

4.13 What do you know about the big-O notation and can you give some examples with respect to different data structures ?

The Big-O notation simply describes how well an algorithm scales or performs in the worst case scenario as the number of elements in a data structure increases. The Big-O notation can also be used to describe other behavior such as memory consumption. Since the collection classes are actually data structures, we usually use the Big-O notation to chose the best implementation to use, based on time, memory and performance. Big-O notation can give a good indication about performance for large amounts of data.

4.14 What is the tradeoff between using an unordered array versus an ordered array ?

The major advantage of an ordered array is that the search times have time complexity of O(log n), compared to that of an unordered array, which is O (n). The disadvantage of an ordered array is that the insertion operation has a time complexity of O(n), because the elements with higher values must be moved to make room for the new element. Instead, the insertion operation for an unordered array takes constant time of O(1).

4.15 What are some of the best practices relating to the Java Collection framework ?

- Choosing the right type of the collection to use, based on the application's needs, is very crucial for its performance. For example if the size of the elements is fixed and know a priori, we shall use an Array, instead of an ArrayList.
- Some collection classes allow us to specify their initial capacity. Thus, if we have an estimation on the number of elements that will be stored, we can use it to avoid rehashing or resizing.
- Always use Generics for type-safety, readability, and robustness. Also, by using Generics you avoid the ClassCastException during runtime.
- Use immutable classes provided by the Java Development Kit (JDK) as a key in a Map, in order to avoid the implementation of the hashCode and equals methods for our custom class.
- Program in terms of interface not implementation.
- Return zero-length collections or arrays as opposed to returning a null in case the underlying collection is actually empty.

4.16 What's the difference between Enumeration and Iterator interfaces ?

Enumeration is twice as fast as compared to an Iterator and uses very less memory. However, the Iterator is much safer compared to Enumeration, because other threads are not able to modify the collection object that is currently traversed by the iterator. Also, Iteratorsallow the caller to remove elements from the underlying collection, something which is not possible with Enumerations.

4.17 What is the difference between HashSet and TreeSet ?

The HashSet is Implemented using a hash table and thus, its elements are not ordered. The add, remove, and contains methods of a HashSet have constant time complexity O(1). On the other hand, a TreeSet is implemented using a tree structure. The elements in a TreeSet are sorted, and thus, the add, remove, and contains methods have time complexity of O(logn).

Chapter 5

Garbage Collectors

5.1 What is the purpose of garbage collection in Java, and when is it used ?

The purpose of garbage collection is to identify and discard those objects that are no longer needed by the application, in order for the resources to be reclaimed and reused.

5.2 What does System.gc() and Runtime.gc() methods do ?

These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

5.3 When is the finalize() called ? What is the purpose of finalization ?

The finalize method is called by the garbage collector, just before releasing the object's memory. It is normally advised to release resources held by the object inside the finalize method.

5.4 If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object ?

No, the object will be available for garbage collection in the next cycle of the garbage collector.

5.5 What is structure of Java Heap ? What is Perm Gen space in Heap ?

The JVM has a heap that is the runtime data area from which memory for all class instances and arrays is allocated. It is created at the JVM start-up. Heap memory for objects is reclaimed by an automatic memory management system which is known as a garbage collector. Heap memory consists of live and dead objects. Live objects are accessible by the application and will not be a subject of garbage collection. Dead objects are those which will never be accessible by the application, but have not been collected by the garbage collector yet. Such objects occupy the heap memory space until they are eventually collected by the garbage collector.

5.6 What is the difference between Serial and Throughput Garbage collector ?

The throughput garbage collector uses a parallel version of the young generation collector and is meant to be used with applications that have medium to large data sets. On the other hand, the serial collector is usually adequate for most small applications (those requiring heaps of up to approximately 100MB on modern processors).

5.7 When does an Object becomes eligible for Garbage collection in Java ?

A Java object is subject to garbage collection when it becomes unreachable to the program in which it is currently used.

5.8 Does Garbage collection occur in permanent generation space in JVM ?

Garbage Collection does occur in PermGen space and if PermGen space is full or cross a threshold, it can trigger a full garbage collection. If you look carefully at the output of the garbage collector, you will find that PermGen space is also garbage collected. This is the reason why correct sizing of PermGen space is important to avoid frequent full garbage collections. Also check our article Java 8: PermGen to Metaspace.

Chapter 6

Exception Handling

6.1 What are the two types of Exceptions in Java ? Which are the differences between them ?

Java has two types of exceptions: checked exceptions and unchecked exceptions. Unchecked exceptions do not need to be declared in a method or a constructor's throws clause, if they can be thrown by the execution of the method or the constructor, and propagate outside the method or constructor boundary. On the other hand, checked exceptions must be declared in a method or a constructor's throws clause. See here for tips on Java exception handling.

6.2 What is the difference between Exception and Error in java ?

Exception and Error classes are both subclasses of the Throwable class. The Exception class is used for exceptional conditions that a user's program should catch. The Error class defines exceptions that are not excepted to be caught by the user program.

6.3 What is the difference between throw and throws ?

The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

6.4 What is the importance of finally block in exception handling ?

A finally block will always be executed, whether or not an exception is actually thrown. Even in the case where the catch statement is missing and an exception is thrown, the finally block will still be executed. Last thing to mention is that the finally block is used to release resources like I/O buffers, database connections, etc.

6.5 What will happen to the Exception object after exception handling ?

The Exception object will be garbage collected in the next garbage collection.

6.6 How does finally block differ from finalize() method ?

A finally block will be executed whether or not an exception is thrown and is used to release those resources held by the application. Finalize is a protected method of the Object class, which is called by the Java Virtual Machine (JVM) just before an object is garbage collected.

Chapter 7

Java Applets

7.1 What is an Applet ?

A java applet is program that can be included in a HTML page and be executed in a java enabled client browser. Applets are used for creating dynamic and interactive web applications.

7.2 Explain the life cycle of an Applet.

An applet may undergo the following states:

- **Init**: An applet is initialized each time is loaded.
- **Start**: Begin the execution of an applet.
- **Stop**: Stop the execution of an applet.
- **Destroy**: Perform a final cleanup, before unloading the applet.

7.3 What happens when an applet is loaded ?

First of all, an instance of the applet's controlling class is created. Then, the applet initializes itself and finally, it starts running.

7.4 What is the difference between an Applet and a Java Application ?

Applets are executed within a java enabled browser, but a Java application is a standalone Java program that can be executed outside of a browser. However, they both require the existence of a Java Virtual Machine (JVM). Furthermore, a Java application requires a main method with a specific signature, in order to start its execution. Java applets don't need such a method to start their execution. Finally, Java applets typically use a restrictive security policy, while Java applications usually use more relaxed security policies.

7.5 What are the restrictions imposed on Java applets ?

Mostly due to security reasons, the following restrictions are imposed on Java applets:

- An applet cannot load libraries or define native methods.

- An applet cannot ordinarily read or write files on the execution host.
- An applet cannot read certain system properties.
- An applet cannot make network connections except to the host that it came from.
- An applet cannot start any program on the host that's executing it.

7.6 What are untrusted applets ?

Untrusted applets are those Java applets that cannot access or execute local system files. By default, all downloaded applets are considered as untrusted.

7.7 What is the difference between applets loaded over the internet and applets loaded via the file system ?

Regarding the case where an applet is loaded over the internet, the applet is loaded by the applet classloader and is subject to the restrictions enforced by the applet security manager. Regarding the case where an applet is loaded from the client's local disk, the applet is loaded by the file system loader. Applets loaded via the file system are allowed to read files, write files and to load libraries on the client. Also, applets loaded via the file system are allowed to execute processes and finally, applets loaded via the file system are not passed through the byte code verifier.

7.8 What is the applet class loader, and what does it provide ?

When an applet is loaded over the internet, the applet is loaded by the applet classloader. The class loader enforces the Java name space hierarchy. Also, the class loader guarantees that a unique namespace exists for classes that come from the local file system, and that a unique namespace exists for each network source. When a browser loads an applet over the net, that applet's classes are placed in a private namespace associated with the applet's origin. Then, those classes loaded by the class loader are passed through the verifier.The verifier checks that the class file conforms to the Java language specification . Among other things, the verifier ensures that there are no stack overflows or underflows and that the parameters to all bytecode instructions are correct.

7.9 What is the applet security manager, and what does it provide ?

The applet security manager is a mechanism to impose restrictions on Java applets. A browser may only have one security manager. The security manager is established at startup, and it cannot thereafter be replaced, overloaded, overridden, or extended.

Chapter 8

Swing

8.1 What is the difference between a Choice and a List ?

A Choice is displayed in a compact form that must be pulled down, in order for a user to be able to see the list of all available choices. Only one item may be selected from a Choice. A List may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

8.2 What is a layout manager ?

A layout manager is the used to organize the components in a container.

8.3 What is the difference between a Scrollbar and a JScrollPane ?

A Scrollbar is a Component, but not a Container. A ScrollPane is a Container. A ScrollPane handles its own events and performs its own scrolling.

8.4 Which Swing methods are thread-safe ?

There are only three thread-safe methods: repaint, revalidate, and invalidate.

8.5 Name three Component subclasses that support painting.

The Canvas, Frame, Panel, and Applet classes support painting.

8.6 What is clipping ?

Clipping is defined as the process of confining paint operations to a limited area or shape.

8.7 What is the difference between a MenuItem and a CheckboxMenuItem ?

The CheckboxMenuItem class extends the MenuItem class and supports a menu item that may be either checked or unchecked.

8.8 How are the elements of a BorderLayout organized ?

The elements of a BorderLayout are organized at the borders (North, South, East, and West) and the center of a container.

8.9 How are the elements of a GridBagLayout organized ?

The elements of a GridBagLayout are organized according to a grid. The elements are of different sizes and may occupy more than one row or column of the grid. Thus, the rows and columns may have different sizes.

8.10 What is the difference between a Window and a Frame ?

The Frame class extends the Window class and defines a main application window that can have a menu bar.

8.11 What is the relationship between clipping and repainting ?

When a window is repainted by the AWT painting thread, it sets the clipping regions to the area of the window that requires repainting.

8.12 What is the relationship between an event-listener interface and an event-adapter class ?

An event-listener interface defines the methods that must be implemented by an event handler for a particular event. An event adapter provides a default implementation of an event-listener interface.

8.13 How can a GUI component handle its own events ?

A GUI component can handle its own events, by implementing the corresponding event-listener interface and adding itself as its own event listener.

8.14 What advantage do Java's layout managers provide over traditional windowing systems ?

Java uses layout managers to lay out components in a consistent manner, across all windowing platforms. Since layout managers aren't tied to absolute sizing and positioning, they are able to accomodate platform-specific differences among windowing systems.

8.15 What is the design pattern that Java uses for all Swing components ?

The design pattern used by Java for all Swing components is the Model View Controller (MVC) pattern.

Chapter 9

JDBC

9.1 What is JDBC ?

JDBC is an abstraction layer that allows users to choose between databases. JDBC enables developers to write database applications in Java, without having to concern themselves with the underlying details of a particular database.

9.2 Explain the role of Driver in JDBC.

The JDBC Driver provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each driver must provide implementations for the following classes of the java.sql package:Connection, Statement, PreparedStatement, CallableStatement, ResultSet and Driver.

9.3 What is the purpose Class.forName method ?

This method is used to method is used to load the driver that will establish a connection to the database.

9.4 What is the advantage of PreparedStatement over Statement ?

PreparedStatements are precompiled and thus, their performance is much better. Also, PreparedStatement objects can be reused with different input values to their queries.

9.5 What is the use of CallableStatement ? Name the method, which is used to prepare a CallableStatement.

A CallableStatement is used to execute stored procedures. Stored procedures are stored and offered by a database. Stored procedures may take input values from the user and may return a result. The usage of stored procedures is highly encouraged, because it offers security and modularity.The method that prepares a CallableStatement is the following: `CallableStament.prepareCall();`

9.6 What does Connection pooling mean ?

The interaction with a database can be costly, regarding the opening and closing of database connections. Especially, when the number of database clients increases, this cost is very high and a large number of resources is consumed.A pool of database connections is obtained at start up by the application server and is maintained in a pool. A request for a connection is served by a connection residing in the pool. In the end of the connection, the request is returned to the pool and can be used to satisfy future requests.

Chapter 10

Remote Method Invocation (RMI)

10.1 What is RMI ?

The Java Remote Method Invocation (Java RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection. Remote Method Invocation (RMI) can also be seen as the process of activating a method on a remotely running object. RMI offers location transparency because a user feels that a method is executed on a locally running object. Check some RMI Tips here.

10.2 What is the basic principle of RMI architecture ?

The RMI architecture is based on a very important principle which states that the definition of the behavior and the implementation of that behavior, are separate concepts. RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.

10.3 What are the layers of RMI Architecture ?

The RMI architecture consists of the following layers:

- **Stub and Skeleton layer**: This layer lies just beneath the view of the developer. This layer is responsible for intercepting method calls made by the client to the interface and redirect these calls to a remote RMI Service.
- **Remote Reference Layer**: The second layer of the RMI architecture deals with the interpretation of references made from the client to the server's remote objects. This layer interprets and manages references made from clients to the remote service objects. The connection is a one-to-one (unicast) link.
- **Transport layer**: This layer is responsible for connecting the two JVM participating in the service. This layer is based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies.

10.4 What is the role of Remote Interface in RMI ?

The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. Any object that is a remote object must directly or indirectly implement this interface. A class that implements a remote interface should declare the remote interfaces being implemented, define the constructor for each remote object and provide an implementation for each remote method in all remote interfaces.

10.5 What is the role of the java.rmi.Naming Class ?

The java.rmi.Naming class provides methods for storing and obtaining references to remote objects in the remote object registry. Each method of the Naming class takes as one of its arguments a name that is a String in URL format.

10.6 What is meant by binding in RMI ?

Binding is the process of associating or registering a name for a remote object, which can be used at a later time, in order to look up that remote object. A remote object can be associated with a name using the bind or rebind methods of the Naming class.

10.7 What is the difference between using bind() and rebind() methods of Naming Class ?

The bind method bind is responsible for binding the specified name to a remote object, while the rebind method is responsible for rebinding the specified name to a new remote object. In case a binding exists for that name, the binding is replaced.

10.8 What are the steps involved to make work a RMI program ?

The following steps must be involved in order for a RMI program to work properly:

- Compilation of all source files.
- Generatation of the stubs using rmic.
- Start the rmiregistry.
- Start the RMIServer.
- Run the client program.

10.9 What is the role of stub in RMI ?

A stub for a remote object acts as a client's local representative or proxy for the remote object. The caller invokes a method on the local stub, which is responsible for executing the method on the remote object. When a stub's method is invoked, it undergoes the following steps:

- It initiates a connection to the remote JVM containing the remote object.
- It marshals the parameters to the remote JVM.
- It waits for the result of the method invocation and execution.
- It unmarshals the return value or an exception if the method has not been successfully executed.
- It returns the value to the caller.

10.10 What is DGC ? And how does it work ?

DGC stands for Distributed Garbage Collection. Remote Method Invocation (RMI) uses DGC for automatic garbage collection. Since RMI involves remote object references across JVM's, garbage collection can be quite difficult. DGC uses a reference counting algorithm to provide automatic memory management for remote objects.

10.11 What is the purpose of using RMISecurityManager in RMI ?

RMISecurityManager provides a security manager that can be used by RMI applications, which use downloaded code. The class loader of RMI will not download any classes from remote locations, if the security manager has not been set.

10.12 Explain Marshalling and demarshalling.

When an application wants to pass its memory objects across a network to another host or persist it to storage, the in-memory representation must be converted to a suitable format. This process is called marshalling and the revert operation is called demarshalling.

10.13 Explain Serialization and Deserialization.

Java provides a mechanism, called object serialization where an object can be represented as a sequence of bytes and includes the object's data, as well as information about the object's type, and the types of data stored in the object. Thus, serialization can be seen as a way of flattening objects, in order to be stored on disk, and later, read back and reconstituted. Deserialisation is the reverse process of converting an object from its flattened state to a live object.

Chapter 11

Servlets

11.1 What is a Servlet ?

The servlet is a Java programming language class used to process client requests and generate dynamic web content. Servlets are mostly used to process or store data submitted by an HTML form, provide dynamic content and manage state information that does not exist in the stateless HTTP protocol.

11.2 Explain the architechure of a Servlet.

The core abstraction that must be implemented by all servlets is the javax.servlet.Servlet interface. Each servlet must implement it either directly or indirectly, either by extending javax.servlet.GenericServlet or javax.servlet.http.HTTPServlet. Finally, each servlet is able to serve multiple requests in parallel using multithreading.

11.3 What is the difference between an Applet and a Servlet ?

An Applet is a client side java program that runs within a Web browser on the client machine. On the other hand, a servlet is a server side component that runs on the web server.An applet can use the user interface classes, while a servlet does not have a user interface. Instead, a servlet waits for client's HTTP requests and generates a response in every request.

11.4 What is the difference between GenericServlet and HttpServlet ?

GenericServlet is a generalized and protocol-independent servlet that implements the Servlet and ServletConfig interfaces. Those servlets extending the GenericServlet class shall override the service method. Finally, in order to develop an HTTP servlet for use on the Web that serves requests using the HTTP protocol, your servlet must extend the HttpServlet instead. Check Servlet examples here.

11.5 Explain the life cycle of a Servlet.

On every client's request, the Servlet Engine loads the servlets and invokes its init methods, in order for the servlet to be initialized. Then, the Servlet object handles all subsequent requests coming from that client, by invoking the service method for each request separately. Finally, the servlet is removed by calling the server's destroy method.

11.6 What is the difference between doGet() and doPost() ?

`doGET`: The GET method appends the name-value pairs on the request's URL. Thus, there is a limit on the number of characters and subsequently on the number of values that can be used in a client's request. Furthermore, the values of the request are made visible and thus, sensitive information must not be passed in that way.

`doPOST`: The POST method overcomes the limit imposed by the GET request, by sending the values of the request inside its body. Also, there is no limitations on the number of values to be sent across. Finally, the sensitive information passed through a POST request is not visible to an external client.

11.7 What is meant by a Web Application ?

A Web application is a dynamic extension of a Web or application server. There are two types of web applications: presentation-oriented and service-oriented. A presentation-oriented Web application generates interactive web pages, which contain various types of markup language and dynamic content in response to requests. On the other hand, a service-oriented web application implements the endpoint of a web service. In general, a Web application can be seen as a collection of servlets installed under a specific subset of the server's URL namespace.

11.8 What is a Server Side Include (SSI) ?

Server Side Includes (SSI) is a simple interpreted server-side scripting language, used almost exclusively for the Web, and is embedded with a servlet tag. The most frequent use of SSI is to include the contents of one or more files into a Web page on a Web server. When a Web page is accessed by a browser, the Web server replaces the servlet tag in that Web page with the hyper text generated by the corresponding servlet.

11.9 What is Servlet Chaining ?

Servlet Chaining is the method where the output of one servlet is sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. The last servlet in the chain is responsible for sending the response to the client.

11.10 How do you find out what client machine is making a request to your servlet ?

The ServletRequest class has functions for finding out the IP address or host name of the client machine. getRemoteAddr() gets the IP address of the client machine and getRemoteHost() gets the host name of the client machine. See example here.

11.11 What is the structure of the HTTP response ?

The HTTP response consists of three parts:

- **Status Code**: describes the status of the response. It can be used to check if the request has been successfully completed. In case the request failed, the status code can be used to find out the reason behind the failure. If your servlet does not return a status code, the success status code, HttpServletResponse.SC_OK, is returned by default.
- **HTTP Headers**: they contain more information about the response. For example, the headers may specify the date/time after which the response is considered stale, or the form of encoding used to safely transfer the entity to the user. See how to retrieve headers in Servlet here.
- **Body**: it contains the content of the response. The body may contain HTML code, an image, etc. The body consists of the data bytes transmitted in an HTTP transaction message immediately following the headers.

11.12 What is a cookie ? What is the difference between session and cookie ?

A cookie is a bit of information that the Web server sends to the browser. The browser stores the cookies for each Web server in a local file. In a future request, the browser, along with the request, sends all stored cookies for that specific Web server.The differences between session and a cookie are the following:

- The session should work, regardless of the settings on the client browser. The client may have chosen to disable cookies. However, the sessions still work, as the client has no ability to disable them in the server side.
- The session and cookies also differ in the amount of information the can store. The HTTP session is capable of storing any Java object, while a cookie can only store String objects.

11.13 Which protocol will be used by browser and servlet to communicate ?

The browser communicates with a servlet by using the HTTP protocol.

11.14 What is HTTP Tunneling ?

HTTP Tunneling is a technique by which, communications performed using various network protocols are encapsulated using the HTTP or HTTPS protocols. The HTTP protocol therefore acts as a wrapper for a channel that the network protocol being tunneled uses to communicate. The masking of other protocol requests as HTTP requests is HTTP Tunneling.

11.15 What's the difference between sendRedirect and forward methods ?

The sendRedirect method creates a new request, while the forward method just forwards a request to a new target. The previous request scope objects are not available after a redirect, because it results in a new request. On the other hand, the previous request scope objects are available after forwarding. FInally, in general, the sendRedirect method is considered to be slower compare to the forward method.

11.16 What is URL Encoding and URL Decoding ?

The URL encoding procedure is responsible for replacing all the spaces and every other extra special character of a URL, into their corresponding Hex representation. In correspondence, URL decoding is the exact opposite procedure.

Chapter 12

JSP

12.1 What is a JSP Page ?

A Java Server Page (JSP) is a text document that contains two types of text: static data and JSP elements. Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically-generated content. See JSP example here.

12.2 How are the JSP requests handled ?

On the arrival of a JSP request, the browser first requests a page with a .jsp extension. Then, the Web server reads the request and using the JSP compiler, the Web server converts the JSP page into a servlet class. Notice that the JSP file is compiled only on the first request of the page, or if the JSP file has changed.The generated servlet class is invoked, in order to handle the browser's request. Once the execution of the request is over, the servlet sends a response back to the client. See how to get Request parameters in a JSP.

12.3 What are the advantages of JSP ?

The advantages of using the JSP technology are shown below:

- JSP pages are dynamically compiled into servlets and thus, the developers can easily make updates to presentation code.
- JSP pages can be pre-compiled.
- JSP pages can be easily combined to static templates, including HTML or XML fragments, with code that generates dynamic content.
- Developers can offer customized JSP tag libraries that page authors access using an XML-like syntax.
- Developers can make logic changes at the component level, without editing the individual pages that use the application's logic.

12.4 What are Directives ? What are the different types of Directives available in JSP ?

Directives are instructions that are processed by the JSP engine, when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries. Directives are defined between `< %@ and % >`. The different types of directives are shown below:

- `Include directive`: it is used to include a file and merges the content of the file with the current page.
- `Page directive`: it is used to define specific attributes in the JSP page, like error page and buffer.
- `Taglib`: it is used to declare a custom tag library which is used in the page.

12.5 What are JSP actions ?

JSP actions use constructs in XML syntax to control the behavior of the servlet engine. JSP actions are executed when a JSP page is requested. They can be dynamically inserted into a file, re-use JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.Some of the available actions are listed below:

- `jsp:include` - includes a file, when the JSP page is requested.
- `jsp:useBean` - finds or instantiates a JavaBean.
- `jsp:setProperty` - sets the property of a JavaBean.
- `jsp:getProperty` - gets the property of a JavaBean.
- `jsp:forward` - forwards the requester to a new page.
- `jsp:plugin` - generates browser-specific code.

12.6 What are Scriptlets ?

In Java Server Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in a JSP page. The scriptlet is everything inside the tags. Between these tags, a user can add any valid scriplet.

12.7 What are Decalarations ?

Declarations are similar to variable declarations in Java. Declarations are used to declare variables for subsequent use in expressions or scriptlets. To add a declaration, you must use the sequences to enclose your declarations.

12.8 What are Expressions ?

A JSP expression is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client, by the web server. Expressions are defined between `<% =and %>` tags.

12.9 What is meant by implicit objects and what are they ?

JSP implicit objects are those Java objects that the JSP Container makes available to developers in each page. A developer can call them directly, without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.The following objects are considered implicit in a JSP page:

- `application`
- `page`
- `request`
- `response`

- `session`
- `exception`
- `out`
- `config`
- `pageContext`

Java Programming Interview Questions and Answers for freshers

1) What is the difference between an Inner Class and a Sub-Class?

An Inner class is a class which is nested within another class. An Inner class has access rights for the class which is nesting it and it can access all variables and methods defined in the outer class.

A sub-class is a class which inherits from another class called super class. Sub-class can access all public and protected methods and fields of its super class.

2) What are the various access specifiers for Java classes?

In Java, access specifiers are the keywords used before a class name which defines the access scope. The types of access specifiers for classes are:

1) Public: Class,Method,Field is accessible from anywhere.

2) Protected: Method,Field can be accessed from the same class to which they belong or from the sub-classes, and from the class of same package, but not from outside.

3) Default: Method,Field,class can be accessed only from the same package and not from outside of it's native package.

4) Private: Method,Field can be accessed from the same class to which they bel

3) What's the purpose of Static methods and static variables?

When there is a requirement to share a method or a variable between multiple objects of a class instead of creating separate copies for each object, we use static keyword to make a method or variable shared for all objects.

4) What is data encapsulation and what's its significance?

Encapsulation is a concept in Object Oriented Programming for combining properties and methods in a single unit.

Encapsulation helps programmers to follow a modular approach for software development as each object has its own set of methods and variables and serves its functions independent of other objects. Encapsulation also serves data hiding purpose.

5) What is a singleton class? Give a practical example of its usage.

A singleton class in java can have only one instance and hence all its methods and variables belong to just one instance. Singleton class concept is useful for the situations when there is a need to limit the number of objects for a class.

The best example of singleton usage scenario is when there is a limit of having only one connection to a database due to some driver limitations or because of any licensing issues.

6) What are Loops in Java? What are three types of loops?

Looping is used in programming to execute a statement or a block of statement repeatedly. There are three types of loops in Java:

1) For Loops

For loops are used in java to execute statements repeatedly for a given number of times. For loops are used when number of times to execute the statements is known to programmer.

2) While Loops

While loop is used when certain statements need to be executed repeatedly until a condition is fulfilled. In while loops, condition is checked first before execution of statements.

3) Do While Loops

Do While Loop is same as While loop with only difference that condition is checked after execution of block of statements. Hence in case of do while loop, statements are executed at least once.

7) What is an infinite Loop? How infinite loop is declared?

An infinite loop runs without any condition and runs infinitely. An infinite loop can be broken by defining any breaking logic in the body of the statement blocks.

Infinite loop is declared as follows:

```
for (;;)
{
    // Statements to execute

    // Add any loop breaking logic
}
```

8) What is the difference between continue and break statement?

break and continue are two important keywords used in Loops. When a break keyword is used in a loop, loop is broken instantly while when continue keyword is used, current iteration is broken and loop continues with next iteration.

In below example, Loop is broken when counter reaches 4.

```
for (counter = 0; counter & lt; 10; counter++)
    system.out.println(counter);

if (counter == 4) {

    break;
}

}
```

In the below example when counter reaches 4, loop jumps to next iteration and any statements after the continue keyword are skipped for current iteration.

```
for (counter = 0; counter < 10; counter++)
    system.out.println(counter);

if (counter == 4) {
```

```
    continue;
}
system.out.println("This will not get printed when counter is 4");
}
```

9) What is the difference between double and float variables in Java?

In java, float takes 4 bytes in memory while Double takes 8 bytes in memory. Float is single precision floating point decimal number while Double is double precision decimal number.

10) What is Final Keyword in Java? Give an example.

In java, a constant is declared using the keyword Final. Value can be assigned only once and after assignment, value of a constant can't be changed.

In below example, a constant with the name const_val is declared and assigned avalue:

Private Final int const_val=100

When a method is declared as final, it can NOT be overridden by the subclasses. This method are faster than any other method, because they are resolved at complied time.

When a class is declares as final, it cannot be subclassed. Example String, Integer and other wrapper classes.

11) What is ternary operator? Give an example.

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

In the below example, if rank is 1, status is assigned a value of "Done" else "Pending".

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

12) How can you generate random numbers in Java?

- Using Math.random() you can generate random numbers in the range greater than or equal to 0.1 and less than 1.0
- Using Random class in package java.util

13) What is default switch case? Give example.

In a switch statement, default case is executed when no other switch condition matches. Default case is an optional case .It can be declared only once all other switch cases have been coded.

In the below example, when score is not 1 or 2, default case is used.

```
public class switchExample {
    int score = 4;
    public static void main(String args[]) {
        switch (score) {
            case 1:
                system.out.println("Score is 1");
                break;
            case 2:
                system.out.println("Score is 2");
                break;
            default:
                system.out.println("Default Case");
        }
    }
}
```

14) What's the base class in Java from which all classes are derived?

`java.lang.object`

15) Can main() method in Java can return any data?

In java, main() method can't return any data and hence, it's always declared with a void return type.

16) What are Java Packages? What's the significance of packages?

In Java, package is a collection of classes and interfaces which are bundled together as they are related to each other. Use of packages helps developers to modularize the code and group the code for proper re-use. Once code has been packaged in Packages, it can be imported in other classes and used.

17) Can we declare a class as Abstract without having any abstract method?

Yes we can create an abstract class by using abstract keyword before class name even if it doesn't have any abstract method. However, if a class has even one abstract method, it must be declared as abstract otherwise it will give an error.

18) What's the difference between an Abstract Class and Interface in Java?

The primary difference between an abstract class and interface is that an interface can only possess declaration of public static methods with no concrete implementation while an abstract class can have members with any access specifiers (public, private etc) with or without concrete implementation.

Another key difference in the use of abstract classes and interfaces is that a class which implements an interface must implement all the methods of the interface while a class which inherits from an abstract class doesn't require implementation of all the methods of its super class.

A class can implement multiple interfaces but it can extend only one abstract class.

19) What are the performance implications of Interfaces over abstract classes?

Interfaces are slower in performance as compared to abstract classes as extra indirections are required for interfaces. Another key factor for developers to take into consideration is that any class can extend only one abstract class while a class can implement many interfaces.

Use of interfaces also puts an extra burden on the developers as any time an interface is implemented in a class; developer is forced to implement each and every method of interface.

20) Does Importing a package imports its sub-packages as well in Java?

In java, when a package is imported, its sub-packages aren't imported and developer needs to import them separately if required.

For example, if a developer imports a package university.*, all classes in the package named university are loaded but no classes from the sub-package are loaded. To load the classes from its sub-package (say department), developer has to import it explicitly as follows:

Import university.department.*

21) Can we declare the main method of our class as private?

In java, main method must be public static in order to run any application correctly. If main method is declared as private, developer won't get any compilation error however, it will not get executed and will give a runtime error.

22) How can we pass argument to a function by reference instead of pass by value?

In java, we can pass argument to a function only by value and not by reference.

23) How an object is serialized in java?

In java, to convert an object into byte stream by serialization, an interface with the name Serializable is implemented by the class. All objects of a class implementing serializable interface get serialized and their state is saved in byte stream.

24) When we should use serialization?

Serialization is used when data needs to be transmitted over the network. Using serialization, object's state is saved and converted into byte stream .The byte stream is transferred over the network and the object is re-created at destination.

25) Is it compulsory for a Try Block to be followed by a Catch Block in Java for Exception handling?

Try block needs to be followed by either Catch block or Finally block or both. Any exception thrown from try block needs to be either caught in the catch block or else any specific tasks to be performed before code abortion are put in the Finally block.

Java Interview Questions and Answers for Experienced

26) Is there any way to skip Finally block of exception even if some exception occurs in the exception block?

If an exception is raised in Try block, control passes to catch block if it exists otherwise to finally block. Finally block is always executed when an exception occurs and the only way to avoid execution of any

statements in Finally block is by aborting the code forcibly by writing following line of code at the end of try block:

```
System.exit(0);
```

27) When the constructor of a class is invoked?

The constructor of a class is invoked every time an object is created with new keyword.

For example, in the following class two objects are created using new keyword and hence, constructor is invoked two times.

```
public class const_example {

    const_example() {

        system.out.println("Inside constructor");
    }
    public static void main(String args[]) {

        const_example c1 = new const_example();

        const_example c2 = new const_example();
    }
}
```

28) Can a class have multiple constructors?

Yes, a class can have multiple constructors with different parameters. Which constructor gets used for object creation depends on the arguments passed while creating the objects.

29) Can we override static methods of a class?

We cannot override static methods. Static methods belong to a class and not to individual objects and are resolved at the time of compilation (not at runtime).Even if we try to override static method, we will not get an complitaion error,nor the impact of overriding when running the code.

30) In the below example, what will be the output?

```
public class superclass {

    public void displayResult() {

        system.out.println("Printing from superclass");

    }

}

public class subclass extends superclass {

    public void displayResult() {

        system.out.println("Displaying from subClass");

        super.displayResult();
```

```
    }

    public static void main(String args[]) {

        subclass obj = new subclass();

        obj.displayResult();

    }

}
```

Ans: Output will be:

Displaying from subClass

Printing from superclass

31) Is String a data type in java?

String is not a primitive data type in java. When a string is created in java, it's actually an object of Java.Lang.String class that gets created. After creation of this string object, all built-in methods of String class can be used on the string object.

32) In the below example, how many String Objects are created?

```
String s1="I am Java Expert";

String s2="I am C Expert";

String s3="I am Java Expert";
```

In the above example, two objects of Java.Lang.String class are created. s1 and s3 are references to same object.

33) Why Strings in Java are called as Immutable?

In java, string objects are called immutable as once value has been assigned to a string, it can't be changed and if changed, a new object is created.

In below example, reference str refers to a string object having value "Value one".

```
String str="Value One";
```

When a new value is assigned to it, a new String object gets created and the reference is moved to the new object.

```
str="New Value";
```

34) What's the difference between an array and Vector?

An array groups data of same primitive type and is static in nature while vectors are dynamic in nature and can hold data of different data types.

35) What is multi-threading?

Multi threading is a programming concept to run multiple tasks in a concurrent manner within a single program. Threads share same process stack and running in parallel. It helps in performance improvement of any program.

36) Why Runnable Interface is used in Java?

Runnable interface is used in java for implementing multi threaded applications. Java.Lang.Runnable interface is implemented by a class to support multi threading.

37) What are the two ways of implementing multi-threading in Java?

Multi threaded applications can be developed in Java by using any of the following two methodologies:

1) By using Java.Lang.Runnable Interface. Classes implement this interface to enable multi threading. There is a Run() method in this interface which is implemented.

2) By writing a class that extend Java.Lang.Thread class.

38) When a lot of changes are required in data, which one should be a preference to be used? String or StringBuffer?

Since StringBuffers are dynamic in nature and we can change the values of StringBuffer objects unlike String which is immutable, it's always a good choice to use StringBuffer when data is being changed too much. If we use String in such a case, for every data change a new String object will be created which will be an extra overhead.

39) What's the purpose of using Break in each case of Switch Statement?

Break is used after each case (except the last one) in a switch so that code breaks after the valid case and doesn't flow in the proceeding cases too.

If break isn't used after each case, all cases after the valid case also get executed resulting in wrong results.

40) How garbage collection is done in Java?

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

41) How we can execute any code even before main method?

If we want to execute any statements before even creation of objects at load time of class, we can use a static block of code in the class. Any statements inside this static block of code will get executed once at the time of loading the class even before creation of objects in the main method.

42) Can a class be a super class and a sub-class at the same time? Give example.

If there is a hierarchy of inheritance used, a class can be a super class for another class and a sub-class for another one at the same time.

In the example below, continent class is sub-class of world class and it's super class of country class.

```
public class world {

..........

}
public class continenet extends world {

............

}
public class country extends continent {

.....................

}
```

43) How objects of a class are created if no constructor is defined in the class?

Even if no explicit constructor is defined in a java class, objects get created successfully as a default constructor is implicitly used for object creation. This constructor has no parameters.

44) In multi-threading how can we ensure that a resource isn't used by multiple threads simultaneously?

In multi-threading, access to the resources which are shared among multiple threads can be controlled by using the concept of synchronization. Using synchronized keyword, we can ensure that only one thread can use shared resource at a time and others can get control of the resource only once it has become free from the other one using it.

45) Can we call the constructor of a class more than once for an object?

Constructor is called automatically when we create an object using new keyword. It's called only once for an object at the time of object creation and hence, we can't invoke the constructor again for an object after its creation.

46) There are two classes named classA and classB. Both classes are in the same package. Can a private member of classA can be accessed by an object of classB?

Private members of a class aren't accessible outside the scope of that class and any other class even in the same package can't access them.

47) Can we have two methods in a class with the same name?

We can define two methods in a class with the same name but with different number/type of parameters. Which method is to get invoked will depend upon the parameters passed.

For example in the class below we have two print methods with same name but different parameters. Depending upon the parameters, appropriate one will be called:

```
public class methodExample {

    public void print() {
```

```
        system.out.println("Print method without parameters.");
    }
    public void print(String name) {
        system.out.println("Print method with parameter");
    }
    public static void main(String args[]) {
        methodExample obj1 = new methodExample();
        obj1.print();
        obj1.print("xx");
    }
}
```

48) How can we make copy of a java object?

We can use the concept of cloning to create copy of an object. Using clone, we create copies with the actual state of an object.

Clone() is a method of Cloneable interface and hence, Cloneable interface needs to be implemented for making object copies.

49) What's the benefit of using inheritance?

Key benefit of using inheritance is reusability of code as inheritance enables sub-classes to reuse the code of its super class. Polymorphism (Extensibility) is another great benefit which allow new functionality to be introduced without effecting existing derived classes.

50) What's the default access specifier for variables and methods of a class?

Default access specifier for variables and method is package protected i.e variables and class is available to any other class but in the same package, not outside the package.

51) Give an example of use of Pointers in Java class.

There are no pointers in Java. So we can't use concept of pointers in Java.

52) How can we restrict inheritance for a class so that no class can be inherited from it?

If we want a class not to be extended further by any class, we can use the keyword **Final** with the class name.

In the following example, Stone class is Final and can't be extend

```
public Final Class Stone {
        // Class methods and Variables
```

```
}
```

53) What's the access scope of Protected Access specifier?

When a method or a variable is declared with Protected access specifier, it becomes accessible in the same class, any other class of the same package as well as a sub-class.

Modifier	Class	Package	Subclass
public	Y	Y	Y
protected	Y	Y	Y
no modifier	Y	Y	N
private	Y	N	N

54) What's difference between Stack and Queue?

Stack and Queue both are used as placeholder for a collection of data. The primary difference between a stack and a queue is that stack is based on Last in First out (LIFO) principle while a queue is based on FIFO (First In First Out) principle.

55) In java, how we can disallow serialization of variables?

If we want certain variables of a class not to be serialized, we can use the keyword **transient** while declaring them. For example, the variable trans_var below is a transient variable and can't be serialized:

```
public class transientExample {
        private transient trans_var;
    // rest of the code
}
```

56) How 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.

57) Which types of exceptions are caught at compile time?

Checked exceptions can be caught at the time of program compilation. Checked exceptions must be handled by using try catch block in the code in order to successfully compile the code.

58) Describe different states of a thread.

A thread in Java can be in either of the following states:

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

59) Can we use a default constructor of a class even if an explicit constructor is defined?

Java provides a default no argument constructor if no explicit constructor is defined in a Java class. But if an explicit constructor has been defined, default constructor can't be invoked and developer can use only those constructors which are defined in the class.

60) Can we override a method by using same method name and arguments but different return types?

The basic condition of method overriding is that method name, arguments as well as return type must be exactly same as is that of the method being overridden. Hence using a different return type doesn't override a method.

61) What will be the output of following piece of code?

```
public class operatorExample {

    public static void main(String args[]) {

        int x = 4;

        system.out.println(x++);
        }
}
```

In this case postfix ++ operator is used which first returns the value and then increments. Hence it's output will be 4.

61) A person says that he compiled a java class successfully without even having a main method in it? Is it possible?

main method is an entry point of Java class and is required for execution of the program however; a class gets compiled successfully even if it doesn't have a main method. It can't be run though.

62) Can we call a non-static method from inside a static method?

Non-Static methods are owned by objects of a class and have object level scope and in order to call the non-Static methods from a static block (like from a static main method), an object of the class needs to be created first. Then using object reference, these methods can be invoked.

63) What are the two environment variables that must be set in order to run any Java programs?

Java programs can be executed in a machine only once following two environment variables have been properly set:

1. PATH variable
2. CLASSPATH variable

64) Can variables be used in Java without initialization?

In Java, if a variable is used in a code without prior initialization by a valid value, program doesn't compile and gives an error as no default value is assigned to variables in Java.

65) Can a class in Java be inherited from more than one class?

In Java, a class can be derived from only one class and not from multiple classes. Multiple inheritances is not supported by Java.

66) Can a constructor have different name than a Class name in Java?

Constructor in Java must have same name as the class name and if the name is different, it doesn't act as a constructor and compiler thinks of it as a normal method.

67) What will be the output of Round(3.7) and Ceil(3.7)?

Round(3.7) returns 4 and Ceil(3.7) returns 4.

68) Can we use goto in Java to go to a particular line?

In Java, there is not goto keyword and java doesn't support this feature of going to a particular labeled line.

69) Can a dead thread be started again?

In java, a thread which is in dead state can't be started again. There is no way to restart a dead thread.

70) Is the following class declaration correct?

```
public abstract final class testClass {
        // Class methods and variables
}
```

Ans: The above class declaration is incorrect as an abstract class can't be declared as Final.

71) Is JDK required on each machine to run a Java program?

JDK is development Kit of Java and is required for development only and to run a Java program on a machine, JDK isn't required. Only JRE is required.

72) What's the difference between comparison done by equals method and == operator?

In Java, equals() method is used to compare the contents of two string objects and returns true if the two have same value while == operator compares the references of two string objects.

In the following example, equals() returns true as the two string objects have same values. However == operator returns false as both string objects are referencing to different objects:

```
public class equalsTest {

    public static void main(String args[]) {

        String str1 = new String("Hello World");

        String str2 = new String("Hello World");
```

```
        if (str1.equals(str2))

        { // this condition is true

            System.out.println("str1 and str2 are equal in terms of values");

        }

        if (str1 == str2) {

            //This condition is true

            System.out.println("Both strings are referencing same object");

        } else

        {

            // This condition is NOT true

            System.out.println("Both strings are referencing different
objects");

        }

    }

}
```

73) Is it possible to define a method in Java class but provide it's implementation in the code of another language like C?

Yes, we can do this by use of native methods. In case of native method based development, we define public static methods in our Java class without its implementation and then implementation is done in another language like C separately.

74) How are destructors defined in Java?

In Java, there are no destructors defined in the class as there is no need to do so. Java has its own garbage collection mechanism which does the job automatically by destroying the objects when no longer referenced.

Java Interview Questions and Answers for 5+ Years Experience

75) Can a variable be local and static at the same time?

No a variable can't be static as well as local at the same time. Defining a local variable as static gives compilation error.

76) Can we have static methods in an Interface?

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

77) In a class implementing an interface, can we change the value of any variable defined in the interface?

No, we can't change the value of any variable of an interface in the implementing class as all variables defined in the interface are by default public, static and Final and final variables are like constants which can't be changed later.

78) Is it correct to say that due to garbage collection feature in Java, a java program never goes out of memory?

Even though automatic garbage collection is provided by Java, it doesn't ensure that a Java program will not go out of memory as there is a possibility that creation of Java objects is being done at a faster pace compared to garbage collection resulting in filling of all the available memory resources.

So, garbage collection helps in reducing the chances of a program going out of memory but it doesn't ensure that.

79) Can we have any other return type than void for main method?

No, Java class main method can have only void return type for the program to get successfully executed.

Nonetheless , if you absolutely must return a value to at the completion of main method , you can use System.exit(int status)

80) I want to re-reach and use an object once it has been garbage collected. How it's possible?

Once an object has been destroyed by garbage collector, it no longer exists on the heap and it can't be accessed again. There is no way to reference it again.

81) In Java thread programming, which method is a must implementation for all threads?

Run() is a method of Runnable interface that must be implemented by all threads.

82) I want to control database connections in my program and want that only one thread should be able to make database connection at a time. How can I implement this logic?

Ans: This can be implemented by use of the concept of synchronization. Database related code can be placed in a method which hs **synchronized** keyword so that only one thread can access it at a time.

83) How can an exception be thrown manually by a programmer?

In order to throw an exception in a block of code manually, **throw** keyword is used. Then this exception is caught and handled in the catch block.

```
public void topMethod() {
    try {
        excMethod();
    } catch (ManualException e) {}
}
```

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

84) I want my class to be developed in such a way that no other class (even derived class) can create its objects. How can I do so?

If we declare the constructor of a class as private, it will not be accessible by any other class and hence, no other class will be able to instantiate it and formation of its object will be limited to itself only.

85) How objects are stored in Java?

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

86) How can we find the actual size of an object on the heap?

In java, there is no way to find out the exact size of an object on the heap.

87) Which of the following classes will have more memory allocated?

Class A: Three methods, four variables, no object

Class B: Five methods, three variables, no object

Memory isn't allocated before creation of objects. Since for both classes, there are no objects created so no memory is allocated on heap for any class.

88) What happens if an exception is not handled in a program?

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

89) I have multiple constructors defined in a class. Is it possible to call a constructor from another constructor's body?

If a class has multiple constructors, it's possible to call one constructor from the body of another one using **this()**.

90) What's meant by anonymous class?

An anonymous class is a class defined without any name in a single line of code using new keyword.

For example, in below code we have defined an anonymous class in one line of code:

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

91) Is there a way to increase the size of an array after its declaration?

Arrays are static and once we have specified its size, we can't change it. If we want to use such collections where we may require a change of size (no of items), we should prefer vector over array.

92) If an application has multiple classes in it, is it okay to have a main method in more than one class?

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

93) I want to persist data of objects for later use. What's the best approach to do so?

The best way to persist data for future use is to use the concept of serialization.

94) What is a Local class in Java?

In Java, if we define a new class inside a particular block, it's called a local class. Such a class has local scope and isn't usable outside the block where its defined.

95) String and StringBuffer both represent String objects. Can we compare String and StringBuffer in Java?

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

96) Which API is provided by Java for operations on set of objects?

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

97) Can we cast any other type to Boolean Type with type casting?

No, we can neither cast any other primitive type to Boolean data type nor can cast Boolean data type to any other primitive data type.

98) Can we use different return types for methods when overridden?

The basic requirement of method overriding in Java is that the overridden method should have same name, and parameters.But a method can be overridden with a different return type as long as the new return type extends the original.

For example , method is returning a reference type.

```
Class B extends A {

    A method(int x) {

        //original method

    }

    B method(int x) {

        //overridden method

    }

}
```

99) What's the base class of all exception classes?

In Java, `Java.lang.Throwable` is the super class of all exception classes and all exception classes are derived from this base class.

100) What's the order of call of constructors in inheritance?

In case of inheritance, when a new object of a derived class is created, first the constructor of the super class is invoked and then the constructor of the derived class is invoked.

What is a classloader?

Classloader is the part of JRE(Java Runtime Environment), during the execution of the bytecode or created .class file classloader is responsible for dynamically loading

the java classes and interfaces to JVM(Java Virtual Machine). Because of classloaders Java run time system does not need to know about files and file systems.
To know more about the topic refer to ClassLoader in Java.

7. Difference between JVM, JRE, and JDK.

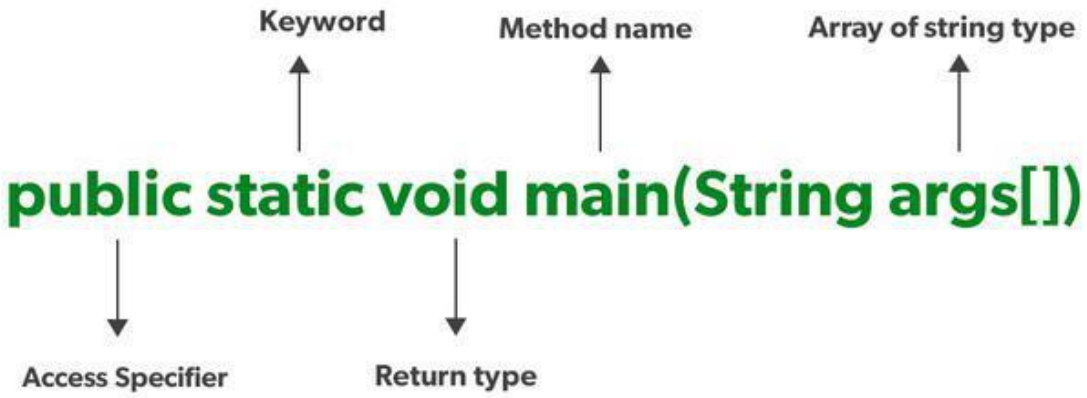
JVM: JVM also known as Java Virtual Machine is a part of JRE. JVM is a type of interpreter responsible for converting bytecode into machine-readable code. JVM itself is platform dependent but it interprets the bytecode which is the platform-independent reason why Java is platform-independent.
JRE: JRE stands for Java Runtime Environment, it is an installation package that provides an environment to run the Java program or application on any machine.
JDK: JDK stands for Java Development Kit which provides the environment to develop and execute Java programs. JDK is a package that includes two things Development Tools to provide an environment to develop your Java programs and, JRE to execute Java programs or applications.
To know more about the topic refer to the Differences between JVM, JRE, and JDK.

8. What are the differences between Java and C++?

Basis	**C++**	**Java**
Platform	C++ is Platform Dependent	Java is Platform Independent
Application	C++ is mainly used for System Programming	Java is Mainly used for Application Programming
Hardware	C++ is nearer to hardware	Java is not so interactive with hardware
Global Scope	C++ supports global and namespace scope.	Java doesn't support global scope.
Not Supporting	Functionality supported in Java but not in C++ are: • thread support • documentation comment • unsigned right shift(>>>)	Functionality supported in C++ but not in Java are: • goto • Pointers • Call by reference • Structures and Unions • Multiple Inheritance

Basis	C++	Java
OOPS	C++ is an object-oriented language. It is not a single root hierarchy .	Java is also an object-oriented language. It is a single root hierarchy as everything gets derived from a single class (java.lang.Object).
Inheritance Tree	C++ always creates a new inheritance tree.	Java uses a Single inheritance tree as classes in Java are the child of object classes in Java.
	• Virtual Functions	

9. Explain public static void main(String args[]) in Java.



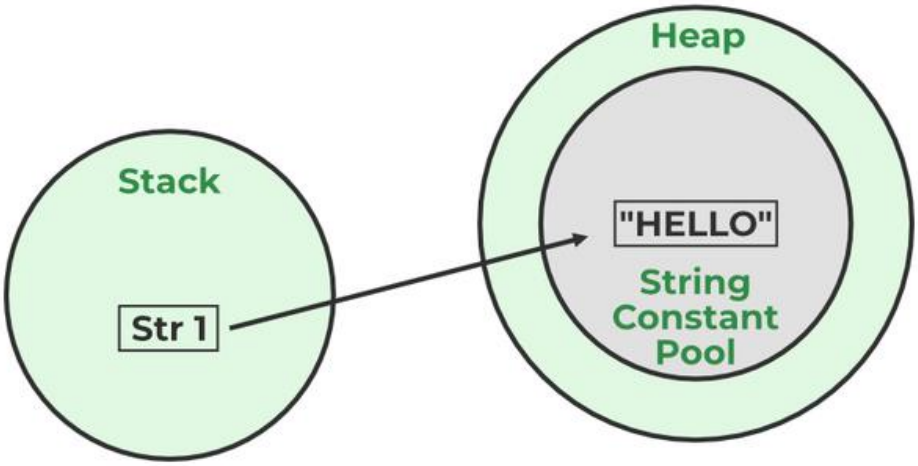
Unlike any other programming language like C, C++, etc. In Java, we declared the main function as a public static void main (String args[]). The meanings of the terms are mentioned below:

1. **public**: the public is the access modifier responsible for mentioning who can access the element or the method and what is the limit. It is responsible for making the main function globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class.
2. **static**: static is a keyword used so that we can use the element without initiating the class so to avoid the unnecessary allocation of the memory.

3. **void**: void is a keyword and is used to specify that a method doesn't return anything. As the main function doesn't return anything we use void.
4. **main**: main represents that the function declared is the main function. It helps JVM to identify that the declared function is the main function.
5. **String args[]**: It stores Java command-line arguments and is an array of type java.lang.String class.

10. What is Java String Pool?

A Java String Pool is a place in heap memory where all the strings defined in the program are stored. A separate place in a stack is there where the variable storing the string is stored. Whenever we create a new string object, JVM checks for the presence of the object in the String pool, If String is available in the pool, the same object reference is shared with the variable, else a new object is created.



Example:

```
String str1="Hello";
// "Hello" will be stored in String Pool
// str1 will be stored in stack memory
```

11. What will happen if we declare don't declare the main as static?

We can declare the main method without using static and without getting any errors. But, the main method will not be treated as the entry point to the application or the program.

12. What are Packages in Java?

Packages in Java can be defined as the grouping of related types of classes, interfaces, etc providing access to protection and namespace management.

13. Why Packages are used?

Packages are used in Java in order to prevent naming conflicts, control access, and make searching/locating and usage of classes, interfaces, etc easier.

14. What are the advantages of Packages in Java?

There are various advantages of defining packages in Java.

- Packages avoid name clashes.
- The Package provides easier access control.
- We can also have the hidden classes that are not visible outside and are used by the package.
- It is easier to locate the related classes.

15. How many types of packages are there in Java?

There are two types of packages in Java

- User-defined packages
- Build In packages

16. Explain different data types in Java.

There are 2 types of data types in Java as mentioned below:

1. Primitive Data Type
2. Non-Primitive Data Type or Object Data type

Primitive Data Type: Primitive data are single values with no special capabilities. There are 8 primitive data types:

- **boolean**: stores value true or false
- **byte**: stores an 8-bit signed two's complement integer
- **char**: stores a single 16-bit Unicode character
- **short**: stores a 16-bit signed two's complement integer
- **int**: stores a 32-bit signed two's complement integer
- **long**: stores a 64-bit two's complement integer
- **float**: stores a single-precision 32-bit IEEE 754 floating-point
- **double**: stores a double-precision 64-bit IEEE 754 floating-point

Non-Primitive Data Type: Reference Data types will contain a memory address of the variable's values because it is not able to directly store the values in the memory. Types of Non-Primitive are mentioned below:

- Strings
- Array
- Class
- Object
- Interface

17. When a byte datatype is used?

A byte is an 8-bit signed two-complement integer. The minimum value supported by bytes is -128 and 127 is the maximum value. It is used in conditions where we need to save memory and the limit of numbers needed is between -128 to 127.

18. Can we declare Pointer in Java?

No, Java doesn't provide the support of Pointer. As Java needed to be more secure because which feature of the pointer is not provided in Java.

19. What is the default value of byte datatype in Java?

The default value of the byte datatype in Java is 0.

20. What is the default value of float and double datatype in Java?

The default value of the float is 0.0f and of double is 0.0d in Java.

21. What is the Wrapper class in Java?

Wrapper, in general, is referred to a larger entity that encapsulates a smaller entity. Here in Java, the wrapper class is an object class that encapsulates the primitive data types.

The primitive data types are the ones from which further data types could be created. For example, integers can further lead to the construction of long, byte, short, etc. On the other hand, the string cannot, hence it is not primitive.

Getting back to the wrapper class, Java contains 8 wrapper classes. They are Boolean, Byte, Short, Integer, Character, Long, Float, and Double. Further, custom wrapper classes can also be created in Java which is similar to the concept of Structure in the C programming language. We create our own wrapper class with the required data types.

22. Why do we need wrapper classes?

The wrapper class is an object class that encapsulates the primitive data types, and we need them for the following reasons:

1. Wrapper classes are final and immutable
2. Provides methods like valueOf(), parseInt(), etc.
3. It provides the feature of autoboxing and unboxing.

23. Differentiate between instance and local variables.

Instance Variable	**Local Variable**
Declared outside the method, directly invoked by the method.	Declared within the method.
Has a default value.	No default value
It can be used throughout the class.	The scope is limited to the method.

24. What are the default values assigned to variables and instances in Java?

In Java When we haven't initialized the instance variables then the compiler initializes them with default values. The default values for instances and variables depend on their data types. Some common types of default data types are:

- The default value for numeric types (byte, short, int, long, float, and double) is 0.
- The default value for the boolean type is false.
- The default value for object types (classes, interfaces, and arrays) is null.
- The null character, "u0000, " is the default value for the char type.

Example:

- Java

```
// Java Program to demonstrate use of default values
```

```java
import java.io.*;
class GFG {
    // static values
    static byte b;
    static int i;
    static long l;
    static short s;
    static boolean bool;
    static char c;
    static String str;
    static Object object;
    static float f;
    static double d;
    static int[] Arr;
    public static void main(String[] args)
    {
        // byte value
        System.out.println("byte value" + b);
        // short value
        System.out.println("short value" + s);
        // int value
        System.out.println("int value" + i);
        // long value
        System.out.println("long value" + l);
        System.out.println("boolean value" + bool);
        System.out.println("char value" + c);
        System.out.println("float value" + f);
        System.out.println("double value" + d);
        System.out.println("string value" + str);
        System.out.println("object value" + object);
```

```
        System.out.println("Array value" + Arr);
    }
}
```

Output

```
byte value0
short value0
int value0
long value0
boolean valuefalse
char value
float value0.0
double value0.0
string valuenull
object valuenull
Array valuenull
```

25. What is a Class Variable?

In Java, a class variable (also known as a static variable) is a variable that is declared within a class but outside of any method, constructor, or block. Class variables are declared with the static keyword, and they are shared by all instances (objects) of the class as well as by the class itself. No matter how many objects are derived from a class, each class variable would only exist once.

Example:

- Java

```java
// Java program to demonstrate use of Clas Variable
class GFG {
    public static int ctr = 0;
    public GFG() { ctr++; }
    public static void main(String[] args)
    {
        GFG obj1 = new GFG();
        GFG obj2 = new GFG();
        GFG obj3 = new GFG();
        System.out.println("Number of objects created are "
                           + GFG.ctr);
    }
}
```

Output

```
Number of objects created are 3
```

26. What is the default value stored in Local Variables?

There is no default value stored with local variables. Also, primitive variables and objects don't have any default values.

27. Explain the difference between instance variable and a class variable.

Instance Variable: A class variable without a static modifier known as an instance variable is typically shared by all instances of the class. These variables can have

distinct values among several objects. The contents of an instance variable are completely independent of one object instance from another because they are related to a specific object instance of the class.

Example:

- Java

```java
// Java Program to demonstrate Instance Variable
import java.io.*;
class GFG {
    private String name;
    public void setName(String name) { this.name = name; }
    public String getName() { return name; }
    public static void main(String[] args)
    {
        GFG obj = new GFG();
        obj.setName("John");
        System.out.println("Name " + obj.getName());
    }
}
```

Output

```
Name John
```

Class Variable: Class Variable variable can be declared anywhere at the class level using the keyword static. These variables can only have one value when applied to various objects. These variables can be shared by all class members since they are not connected to any specific object of the class.

Example:

- Java

```java
// Java Program to demonstrate Class Variable

import java.io.*;

class GFG {

    // class variable

    private static final double PI = 3.14159;

    private double radius;

    public GFG(double radius) { this.radius = radius; }

    public double getArea() { return PI * radius * radius; }

    public static void main(String[] args)

    {

        GFG obj = new GFG(5.0);

        System.out.println("Area of circle: "

                           + obj.getArea());

    }

}
```

Output

```
Area of circle: 78.53975
```

28. What is a static variable?

The static keyword is used to share the same variable or method of a given class. Static variables are the variables that once declared then a single copy of the variable is created and shared among all objects at the class level.

29. What is the difference between System.out, System.err, and System.in?

System.out – It is a PrintStream that is used for writing characters or can be said it can output the data we want to write on the Command Line Interface console/terminal.

Example:

- Java

```
// Java Program to implement
// System.out
import java.io.*;
// Driver Class
class GFG {
    // Main Function
    public static void main(String[] args)
    {
        // Use of System.out
        System.out.println("");
    }
}
```

System.err – It is used to display error messages.

Example:

- Java

```
// Java program to demonstrate
// System.err
```

```java
import java.io.*;

// Driver Class

class GFG {

    // Main function

    public static void main(String[] args)

    {

        // Printing error

        System.err.println(

            "This is how we throw error with System.err");

    }

}
```

Output:

```
This is how we throw error with System.err
```

Although, System.err have many similarities both of them have quite a lot of difference also, let us check them.

System.out	System.err
It will print to the standard out of the system.	It will print to the standard error.
It is mostly used to display results on the console.	It is mostly used to output error texts.
It gives output on the console with the default(black) color.	It also gives output on the console but most of the IDEs give it a red color to differentiate.

System.in – It is an InputStream used to read input from the terminal Window. We can't use the System.in directly so we use Scanner class for taking input with the system.in.

Example:

- Java

```java
// Java Program to demonstrate
// System.in
import java.util.*;
// Driver Class
class Main {
    // Main Function
    public static void main(String[] args)
    {
        // Scanner class with System.in
        Scanner sc = new Scanner(System.in);
        // Taking input from the user
        int x = sc.nextInt();
        int y = sc.nextInt();
        // Printing the output
        System.out.printf("Addition: %d", x + y);
    }
}
```

Output:

```
3
4
Addition: 7
```

30. What do you understand by an IO stream?

Java brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all types of objects, data types, characters, files, etc to fully execute the I/O operations.

31. What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?

The key difference between them is that byte stream data is read and written by input/output stream classes. Characters are handled by the Reader and Writer classes.

In contrast to Reader/Writer classes, which accept character arrays as parameters, input/output stream class methods accept byte arrays. In comparison to input/output streams, the Reader/Writer classes are more efficient, handle all Unicode characters, and are useful for internalization. Use Reader/Writer classes instead of binary data, such as pictures, unless you do so.

Example:

- Java

```
// Java Program to demonstrate Reading Writing Binary Data
// with InputStream/OutputSream
import java.io.*;
class GFG {
    public static void main(String[] args)
    {
        try {
            // Writing binary data to a file using
            // OutputStream
            byte[] data = { 3, 334, 234, 234, 324, 234 };
            OutputStream os
                = new FileOutputStream("data.bin");
            os.write(data);
            os.close();
            // Reading binary data from a file using
            // InputStream
            InputStream is
                = new FileInputStream("data.bin");
            byte[] Buffer = new byte[5];
            is.read(buffer);
            is.close();
            // Printing the read data
            for (byte b : Buffer) {
                System.out.println(b);
```

```
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

32. What are the super most classes for all the streams?

All the stream classes can be divided into two types of classes that are ByteStream classes and CharacterStream Classes. The ByteStream classes are further divided into InputStream classes and OutputStream classes. CharacterStream classes are also divided into Reader classes and Writer classes. The SuperMost classes for all the InputStream classes is java.io.InputStream and for all the output stream classes is java.io.OutPutStream. Similarly, for all the reader classes, the super-most class is java.io.Reader, and for all the writer classes, it is java.io.Writer.

33. What are the FileInputStream and FileOutputStream?

To read and write data, Java offers I/O Streams. A Stream represents an input source or an output destination, which could be a file, an i/o device, another program, etc. **FileInputStream** in Java is used to read data from a file as a stream of bytes. It is mostly used for reading binary data such as images, audio files, or serialized objects.

Example:

```
File file = new File("path_of_the_file");
FileInputStream inputStream = new FileInputStream(file);
```

In Java, the **FileOutputStream** function is used to write data byte by byte into a given file or file descriptor. Usually, raw byte data, such as pictures, is written into a file using FileOutputStream.

Example:

```
File file = new File("path_of_the_file");
FileOutputStream outputStream = new FileOutputStream(file);
```

34. What is the purpose of using BufferedInputStream and BufferedOutputStream classes?

When we are working with the files or stream then to increase the Input/Output performance of the program we need to use the BufferedInputStream and BufferedOutputStream classes. These both classes provide the capability of buffering which means that the data will be stored in a buffer before writing to a file or reading it from a stream. It also reduces the number of times our OS needs to interact with the network or the disk. Buffering allows programs to write a big amount of data instead

of writing it in small chunks. This also reduces the overhead of accessing the network or the disk.

```
BufferedInputStream(InputStream inp);
// used to create the bufferinput stream and save the arguments.

BufferedOutputStream(OutputStream output);
// used to create a new buffer with the default size.
```

35. What are FilterStreams?

Stream filter or Filter Streams returns a stream consisting of the elements of this stream that match the given predicate. While working filter() it doesn't actually perform filtering but instead creates a new stream that, when traversed, contains the elements of initial streams that match the given predicate.

36. What is an I/O filter?

An I/O filter also defined as an Input Output filter is an object that reads from one stream and writes data to input and output sources. It used java.io package to use this filter.

37. How many ways you can take input from the console?

There are two methods to take input from the console in Java mentioned below:

1. Using Command line argument
2. Using Buffered Reader Class
3. Using Console Class
4. Using Scanner Class

The program demonstrating the use of each method is given below.

Example:

- Java

```java
// Java Program to implement input

// using Command line argument

import java.io.*;

class GFG {

    public static void main(String[] args)

    {

        // check if length of args array is

        // greater than 0

        if (args.length > 0) {

            System.out.println(

                "The command line arguments are:");
```

```
            // iterating the args array and printing
            // the command line arguments
            for (String val : args)
                System.out.println(val);
        }
        else
            System.out.println("No command line "
                               + "arguments found.");
    }
}
// Use below commands to run the code
// javac GFG.java
// java Main GeeksforGeeks
```

- Java

```
// Java Program to implement
// Buffer Reader Class
import java.io.*;
class GFG {
    public static void main(String[] args)
        throws IOException
    {
        // Enter data using BufferReader
        BufferedReader read = new BufferedReader(
            new InputStreamReader(System.in));
        // Reading data using readLine
        String x = read.readLine();
        // Printing the read line
        System.out.println(x);
```

```
    }
}
```

- Java

```
// Java program to implement input
// Using Console Class
public class GfG {
    public static void main(String[] args)
    {
        // Using Console to input data from user
        String x = System.console().readLine();
        System.out.println("You entered string " + x);
    }
}
```

- Java

```
// Java program to demonstrate
// working of Scanner in Java
import java.util.Scanner;
class GfG {
    public static void main(String args[])
    {
        // Using Scanner for Getting Input from User
        Scanner in = new Scanner(System.in);
        String str = in.nextLine();
        System.out.println("You entered string " + str);
    }
```

```
}
```

Output:

```
GeeksforGeeks
```

38. Difference in the use of print, println, and printf.

print, println, and printf all are used for printing the elements but print prints all the elements and the cursor remains in the same line. println shifts the cursor to next line. And with printf we can use format identifiers too.

39. What are operators?

Operators are the special types of symbols used for performing some operations over variables and values.

40. How many types of operators are available in Java?

All types of operators in Java are mentioned below:

1. Arithmetic Operators
2. Unary Operators
3. Assignment Operator
4. Relational Operators
5. Logical Operators
6. Ternary Operator
7. Bitwise Operators
8. Shift Operators
9. instance of operator

Postfix operators are considered as the highest precedence according to Java operator precedence.

41. Explain the difference between >> and >>> operators.

Operators like >> and >>> seem to be the same but act a bit differently. >> operator shifts the sign bits and the >>> operator is used in shifting out the zero-filled bits.

Example:

- Java

```
// Java Program to demostrate

// >> and >>> operators

import java.io.*;

// Driver

class GFG {

    public static void main(String[] args)

    {
```

```
        int a = -16, b = 1;

        // Use of >>

        System.out.println(a >> b);

        a = -17;

        b = 1;

        // Use of >>>

        System.out.println(a >>> b);

    }
}
```

Output

```
-8

2147483639
```

42. Which Java operator is right associative?

There is only one operator which is right associative which is = operator.

43. What is dot operator?

The Dot operator in Java is used to access the instance variables and methods of class objects. It is also used to access classes and sub-packages from the package.

44. What is covariant return type?

The covariant return type specifies that the return type may vary in the same direction as the subclass. It's possible to have different return types for an overriding method in the child class, but the child's return type should be a subtype of the parent's return type and because of that overriding method becomes variant with respect to the return type.

We use covariant return type because of the following reasons:

- Avoids confusing type casts present in the class hierarchy and makes the code readable, usable, and maintainable.
- Gives liberty to have more specific return types when overriding methods.
- Help in preventing run-time ClassCastExceptions on returns.

45. What is the transient keyword?

The transient keyword is used at the time of serialization if we don't want to save the value of a particular variable in a file. When JVM comes across a transient keyword, it ignores the original value of the variable and saves the default value of that variable data type.

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

Sleep()	**Wait()**
The sleep() method belongs to the thread class.	Wait() method belongs to the object class.
Sleep does not release the lock that the current thread holds.	wait() release the lock which allows other threads to acquire it.
This method is a static method.	This method is not a static method.
Sleep() does not throw an InterruptedException.	InterruptedException is shown if the thread is interrupted while waiting.
Mainly used to delay a thread for some specific time duration.	Mainly used to pause a thread until notified by another thread.
Sleep() Has Two Overloaded Methods: • sleep(long millis)millis: milliseconds • sleep(long millis, int nanos) nanos: Nanoseconds	Wait() Has Three Overloaded Methods: • wait() • wait(long timeout) • wait(long timeout, int nanos)

47. What are the differences between String and StringBuffer?

String	**StringBuffer**
Store of a sequence of characters.	Provides functionality to work with the strings.

String	StringBuffer
It is immutable.	It is mutable (can be modified and other string operations could be performed on them.)
No thread operations in a string.	It is thread-safe (two threads can't call the methods of StringBuffer simultaneously)

48. What are the differences between StringBuffer and StringBuilder?

StringBuffer	StringBuilder
StringBuffer provides functionality to work with the strings.	StringBuilder is a class used to build a mutable string.
It is thread-safe (two threads can't call the methods of StringBuffer simultaneously)	It is not thread-safe (two threads can call the methods concurrently)
Comparatively slow as it is synchronized.	Being non-synchronized, implementation is faster

49. Which among String or String Buffer should be preferred when there are a lot of updates required to be done in the data?

The string is preferred over StringBuffer as StringBuilder is faster than StringBuffer, it has the capability to utilize it whenever possible.

50. Why is StringBuffer called mutable?

StringBuffer class in Java is used to represent a changeable string of characters. It offers an alternative to the immutable String class by enabling you to change a string's contents without constantly creating new objects. Mutable (modifiable) strings are created with the help of the StringBuffer class. The StringBuffer class in Java is identical to the String class except that it is changeable.

Example:

- Java

```
// Java Program to demonstrate use of stringbuffer
```

```
public class StringBufferExample {

    public static void main(String[] args)

    {

        StringBuffer s = new StringBuffer();

        s.append("Geeks");

        s.append("for");

        s.append("Geeks");

        String message = s.toString();

        System.out.println(message);

    }

}
```

Output

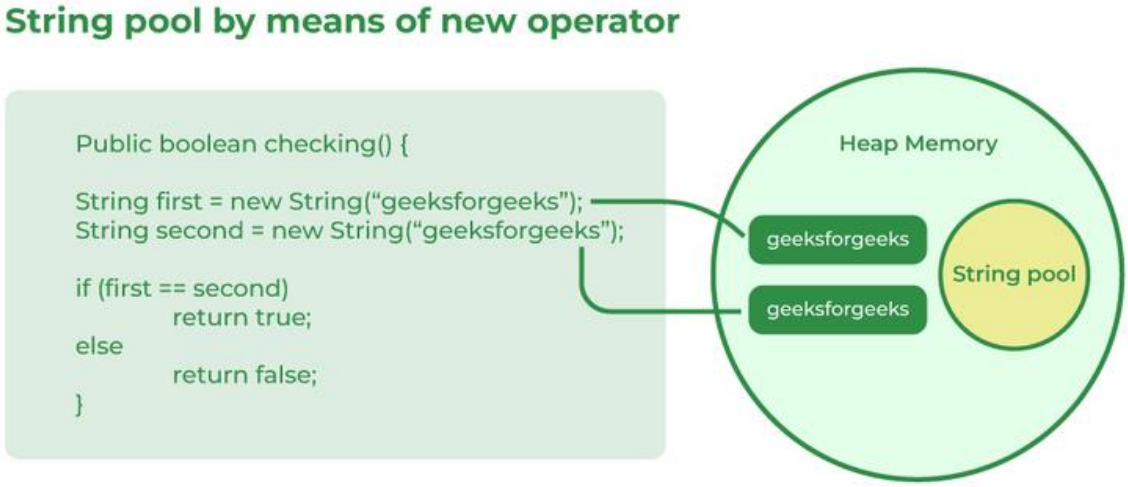
```
GeeksforGeeks
```

51. How is the creation of a String using new() different from that of a literal?

String using new() is different from the literal as when we declare string it stores the elements inside the stack memory whereas when it is declared using new() it allocates a dynamic memory in the heap memory. The object gets created in the heap memory even if the same content object is present.

Syntax:

```
String x = new String("ABC");
```



52. What is an array in Java?

An Array in Java is a data structure that is used to store a fixed-size sequence of elements of the same type. Elements of an array can be accessed by their index, which starts from 0 and goes up to a length of minus 1. Array declaration in Java is done with the help of square brackets and size is also specified during the declaration.

Syntax:

```
int[] Arr = new int[5];
```

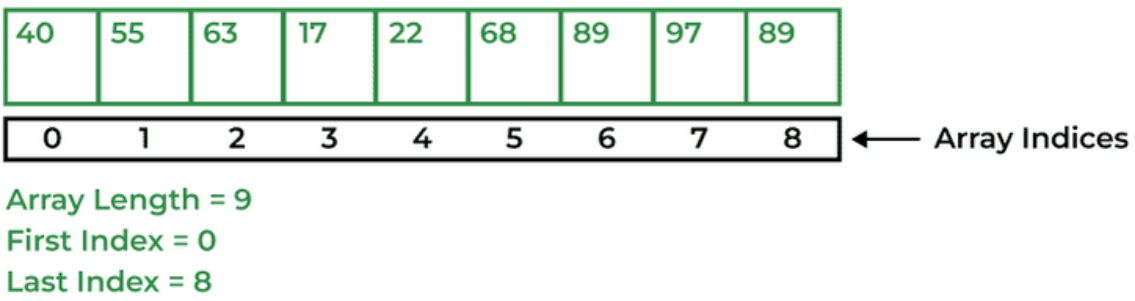
53. On which memory arrays are created in Java?

Arrays in Java are created in heap memory. When an array is created with the help of a new keyword, memory is allocated in the heap to store the elements of the array. In Java, the heap memory is managed by the Java Virtual Machine(JVM) and it is also shared between all threads of the Java Program. The memory which is no longer in use by the program, JVM uses a garbage collector to reclaim the memory. Arrays in Java are created dynamically which means the size of the array is determined during the runtime of the program. The size of the array is specified during the declaration of the array and it cannot be changed once the array is created.

54. What are the types of an array?

There are two types of arrays i.e., Primitive arrays and References Arrays.

- **Single-Dimensional Arrays:** Arrays that have only one dimension i.e., an array of integers or an array of strings are known as single-dimensional arrays.



Syntax:

```
data_type[] Array_Name = new data_type[ArraySize];
```

- **Multi-Dimensional Arrays:** Arrays that have two or more dimensions such as two-dimensional or three-dimensional arrays.

55. Why does the Java array index start with 0?

The index of an array signifies the distance from the start of the array. So, the first element has 0 distance therefore the starting index is 0.

Syntax:

```
[Base Address + (index * no_of_bytes)]
```

56. What is the difference between int array[] and int[] array?

Both int array[] and int[] array are used to declare an array of integers in java. The only difference between them is on their syntax no functionality difference is present between them.

```
int arr[] is a C-Style syntax to declare an Array.

int[] arr is a Java-Style syntax to declare an Array.
```

However, it is generally recommended to use Java-style syntax to declare an Array. As it is easy to read and understand also it is more consistent with other Java language constructs.

57. How to copy an array in Java?

In Java there are multiple ways to copy an Array based on the requirements.

- **clone() method in Java:** This method in Java is used to create a shallow copy of the given array which means that the new array will share the same memory as the original array.

```
int[] Arr = { 1, 2, 3, 5, 0};
int[] tempArr = Arr.clone();
```

- **arraycopy() method:** To create a deep copy of the array we can use this method which creates a new array with the same values as the original array.

```
int[] Arr = {1, 2, 7, 9, 8};
int[] tempArr = new int[Arr.length];
System.arraycopy(Arr, 0, tempArr, 0, Arr.length);
```

- **copyOf() method:** This method is used to create a new array with a specific length and copies the contents of the original array to the new array.

```
int[] Arr = {1, 2, 4, 8};
int[] tempArr = Arrays.copyOf(Arr, Arr.length);
```

- **copyOfRange() method:** This method is very similar to the copyOf() method in Java, but this method also allows us to specify the range of the elements to copy from the original array.

```
int[] Arr = {1, 2, 4, 8};
int[] temArr = Arrays.copyOfRange(Arr, 0, Arr.length);
```

58. What do you understand by the jagged array?

A jagged Array in Java is just a two-dimensional array in which each row of the array can have a different length. Since all the rows in a 2-d Array have the same length but a jagged array allows more flexibility in the size of each row. This feature is very useful in conditions where the data has varying lengths or when memory usage needs to be optimized.

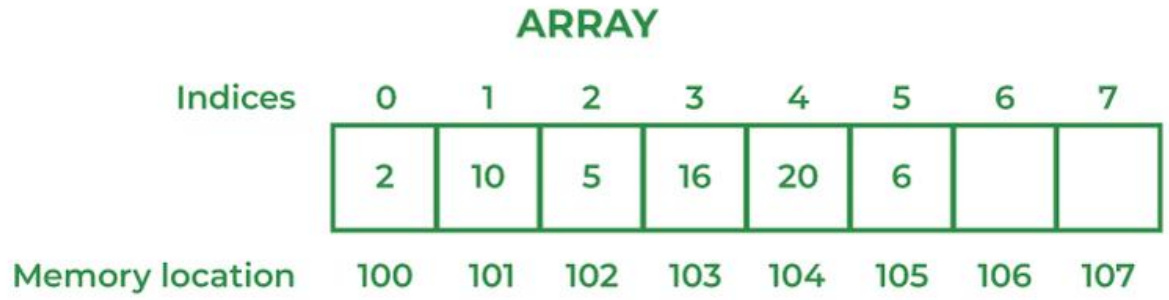
Syntax:

```
int[][] Arr = new int[][] {
    {1, 2, 8},
    {7, 5},
    {6, 7, 2, 6}
};
```

59. Is it possible to make an array volatile?

In Java, it is not possible to make a volatile. Volatile keywords in Java can only be applied to individual variables but not to arrays or collections. The value of the Variable is always read from and written to the main memory when it is defined as volatile rather than being cached in a thread's local memory. This makes it easier to make sure that all threads that access the variable can see changes made to it.

60. What are the advantages and disadvantages of an array?



The advantages of Arrays are:

- Direct and effective access to any element in the collection is made possible by arrays. An array's elements can be accessed using an O(1) operation, which means that the amount of time needed to do so is constant and independent of the array's size.
- Data can be stored effectively in memory using arrays. The size of an array is known at compile time since its elements are stored in contiguous memory regions.
- Due to the fact that the data is stored in contiguous memory areas, arrays provide quick data retrieval.
- Arrays are easy to implement and understand, making them an ideal choice for beginners learning computer programming.

Disadvantages of Arrays are:

- Arrays are created with a predetermined size that is chosen at that moment. This means that if the array's size needs to be extended, a new array will

need to be made, and the data will need to be copied from the old array to the new array, which can take a lot of time and memory.
- There may be unused memory space in an array's memory space if the array is not completely occupied. If you have poor recall, this can be a problem.
- Compared to other data structures like linked lists and trees, arrays might be rigid due to their fixed size and limited support for sophisticated data types.
- Because an array's elements must all be of the same data type, it does not support complex data types like objects and structures.

61. What is an object-oriented paradigm?

Paradigm literally means a pattern or a method. Programming paradigms are the methods to solve a program that is of four types namely, Imperative, logical, functional, and object-oriented. When objects are used as base entities upon which the methods are applied, encapsulation or inheritance functionalities are performed, it is known as an object-oriented paradigm.

62. What are the main concepts of OOPs in Java?

The main concepts of OOPs in Java are mentioned below:

- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

63. What is the difference between an object-oriented programming language and an object-based programming language?

Object-Oriented Programming Language	**Object-Based Programming Language**
Object-oriented programming language covers larger concepts like inheritance, polymorphism, abstraction, etc.	The scope of object-based programming is limited to the usage of objects and encapsulation.
It supports all the built-in objects	It doesn't support all the built-in objects
Examples: Java, C#, etc.	Examples: Java script, visual basics, etc.

64. How is the 'new' operator different from the 'newInstance()' operator in Java?

the new operator is used to create objects, but if we want to decide the type of object to be created at runtime, there is no way we can use the new operator. In this case, we have to use the newInstance() method.

65. What are Classes in Java?

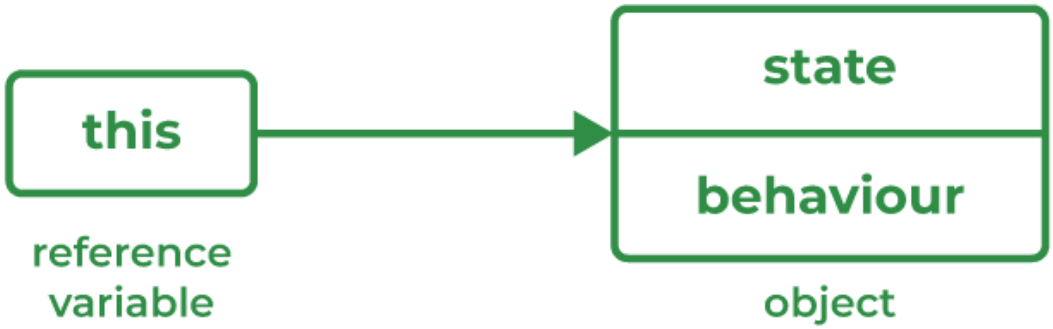
In Java, Classes are the collection of objects sharing similar characteristics and attributes. Classes represent the blueprint or template from which objects are

created. Classes are not real-world entities but help us to create objects which are real-world entities.

66. What is the difference between static (class) method and instance method?

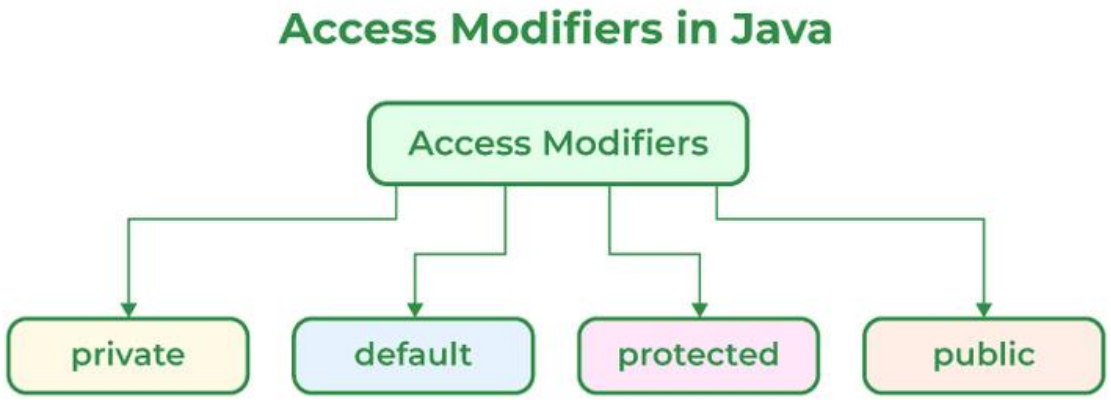
Static(Class) method	**Instance method**
Static method is associated with a class rather than an object.	The instance method is associated with an object rather than a class.
Static methods can be called using the class name only without creating an instance of a class.	The instance method can be called on a specific instance of a class using the object reference.
Static methods do not have access to **this** keyword**.**	Instance methods have access to **this** keyword**.**
This method can access only static members of the class	This method can access both static and non-static methods of the class.

67. What is this keyword in Java?



'this' is a keyword used to reference a variable that refers to the current object.

68. What are Brief Access Specifiers and Types of Access Specifiers?



Access Specifiers in Java help to restrict the scope of a class, constructor, variable, method, or data member. There are four types of Access Specifiers in Java mentioned below:

1. Public
2. Private
3. Protected
4. Default

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

The initial value of an object reference which is defined as an instance variable is a NULL value.

70. What is an object?

The object is a real-life entity that has certain properties and methods associated with it. The object is also defined as the instance of a class. An object can be declared using a new keyword.

71. What are the different ways to create objects in Java?

Methods to create objects in Java are mentioned below:

1. Using new keyword
2. Using new instance
3. Using clone() method
4. Using deserialization
5. Using the newInstance() method of the Constructor class

To know more about methods to create objects in Java refer to this article.

72. What are the advantages and disadvantages of object cloning?

There are many advantages and disadvantages of using object cloning as mentioned below:

Advantages:

- In Java, the '=' assignment operator cannot be used for cloning as it simply creates a copy of reference variables. To overcome such discrepancy the clone() method of Object class can be used over the assignment operator.
- The clone() method is a protected method of class Object which means that only the Employee class can clone Employee objects. This means no class

other than Employee can clone Employee objects since it does not know the Employee class' attributes.
- Code size decreases as repetition decreases.

Disadvantages:

- As the Object.clone() method is protected, so need to provide our own clone() and indirectly call Object.clone() from it.
- If we don't have any methods then we need to provide a Cloneable interface as we need to provide JVM information so that we can perform a clone() on our object.

73. What are the advantages of passing this into a method instead of the current class object itself?

There are a few advantages of passing this into a method instead of the current class object itself these are:

- this is the final variable because of which this cannot be assigned to any new value whereas the current class object might not be final and can be changed.
- this can be used in the synchronized block.

74. What is the constructor?

Constructor is a special method that is used to initialize objects. Constructor is called when a object is created. The name of constructor is same as of the class.

Example:

```
// Class Created
class XYZ{
      private int val;

      // Constructor
      XYZ(){
            val=0;
      }
};
```

75. What happens if you don't provide a constructor in a class?

If you don't provide a constructor in a class in Java, the compiler automatically generates a default constructor with no arguments and no operation which is a default constructor.

76. How many types of constructors are used in Java?

There are two types of constructors in Java as mentioned below:

1. Default Constructor
2. Parameterized Constructor

Default Constructor: It is the type that does not accept any parameter value. It is used to set initial values for object attributes.

```
class_Name();
// Default constructor called
```

Parameterized Constructor: It is the type of constructor that accepts parameters as arguments. These are used to assign values to instance variables during the initialization of objects.

```
class_Name(parameter1, parameter2......);
// All the values passed as parameter will be
// allocated accordingly
```

77. What is the purpose of a default constructor?

Constructors help to create instances of a class or can be said to create objects of a class. Constructor is called during the initialization of objects. A default constructor is a type of constructor which do not accept any parameter, So whatever value is assigned to properties of the objects are considered default values.

78. What do you understand by copy constructor in Java?

The copy constructor is the type of constructor in which we pass another object as a parameter because which properties of both objects seem the same, that is why it seems as if constructors create a copy of an object.

79. Where and how can you use a private constructor?

A private constructor is used if you don't want any other class to instantiate the object to avoid subclassing. The use private constructor can be seen as implemented in the example.

Example:

- Java

```java
// Java program to demonstrate implementation of Singleton
// pattern using private constructors.
import java.io.*;
class GFG {
    static GFG instance = null;
    public int x = 10;
    // private constructor can't be accessed outside the
    // class
    private GFG() {}
    // Factory method to provide the users with instances
    static public GFG getInstance()
    {
        if (instance == null)
```

```
            instance = new GFG();
        return instance;
    }
}
// Driver Class
class Main {
    public static void main(String args[])
    {
        GFG a = GFG.getInstance();
        GFG b = GFG.getInstance();
        a.x = a.x + 10;
        System.out.println("Value of a.x = " + a.x);
        System.out.println("Value of b.x = " + b.x);
    }
}
```

Output

```
Value of a.x = 20
Value of b.x = 20
```

80. What are the differences between the constructors and methods?

Java constructors are used for initializing objects. During creation, constructors are called to set attributes for objects apart from this few basic differences between them are:

1. Constructors are only called when the object is created but other methods can be called multiple times during the life of an object.
2. Constructors do not return anything, whereas other methods can return anything.
3. Constructors are used to setting up the initial state but methods are used to perform specific actions.

81. What is an Interface?

An interface in Java is a collection of static final variables and abstract methods that define the contract or agreement for a set of linked classes. Any class that implements an interface is required to implement a specific set of methods. It specifies the behavior that a class must exhibit but not the specifics of how it should be implemented.

Syntax:

```
interface
{
    // constant fields
    // methds that are abstract by default
}
```

Example:

- Java

```java
// Java Program to demonstrate Interface

import java.io.*;

interface Shape {

    double getArea();

    double getPerimeter();

}

class Circle implements Shape {

    private double radius;

    public Circle(double radius) { this.radius = radius; }

    public double getArea()

    {

        return Math.PI * radius * radius;

    }

    public double getPerimeter()
```

```
    {
        return 2 * Math.PI * radius;
    }
}
class GFG {
    public static void main(String[] args)
    {
        Circle circle = new Circle(5.0);
        System.out.println("Area of circle is "
                           + circle.getArea());
        System.out.println("Perimeter of circle is"
                           + circle.getPerimeter());
    }
}
```

Output

```
Area of circle is 78.53981633974483
Perimeter of circle is31.41592653589793
```

82. Give some features of the Interface.

An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a behavior. A Java interface contains static constants and abstract methods.

Features of the Interface are mentioned below:

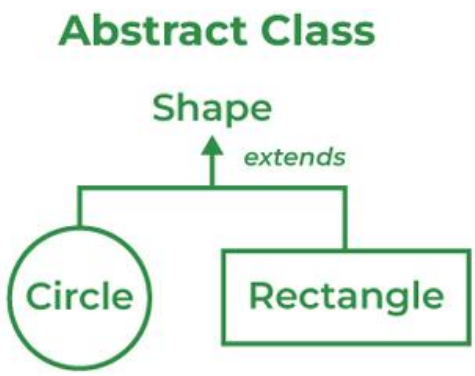
- The interface can help to achieve total abstraction.
- Allows us to use multiple inheritances in Java.

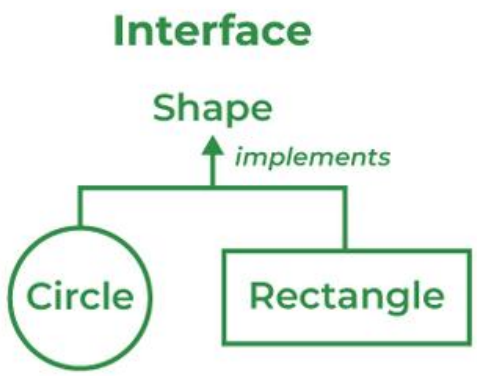
- Any class can implement multiple interfaces even when one class can extend only one class.
- It is also used to achieve loose coupling.

83. What is a marker interface?

An Interface is recognized as an empty interface (no field or methods) it is called a marker interface. Examples of marker interfaces are Serializable, Cloneable, and Remote interfaces.

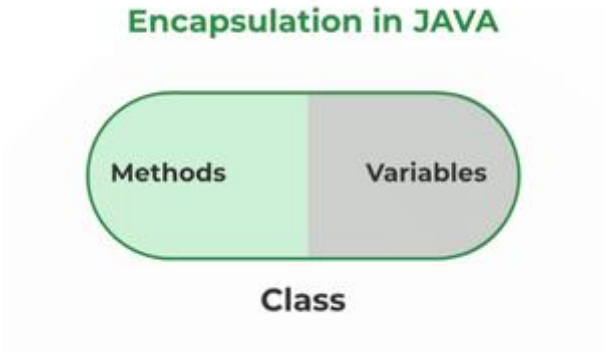
84. What are the differences between abstract class and interface?





Abstract Class	Interface Class
Both abstract and non-abstract methods may be found in an abstract class.	The interface contains only abstract methods.
Abstract Class supports Final methods.	The interface class does not support Final methods.
Multiple inheritance is not supported by the Abstract class.	Multiple inheritances is supported by Interface Class.
Abstract Keyword is used to declare Abstract class.	Interface Keyword is used to declare the interface class.
extend keyword is used to extend an Abstract Class.	**implements** Keyword is used to implement the interface.
Abstract Class has members like protected, private, etc.	All class members are public by default.

85. What do you mean by data encapsulation?



Data Encapsulation is the concept of OOPS properties and characteristics of the classes that The interface is binded together. Basically, it bundles data and methods that operate on that data within a single unit. Encapsulation is achieved by declaring the instance variables of a class as private, which means they can only be accessed within the class.

86. What are the advantages of Encapsulation in Java?

The advantages of Encapsulation in Java are mentioned below:

1. Data Hiding: it is a way of restricting the access of our data members by hiding the implementation details. Encapsulation also provides a way for data hiding. The user will have no idea about the inner implementation of the class.
2. Increased Flexibility: We can make the variables of the class read-only or write-only depending on our requirements.
3. Reusability: Encapsulation also improves the re-usability and is easy to change with new requirements.
4. Testing code is easy: Code is made easy to test for unit testing.

87. What is the primary benefit of Encapsulation?

The main advantage of Encapsulation in Java is its ability to protect the internal state of an object from external modification or access. It is the is a way of hiding the implementation details of a class from outside access and only exposing a public interface that can be used to interact with the class. The main benefit is of providing a way to control and manage the state and the behavior of an object and also protecting it from modification and unauthorized access at the same time.

Example:

- Java

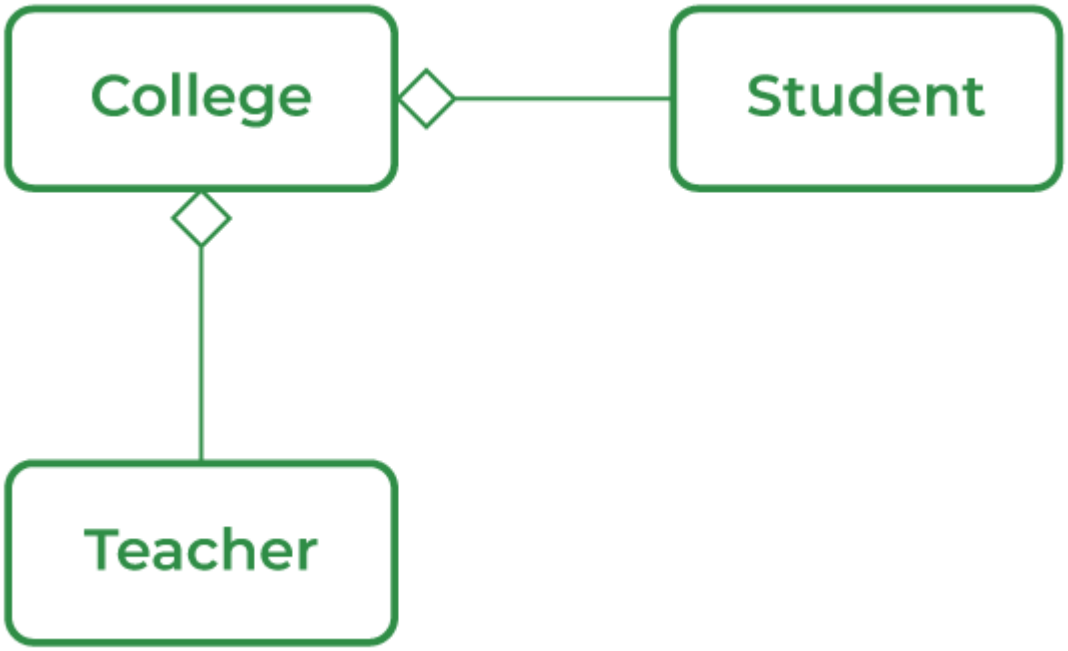
```
// Java Program to demonstrate use of Encapsulation

import java.io.*;

class Person {

    private String Name;
```

```
    private int age;
    public String getName() { return Name; }
    public void setName(String Name) { this.Name = Name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
// Driver class
class GFG {
    // main function
    public static void main(String[] args)
    {
        Person p = new Person();
        p.setName("Rohan");
        p.setAge(29);
        System.out.println("Name is " + p.getName());
        System.out.println("Age is " + p.getAge());
    }
}
```

Output

```
Name is Rohan
Age is 29
```

88. What do you mean by aggregation?



Aggregation is a term related to the relationship between two classes best described as a "has-a" relationship. This kind is the most specialized version of association. It is a unidirectional association means it is a one-way relationship. It contains the reference to another class and is said to have ownership of that class.

89. What is the 'IS-A ' relationship in OOPs Java?

'IS-A' is a type of relationship in OOPs Java where one class inherits another class.

90. Define Inheritance.

When an object that belongs to a subclass acquires all the properties and behavior of a parent object that is from the superclass, it is known as inheritance. A class within a class is called the subclass and the latter is referred to as the superclass. Sub class or the child class is said to be specific whereas the superclass or the parent class is generic. Inheritance provides code reusability.

91. What are the different types of inheritance in Java?

Inheritance is the method by which the Child class can inherit the features of the Super or Parent class. In Java, Inheritance is of four types:

- **Single Inheritance:** When a child or subclass extends only one superclass, it is known to be single inheritance. Single-parent class properties are passed down to the child class.
- **Multilevel Inheritance:** When a child or subclass extends any other subclass a hierarchy of inheritance is created which is known as multilevel inheritance. In other words, one subclass becomes the parent class of another.
- **Hierarchical Inheritance:** When multiple subclasses derive from the same parent class is known as Hierarchical Inheritance. In other words, a class that has a single parent has many subclasses.

- **Multiple Inheritance:** When a child class inherits from multiple parent classes is known as Multiple Inheritance. In Java, it only supports multiple inheritance of interfaces, not classes.

92. What is multiple inheritance? Is it supported by Java?

A component of the object-oriented notion known as multiple inheritances allows a class to inherit properties from many parent classes. When methods with the same signature are present in both superclasses and subclasses, an issue arises. The method's caller cannot specify to the compiler which class method should be called or even which class method should be given precedence.

Example:

- Java

```java
// Java Program to show multiple Inheritance
import java.io.*;
interface Animal {
    void eat();
}
interface Mammal {
    void drink();
}
class Dog implements Animal, Mammal {
    public void eat() { System.out.println("Eating"); }
    public void drink() { System.out.println("Drinking"); }
    void bark() { System.out.println("Barking"); }
}
class GFG {
    public static void main(String[] args)
    {
        Dog d = new Dog();
        d.eat();
        d.drink();
        d.bark();
```

```
    }
}
```

Output

```
Eating
Drinking
Barking
```

93. How is inheritance in C++ different from Java?

Inheritance in C++	Inheritance in Java
C++ lets the user to inherit multiple classes.	Java doesn't support multiple inheritances.
When a class is created in C++, it doesn't inherit from the object class, instead exists on its own.	Java is always said to have a single inheritance as all the classes inherit in one or the other way from the object class.

94. Is there any limitation to using Inheritance?

Yes, there is a limitation of using Inheritance in Java, as because of inheritance one can inherit everything from super class and interface because of which subclass is too clustered and sometimes error-prone when dynamic overriding or dynamic overloading is done in certain situations.

95. Although inheritance is a popular OOPs concept, it is less advantageous than composition. Explain.

Inheritance is a popular concept of Object-Oriented Programming (OOP), in which a class can inherit the properties and methods from any other class, which is referred to as a Parent or superclass. On the other hand in Composition, a class can contain an

instance of another class as a member variable which is often referred to as part or a component. Below are some reasons why composition is more advantageous than inheritance:

- **Tight Coupling:** Whenever any changes are made to the superclass, these changes can affect the behavior of all its child or Subclasses. This problem makes code less flexible and also creates issues during maintenance. This problem also leads to the Tight coupling between the classes.
- **Fragile Base Class Problem:** When the changes to the base class can break the functionality of its derived classes. This problem can make it difficult to add new features or modify the existing ones. This problem is known as the Fragile Base class problem.
- **Limited Reuse:** Inheritance in Java can lead to limited code reuse and also code duplication. As a subclass inherits all the properties and methods of its superclass, sometimes it may end up with unnecessary code which is not needed. This leads to a less maintainable codebase.

96. What is an association?

The association is a relation between two separate classes established through their Objects. It represents Has-A's relationship.

97. What do you mean by aggregation?

Composition is a restricted form of Aggregation in which two entities are highly dependent on each other. It represents **part-of** the relationship.

98. What is the composition of Java?

Composition implies a relationship where the child **cannot exist independently** of the parent. For example Human heart, the heart doesn't exist separately from a Human.

99. State the difference between Composition and Aggregation.

Aggregation	**Composition**
It defines a "has a" relationship between the objects	It represents the part-of relationship
Objects are independent of each other.	Objects are dependent on each other.
Represent it by using the filled diamond.	Represent it by using the empty diamond.
Child objects don't have a lifetime.	Child objects have a lifetime.

100. Can the constructor be inherited?

No, we can't inherit a constructor.

101. What is Polymorphism?

Polymorphism is defined as the ability to take more than one form It is of two types namely, Compile time polymorphism or method overloading- a function called during compile time. For instance, take a class 'area'. Based on the number of parameters it

may calculate the area of a square, triangle, or circle. Run time polymorphism or method overriding- links during run time. The method inside a class overrides the method of the parent class.

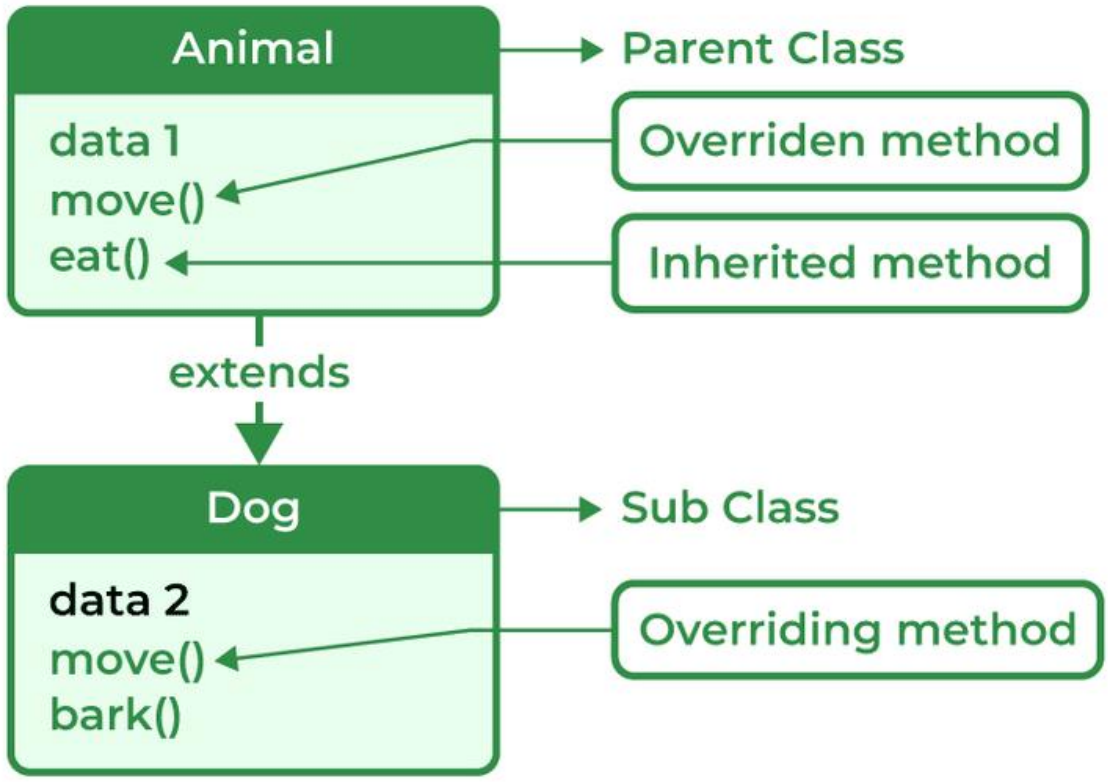
102. What is runtime polymorphism or dynamic method dispatch?

Dynamic method dispatch is a resolving mechanism for method overriding during the run time. Method overriding is the one where the method in a subclass has the same name, parameters, and return type as a method in the superclass. When the over-ridden method is called through a superclass reference, java determines which version (superclass or subclass) of that method is to be executed based upon the type of an object being referred to at the time the call occurs. Thus the decision is made at run time. This is referred to as dynamic method dispatch.

103. What is method overriding?

Method overriding, also known as run time polymorphism is one where the child class contains the same method as the parent class. For instance, we have a method named 'gfg()' in the parent class. A method gfg() is again defined in the sub-class. Thus when gfg() is called in the subclass, the method within the class id executed. Here, gfg() within the class overridden the method outside.

104. What is method overloading?



Method overriding is a method to achieve Run-time polymorphism in Java. Method overriding is a feature that allows a child class to provide a specific implementation of a method that is already provided by one of its parent classes. When a method in a child class has the same name, the same parameters or signature, and the same return

type(or sub-type) as a method in its parent class, then the method in the subclass is said to override the method in the superclass.

105. Can we override the static method?

No, as static methods are part of the class rather than the object so we can't override them.

106. Can we override the overloaded method?

Yes, since the overloaded method is a completely different method in the eyes of the compiler. Overriding isn't the same thing at all. The decision as to which method to call is deferred to runtime.

107. Can we overload the main() method?

Yes in Java we can overload the main method to call the main method with the help of its predefined calling method.

108. What are method overloading and method overriding?

Method Overloading: It is also known as Compile Time Polymorphism. In method overloading two or more methods are shared in the same class with a different signature.

Example:

- Java

```java
// Java Program to demonstrate use of Method Overloading

import java.io.*;

class GFG {

    static int multiply(int a, int b) { return a * b; }

    static int multiply(int a, int b, int c)

    {

        return a * b * c;

    }

    static int multiply(int a, int b, int c, int d)

    {

        return a * b * c * d;

    }

    public static void main(String[] args)

    {

        System.out.println("multiply() with 2 parameters");

        System.out.println(multiply(4, 5));
```

```
        System.out.println("multiply() with 3 parameters");
        System.out.println(multiply(2, 3, 4));
        System.out.println("multiply() with 4 parameters");
        System.out.println(multiply(2, 3, 4, 1));
    }
}
```

Output

```
multiply() with 2 parameters
20
multiply() with 3 parameters
24
multiply() with 4 parameters
24
```

Method Overriding: Method Overriding occurs when a subclass can provide the implementation of a method which is already defined in the parent class or superclass. The return type, name and arguments must be similar to the methods in superclass.

Example:

- Java

```
// Java Program to demonstrate use of Method Overriding
import java.io.*;
class Vehicle {
```

```
    void drive()
    {
        System.out.println("drive() method of base class");
        System.out.println("driving the Car.");
    }
}
class Car extends Vehicle {
    void drive()
    {
        System.out.println(
            "drive() method of derived class");
        System.out.println("Car is driving.");
    }
}
class GFG {
    public static void main(String[] args)
    {
        Car c1 = new Car();
        Vehicle v1 = new Vehicle();
        c1.drive();
        v1.drive();
        Vehicle vehicle = new Car();
        // drive() method of Vehicle class is overridden by
        // Car class drive()
        vehicle.drive();
    }
}
```

Output

```
drive() method of derived class
```

```
Car is driving.
drive() method of base class
driving the Car.
drive() method of derived class
Car is driving.
```

Method Overloading	Method Overriding
When two or multiple methods are in the same class with different parameters but the same name.	When a subclass provides its own implementation of a method that is already defined in the parent class.
Method overloading can only happen in the same class or between a subclass or parent class.	Method overriding can only happen in Subclass.
When an error occurs it is caught at the compile time of the program.	When an error occurs it is caught at Runtime of the program.
Example of Compile Time Polymorphism.	Example of Run Time Polymorphism.
Method Overloading may or may not require Inheritance.	Method overriding always needs Inheritance.
It occurs within the class.	It is performed in two classes with an inheritance relationship.

109. Can we override the private methods?

It is not possible to override the private methods in Java. Method overriding is where the method in the subclass is implemented instead of the method from the parent class.

The private methods are accessible only within the class in which it is declared. Since this method is not visible to other classes and cannot be accessed, it cannot be overridden.

110. Can we change the scope of the overridden method in the subclass?

In Java, it is not possible to modify the overridden method's scope. The subclass method's scope must be equal to or wider than the Superclass method's overridden method's scope. The overridden method in the subclass, for instance, can have a public scope or a more accessible scope like protected or default if the overridden method in the superclass has a public scope. It cannot, however, have a more exclusive scope like private.

111. Can we modify the throws clause of the superclass method while overriding it in the subclass?

We can modify the throws clause of the Superclass method with some limitations, we can change the throws clause of the superclass method while overriding it in the subclass. The subclass overridden method can only specify unchecked exceptions if the superclass method does not declare any exceptions. If the superclass method declares an exception, the subclass method can declare the same exception, a subclass exception, or no exception at all. However, the subclass method cannot declare a parent exception that is broader than the ones declared in the superclass method.

112. Can you have virtual functions in Java?

Yes, Java supports virtual functions. Functions are by default virtual and can be made non-virtual using the final keyword.

113. What is Abstraction?

Abstraction refers to the act of representing essential features without including background details. The detailed information or the implementation is hidden. The most common example of abstraction is a car, we know how to turn on the engine, accelerate and move, however, the way engine works, and its internal components are complex logic hidden from the general users. This is usually done to handle the complexity.

114. What is Abstract class?

A class declared as abstract, cannot be instantiated i.e., the object cannot be created. It may or may not contain abstract methods but if a class has at least one abstract method, it must be declared abstract.

Example of an abstract class with abstract method:

- Java

```
// Java Program to implement

// abstract method

import java.io.*;

// Abstract class
```

```java
abstract class Fruits {

    abstract void run();

}

// Driver Class

class Apple extends Fruits {

    void run()

    {

        System.out.println("Abstract class example");

    }

    // main method

    public static void main(String args[])

    {

        Fruits obj = new Apple();

        obj.run();

    }

}
```

115. When Abstract methods are used?

An abstract method is used when we want to use a method but want to child classes to decide the implementation in that case we use Abstract methods with the parent classes.

116. How can you avoid serialization in the child class if the base class is implementing the Serializable interface?

Serialization in the child class if the base class is implementing the Serializable interface then we can avoid it by defining the writeObject() method and throwing NotSerializableException().

117. What is Collection Framework in Java?

Collections are units of objects in Java. The collection framework is a set of interfaces and classes in Java that are used to represent and manipulate collections of objects in a variety of ways. The collection framework contains classes(ArrayList, Vector, LinkedList, PriorityQueue, TreeSet) and multiple interfaces (Set, List, Queue, Deque) where every interface is used to store a specific type of data.

118. Explain various interfaces used in the Collection framework.

Collection framework implements

1. Collection Interface
2. List Interface
3. Set Interface

4. Queue Interface
5. Deque Interface
6. Map Interface

Collection interface: Collection is the primary interface available that can be imported using java.util.Collection.

Syntax:

```
public interface Collection<E> extends iterable
```

119. How can you synchronize an ArrayList in Java?

An ArrayList can be synchronized using two methods mentioned below:

1. Using Collections.synchronizedList()
2. Using CopyOnWriteArrayList

Using Collections.synchronizedList():

```
public static List<T> synchronizedList(List<T> list)
```

Using CopyOnWriteArrayList:

1. Create an empty List.
2. It implements the List interface
3. It is a thread-safe variant of ArrayList
4. T represents generic

120. Why do we need a synchronized ArrayList when we have Vectors (which are synchronized) in Java?

ArrayList is in need even when we have Vectors because of certain reasons:

1. ArrayList is faster than Vectors.
2. ArrayList supports multithreading whereas Vectors only supports single-thread use.
3. ArrayList is safer to use, as Vectors supports single threads and individual operations are less safe and take longer to synchronize.
4. Vectors are considered outdated in Java because of their synchronized nature.

121. Why can’t we create a generic array?

Generic arrays can’t be created because an **array** carries type information of its elements at runtime because of which during runtime it throw ‘ArrayStoreException’ if the elements’ type is not similar. Since generics type information gets erased at compile time by Type Erasure, the array store check would have been passed where it should have failed.

122. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.

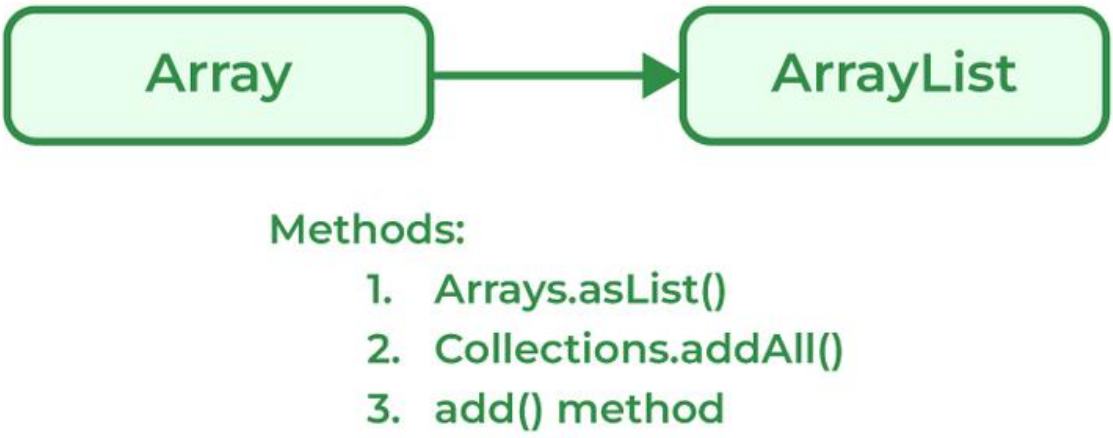
The elements of an array are stored in contiguous memory locations, which means that each element is stored in a separate block based on it located within the array. Since the elements of the array are stored in contiguous locations, it can be relatively easy to access any element by its index, as the element address can be calculated based on the location of the element. But Java implements ArrayLists as dynamic arrays, which means that the size can change as elements are removed or added. ArrayList elements are not stored in contiguous memory locations in order to accommodate this

dynamic nature. Instead, the ArrayList makes use of a method known as an expandable array in which the underlying array is expanded to a larger size as needed and the elements are then copied to the new location. In contrast to an ArrayList, which has a dynamic size and does not store its elements in contiguous memory locations, an array has a fixed size and its elements are stored there.

123. Explain the method to convert ArrayList to Array and Array to ArrayList.

Conversion of List to ArrayList

There are multiple methods to convert List into ArrayList



Programmers can convert an Array to ArrayList using asList() method of the Arrays class. It is a static method of the Arrays class that accepts the List object.

Syntax:

```
Arrays.asList(item)
```

Example:

- Java

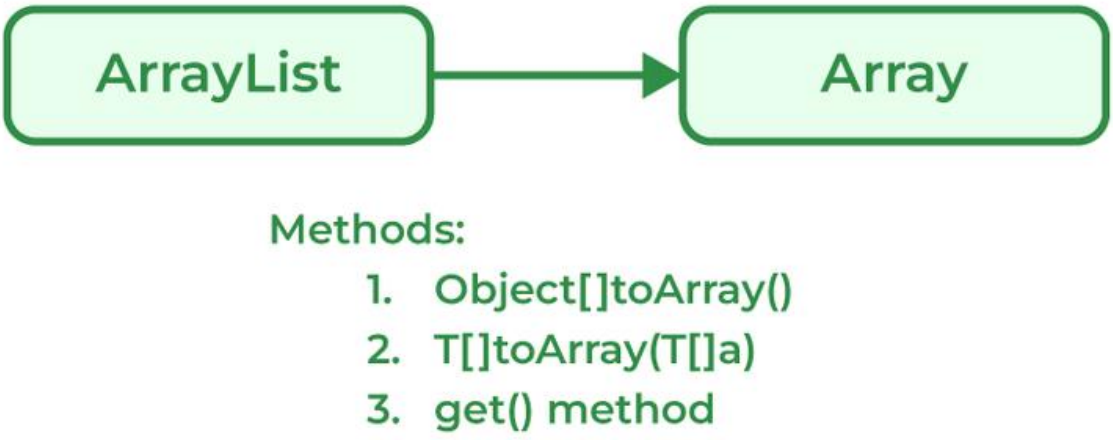
```java
// Java program to demonstrate conversion of

// Array to ArrayList of fixed-size.

import java.util.*;

// Driver Class

class GFG {

    // Main Function

    public static void main(String[] args)

    {

        String[] temp = { "Abc", "Def", "Ghi", "Jkl" };
```

```
        // Conversion of array to ArrayList
        // using Arrays.asList
        List conv = Arrays.asList(temp);
        System.out.println(conv);
    }
}
```

Output

```
[Abc, Def, Ghi, Jkl]
```

Conversion of ArrayList to Array



Java programmers can convert ArrayList to

Syntax:

```
List_object.toArray(new String[List_object.size()])
```

Example:

- Java

```java
// Java program to demonstrate working of
// Objectp[] toArray()
import java.io.*;
import java.util.List;
import java.util.ArrayList;
// Driver Class
class GFG {
    // Main Function
    public static void main(String[] args)
    {
        // List declared
        List<Integer>
            arr = new ArrayList<Integer>();
        arr.add(1);
        arr.add(2);
        arr.add(3);
        arr.add(2);
        arr.add(1);
        // Conversion
        Object[] objects = arr.toArray();
        // Printing array of objects
        for (Object obj : objects)
            System.out.print(obj + " ");
    }
}
```

Output

```
1 2 3 2 1
```

124. How does the size of ArrayList grow dynamically? And also state how it is implemented internally.

Due to ArrayLists array-based nature, it grows dynamically in size ensuring that there is always enough room for elements. When an ArrayList element is first created, the default capacity is around 10-16 elements which basically depends on the Java version. ArrayList elements are copied over from the original array to the new array when the capacity of the original array is full. As the ArrayList size increases dynamically, the class creates a new array of bigger sizes and it copies all the elements from the old array to the new array. Now, the reference of the new array is used internally. This process of dynamically growing an array is known as resizing.

125. What is a Vector in Java?

Vectors in Java are similar and can store multiple elements inside them. Vectors follow certain rules mentioned below:

1. Vector can be imported using Java.util.Vector.
2. Vector is implemented using a dynamic array as the size of the vector increases and decreases depending upon the elements inserted in it.
3. Elements of the Vector using index numbers.
4. Vectors are synchronized in nature means they only used a single thread (only one process is performed at a particular time).
5. The vector contains many methods that are not part of the collections framework.

Syntax:

```
Vector gfg = new Vector(size, increment);
```

126. How to make Java ArrayList Read-Only?

An ArrayList can be made ready only using the method provided by Collections using the Collections.unmodifiableList() method.

Syntax:

```
array_readonly = Collections.unmodifiableList(ArrayList);
```

Example:

- Java

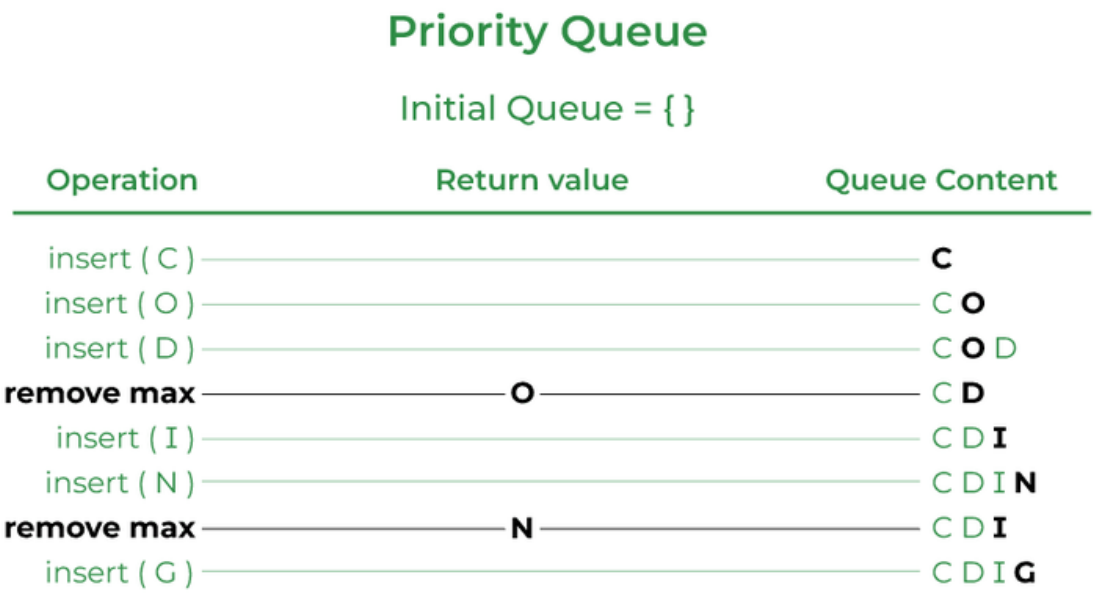
```java
// Java program to demonstrate
// unmodifiableList() method
import java.util.*;
public class Main {
    public static void main(String[] argv) throws Exception
    {
        try {
            // creating object of ArrayList
            <Character> ArrayList<Character>
                temp
                = new ArrayList<Character>();
            // populate the list
            temp.add('X');
            temp.add('Y');
            temp.add('Z');
            // printing the list
            System.out.println("Initial list: " + temp);
            // getting readonly list
            // using unmodifiableList() method
            List<Character>
                new_array
                = Collections.unmodifiableList(temp);
            // printing the list
            System.out.println("ReadOnly ArrayList: "
                               + new_array);
            // Adding element to new Collection
            System.out.println("\nIf add element in "
                               + " the ReadOnly ArrayList");
            new_array.add('A');
        }
```

```
        catch (UnsupportedOperationException e) {
            System.out.println("Exception is thrown : "
                               + e);
        }
    }
}
```

Output

```
Initial list: [X, Y, Z]
ReadOnly ArrayList: [X, Y, Z]

If add element in  the ReadOnly ArrayList
Exception is thrown : java.lang.UnsupportedOperationException
```

127. What is a priority queue in Java?



A priority queue is an abstract data type similar to a regular queue or stack data structure. Elements stored in elements are depending upon the priority defined from low to high. The PriorityQueue is based on the priority heap.

Syntax:

- Java

```
// Java program to demonstrate the
// working of PriorityQueue
import java.util.*;
class PriorityQueueDemo {
    // Main Method
    public static void main(String args[])
    {
        // Creating empty priority queue
        PriorityQueue<Integer>
            var1
            = new PriorityQueue<Integer>();
        // Adding items to the pQueue using add()
        var1.add(10);
```

```
        var1.add(20);
        var1.add(15);
        // Printing the top element of PriorityQueue
        System.out.println(var1.peek());
    }
}
```

Output

```
10
```

128. Explain the LinkedList class.

LinkedList class is Java that uses a doubly linked list to store elements. It inherits the AbstractList class and implements List and Deque interfaces. Properties of the LinkedList Class are mentioned below:

1. LinkedList classes are non-synchronized.
2. Maintains insertion order.
3. It can be used as a list, stack, or queue.

Syntax:

```
LinkedList<class> list_name=new LinkedList<class>();
```

129. What is the Stack class in Java and what are the various methods provided by it?

A Stack class in Java is a LIFO data structure that implements the Last In First Out data structure. It is derived from a Vector class but has functions specific to stacks. The Stack class in java provides the following methods:

- **peek():** returns the top item from the stack without removing it
- **empty():** returns true if the stack is empty and false otherwise
- **push():** pushes an item onto the top of the stack
- **pop():** removes and returns the top item from the stack

- **search():** returns the 1, based position of the object from the top of the stack. If the object is not in the stack, it returns -1

130. What is Set in the Java Collections framework and list down its various implementations?

Sets are collections that don't store duplicate elements. They don't keep any order of the elements. The Java Collections framework provides several implementations of the Set interface, including:

- **HashSet:** HashSet in Java, stores the elements in a has table which provides faster lookups and faster insertion. HashSet is not ordered.
- **LinkedHashSet:** LinkedHashSet is an implementation of HashSet which maintains the insertion order of the elements.
- **TreeSet:** TreeSet stores the elements in a sorted order that is determined by the natural ordering of the elements or by a custom comparator provided at the time of creation.

131. What is the HashSet class in Java and how does it store elements?

The HashSet class implements the Set interface in the Java Collections Framework and is a member of the HashSet class. Unlike duplicate values, it stores a collection of distinct elements. In this implementation, each element is mapped to an index in an array using a hash function, and the index is used to quickly access the element. It produces an index for the element in the array where it is stored based on the input element. Assuming the hash function distributes the elements among the buckets appropriately, the HashSet class provides constant-time performance for basic operations (add, remove, contain, and size).

132. What is LinkedHashSet in Java Collections Framework?

The LinkedHashSet is an ordered version of Hashset maintained by a doubly-linked List across all the elements. It is very helpful when iteration order is needed. During Iteration in LinkedHashSet, elements are returned in the same order they are inserted.

Syntax:

```
LinkedHashSet<E> hs = new LinkedHashSet<E>();
```

Example:

- Java

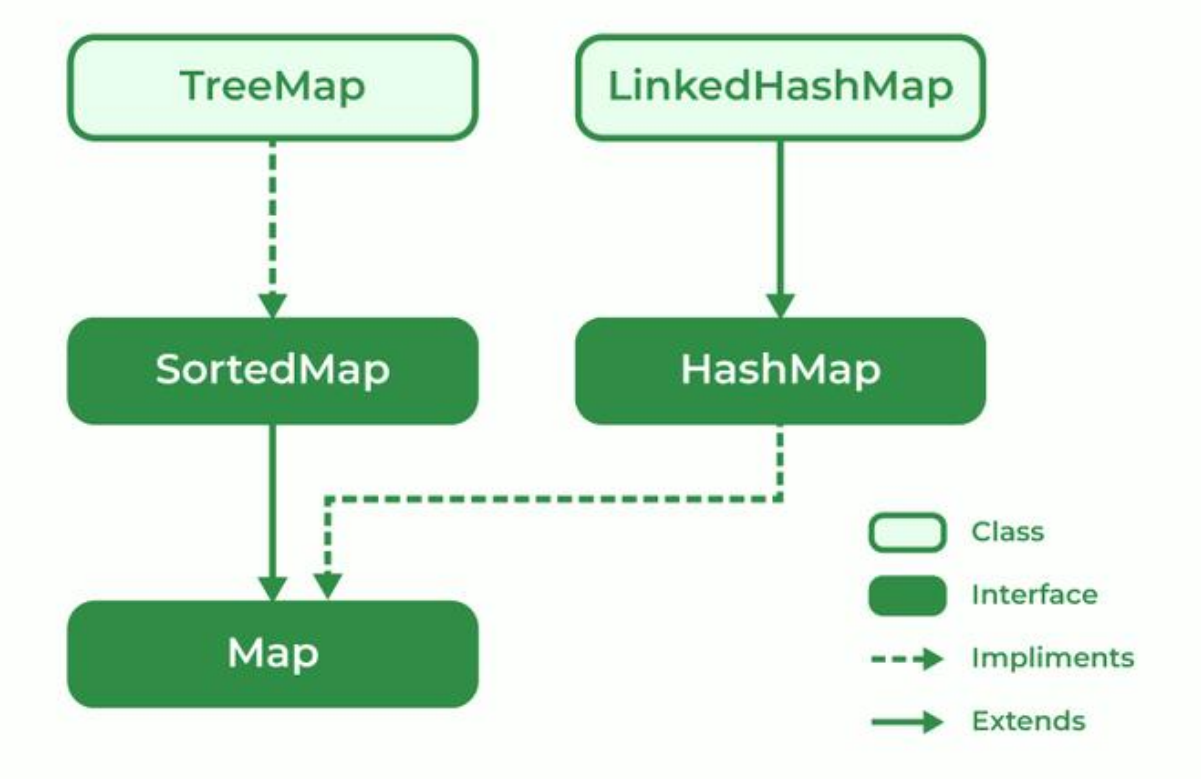
```java
// Java Program to implement

// LinkedHashSet

import java.io.*;

import java.util.*;

// Driver Class

class GFG {
```

```
    // Main Function
    public static void main(String[] args)
    {
        // LinkedHashSet declared
        LinkedHashSet<Integer>
            hs = new LinkedHashSet<Integer>();
        // Add elements in HashSet
        hs.add(1);
        hs.add(2);
        hs.add(5);
        hs.add(3);
        // Print values
        System.out.println("Values:" + hs);
    }
}
```

Output

```
Values:[1, 2, 5, 3]
```

133. What is a Map interface in Java?



The map interface is present in the Java collection and can be used with Java.util package. A map interface is used for mapping values in the form of a key-value form. The map contains all unique keys. Also, it provides methods associated with it like containsKey(), contains value (), etc.

There are multiple types of maps in the map interface as mentioned below:

1. SortedMap
2. TreeMap
3. HashMap
4. LinkedHashMap

134. Explain Treemap in Java

TreeMap is a type of map that stores data in the form of key-value pair. It is implemented using the red-black tree. Features of TreeMap are :

1. It contains only unique elements.
2. It cannot have a NULL key
3. It can have multiple NULL values.
4. It is non-synchronized.
5. It maintains ascending order.

135. What is EnumSet?

EnumSet is a specialized implementation of the Set interface for use with enumeration type. A few features of EnumSet are:

1. It is non-synchronized.
2. Faster than HashSet.
3. All of the elements in an EnumSet must come from a single enumeration type.

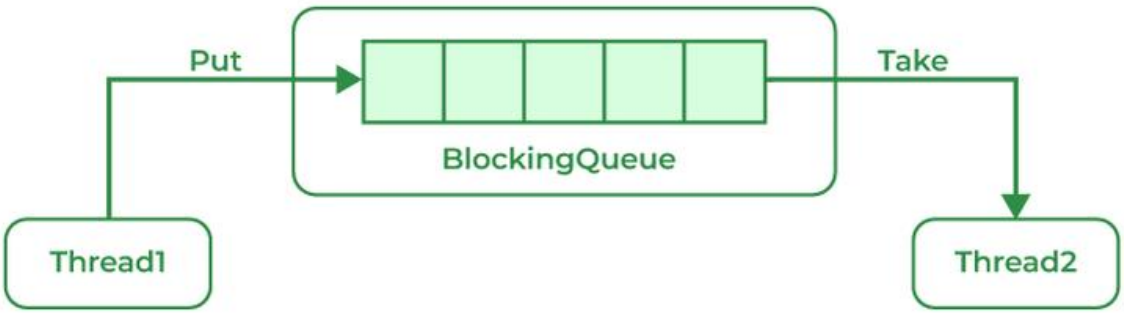
4. It doesn't allow null Objects and throws NullPointerException for exceptions.
5. It uses a fail-safe iterator.

Syntax:

```
public abstract class EnumSet<E extends Enum<E>>
```

Parameter: E specifies the elements.

136. What is BlockingQueue?



A blocking queue is a Queue that supports the operations that wait for the queue to become non-empty while retrieving and removing the element, and wait for space to become available in the queue while adding the element.

Syntax:

```
public interface BlockingQueue<E> extends Queue<E>
```

Parameters: E is the type of elements stored in the Collection

137. What is the ConcurrentHashMap in Java and do you implement it?

ConcurrentHashMap is implemented using Hashtable.

Syntax:

```
public class ConcurrentHashMap<K, V>
extends AbstractMap<K, V>
implements ConcurrentMap<K, V>, Serializable
```

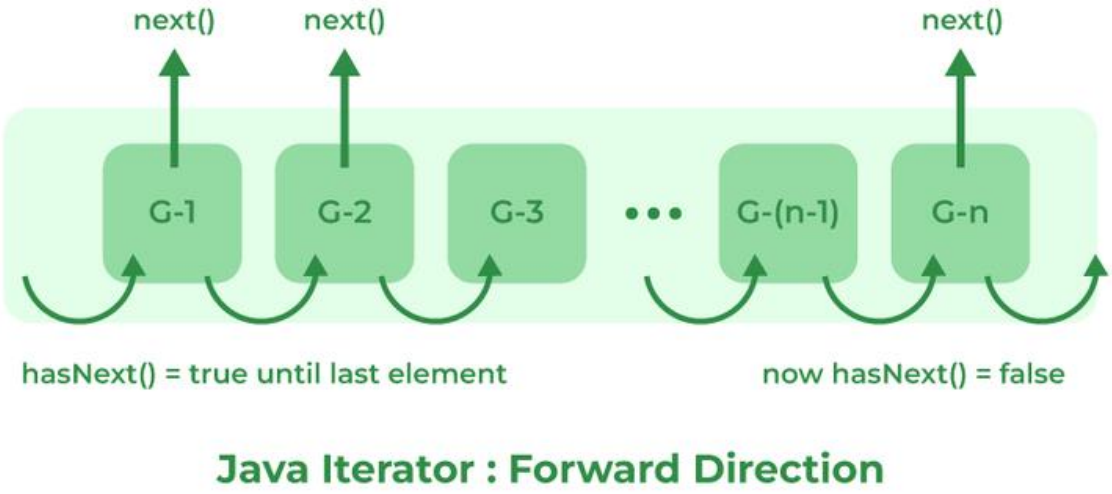
Parameters: K is the key Object type and V is the value Object type

138. Can you use any class as a Map key?

Yes, we can use any class as a Map Key if it follows certain predefined rules mentioned below:

1. The class overriding the equals() method must also override the hashCode() method
2. The concurrentHashMap class is thread-safe.
3. The default concurrency level of ConcurrentHashMap is 16.
4. Inserting null objects in ConcurrentHashMap is not possible as a key or as value.

139. What is an Iterator?



The Iterator interface provides methods to iterate over any Collection in Java. Iterator is the replacement of Enumeration in the Java Collections Framework. It can get an iterator instance from a Collection using the _iterator()_ method. It also allows the caller to remove elements from the underlying collection during the iteration.

140. What is an enumeration?

Enumeration is a user-defined data type. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain. The main objective of the enum is to define user-defined data types.

Example:

```
// A simple enum example where enum is declared
// outside any class (Note enum keyword instead of
// class keyword)
enum Color
{
    RED, GREEN, BLUE;
}
```

141. What is the difference between Collection and Collections?

Collection	**Collections**
The Collection is an Interface.	Collections is a class.
It provides the standard functionality of data structure.	It is to sort and synchronize the collection elements.
It provides the methods that can be used for the data structure.	It provides static methods that can be used for various operations.

142. Differentiate between Array and ArrayList in Java.

Array	ArrayList
Single-dimensional or multidimensional	Single-dimensional
For and for each used for iteration	Here iterator is used to traverse riverArrayList
length keyword returns the size of the array.	size() method is used to compute the size of ArrayList.
The array has Fixed-size.	ArrayList size is dynamic and can be increased or decreased in size when required.
It is faster as above we see it of fixed size	It is relatively slower because of its dynamic nature
Primitive data types can be stored directly in unlikely objects.	Primitive data types are not directly added to unlikely arrays, they are added indirectly with help of autoboxing and unboxing
They can not be added here hence the type is in the unsafe.	They can be added here hence makingArrayList type-safe.
The assignment operator only serves the purpose	Here a special method is used known as add() method

143. What is the difference between Array and Collection in Java?

Array	Collections
Array in Java has a fixed size.	Collections in Java have dynamic sizes.
In an Array, Elements are stored in contiguous memory locations.	In Collections, Elements are not necessarily stored in contiguous memory locations.
Objects and primitive data types can be stored in an array.	We can only store objects in collections.

Array	Collections
Manual manipulation is required for resizing the array.	Resizing in collections is handled automatically.
The array has basic methods for manipulation.	Collections have advanced methods for manipulation and iteration.
The array is available since the beginning of Java.	Collections were introduced in Java 1.2.

144. Difference between ArrayList and LinkedList.

ArrayList	LinkedList
ArrayList is Implemented as an expandable Array.	LinkedList is Implemented as a doubly-linked list.
In ArrayList, Elements are stored in contiguous memory locations	LinkedList Elements are stored in non-contiguous memory locations as each element has a reference to the next and previous elements.
ArrayLists are faster for random access.	LinkedLists are faster for insertion and deletion operations
ArrayLists are more memory efficient.	LinkedList is less memory efficient
ArrayLists Use more memory due to maintaining the array size.	LinkedList Uses less memory as it only has references to elements
The search operation is faster in ArrayList.	The search operation is slower in LinkedList

145. Differentiate between ArrayList and Vector in Java.

ArrayList	Vector
ArrayLists are implemented as an expandable array.	Vector is Implemented as a growable array.
ArrayList is not synchronized.	The vector is synchronized.

ArrayList	Vector
ArrayLists are Faster for non-concurrent operations.	Vector is Slower for non-concurrent operations due to added overhead of synchronization.
ArrayLists were Introduced in Java 1.2.	Vector was Introduced in JDK 1.0.
Recommended for use in a single-threaded environment.	Vectors are Recommended for use in a multi-threaded environment.
The default initial capacity of ArrayLists is 10.	In Vectors, the default initial capacity is 10 but the default increment is twice the size.
ArrayList performance is high.	Vector performance is low.

146. What is the difference between Iterator and ListIterator?

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Used to traverse Map, List, and Set.	Can only traverse List and not the other two.
Indexes can’t be obtained using Iterator	It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing the List.
Can’t modify or replace elements present in Collection	Can modify or replace elements with the help of set(E e)
Can’t add elements, and also throws ConcurrentModificationException.	Can easily add elements to a collection at any time.
Certain methods of Iterator are next(), remove(), and hasNext().	Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

147. Differentiate between HashMap and HashTable.

HashMap	HashTable
HashMap is not synchronized	HashTable is synchronized
One key can be a NULL value	NULL values not allowed
The iterator is used to traverse HashMap.	Both Iterator and Enumertar can be used
HashMap is faster.	HashTable is slower as compared to HashMap.

148. What is the difference between Iterator and Enumeration?

Iterator	Enumeration
The Iterator can traverse both legacies as well as non-legacy elements.	Enumeration can traverse only legacy elements.
The Iterator is fail-fast.	Enumeration is not fail-fast.
The Iterators are slower.	Enumeration is faster.
The Iterator can perform a remove operation while traversing the collection.	The Enumeration can perform only traverse operations on the collection.

149. What is the difference between Comparable and Comparator?

Comparable	Comparator
The interface is present in java.lang package.	The Interface is present in java.util package.
Provides compareTo() method to sort elements.	Provides compare() method to sort elements.
It provides single sorting sequences.	It provides multiple sorting sequences.
The logic of sorting must be in the same class whose object you are going to sort.	The logic of sorting should be in a separate class to write different sorting based on different attributes of objects.

Comparable	Comparator
Method sorts the data according to fixed sorting order.	Method sorts the data according to the customized sorting order.
It affects the original class.	It doesn’t affect the original class.
Implemented frequently in the API by Calendar, Wrapper classes, Date, and String.	It is implemented to sort instances of third-party classes.

150. What is the difference between Set and Map?

Set	Map
The Set interface is implemented using java.util package.	The map is implemented using java.util package.
It can extend the collection interface.	It does not extend the collection interface.
It does not allow duplicate values.	It allows duplicate values.
The set can sort only one null value.	The map can sort multiple null values.

Java Intermediate Interview Questions

151. Explain the FailFast iterator and FailSafe iterator along with examples for each.

A FailFast iterator is an iterator that throws a **ConcurrentModificationException** if it detects that the underlying collection has been modified while the iterator is being used. This is the default behavior of iterators in the Java Collections Framework. For example, the iterator for a HashMap is FailFast.

Example:

- Java

```
// Java Program to demonstrate FailFast iterator

import java.io.*;

import java.util.HashMap;

import java.util.Iterator;
```

```
import java.util.Map;
class GFG {
    public static void main(String[] args)
    {
        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "one");
        map.put(2, "two");
        Iterator<Map.Entry<Integer, String> > iterator
            = map.entrySet().iterator();
        while (iterator.hasNext()) {
            Map.Entry<Integer, String> entry
                = iterator.next();
            // this will throw a
            // ConcurrentModificationException
            if (entry.getKey() == 1) {
                map.remove(1);
            }
        }
    }
}
```

Output:

```
Exception in thread "main" java.util.ConcurrentModificationException
```

A FailSafe iterator does not throw a **ConcurrentModificationException** if the underlying collection is modified while the iterator is being used. Alternatively, it creates a snapshot of the collection at the time the iterator is created and iterates over the snapshot. For example, the iterator for a ConcurrentHashMap is FailSafe.

Example:

- Java

```
// Java Program to demonstrate FailSafe
```

```java
import java.io.*;
import java.util.Iterator;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
class GFG {
    public static void main(String[] args)
    {
        ConcurrentHashMap<Integer, String> map
            = new ConcurrentHashMap<>();
        map.put(1, "one");
        map.put(2, "two");
        Iterator<Map.Entry<Integer, String> > iterator
            = map.entrySet().iterator();
        while (iterator.hasNext()) {
            Map.Entry<Integer, String> entry = iterator.next();
            // this will not throw an exception
            if (entry.getKey() == 1) {
                map.remove(1);
            }
        }
    }
}
```

152. What is Exception Handling?

An Exception is an Event that interrupts the normal flow of the program and requires special processing. During the execution of a program, errors and unplanned occurrences can be dealt with by using the Java Exception Handling mechanism. Below are some reasons why Exceptions occur in Java:

- Device failure
- Loss of Network Connection
- Code Errors
- Opening an Unavailable file
- Invalid User Input
- Physical Limitations (out of disk memory)

153. How many types of exceptions can occur in a Java program?

There are generally two types of exceptions in Java:

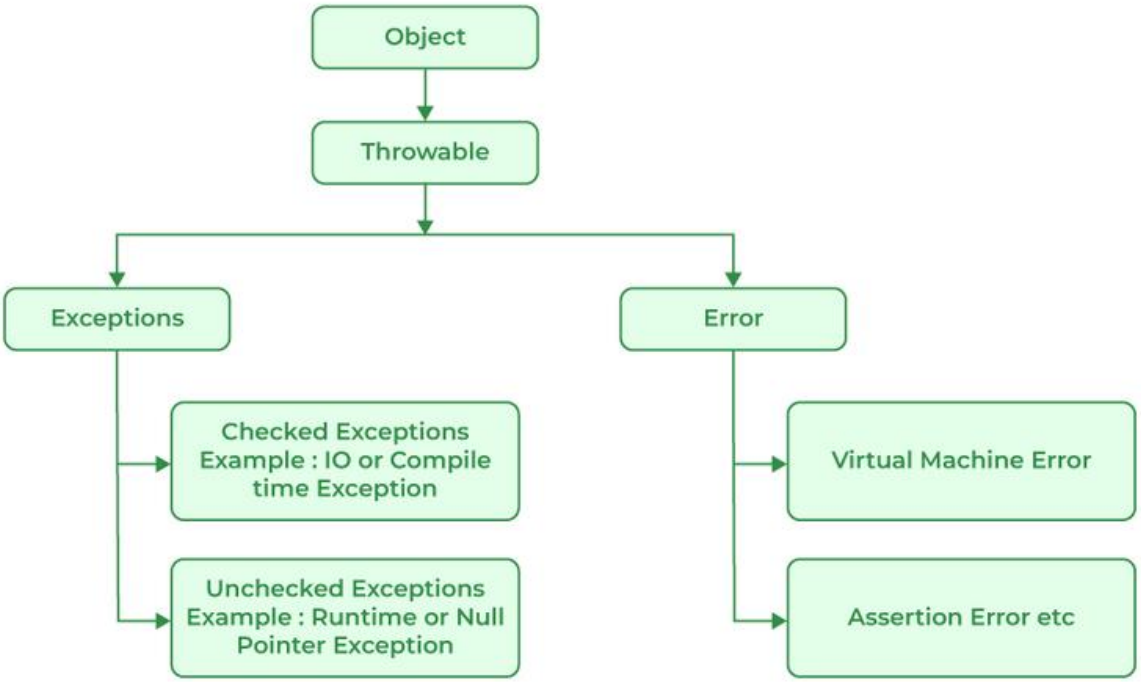
- **Built-in Exceptions:** Built-in exceptions in Java are provided by the Java Libraries. These exceptions can be further divided into two subcategories i.e., checked and unchecked Exceptions. Below are some of the built-in exceptions in Java:
  - ArrayIndexOutOfBoundsExceptions
  - ClassNotFoundException
  - FileNotFoundException
  - IOException
  - NullPointerException
  - ArithmeticException
  - InterruptedException
  - RuntimeException
- **User-Defined Exceptions:** User-defined exceptions are defined by the programmers themselves to handle some specific situations or errors which are not covered by built-in exceptions. To define user-defined exceptions a new class that extends the appropriate exception class must be defined. User-defined Exceptions in Java are used when the built-in exceptions are in Java.

154. Difference between an Error and an Exception.

Errors	**Exceptions**
Recovering from Errors is not possible.	Recover from exceptions by either using a try-catch block or throwing exceptions back to the caller.
Errors are all unchecked types in Java.	It includes both checked as well as unchecked types that occur.
Errors are mostly caused by the environment in which the program is running.	The program is mostly responsible for causing exceptions.

Errors	Exceptions
Errors can occur at compile time as well as run time. Compile Time: Syntax Error, Run Time: Logical Error.	All exceptions occur at runtime but checked exceptions are known to the compiler while unchecked are not.
They are defined in java.lang.Error package. **Examples**: java.lang.StackOverflowError, java.lang.OutOfMemoryError	They are defined in java.lang.Exception package **Examples**: Checked Exceptions: SQLException, IOException Unchecked Exceptions: ArrayIndexOutOfBoundException, NullPointerException, ArithmeticException.

155. Explain the hierarchy of Java Exception classes.



All exception and error types in Java are subclasses of the class throwable, which is the base class of the hierarchy. This class is then used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, error is used by the Java run-time system to indicate errors having to do with the JRE. StackOverflowError is an example of one of such error.

156. Explain Runtime Exceptions.

Runtime Exceptions are exceptions that occur during the execution of a code, as opposed to compile-time exceptions that occur during compilation. Runtime exceptions are unchecked exceptions, as they aren’t accounted for by the JVM.

Examples of runtime exceptions in Java include:

- NullPointerException: This occurs when an application attempts to use a null object reference.
- ArrayIndexOutOfBoundsException: This occurs when an application attempts to access an array index that is out of bounds.
- ArithmeticException: This occurs when an application attempts to divide by zero.
- IllegalArgumentException: This occurs when a method is passed on an illegal or inappropriate argument.

Unlike checked exceptions, runtime exceptions do not require a declaration in the throws clause or capture in a try-catch block. However, handling runtime exceptions is advisable in order to provide meaningful error messages and prevent a system crash. Because runtime exceptions provide more specific information about the problem than checked exceptions, they enable developers to detect and correct programming errors more easily and quickly.

157. What is NullPointerException?

It is a type of run-time exception that is thrown when the program attempts to use an object reference that has a null value. The main use of NullPointerException is to indicate that no value is assigned to a reference variable, also it is used for implementing data structures like linked lists and trees.

158. When is the ArrayStoreException thrown?

ArrayStoreException is thrown when an attempt is made to store the wrong type of object in an array of objects.

Example:

- Java

```java
// Java Program to implement
// ArrayStoreException
public class GFG {
    public static void main(String args[])
    {
        // Since Double class extends Number class
        // only Double type numbers
        // can be stored in this array
        Number[] a = new Double[2];
        // Trying to store an integer value
        // in this Double type array
```

```
        a[0] = new Integer(4);

    }

}
```

Example:

```
Exception in thread "main" java.lang.ArrayStoreException:
java.lang.Integer
    at GFG.main(GFG.java:6)
```

159. What is the difference between Checked Exception and Unchecked Exception?

Checked Exception:

Checked Exceptions are the exceptions that are checked during compile time of a program. In a program, if some code within a method throws a checked exception, then the method must either handle the exception or must specify the exception using the throws keyword.

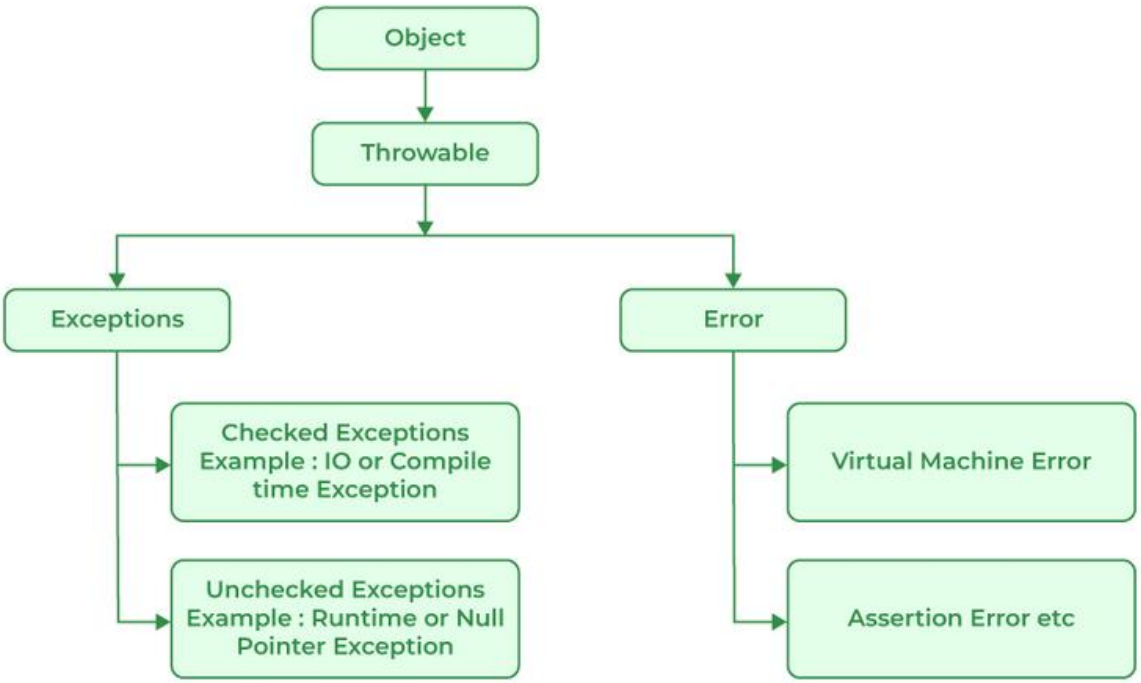
Checked exceptions are of two types:

- Fully checked exceptions: all its child classes are also checked, like IOException, and InterruptedException.
- Partially checked exceptions: some of its child classes are unchecked, like an Exception.

Unchecked Exception:

Unchecked are the exceptions that are not checked at compile time of a program. Exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.

160. What is the base class for Error and Exception?



Error is an illegal operation performed by the user which causes abnormality in the program. Exceptions are the unexpected events or conditions that comes while running the program, exception disrupts the normal flow of the program's instructions.
Errors and Exceptions both have a common parent class which is java.lang.Throwable class.

161. Is it necessary that each try block must be followed by a catch block?

No, It is not necessary to use catch block after try block in Java as we can create another combination with finally block. Finally is the block which runs despite the fact that the exception is thrown or not.

162. What is exception propagation?

Exception propagation is a process in which the exception is dropped from to the top to the bottom of the stack. If not caught once, the exception again drops down to the previous method, and so on until it gets caught or until it reaches the very bottom of the call stack.

163. What will happen if you put System.exit(0) on the try or catch block? Will finally block execute?

System.exit(int) has the capability to throw SecurityException. So, if in case of security, the exception is thrown then finally block will be executed otherwise JVM will be closed while calling System. exit(0) because of which finally block will not be executed.

164. What do you understand by Object Cloning and how do you achieve it in Java?

It is the process of creating an exact copy of any object. In order to support this, a java class has to implement the Cloneable interface of java.lang package and override the clone() method provided by the Object class the syntax of which is:
Protected Object clone() throws CloneNotSupportedException{ return (Object)super.clone();}In case the Cloneable interface is not implemented and just the method is overridden, it results in CloneNotSupportedException in Java.

165. How do exceptions affect the program if it doesn't handle them?

Exceptions are responsible for abruptly terminating the running of the program while executing and the code written after the exception occurs is not executed.

166. What is the use of the final keyword?

The final keyword is used to make functions non-virtual. By default, all the functions are virtual so to make it non-virtual we use the final keyword.

167. What purpose do the keywords final, finally, and finalize fulfill?

i). final:
final is a keyword is used with the variable, method, or class so that they can't be overridden.
Example:

- Java

```java
// Java Program to use final

// keyword

import java.io.*;

// Driver Class

class GFG {

    // Main function

    public static void main(String[] args)

    {

        final int x = 100;

        x = 50;

    }

}
```

Output:

```
./GFG.java:6: error: cannot assign a value to final variable x
          x=50;
          ^
1 error
```

ii). finally

finally is a block of code used with “try-catch” in exception handling. Code written in finally block runs despite the fact exception is thrown or not.

Example:

- Java

```java
// Java Program to implement finally

import java.io.*;

// Driver class

class GFG {

    // Main function

    public static void main(String[] args)

    {

        int x = 10;
```

```
        // try block
        try {
            System.out.println("Try block");
        }
        // finally block
        finally {
            System.out.println(
                "Always runs even without exceptions");
        }
    }
}
```

Output

```
Try block
Always runs even without exceptions
```

iii). finalize

It is a method that is called just before deleting/destructing the objects which are eligible for Garbage collection to perform clean-up activity.

Example:

- Java

```
/*package whatever // do not write package name here */
```

```java
import java.io.*;
class GFG {
    public static void main(String[] args)
    {
        System.out.println("Main function running");
        System.gc();
    }
    // Here overriding finalize method
    public void finalize()
    {
        System.out.println("finalize method overridden");
    }
}
```

Output

```
Main function running
```

168. What is the difference between this() and super() in Java?

this( )	**super( )**
It represents the current instance of the class.	It represents the current instance of the parent class.
Calls the default constructor of the same class.	Calls the default constructor of the base class.

this( )	super( )
Access the methods of the same class.	Access the methods of the parent class.
Points current class instance.	Points the superclass instance.

169. What is multitasking?

Multitasking in Java refers to a program's capacity to carry out several tasks at once. Threads, which are quick operations contained within a single program, can do this. Executing numerous things at once is known as multitasking.

Example:

- Java

```java
// Java program for multitasking
import java.io.*;
public class MyThread extends Thread {
    public void run()
    {
        // Code to be executed in this thread
        for (int i = 0; i < 10; i++) {
            System.out.println(
                "Thread " + Thread.currentThread().getId()
                + ": " + i);
        }
    }
}
public class GFG {
    public static void main(String[] args)
    {
        MyThread thread1 = new MyThread();
        MyThread thread2 = new MyThread();
        // Start the threads
```

```
        thread1.start();

        thread2.start();

    }

}
```

170. What do you mean by a Multithreaded program?

Multithreaded programs in Java contain threads that run concurrently instead of running sequentially. A computer can use its resources more efficiently by combining multiple tasks at once. Any program with multithreading allows more than one user to simultaneously use the program without running multiple copies. A multithreaded program is designed to run multiple processes at the same time which can improve the performance of a program and allows the program to utilize multiple processors and improves the overall throughput.

171. What are the advantages of multithreading?

There are multiple advantages of using multithreading which are as follows:

- Responsiveness: User Responsiveness increases because multithreading interactive application allows running code even when the section is blocked or executes a lengthy process.
- Resource Sharing: The process can perform message passing and shared memory because of multithreading.
- Economy: We are able to share memory because of which the processes are economical.
- Scalability: Multithreading on multiple CPU machines increases parallelism.
- Better Communication: Thread synchronization functions improves inter-process communication.
- Utilization of multiprocessor architecture
- Minimized system resource use

172. What are the two ways in which Thread can be created?

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of the CPU. In general, threads are small, lightweight processes with separate paths of execution. These threads use shared memory, but they act independently, thus if any one thread fails it does not affect the other threads. There are two ways to create a thread:

- By extending the Thread class
- By implementing a Runnable interface.

By extending the Thread class

We create a class that extends the **java.lang.Thread class**. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method.

Syntax:

```
public class MyThread extends Thread {
public void run() {
        // thread code goes here
```

```
    }
}
```

By implementing the Runnable interface

We create a new class that implements **java.lang.Runnable** interface and override run() method. Then we instantiate a Thread object and call the start() method on this object.

Syntax:

```
public class MyRunnable implements Runnable {
public void run() {
        // thread code goes here
    }
}
```

173. What is a thread?

Threads in Java are subprocess with lightweight with the smallest unit of processes and also has separate paths of execution. These threads use shared memory but they act independently hence if there is an exception in threads that do not affect the working of other threads despite them sharing the same memory. A thread has its own program counter, execution stack, and local variables, but it shares the same memory space with other threads in the same process. Java provides built-in support for multithreading through the **Runnable interface** and the **Thread class**.

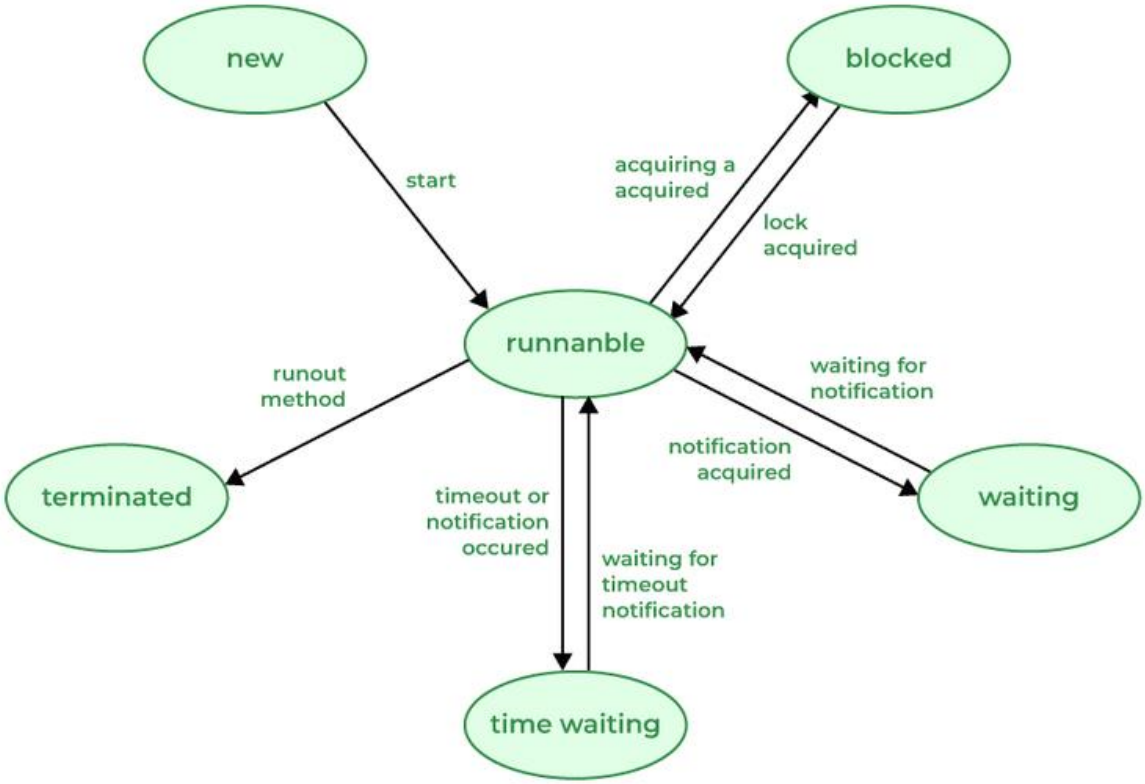
174. Differentiate between process and thread?

A process and a thread are both units of execution in a computer system, but they are different in several ways:

Process	**Thread**
A process is a program in execution.	A thread is a single sequence of instructions within a process.
The process takes more time to terminate.	The thread takes less time to terminate.
The process takes more time for context switching.	The thread takes less time for context switching.
The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
The process is isolated.	Threads share memory.
The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.

Process	Thread
The process does not share data with each other.	Threads share data with each other.

175. Describe the life cycle of the thread?



A thread in Java at any point in time exists in any one of the following states. A thread lies only in one of the shown states at any instant:

1. **New:** The thread has been created but has not yet started.
2. **Runnable:** The thread is running, executing its task, or is ready to run if there are no other higher-priority threads.
3. **Blocked:** The thread is temporarily suspended, waiting for a resource or an event.
4. **Waiting:** The thread is waiting for another thread to perform a task or for a specified amount of time to elapse.
5. **Terminated:** The thread has completed its task or been terminated by another thread.

176. Explain suspend() method under the Thread class.

The suspend() method of the Thread class in Java temporarily suspends the execution of a thread. When a thread is suspended it goes into a blocked state and it would not be scheduled by the operating system which means that it will not be able to execute its task until it is resumed. There are more safer and flexible alternatives to the suspend() methods in the modern java programming language. This method does not return any value.

Syntax:

```
public final void suspend();
```

Example:

- Java

```
// Java program to show thread suspend() method
import java.io.*;
class MyThread extends Thread {
    public void run()
    {
        for (int i = 0; i < 10; i++) {
            System.out.println(" Running thread : " + i);
            try {
                Thread.sleep(1000);
            }
            catch (Interrupted_Exception e) {
                e.printStackTrace();
            }
        }
    }
}
class GFG {
    public static void main(String[] args)
    {
        MyThread t1 = new MyThread();
        t1.start();
        try {
            Thread.sleep(3000);
        }
        catch (InterruptedException e) {
```

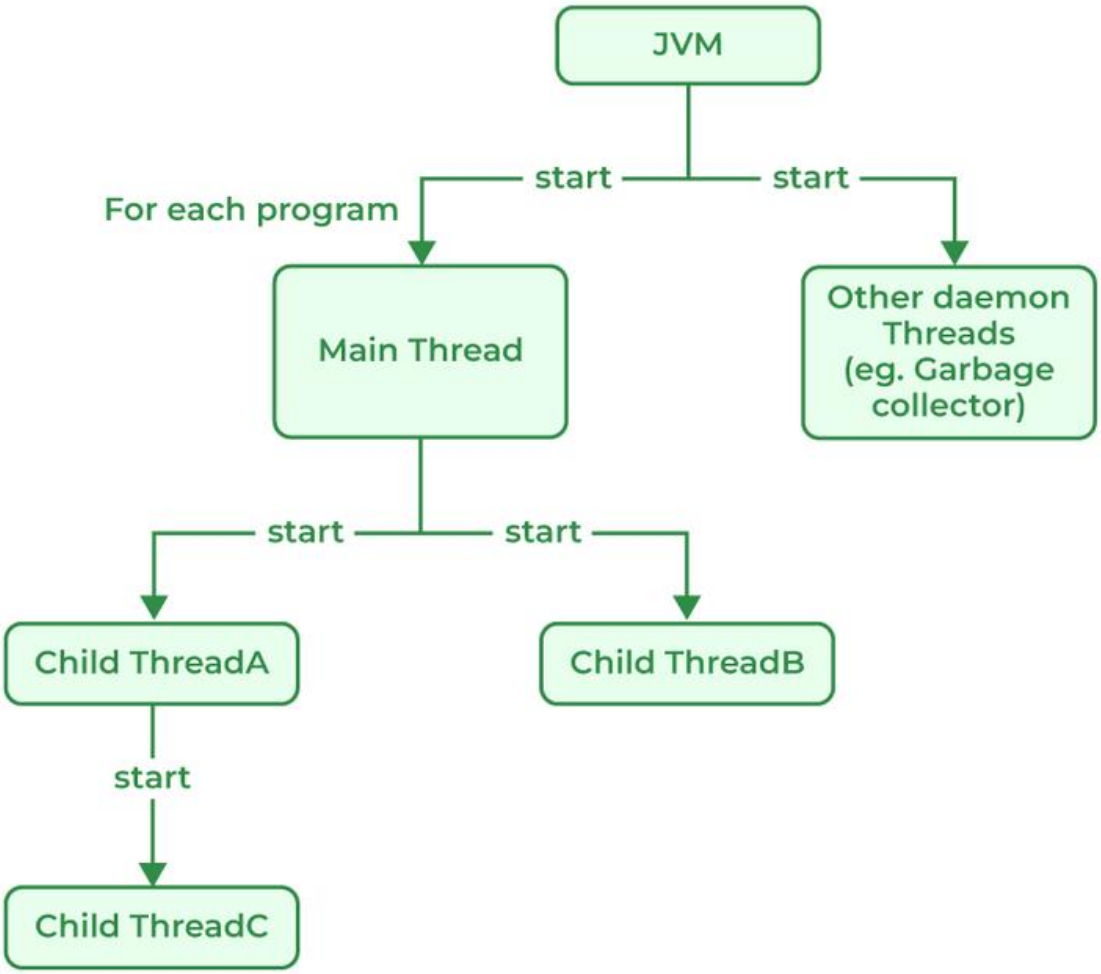
```
            e.printStackTrace();
        }
        // suspend the execution of the thread
        t1.suspend();
        System.out.println("Suspended thread ");
        try {
            Thread.sleep(3000);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
        // resume the execution of the thread
        t1.resume();
        System.out.println("Resumed thread");
    }
}
```

Output:

```
Thread running: 0
Thread running: 1
Thread running: 2
Suspended thread
Resumed thread
Thread running: 3
Thread running: 4
Thread running: 5
Thread running: 6
Thread running: 7
Thread running: 8
Thread running: 9
```

177. Explain the main thread under Thread class execution.

Java provides built-in support for multithreaded programming. The main thread is considered the parent thread of all the other threads that are created during the program execution. The main thread is automatically created when the program starts running. This thread executes the main method of the program. It is responsible for executing the main logic of the Java program as well as handling the user input operations. The main thread serves as the base thread from which all other child threads are spawned.



178. What is a daemon thread?

A daemon thread in Java is a low-priority thread that is used to perform background operations or tasks which are used to perform continuously. such as Garbage collection, Signal dispatches, Action listeners, etc. Daemon threads in Java have lower priority than user threads, which means they can only execute when no user threads are running. Daemon threads in Java are useful features that are required for background tasks that do not require explicit shutdown or finalization. It allows more efficient use of system resource and are used to simplify resources and can simplify long-running tasks.

179. What are the ways in which a thread can enter the waiting state?

Thread is a lightweight process that runs concurrently with the other thread inside a single process. Each thread can execute a different task and share the resources within a single process. Thread in Java can enter the waiting state in many different ways:

- **Sleep() method Call: The sleep** () method is used to pause the execution of the thread for a specific amount of time. While the thread is paused it goes into the waiting state.
- **Wait() method:** This method is used to wait a thread until the other thread signals it to wake up. Thread goes into the waiting state until it receives a notification from another thread.

- **Join() method:** Join() method can be used to wait for thread to finish the execution. Calling thread goes into the waiting state until the target thread is completed.
- **Waiting for I/O operations:** If the thread is waiting for Input/Output operation to complete, it goes into the waiting state until the operation is finished.
- **Synchronization Issues:** If there are any synchronization issues in a multi-threaded application, threads may go into the waiting state until the synchronization issues are resolved.

180. How does multi-threading take place on a computer with a single CPU?

Java uses a technique called time-sharing, commonly referred to as time-slicing, to implement multi-threading on computers with a single CPU. The appearance of parallel execution is created by the CPU switching between active threads. The operating system is in charge of allocating CPU time to each thread sequentially and scheduling the threads.

In order to stop threads from interacting with one another and creating race situations or other issues, Java has a number of ways to govern the behavior of threads, including synchronization and locking. It is feasible to create multi-threaded programmers that operate correctly and effectively on a machine with a single CPU by regulating the interaction between threads and making sure that crucial code parts are synchronized.

In contrast to running the same program on a computer with multiple CPUs or cores, multi-threading on a single CPU can only give the appearance of parallelism, and actual performance gains may be modest. The operating system divides the CPU time that is available when numerous threads are running on a single CPU into small time slices and gives each thread a time slice to execute. Rapid switching between the threads by the operating system creates the appearance of parallel execution. The switching between threads appears to be immediate because the time slices are often very tiny, on the order of milliseconds or microseconds.

Java Interview Questions For Experienced

181. What are the different types of Thread Priorities in Java? And what is the default priority of a thread assigned by JVM?

Priorities in threads is a concept where every thread is having a priority which in layman's language one can say every object is having priority here which is represented by numbers ranging from 1 to 10. There are different types of thread properties in Java mentioned below:

- MIN_PRIORITY
- MAX_PRIORITY
- NORM_PRIORITY

By default, the thread is assigned NORM_PRIORITY.

182. Why Garbage Collection is necessary in Java?

For Java, Garbage collection is necessary to avoid memory leaks which can cause the program to crash and become unstable. There is no way to avoid garbage collection in Java. Unlike C++, Garbage collection in Java helps programmers to focus on the development of the application instead of managing memory resources and worrying about memory leakage. Java Virtual Machine (JVM) automatically manages the

memory periodically by running a garbage collector which frees up the unused memory in the application. Garbage collection makes Java memory efficient because it removes unreferenced objects from the heap memory.

183. What is the drawback of Garbage Collection?

Apart from many advantages, Garbage Collector has certain drawbacks mentioned below:

1. The main drawback to Garbage collection is that it can cause pauses in an application's execution as it works to clear the memory which slows down the performance of the application.
2. The Process of Garbage collection is non-deterministic which makes it difficult to predict when garbage collection occurs which causes unpredictable behavior in applications. For Example, if we write any program then it is hard for programmers to decide if the issue is caused by garbage collection or by any other factors in the program.
3. Garbage collection can also increase memory usage if the program creates and discards a lot of short-lived objects.

184. Explain the difference between a minor, major, and full garbage collection.

The Java Virtual Machine (JVM) removes objects that are no longer in use using a garbage collector which periodically checks and removes these objects. There are different types of garbage collection in the JVM, each with different characteristics and performance implications. The main types of garbage collection are:

- **Minor garbage collection:** Also known as young generation garbage collection, this type of garbage collection is used to collect and reclaim memory that is used by short-lived objects (objects that are quickly created and discarded).
- **Major garbage collection:** Also known as old-generation garbage collection, this type of garbage collection is used to collect and reclaim memory that is used by long-lived objects (objects that survive multiple minor garbage collections and are promoted to the old generation).
- **Full garbage collection:** During full garbage collection, memories from all generations are collected and reclaimed, including memories of young and old. A full garbage collection normally takes longer to complete than a minor or major garbage collection which causes that app to pause temporarily.

185. How will you identify major and minor garbage collections in Java?

Major garbage collection works on the survivor space and Minor garbage collection works on the Eden space to perform a mark-and-sweep routine. And we can identify both of them based on the output where the minor collection prints "GC", whereas the major collection prints "Full GC" for the case where the garbage collection logging is enabled with "-XX:PrintGCDetails" or "verbose:gc".

186. What is a memory leak, and how does it affect garbage collection?

In Java Memory leaks can be caused by a variety of factors, such as not closing resources properly, holding onto object references longer than necessary, or creating too many objects unnecessarily. There are situations in which garbage collector does not collect objects because there is a reference to those objects. In these situations where the application creates lots of objects and does not use them and every object

has some valid references, a Garbage collector in Java cannot destroy the objects. These useless objects which do not provide any value to the program are known as Memory leaks. Memory leaks can impact garbage collection negatively by preventing the garbage collector from reclaiming unused memory. This behavior will lead to slow performance or sometimes system failure. In a program, it is important to avoid memory leaks by managing resources and object references properly.

Example:

- Java

```java
// Java Program to demonstrate memory leaks

import java.io.*;

import java.util.Vector;

class GFG {

    public static void main(String[] args)

    {

        Vector a = new Vector(21312312);

        Vector b = new Vector(2147412344);

        Vector c = new Vector(219944);

        System.out.println("Memory Leak in Java");

    }

}
```

Output:

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
    at java.base/java.util.Vector.<init>(Vector.java:142)
    at java.base/java.util.Vector.<init>(Vector.java:155)
    at GFG.main(GFG.java:9)
```

187. Name some classes present in java.util.regex package.

Regular Expressions or Regex in Java is an API used for searching and manipulating of strings in Java. It creates String patterns that can extract the data needed from the strings or can generalize a pattern.

There are 3 Classes present in java.util.regex mentioned below:

- Pattern Class: Can define patterns
- Matcher Class: Can perform match operations on text using patterns
- PatternSyntaxException Class: Can indicate a syntax error in a regular expression pattern.

Also, apart from the 3 classes package consists of a single interface MatchResult Interface which can be used for representing the result of a match operation.

188. Write a regular expression to validate a password. A password must start with an alphabet and followed by alphanumeric characters; Its length must be in between 8 to 20.

```
regex = "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&-+=()])(?=\\S+$).{8, 20}$"
```

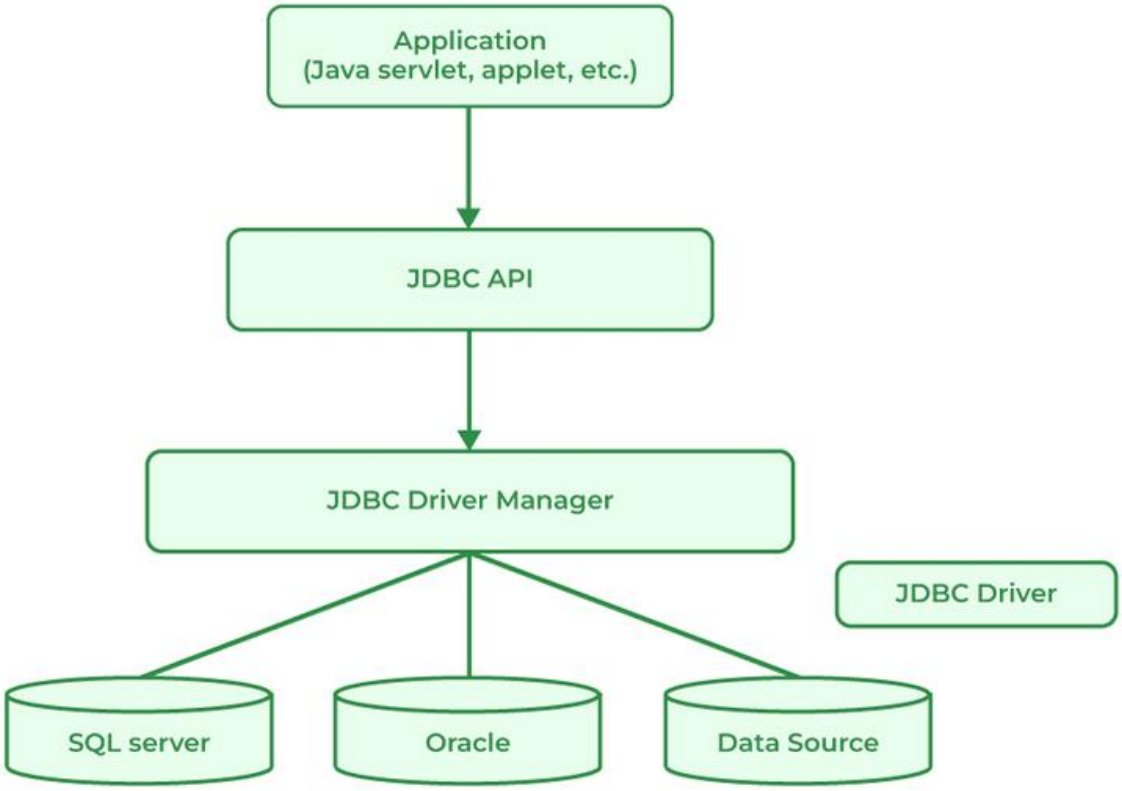
Explanation:

- ^ used for starting character of the string.
- (?=.*[0-9]) used for a digit must occur at least once.
- (?=.*[a-z]) used for a lowercase alphabet must occur at least once.
- (?=.*[A-Z]) used for an upper case alphabet that must occur at least once in the substring.
- (?=.*[@#$%^&-+=()] used for a special character that must occur at least once.
- (?=\\S+$) white spaces don't allow in the entire string.
- .{8, 20} used for at least 8 characters and at most 20 characters.
- $ used for the end of the string.

189. What is JDBC?

JDBC standard API is used to link Java applications and relational databases. It provides a collection of classes and interfaces that let programmers to use the Java programming language to communicate with the database. The classes and interface of JDBC allow the application to send requests which are made by users to the specified database. There are generally four components of JDBC by which it interacts with the database:

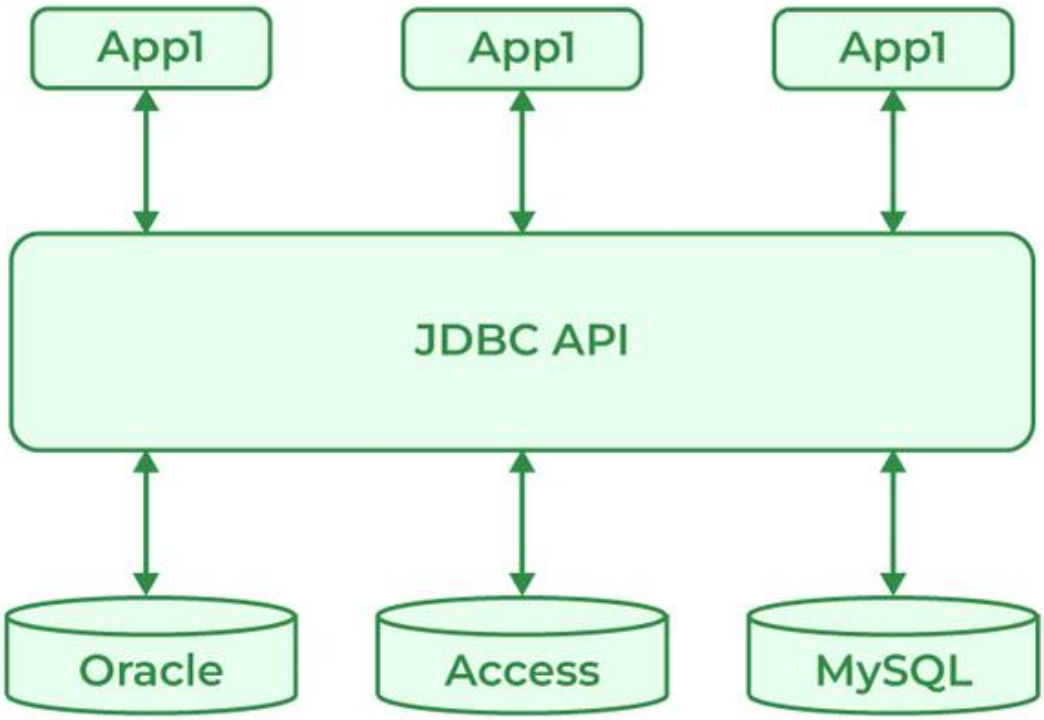
- JDBC API
- JDBC Driver manager
- JDBC Test Suite
- JDBC-ODBC Bridge Drivers



190. What is JDBC Driver?

JDBC Driver is a software component that is used to enable a Java application to interact with the database. JDBC provides the implementation of the JDBC API for a specific database management system, which allows it to connect the database, execute SQL statements and retrieve data. There are four types of JDBC drivers:

- JDBC-ODBC Bridge driver
- Native-API driver
- Network Protocol driver
- Thin driver



191. What are the steps to connect to the database in Java?

There are certain steps to connect the database and Java Program as mentioned below:

- Import the Packages
- Load the drivers using the forName() method
- Register the drivers using DriverManager
- Establish a connection using the Connection class object
- Create a statement
- Execute the query
- Close the connections

192. What are the JDBC API components?

JDBC API components provide various methods and interfaces for easy communication with the databases also it provides packages like java Se and java EE which provides the capability of write once run anywhere (WORA).

Syntax:

```
java.sql.*;
```

193. What is JDBC Connection interface?

Java database connectivity interface (JDBC) is a software component that allows Java applications to interact with databases. To enhance the connection, JDBC requires drivers for each database.

194. What does the JDBC ResultSet interface?

JDBC ResultSet interface is used to store the data from the database and use it in our Java Program. We can also use ResultSet to update the data using updateXXX()

methods. ResultSet object points the cursor before the first row of the result data. Using the next() method, we can iterate through the ResultSet.

195. What is the JDBC Rowset?

A JDBC RowSet provides a way to store the data in tabular form. RowSet is an interface in java that can be used within the java.sql package. The connection between the RowSet object and the data source is maintained throughout its life cycle. RowSets are classified into five categories based on implementation mentioned below:

1. JdbcRowSet
2. CachedRowSet
3. WebRowSet
4. FilteredRowSet
5. JoinRowSet

196. What is the role of the JDBC DriverManager class?

JDBC DriverManager class acts as an interface for users and Drivers. It is used in many ways as mentioned below:

- It is used to create a connection between a Java application and the database.
- Helps to keep track of the drivers that are available.
- It can help to establish a connection between a database and the appropriate drivers.
- It contains all the methods that can register and deregister the database driver classes.
- DriverManager.registerDriver() method can maintain the list of Driver classes that have registered themselves.

Java Difference Interview Questions

197. Differentiate between Iterable and Iterator.

Iterable	**Iterator**
Iterable provides a way to iterate over a sequence of elements.	Iterator helps in iterating over a collection of elements sequentially.
iterator() method returns an Iterator.	**hasNext()** and **next()** methods are required.
remove() method is optional.	**remove()** method is required in the iterator.
Examples are **List, Queue, and Set.**	Examples are **ListIterator, Enumeration, and ArrayIterator.**

198. Differentiate between List and Set.

List	Set
Ordered	Unordered
List allows duplicates.	Set does not allow duplicate values.
List is accessed by index.	Set is accessed by hashcode.
Multiple null elements can be stored.	Null element can store only once.
Examples are ArrayList, LinkedList, etc.	Examples are HashSet and TreeSet. LinkedHashSet etc.

199. Differentiate between List and Map.

List	Map
List interface allows duplicate elements.	Map does not allow duplicate elements.
The list maintains insertion order.	The list maintains insertion order.
Multiple null elements can be stored.	The map allows a single null key at most and any number of null values.
The list provides get() method to get the element at a specified index.	The map does not provide a get method to get the elements at a specified index.
List is Implemented by ArrayList, etc.	Map is Implemented by HashMap, TreeMap, LinkedHashMap

200. Differentiate between Queue and Stack.

Queue	Stack
Queue data structure is used to store elements, and is used to perform operations like enqueue, dequeue from back or end of the queue.	Stack data structure is used to store elements, and is used to perform operations like push, pop from top of the stack.
Queue data structure Implements FIFO order.	Stack data structure Implements LIFO order.

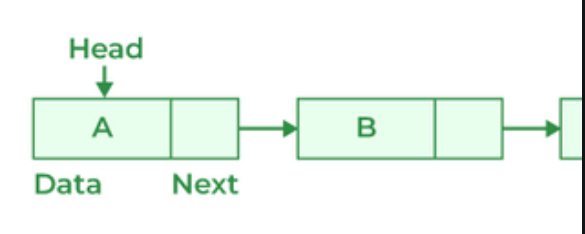
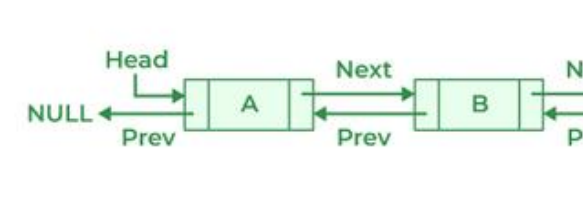
Queue	Stack
Insertion and deletion in queues take place from the opposite ends of the list. Deletion takes place from the front of the list and insertion takes place at the rear of the list.	Insertion and deletion in stacks take place only from one end of the list called the top.
Insert operation is called enqueue operation.	Insert operation is called Push operation.
Queue is generally used to solve problems related to sequential processing.	Stack is generally used to solve problems related to recursion.

201. Differentiate between PriorityQueue and TreeSet.

Priority Queue	TreeSet
It uses Queue as an underlying data structure.	It uses a Set as an underlying data structure.
This data structure allows duplicate elements	This data structure does not allow duplicate elements
Priority Queue is Implemented by PriorityQueue class.	TreeSet is implemented by TreeSet class.
PriorityQueue comes in JDK 1.5.	TreeSet comes in JDK 1.4.
PriorityQueue<Integer> pq = new PriorityQueue<>();	reeSet<Integer> ts = new TreeSet<>();

202. Differentiate between the Singly Linked List and Doubly Linked List.

Singly Linked List	Doubly Linked List
Singly Linked List contain only two segments i.e, Data and Link.	Doubly Linked List contains three segments i.e, Data, and two pointers.
Traversal in a singly linked list is possible in only a forward direction.	Traversal in a doubly linked list is only possible in both directions forward as well as backward.

Singly Linked List	Doubly Linked List
It uses less memory as every single node has only one pointer.	It requires more memory than a singly linked list as each node has two pointers.
Easy to use and insert nodes at the beginning of the list.	Slightly more complex to use and easy to insert at the end of the list.
The time complexity of insertion and deletion is O(n).	The time complexity of insertion and deletion is O(1).
	

203. Differentiate between Failfast and Failsafe.

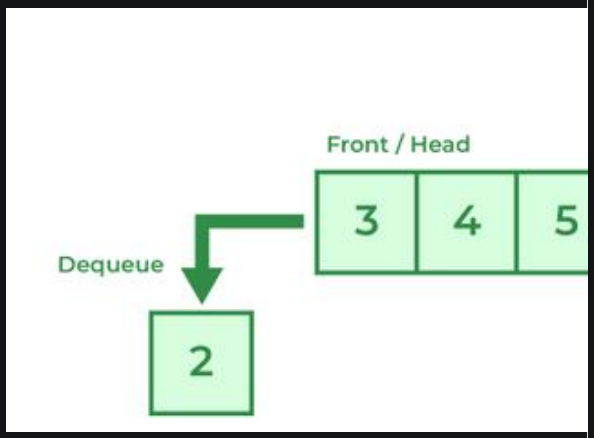
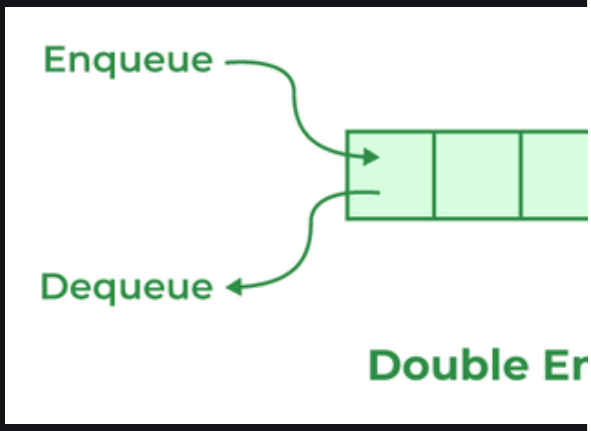
FailFast	FailSafe
Failsfast fails immediately when it detects concurrent modification during the time of iteration.	Failsafe continues to iterate over the original collection and also creates a copy to modify.
Failfast is generally used in single-threaded environments.	Failsafe is used in multithreaded environments.
Failfast does not allow any modification while iteration.	Failsafe allows modification during the time of iteration.
Failfast is fast compared to failsafe as it does not involve the copying of the collection.	Failsafe is generally slow compared to failfast.
FailFast throws **ConcurrentModificationException** if the collection is modified during iteration.	FailSafe does not throws any exception but instead, it creates a copy of the collection to iterate.

204. Differentiate between HashMap and TreeMap.

HashMap	TreeMap
Hasmap uses a hashtable in order to store key-value pairs.	Treemap uses Red-black trees to store key-value pair.
Hashmap does not maintain any specific order for key-value pairs.	Treemap maintains a natural ordering based on the keys.
Order of iteration is not guaranteed in the hashmap.	Iteration is of sorted order based on keys.
Hashmaps are faster for retrieval compared to Treemap.	Retrieval in Treemap is slower as it uses tree traversal to find keys.
Hashmap is implemented by using an Array of linked list.	TreeMap is implemented using a Red-black Tree.
Hashmap uses the equals() method of the Object class to compare keys.	TreeMap uses compareTo() method to compare keys.

205. Differentiate between Queue and Deque.

Queue	Deque
The queue is a linear Data structure that is used to store a collection of elements.	Deque also known as a Double-ended queue is also a linear data structure that stores a collection of elements with operations to remove and add from both ends.
Elements in the queue can only be inserted at the end of the data structure.	Elements can be inserted from both ends of the data structure.
Queue can be implemented using Array or Linked List.	Dequeue can be implemented using Circular Array or Doubly Linked List.
Queues are generally used to implement a waiting list or task queue.	Deque is used to implement a stack or dequeuing elements from both ends.

Queue	Deque
	

206. Differentiate between HashSet and TreeSet.

HashSet	TreeSet
HashSet is unordered.	TreeSet is based on natural ordering.
HashSet allows null elements.	TreeSet does not allow null elements.
HashSet is Implemented by the HashSet class.	TreeSet is Implemented by TreeSet class.
HashSet<String> hs = new HashSet<>();	TreeSet<String> ts = new TreeSet<>();

What is Java?

- Java is a distributed application software
- Java is the high-level, object-oriented programming
- Java is API Document.

What are differences between C, C++ and Java

	c	C++	java
Language	C is a procedural	c++ is a object oriented	object oriented language.
approach	Top-down	Bottom-up	Bottom-up
Develop year	1972	1979	1991
Developer name	Dennis M. Ritchie	Bjarne Stroustrup	James Gosling
Header Files	Stdio.h	Iostream.h	Not support
Platform Independent	No	No	Platform Independent
System Independent	Yes	Yes	Yes
Pointers	Yes	Yes	No
Keywords	32	63	57
Preprocessor directives	Yes #include, #define	Yes #include , #define	No
exception handling	No	Yes	yes
templates	No	Yes	No
Multi Threading	No	No	Yes
Code executed	directly	directly	By jvm
Translation Type	Compiled	Compiled	Compiled& interpreter
Operator Overloading	Not supported	Supported	Not supported
storage allocation	Uses malloc , calloc	Uses new , delete	Uses garbage collector
Database Connectivity	No	No	Yes

Why java or What are the features in JAVA?

Features of Java:

1. **Oops concepts**
    - Class
    - object
    - Inheritance
    - Encapsulation
    - Polymorphism
    - Abstraction
2. **Platform independent:** Java is System independent as well as Platform independent because it works with diff System hw as well as Diff Platforms(diff operating Systems)

3. **High Performance:** JIT (Just In Time compiler) enables high performance in Java. JIT converts the bytecode into machine language and then JVM starts the
4. **Multi-threaded:** A flow of execution is known as a Thread. JVM creates a thread which is called main thread. The user can create multiple threads by extending the thread class or by implementing Runnable
5. **Simple:** Java having simple Syntax rules. easy to learn. Complicated things like pointers and multiple inheritance is not
6. **Portable:** Java supports write-once-run-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.
7. **Secured:** Java is secured because it doesn't use explicit Java also provides the concept of ByteCode and Exception handling which makes it more secured.
8. **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more
9. **Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java.
10. **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program
11. Distributed: Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the
12. **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

What is difference between compiler and interpreter ?

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

Why java not support pointer?

A pointer is a variable which can hold the address of another variable or object.

But, Java does not support pointer due to security reason, because if you get the address of any variable you could access it anywhere from the program without any restriction even variable is private.

Java comments Interview Questions

What is java comments ?

The java comments are statements that are not executed by the compiler and interpreter. The comments can be used to provide information or explanation about the variable, method, class or any statement. It can also be used to hide program code for specific time.

Comments are not consider as java code used build communication between programmers and end users?

What are Types of Java Comments?

There are 3 types of comments in java.
1. **Single Line Comment**

The single line comment is used to comment only one line. EX:
// this is Our application

2. **Multi Line Comment**
The multi line comment is used to comment several lines of code.

EX:
/*
This is multiline comment
User for give big description purpose Give n no line etc
*/

3. Documentation Comment
/**
This is doc comment
User for give big description purpose Give n no line etc
*/

Difference Between Multiline and Document Type Comments in java?

Multiline Comments are used build communication between programmers Document Type Comments are used build communication between programmer and end users. Javadoc tool generates html description for Document Type Comments

---

Java Package Interview Questions

What is a package?

Package is a collection class And Interfaces those are collection of methods they can perform some action.

What is the difference between #include and import?

#include is used in C or C++ programing witch is used to go to standard library and copy the entire header file code in to a C/C++ programs. So the program size increases unnecessarily wasting memory & processor time.

Import statement used in java programming uses to pass Ref for particular package .package is saved once used for N no of times. Witch avoids memory wastage problems.

What Is The Super Class Of All Classes?

Java.lang.Object is a Super class for classes

What are the advantages of java package?

- Packages hide classes & interfaces. Thus they provide protection for them.
- The classes of one Package are isolated from the classes of another Package. So it is possible to use same names for the classes into different packages.
- Using package concept we can create our own Packages & also we can extend already available Packages.
- Packages provide re usability of code.
- Package is used to categorize the classes and interfaces so that they can be easily maintained
- Application development time is less, because reuse the code
- Application memory space is less (main memory)
- Application execution time is less
- Application performance is enhance (improve)
- Redundancy (repetition) of code is minimized
- Package provides access
- Package removes naming

Can we import same package/class twice? Will the JVM load the package twice at runtime?

Yes even though programmer whitens same package class twice in the program Jvm loads it once only

---

Can u Explain Rules to create user defined package?

- package statement should be the first statement of any package program.
- Choose an appropriate class name or interface name and whose modifier must be public.
- Any package program can contain only one public class or only one public interface but it can contain any number of normal classes.
- Package program should not contain any main class (that means it should not contain any main())

- modifier of constructor of the class which is present in the package must be public. (This is not applicable in case of interface because interface have no constructor.)
- The modifier of method of class or interface which is present in the package must be public (This rule is optional in case of interface because interface methods by default public)

- Every package program should be save either with public class name or public Interface name

---

Which package is always imported by default?

By default java.lang package imported

How many ways we can use packages classes

Method-I By passing complete address: Java.util.Scanner sc=new java.util.Scanner(System.in);

Method-II: By using import stmt: import java.util.*;

Scanner sc=new Scanner(System.in);

Java Main Method Interview Questions

Can we execute java program without main method?

No, you can't run java class without main method.

Before Java 7, you can run java class by using static initializers. But, from Java 7 it is not possible.

Can we change return type of main() method?

No, the return type of main() method must be void only. Any other type is not acceptable.

Can we declare main() method as private or protected or with no access modifier?

- No, main() method must be public. You can't define main() method as private or protected or with no access
- This is because to make the main() method accessible to JVM. If you define main() method other than public, compilation will be successful but you will get run time error as no main method

---

Can We Overload main() method?

Can java class Suports more Than One Main Method

Yes, We can overload main() method. A Java class can have any number of main() methods. But to run the java class, class should have main()

- method with signature as "public static void main(String[] args)". If you do any modification to this signature, compilation will be successful.

- But, you can't run the java program. You will get run time error as main method not

Can main() method take an argument other than string array?

No, argument of main() method must be string array.

- But, from the introduction of var args you can pass var args of string type as an argument to main() method. Again, var args are nothing but the arrays.

Java Data type Interview Questions

What are the primitive data types in Java ?

There are eight primitive data types.

- byte
- short
- int
- long
- float
- double
- boolean
- char

What is the default value of variables in Java?

data type	Size	Range	Corresponding Wrapper class	Default value
Byte	1 byte	-27 to 27-1(-128 to 127)	Byte	0
Short	2 bytes	-215 to 215-1 (-32768 to 32767)	Short	0
Int	4 bytes	-231 to 231-1 (-2147483648 to 2147483647)	Integer	0
Long	8 bytes	-263 to 263-1	Long	0
Float	4 bytes	-3.4e38 to 3.4e38	Float	0.0
double	8 bytes	-1.7e308 to 1.7e308	Double	0.0
boolean	Not applicable	Not applicable(but allowed values true|false)	Boolean	false
Char	2 bytes	0 to 65535	Character	0(represents blank space)

What is uni code?

Uni code is a standard to include the alphabet from all human languages in java. Uni code system uses 2 bytes to represent a character

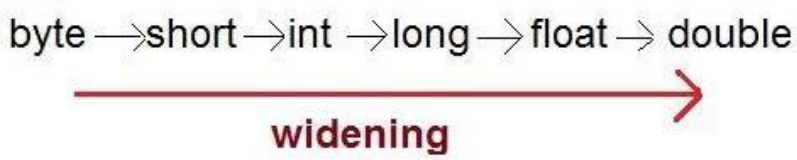
---

What is the default value of local variables in Java?

- There is no default value for local variables

What is Widening?

- Is process of conversion lower data types into higher data types



What is Narrowing ?

- Is process of conversion Higher data types into Lower data types

What is ASCII?

American Standard Code for Information Interchange. It is a standard numarical value asigened to every key in keybord.its range is 0 to 255.

EX: A value=65

What are the Types of Variables in JAVA ?

- Java has 3 kinds of variables. They are Local variables, Instance variables (fields) and Static variables (class variables)

1.LOCAL VARIABLES

- A local variable is defined inside a method block. A block begins with an opening curly brace and ends with a closing curly brace.
- The scope of the variable is limited within the block. In other words, local variables are visible only in the block (method) in which they are declared.

2.INSTANCE VARIABLES

- Instance variables are declared inside a class, but outside of any method / constructor / any block. They are also referred to as fields.
- Objects store their individual states in these fields. Their values are unique to each object (instance of class) and hence they are called as instance variables.

Example – The ‘employeeId’ field of the Employee class will have a unique value for each of its object. (Consider Emp1, Emp2… as objects of the class Employee, then each object will have unique value for the property ‘employee-id’).

3.STATIC VARIABLES (CLASS VARIABLES)

Static variables belong to the class rather than objects in which they are declared. The keyword ‘static’ is prefixed before the variable to represent static variables. Only one copy of this variable is maintained for all the objects.

What is this key word in java?

“this”is a predefined instance variable to hold current object reference

---

Can we use this in static methods?

No we cannot use this in static methods. if we try to use compile time error will come :Cannot use this in a static context

---

What are all the differences between this and super keyword?

- This refers to current class object where as super refers to super class object
- Using this we can access all non static methods and variables. Using super we can access super class variable and methods from sub class.
- Using this(); call we can call other constructor in same class. Using super we can call super class constructor from sub class constructor.

Is it possible to use this in static blocks?

- No its not possible to use this keyword in static block.

Can we use this to refer static members?

- Yes its possible to access static variable of a class using this but its discouraged and as per best practices this should be used on non static reference

---

Can we pass this as parameter of method?

- Yes we can pass this as parameter in a method

Can we return this from a method?

Yes We can return this as current class object. public class B{
int a;
public int getA() { return a;
}
public void setA(int a) { this.a = a;
}

B show(){ return this;
}

public static void main(String[] args) { B obj = new B();
obj.setA(10); System.out.println(obj.getA()); B obj2= obj.show(); System.out.println(obj2.getA());

}
}

Can we call method on this keyword from constructor?

- Yes we can call non static methods from constructor using this keyword.

What is the use of final keyword in java?

- By using final keyword we can make
- Final class
- Final method
- Final variables
- If we declare any class as final we can not extend that class
- If we declare any method as final it can not be overridden in sub class
- If we declare any variable as final its value unchangeable once assigned.

What is the main difference between abstract method and final method?

- Abstract methods must be overridden in sub class where as final methods can not be overridden in sub class

---

What is the actual use of final class in java?

- If a class needs some security and it should not participate in inheritance in this scenario we need to use final class.
- We can not extend final class.

Can we declare interface as final?

- No We can not declare interface as final because interface should be implemented by some class so its not possible to declare interface as final.

Is it possible to declare final variables without initialization?

- No. Its not possible to declare a final variable without initial value assigned.
- While declaring itself we need to initialize some value and that value can not be change at any time.

Can we declare constructor as final?

- No . Constructors can not be final.

What will happen if we try to override final methods in sub classes?

- Compile time error will come :Cannot override the final method from Super Class

---

Can we create object for final class?

- Yes we can create object for final class

What is java static import?

- In Java, static import concept is introduced in 1.5 version access the static members of a class directly without class name or any object

- For example, System class & Math class has static method: import static java.lang.System.*;
  import static java.lang.Math.*; public class MyStaticImportTest { public static void main(String[] a) {
  System.out.println(sqrt(625));
  }
  }
  }

What is Drawback of Static Import in Java?

- If two static members of the same name are imported from multiple different classes, the compiler will throw an error, as it will not be able to determine which member to use in the absence of class name qualification.

EX:

```
import static java.lang.System.*; import static java.lang.Integer.*; import static java.lang.Byte.*;
class Demo
{
public static void main(String[] args)
{
out.println(MAX_VALUE);
}
}
```

Output:

Error:Reference to MAX_VALUE is ambigious

What is the Difference between import and static import?

Import	Static import
we can access classes and interfaces of specified package	we can access all the static members (variables and methods) of a class directly without explicitly calling class name.
import provides accessibility to classes and interface	static import provides accessibility to static members of the class.

Is Java platform independent?

Yes. Java is a platform independent language. We can write java code on one platform and run it on another platform. For e.g. we can write and compile the code on windows and can run the generated bytecode on Linux or any other supported platform. This is one of the main features of java.

What all memory areas are allocated by JVM?

- Classloader, Class area, Heap, Stack, Program Counter Register and Native Method Stack

---

What is javac ?

- The javac is a compiler that compiles the source code of your program and generates bytecode. In simple words javac produces the java byte code from the source code written *.java file. JVM executes the bytecode to run the program.

What is class?

- A class is a blueprint or template or prototype from which you can create the object of that class. A class has set of properties and methods that are common to its objects.

What is a wrapper class in Java?

- A wrapper class converts the primitive data type such as int, byte, char, boolean etc. to the objects of their respective classes such as Integer, Byte, Character, Boolean etc.

What is a path and classPath in Java?

- Path specifies the location of .exe files. Classpath specifies the location of bytecode (.class files).

Basic Java Interview Questions

Q) Is Java platform independent?

Yes. Java is a platform independent language. We can write java code on one platform and run it on another platform. For e.g. we can write and compile the code on windows and can run the generated bytecode on Linux or any other supported platform. This is one of the main features of java.

Q) What all memory areas are allocated by JVM?

Classloader, Class area, Heap, Stack, Program Counter Register and Native Method Stack

Q) Java vs. C ++?

Here are the few differences between Java and C++:

- Platform dependency – C++ is platform dependent while java is platform independent
- No goto support – Java doesn't support **goto statement** while C++ does.
- Multiple inheritance – C++ supports **multiple inheritance** while java does not.
- Multithreading – C++ does not have in-build thread support, on the other hand java supports **multithreading**
- Virtual keyword – C++ has virtual keyword, it determines if a member function of a class can be overridden in its child class. In java there is no concept of virtual keyword.

Q) Explain public static void main(String args[])

Here **public** is an **access modifier**, which means that this method is accessible by any class.

static – static keyword tells that this method can be accessed without creating the instance of the class. Refer: **Static keyword in java**

void – this main method returns no value.

main – It is the name of the method.

String args[] – The args is an array of String type. This contains the command line arguments that we can pass while running the program.

Q) What is javac ?

The javac is a compiler that compiles the source code of your program and generates bytecode. In simple words javac produces the java byte code from the source code written *.java file. JVM executes the bytecode to run the program.

Q) What is class?

A class is a blueprint or template or prototype from which you can create the object of that class. A class has set of properties and methods that are common to its objects.

Q) What is the base class of all classes?

`java.lang.Object` is the base class (super class) of all classes in java.

Q) What is a wrapper class in Java?

A wrapper class converts the primitive data type such as int, byte, char, boolean etc. to the objects of their respective classes such as Integer, Byte, Character, Boolean etc. Refer: **Wrapper class in Java**

Q) What is a path and classPath in Java?

Path specifies the location of .exe files. Classpath specifies the location of bytecode (.class files).

Q) Different Data types in Java.

- byte – 8 bit
- short – 16 bit
- char – 16 bit Unicode
- int – 32 bit (whole number)
- float – 32 bit (real number)
- long – 64 bit (Single precision)
- double – 64 bit (double precision)

Q) What is Unicode?

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

Q) What are Literals?

Any constant value that is assigned to a variable is called literal in Java. For example –

```
// Here 101 is a literal
int num = 101
```

Q) Dynamic Initialization?

Dynamic initialization is process in which initialization value of a variable isn't known at compile-time. It's computed at runtime to initialize the variable.

Q) What is Type casting in Java?

When we assign a value of one data type to the different data type then these two data types may not be compatible and needs a conversion. If the data types are compatible (for example assigning int value to long) then java does automatic conversion and does not require casting. However if the data types are not compatible then they need to be casted for conversion.

For example:

```
//here in the brackets we have mentioned long keyword, this is casting
double num = 10001.99;
long num2 = (long)num;
```

Q) What is an Array?

An array is a collection (group) of fixed number of items. Array is a homogeneous data structure which means we can store multiple values of same type in an array but it can't contain multiple values of different types. For example an array of int type can only hold integer values.

Q) What is BREAK statement in java?

The break statement is used to break the flow sequence in Java.

- break statement is generally used with switch case data structure to come out of the statement once a case is executed.
- It can be used to come out of the loop in Java

Q) Arrays can be defined in different ways. Write them down.

```
int arr[];
int[] arr;
```

OOPs Interview Questions

Q) Four main principles of OOPS Concepts?

- Inheritance
- Polymorphism

- Data Encapsulation
- Abstraction

Q) What is inheritance?

The process by which one class acquires the properties and functionalities of another class is called inheritance. Inheritance brings reusability of code in a java application. **Refer**: Guide to **Inheritance in Java**.

Q) Does Java support Multiple Inheritance?

When a class extends more than one classes then it is called multiple inheritance. Java doesn't support multiple inheritance whereas C++ supports it, this is one of the difference between java and C++. **Refer: Why java doesn't support multiple inheritance?**

Q) What is Polymorphism and what are the types of it?

Polymorphism is the ability of an object to take many forms. The most common use of polymorphism in OOPs is to have more than one method with the same name in a single class. There are two types of polymorphism: static polymorphism and dynamic polymorphism. Refer these guides to understand the polymorphism concept in detail: 1) **Java Polymorphism** 2) **Types of Polymorphism**

Q) What is method overriding in Java?

When a sub class (child class) overrides the method of super class(parent class) then it is called overriding. To override a method, the signature of method in child class must match with the method signature in parent class. **Refer**: **Java – Method Overriding**

Q) Can we override a static method?

No, we cannot override a static method in Java.

Q) What is method overloading?

When a class has more than one methods with the same name but different number, sequence or types of arguments then it is known as method overloading. **Refer: Java – Method Overloading**

Q) Does Java support operator overloading?

Operator overloading is not supported in Java.

Q) Can we overload a method by just changing the return type and without changing the signature of method?

No, We cannot do this. To overload a method, the method signature must be different, return type doesn't play any role in method overloading.

Q) Is it possible to overload main() method of a class?

Yes, we can overload main() method in Java.

Q) What is the difference between method overloading and method overriding?

Refer this guide: **Overloading vs overriding in Java**

Q) What is static and dynamic binding in Java?

Binding refers to the linking of method call to its body. A binding that happens at compile time is known as static binding while binding at runtime is known as dynamic binding. Refer: **Static and Dynamic binding in Java**.

Q) What is Encapsulation?

Wrapping of the data and code together is known as encapsulation. **Refer**: **Java Encapsulation**.

Q) What is an abstract class in Java?

An abstract class is a class which can't be instantiated (we cannot create the object of abstract class), we can only extend such classes. It provides the generalised form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details. We can achieve partial abstraction using **abstract classes**, to achieve full abstraction we use interfaces.

Q) What is Interface in java?

An interface is used for achieving full abstraction. A class implements an interface, thereby inheriting the abstract methods of the interface. **Refer**: **Java Interface**

Q) What is the difference between abstract class and interface?

1) abstract class can have abstract and non-abstract methods. An interface can only have abstract methods.
2) An abstract class can have static methods but an interface cannot have static methods.
3) abstract class can have constructors but an interface cannot have constructors.

Q) Name the access modifiers that can be applied to the inner classes?

public ,private , abstract, final, protected.

Q) What is a constructor in Java?

Constructor is used for creating an instance of a class, they are invoked when an instance of class gets created. Constructor name and class name should be same and it doesn't have a return type. Refer this guide: **Java Constructor**.

Q) Can we inherit the constructors?

No, we cannot inherit constructors.

Q) Can we mark constructors final?

No, Constructor cannot be declared final.

Q) What is default and parameterized constructors?

Default: Constructors with no arguments are known as default constructors, when you don't declare any constructor in a class, compiler creates a default one automatically.

Parameterized: Constructor with arguments are known as parameterized constructors.

Q) Can a constructor call another constructor?

Yes. A constructor can call the another constructor of same class using this keyword. For e.g. this() calls the default constructor.
Note: this() must be the first statement in the calling constructor.

Q) Can a constructor call the constructor of parent class?

Yes. In fact it happens by default. A child class constructor always calls the parent class constructor. However we can still call it using super keyword. For e.g. super() can be used for calling super class default constructor.

Note: super() must be the first statement in a constructor.

Q)THIS keyword?

The `this` keyword is a reference to the current object.

Q) Can this keyword be assigned null value?

No, this keyword cannot have null values assigned to it.

Q) Explain ways to pass the arguments in Java?

In java, arguments can be passed as call by value – Java only supports call by value, there is no concept of call by reference in Java.

Q) What is static variable in java?

Static variables are also known as class level variables. A static variable is same for all the objects of that particular class in which it is declared.

Q) What is static block?

A **static block** gets executed at the time of class loading. They are used for initializing static variables.

Q) What is a static method?

Static methods can be called directly without creating the instance (Object) of the class. A static method can access all the static variables of a class directly but it cannot access non-static variables without creating instance of class.

Q) Explain super keyword in Java?

super keyword references to the parent class. There are several uses of super keyword:

- It can be used to call the superclass(Parent class) constructor.
- It can be used to access a method of the superclass that has been hidden by subclass (Calling parent class version, In case of method overriding).
- To call the constructor of parent class.

Q) Use of final keyword in Java?

Final methods – These methods cannot be overridden by any other method.
Final variable – Constants, the value of these variable can't be changed, its fixed.
Final class – Such classes cannot be inherited by other classes. These type of classes will be used when application required security or someone don't want that particular class. **Final Keyword in Java**.

Q) What is a Object class?

This is a special class defined by java; all other classes are subclasses of object class. Object class is superclass of all other classes. Object class has the following methods

- objectClone () – to creates a new object that is same as the object being cloned.
- boolean equals(Object obj) – determines whether one object is equal to another.
- finalize() – Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. A subclass overrides the finalize method to dispose of system resources or to perform other cleanup.
- toString () – Returns a string representation of the object.

Q) What are Packages in Java?

A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations). Refer: **Package in Java**.

Q)What is the difference between import java.util.Date and java.util.* ?

The star form (java.util.*) includes all the classes of that package and that may increase the compilation time – especially if you import several packages. However it doesn't have any effect run-time performance.

Q) What is static import?

Refer: Static Import in Java.

Q) Garbage collection in java?

Since objects are dynamically allocated by using the new operator, java handles the de-allocation of the memory automatically, when no references to an object exist for a long time. This whole process is called **garbage collection**. The whole purpose of Garbage collection is efficient memory management.

Q) Use of finalize() method in java?

finalize() method is used to free the allocated resource.

Q) How many times does the garbage collector calls the finalize() method for an object?

The **garbage collector** calls the finalize() method only once for an object.

Q) What are two different ways to call garbage collector?

`System.gc()` OR `Runtime.getRuntime().gc()`.

Q) Can the Garbage Collection be forced by any means?

No, its not possible. you cannot force garbage collection. you can call system.gc() methods for garbage collection but it does not guarantee that garbage collection would be done.

Exception handling Interview Questions

Q) What is an exception?

Exceptions are abnormal conditions that arise during execution of the program. It may occur due to wrong user input or wrong logic written by programmer.

Q) Exceptions are defined in which java package? OR which package has definitions for all the exception classes?

`Java.lang.Exception`
This package contains definitions for Exceptions.

Q) What are the types of exceptions?

There are two types of exceptions: **checked and unchecked exceptions**.
Checked exceptions: These exceptions must be handled by programmer otherwise the program would throw a compilation error.
Unchecked exceptions: It is up to the programmer to write the code in such a way to avoid unchecked exceptions. You would not get a compilation error if you do not handle these exceptions. These exceptions occur at runtime.

Q) What is the difference between Error and Exception?

Error: Mostly a system issue. It always occur at run time and must be resolved in order to proceed further.
Exception: Mostly an input data issue or wrong logic in code. Can occur at compile time or run time.

Q) What is throw keyword in exception handling?

The throw keyword is used for throwing user defined or pre-defined exception.

Q) What is throws keyword?

If a method does not handle a checked exception, the method must declare it using the throwskeyword. The throws keyword appears at the end of a method's signature.

Q) Difference between throw and throws in Java

Read the difference here: Java – **throw vs throws**.

Q) Can static block throw exception?

Yes, A static block can throw exceptions. It has its own limitations: It can throw only Runtime exception (Unchecked exceptions), In order to throw checked exceptions you can use a try-catch block inside it.

Q) What is finally block?

Finally block is a block of code that always executes, whether an exception occurs or not. Finally block follows try block or try-catch block.

Q) ClassNotFoundException vs NoClassDefFoundError?
1) ClassNotFoundException occurs when loader could not find the required class in class path.

2) NoClassDefFoundError occurs when class is loaded in classpath, but one or more of the class which are required by other class, are removed or failed to load by compiler.

Q) Can we have a try block without catch or finally block?

No, we cannot have a try block without catch or finally block. We must have either one of them or both.

Q) Can we have multiple catch blocks following a single try block?

Yes we can have **multiple catch blocks** in order to handle more than one exception.

Q) Is it possible to have finally block without catch block?

Yes, we can have try block followed by finally block without even using catch blocks in between.

When a finally block does not get executed?

The only time finally won't be called is if you call System.exit() or if the JVM crashes first.

Q) Can we handle more than one exception in a single catch block?

Yes we can do that using if-else statement but it is not considered as a good practice. We should have one catch block for one exception.

Q) What is a Java Bean?

A JavaBean is a Java class that follows some simple conventions including conventions on the names of certain methods to get and set state called Introspection. Because it follows conventions, it can easily be processed by a software tool that connects Beans together at runtime. JavaBeans are reusable software components.

Java Multithreading Interview Questions

Q) What is Multithreading?

It is a process of executing two or more part of a program simultaneously. Each of these parts is known as threads. In short the process of executing multiple threads simultaneously is known as multithreading.

Q) What is the main purpose of having multithread environment?

Maximizing CPU usage and reducing CPU idle time

Q) What are the main differences between Process and thread? Explain in brief.

1) One process can have multiple threads. A thread is a smaller part of a process.
2) Every process has its own memory space, executable code and a unique process identifier (PID) while every thread has its own stack in Java but it uses process main memory and shares it with other threads.
3) Threads of same process can communicate with each other using keyword like wait and notify etc. This process is known as inter process communication.

Q) How can we create a thread in java?

There are following two ways of creating a thread:
1) By Implementing Runnable interface.
2) By Extending Thread class.

Q) Explain yield and sleep?

yield() – It causes the currently executing thread object to temporarily pause and allow other threads to execute.

sleep() – It causes the current thread to suspend execution for a specified period. When a thread goes into sleep state it doesn't release the lock.

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

sleep() – It causes the current thread to suspend execution for a specified period. When a thread goes into sleep state it doesn't release the lock

wait() – It causes current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

Q) What is a daemon thread?

A daemon thread is a thread, that does not prevent the JVM from exiting when the program finishes but the thread is still running. An example for a daemon thread is the garbage collection.

Q) What does join( ) method do?

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

Q) Preemptive scheduling vs. time slicing?

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

Q) Can we call run() method of a Thread class?

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

Q) What is Starvation?

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

Q) What is deadlock?

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

Serialization interview Questions

Q: What is Serialization and de-serialization?

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

Q) Do we need to implement any method of Serializable interface to make an object serializable?

No. In order to make an object serializable we just need to implement the interface Serializable. We don't need to implement any methods.

Q) What is a transient variable?

1) transient variables are not included in the process of serialization.
2) They are not the part of the object's serialized state.
3) Variables which we don't want to include in serialization are declared as transient.

String interview questions

Q) A string class is immutable or mutable?

String class is immutable that's the reason once its object gets created, it cannot be changed further.

Q) Difference between StringBuffer and StringBuilder class?

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

Q) What is toString() method in Java?

The toString() method returns the string representation of any object.

Java collections interview questions

Q) What is List?

Elements can be inserted or accessed by their position in the list, using a zero-based index.
A list may contain duplicate elements.

Q) What is Map?

Map interface maps unique keys to values. A key is an object that we use to retrieve a value later. A map cannot contain duplicate keys: Each key can map to at most one value.

Q) What is Set?

A Set is a Collection that cannot contain duplicate elements.

Q) Why ArrayList is better than Arrays?

Array can hold fixed number of elements. ArrayList can grow dynamically.

Q) What is the difference between ArrayList and LinkedList?

1) LinkedList store elements within a doubly-linked list data structure. ArrayList store elements within a dynamically resizing array.
2) LinkedList is preferred for add and update operations while ArrayList is a good choice for search operations. **Read more here**.

Q) For addition and deletion. Which one is most preferred: ArrayList or LinkedList?

LinkedList. Because deleting or adding a node in LinkedList is faster than ArrayList.

Q) For searches. Which one is most preferred: ArrayList or LinkedList?

ArrayList. Searching an element is faster in ArrayList compared to LinkedList.

Q) What is the difference between ArrayList and Vector?

1) Vector is synchronized while ArrayList is not synchronized.
2) By default, Vector doubles the size of its array when it is re-sized internally. ArrayList increases by half of its size when it is re-sized. **More details**.

Q) What is the difference between Iterator and ListIterator?

Following are the major differences between them:
1) Iterator can be used for traversing Set, List and Map. ListIterator can only be used for traversing a List.
2) We can traverse only in forward direction using Iterator. ListIterator can be used for traversing in both the directions(forward and backward). Read more at: **ListIterator vs Iterator**.

Q) Difference between TreeSet and SortedSet?

TreeSet implements SortedSet interface.

Q) What is the difference between HashMap and Hashtable?

1) Hashtable is synchronized. HashMap is not synchronized.
2) Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values. **Read more here**.

Q) What is the difference between Iterator and Enumeration?

1) Iterator allows to remove elements from the underlying collection during the iteration using its remove() method. We cannot add/remove elements from a collection when using enumerator.
2) Iterator has improved method names.
Enumeration.hasMoreElement() -> Iterator.hasNext()
Enumeration.nextElement() -> Iterator.next().

Applet Interview Questions

Q) How do you do file I/O from an applet?

Unsigned applets are simply not allowed to read or write files on the local file system .

Unsigned applets can, however, read (but not write) non-class files bundled with your applet on the server, called resource files

Q) What is container ?

A component capable of holding another component is called as container.
Container
Panel
Applet
Window
Frame
Dialog

Q) On Windows, generally frames are invisible, how to make it visible?

```
Frame f = new Frame();
f.setSize(300,200);  //height and width
```

```
f.setVisible(true) ;  // Frames appears
```

Q) Listeners and corresponding Methods?

ActionListerner – actionPerformed();
ItemListerner – itemStateChanged();
TextListener – textValueChanged();
FocusListener – focusLost(); & FocusGained();
WindowListener – windowActified(); windowDEactified(); windowIconified(); windowDeiconified(); windowClosed(); windowClosing(); windowOpened();
MouseMotionListener – mouseDragged(); & mouseMoved();
MouseListener – mousePressed(); mouseReleased(); mouseEntered(); mouseExited(); mouseClicked();

Q) Applet Life cycle?

Following stage of any applets life cycle, starts with init(), start(), paint(), stop() and destroy().

Q) Use of showStatus() method in Java

To display the message at the bottom of the browser when applet is started.

Q) What is Event handling in Java?

Is irrespective of any component, if any action performed/done on Frame, Panel or on window, handling those actions are called Event Handling.

Q) What is Adapter class?

Adapter class is an abstract class.
Advantage of adapter: To perform any window listener, we need to include all the methods used by the window listener whether we use those methods are not in our class like Interfaces whereas with adapter class, its sufficient to include only the methods required to override. Straight opposite to Interface.

240 CORE JAVA INTERVIEW QUESTIONS AND ANSWERS

Table of Contents

1) what are static blocks and static initalizers in Java ?

Static blocks or static initializers are used to initalize static fields in java. we declare static blocks when we want to intialize static fields in our class. Static blocks gets executed exactly once when the class is loaded . Static blocks are executed even before the constructors are executed.

2) How to call one constructor from the other constructor ?

With in the same class if we want to call one constructor from other we use this() method. Based on the number of parameters we pass appropriate this() method is called.
Restrictions for using this method :
1) this must be the first statement in the constructor
2)we cannot use two this() methods in the constructor

3) What is method overriding in java ?

If we have methods with same signature (same name, same signature, same return type) in super class and subclass then we say
subclass method is overridden by superclass.
When to use overriding in java
If we want same method with different behaviour in superclass and subclass then we go for overriding.
When we call overridden method with subclass reference subclass method is called hiding the superclass method.

4) What is super keyword in java ?

Variables and methods of super class can be overridden in subclass . In case of overriding , a subclass object call its own variables and methods. Subclass cannot access the variables and methods of superclass because the overridden variables or methods hides the methods and variables of super class. But still java provides a way to access super class members even if its members are overridden. Super is used to access superclass variables, methods, constructors.
Super can be used in two forms :
1) First form is for calling super class constructor.
2) Second one is to call super class variables,methods.
Super if present must be the first statement.

5) Difference between method overloading and method overriding in java ?

Method Overloading	Method Overriding
1) Method Overloading occurs with in the same class	Method Overriding occurs between two classes superclass and subclass
2) Since it involves with only one class inheritance is not involved.	Since method overriding occurs between superclass and subclass inheritance is involved.
3)In overloading return type need not be the same	3) In overriding return type must be same.
4) Parameters must be different when we do overloading	4) Parameters must be same.
5) Static polymorphism can be acheived using method overloading	5) Dynamic polymorphism can be acheived using method overriding.
6) In overloading one method can't hide the another	6) In overriding subclass method hides that of the superclass method.

6) Difference between abstract class and interface ?

Interface	Abstract Class
1) Interface contains only abstract methods	1) Abstract class can contain abstract methods, concrete methods or both
2) Access Specifiers for methods in interface must be public	2) Except private we can have any access specifier for methods in abstract class.
3) Variables defined must be public , static , final	3) Except private variables can have any access specifiers
4) Multiple Inheritance in java is implemented using interface	4)We cannot achieve multiple inheritance using abstract class.
5) To implement an interface we use implements keyword	5)To implement an interface we use implements keyword

7) Why java is platform independent?

The most unique feature of java is platform independent. In any programming language soruce code is compiled in to executable code . This cannot be run across all platforms. When javac compiles a java program it generates an executable file called .class file.
class file contains byte codes. Byte codes are interpreted only by JVM's . Since these JVM's are made available across all platforms by Sun Microsystems, we can execute this byte code in any platform. Byte code generated in windows environment can also be executed in linux environment. This makes java platform independent.

8) What is method overloading in java ?

A class having two or more methods with same name but with different arguments then we say that those methods are overloaded. Static polymorphism is achieved in java using method overloading.
Method overloading is used when we want the methods to perform similar tasks but with different inputs or values. When an overloaded method is invoked java first checks the method name, and the number of arguments ,type of arguments; based on this compiler executes this method.
Compiler decides which method to call at compile time. By using overloading static polymorphism or static binding can be achieved in java.
Note : Return type is not part of method signature. we may have methods with different return types but return type alone is not sufficient to call a method in java.

9) What is difference between c++ and Java ?

Java	C++
1) Java is platform independent	C++ is platform dependent.
2) There are no pointers in java	There are pointers in C++.
3) There is no operator overloading in java	C ++ has operator overloading.
4) There is garbage collection in java	There is no garbage collection
5) Supports multithreading	Does'nt support multithreading
6) There are no templates in java	There are templates in java
7) There are no global variables in java	There are global variables in c++

10) What is JIT compiler ?

JIT compiler stands for Just in time compiler. JIT compiler compiles byte code in to executable code . JIT a part of JVM .JIT cannot convert complete java program in to executable code it converts as and when it is needed during execution.

11) What is bytecode in java ?

When a javac compiler compiler compiles a class it generates .class file. This .class file contains set of instructions called byte code. Byte code is a machine independent language and contains set of instructions which are to be executed only by JVM. JVM can understand this byte codes.

12) Difference between this() and super() in java ?

this() is used to access one constructor from another with in the same class while super() is used to access superclass constructor. Either this() or super() exists it must be the first statement in the constructor.

13) What is a class ?

Classes are fundamental or basic unit in Object Oriented Programming .A class is kind of blueprint or template for objects. Class defines variables, methods. A class tells what type of objects we are creating. For example take Department class tells us we can create department type objects. We can create any number of department objects.
All programming constructs in java reside in class. When JVM starts running it first looks for the class when we compile. Every Java application must have atleast one class and one main method.
Class starts with class keyword. A class definition must be saved in class file that has same as class name. File name must end with .java extension.

```
public class FirstClass
{public static void main(String[] args)
{System.out.println("My First class");
}
}
```

If we see the above class when we compile JVM loads the FirstClass and generates a .class file(FirstClass.class). When we run the program we are running the class and then executes the main method.

14) What is an object ?

An Object is instance of class. A class defines type of object. Each object belongs to some class.Every object contains state and behavior. State is determined by value of attributes and behavior is called method. Objects are alos called as an instance.
To instantiate the class we declare with the class type.

```
public classFirstClass {public static voidmain(String[] args)
{
FirstClass f=new FirstClass();
System.out.println("My First class");
}

}
```

To instantiate the FirstClass we use this statement
FirstClass f=new FirstClass();
f is used to refer FirstClass object.

15)What is method in java ?

It contains the executable body that can be applied to the specific object of the class.
Method includes method name, parameters or arguments and return type and a body of executable code.

```
Syntax : type methodName(Argument List){
}
```

ex : public float add(int a, int b, int c)
methods can have multiple arguments. Separate with commas when we have multiple arguments.

16) What is encapsulation ?

The process of wrapping or putting up of data in to a single unit class and keeps data safe from misuse is called encapsulation .

Through encapsulation we can hide and protect the data stored in java objects.Java supports encapsulation through access control. There are four access control modifiers in java public , private ,protected and default level.
For example take a car class , In car we have many parts which is not required for driver to know what all it consists inside. He is required to know only about how to start and stop the car. So we can expose what all are required and hide the rest by using encapsulation.

17) Why main() method is public, static and void in java ?

public : "public" is an access specifier which can be used outside the class. When main method is declared public it means it can be used outside class.
static : To call a method we require object. Sometimes it may be required to call a method without the help of object. Then we declare that method as static. JVM calls the main() method without creating object by declaring keyword static.
void : void return type is used when a method does'nt return any value . main() method does'nt return any value, so main() is declared as void.

```
Signature : public static void main(String[] args) {
```

18) Explain about main() method in java ?

Main() method is starting point of execution for all java applications.

```
public static void main(String[] args) {}
```

String args[] are array of string objects we need to pass from command line arguments.
Every Java application must have atleast one main method.

19)What is constructor in java ?

A constructor is a special method used to initialize objects in java.

we use constructors to initialize all variables in the class when an object is created. As and when an object is created it is initialized automatically with the help of constructor in java.
We have two types of constructors
Default Constructor
Parameterized Constructor
Signature : public classname()
{
}

Signature : public classname(parameters list)
{
}

20) What is difference between length and length() method in java ?

length() : In String class we have length() method which is used to return the number of characters in string.
Ex : String str = "Hello World";
System.out.println(str.length());
Str.length() will return 11 characters including space.

length : we have length instance variable in arrays which will return the number of values or objects in array.
For example :
String days[]={" Sun","Mon","wed","thu","fri","sat"};
Will return 6 since the number of values in days array is 6.

21) What is ASCII Code?

ASCII stands for American Standard code for Information Interchange. ASCII character range is 0 to 255. We can't add more characters to the ASCII Character set. ASCII character set supports only English. That

is the reason, if we see C language we can write c language only in English we can't write in other languages because it uses ASCII code.

22) What is Unicode ?

Unicode is a character set developed by Unicode Consortium. To support all languages in the world Java supports Unicode values. Unicode characters were represented by 16 bits and its character range is 0-65,535.
Java uses ASCII code for all input elements except for Strings,identifiers, and comments. If we want to use telugu we can use telugu characters for identifiers.We can enter comments in telugu.

23) Difference between Character Constant and String Constant in java ?

Character constant is enclosed in single quotes. String constants are enclosed in double quotes. Character constants are single digit or character. String Constants are collection of characters.
Ex :'2', 'A'
Ex : "Hello World"

24) What are constants and how to create constants in java?

Constants are fixed values whose values cannot be changed during the execution of program. We create constants in java using final keyword.
Ex : final int number =10;
final String str="java-interview –questions"

25) Difference between '>>' and '>>>' operators in java?

>> is a right shift operator shifts all of the bits in a value to the right to a specified number of times.
int a =15;
a= a >> 3;
The above line of code moves 15 three characters right.
>>> is an unsigned shift operator used to shift right. The places which were vacated by shift are filled with zeroes.

Core java Interview questions on Coding Standards

26) Explain Java Coding Standards for classes or Java coding conventions for classes?

Sun has created Java Coding standards or Java Coding Conventions . It is recommended highly to follow java coding standards.
Classnames should start with uppercase letter. Classnames names should be nouns. If Class name is of multiple words then the first letter of inner word must be capital letter.
Ex : Employee, EmployeeDetails, ArrayList, TreeSet, HashSet

27) Explain Java Coding standards for interfaces?

1) Interface should start with uppercase letters
2) Interfaces names should be adjectives

Example : Runnable, Serializable, Marker, Cloneable

28) Explain Java Coding standards for Methods?

1) Method names should start with small letters.
2) Method names are usually verbs
3) If method contains multiple words, every inner word should start with uppercase letter.

Ex : toString()

4) Method name must be combination of verb and noun

Ex : getCarName(),getCarNumber()

29) Explain Java Coding Standards for variables ?

1) Variable names should start with small letters.
2) Variable names should be nouns
3) Short meaningful names are recommended.
4) If there are multiple words every innerword should start with Uppecase character.

Ex : string,value,empName,empSalary

30) Explain Java Coding Standards for Constants?

Constants in java are created using static and final keywords.
1) Constants contains only uppercase letters.
2) If constant name is combination of two words it should be separated by underscore.

3) Constant names are usually nouns.
Ex:MAX_VALUE, MIN_VALUE, MAX_PRIORITY, MIN_PRIORITY

31) Difference between overriding and overloading in java?

Overriding	Overloading
In overriding method names must be same	In overloading method names must be same
Argument List must be same	Argument list must be different atleast order of arguments.
Return type can be same or we can return covariant type. From 1.5 covariant types are allowed	Return type can be different in overloading.
We cant increase the level of checked exceptions. No restrictions for unchecked exceptions	In overloading different exceptions can be thrown.
A method can only be overridden in subclass	A method can be overloaded in same class or subclass
Private,static and final variables cannot be overridden.	Private , static and final variables can be overloaded.
In overriding which method is called is decided at runtime based on the type of object referenced at run time	In overloading which method to call is decided at compile time based on reference type.
Overriding is also known as Runtime polymorphism, dynamic polymorphism or late binding	Overloading is also known as Compile time polymorphism, static polymorphism or early binding.

32) What is 'IS-A ' relationship in java?

'is a' relationship is also known as inheritance. We can implement 'is a' relationship or inheritance in java using extends keyword. The advantage or inheritance or is a relationship is reusability of code instead of duplicating the code.
Ex : Motor cycle is a vehicle
Car is a vehicle Both car and motorcycle extends vehicle.

33) What is 'HAS A'' relationship in java?

'Has a ' relationship is also known as "composition or Aggregation". As in inheritance we have 'extends' keyword we don't have any keyword to implement 'Has a' relationship in java. The main advantage of 'Has-A' relationship in java code reusability.

34) Difference between 'IS-A' and 'HAS-A' relationship in java?

IS-A relationship	HAS- A RELATIONSHIP
Is a relationship also known as inheritance	Has a relationship also known as composition or aggregation.
For IS-A relationship we uses extends keyword	For Has a relationship we use new keyword
Ex : Car is a vehicle.	Ex : Car has an engine. We cannot say Car is an engine
The main advantage of inheritance is reusability of code	The main advantage of has a relationship is reusability of code.

35) Explain about instanceof operator in java?

Instanceof operator is used to test the object is of which type.
Syntax : <reference expression> instanceof <destination type>
Instanceof returns true if reference expression is subtype of destination type.
Instanceof returns false if reference expression is null.

```
Example : public classInstanceOfExample {public static voidmain(String[] args) {Integer a =
newInteger(5);if (a instanceof java.lang.Integer) {
System.out.println(true);
} else {
System.out.println(false);
}
```

```
}

}
```

Since a is integer object it returns true.
There will be a compile time check whether reference expression is subtype of destination type. If it is not a subtype then compile time error will be shown as Incompatible types

36) What does null mean in java?

When a reference variable doesn't point to any value it is assigned null.
Example : Employee employee;
In the above example employee object is not instantiate so it is pointed no where

37) Can we have multiple classes in single file ?

Yes we can have multiple classes in single file but it people rarely do that and not recommended. We can have multiple classes in File but only one class can be made public. If we try to make two classes in File public we get following compilation error.

```
"The public type must be defined in its own file".
```

38) What all access modifiers are allowed for top class ?

For top level class only two access modifiers are allowed. public and default. If a class is declared as public it is visible everywhere.
If a class is declared default it is visible only in same package.
If we try to give private and protected as access modifier to class we get the below compilation error.

```
Illegal Modifier for the class only public,abstract and final are permitted.
```

39) What are packages in java?

Package is a mechanism to group related classes ,interfaces and enums in to a single module.
Package can be declared using the following statement :
Syntax : package <package-name>
Coding Convention : package name should be declared in small letters.
package statement defines the namespace.
The main use of package is
1) To resolve naming conflicts
2) For visibility control : We can define classes and interfaces that are not accessible outside the class.

40) Can we have more than one package statement in source file ?

We can't have more than one package statement in source file. In any java program there can be atmost only 1 package statement. We will get compilation error if we have more than one package statement in source file.

41) Can we define package statement after import statement in java?

We can't define package statement after import statement in java. package statement must be the first statement in source file. We can have comments before the package statement.

42) What are identifiers in java?

Identifiers are names in java program. Identifiers can be class name, method name or variable name.
Rules for defining identifiers in java:
1) Identifiers must start with letter,Underscore or dollar($) sign.
2) Identifiers can't start with numbers .
3) There is no limit on number of characters in identifier but not recommended to have more than 15 characters
4) Java identifiers are case sensitive.
5) First letter can be alphabet ,or underscore and dollar sign. From second letter we can have numbers .
6) We should'nt use reserve words for identifiers in java.

43) What are access modifiers in java?

The important feature of encapsulation is access control. By preventing access control we can misuse of class, methods and members.

A class, method or variable can be accessed is determined by the access modifier. There are three types of access modifiers in java. public,private,protected. If no access modifier is specified then it has a default access.

44) What is the difference between access specifiers and access modifiers in java?

In C++ we have access specifiers as public,private,protected and default and access modifiers as static, final. But there is no such divison of access specifiers and access modifiers in java. In Java we have access modifiers and non access modifiers.
Access Modifiers : public, private, protected, default
Non Access Modifiers : abstract, final, stricfp.

45) What access modifiers can be used for class ?

We can use only two access modifiers for class public and default.
public: A class with public modifier can be visible
1) In the same class
2) In the same package subclass
3) In the same package nonsubclass
4) In the different package subclass
5) In the different package non subclass.

default : A class with default modifier can be accesed
1) In the same class
2) In the same package subclass
3) In the same package nonsubclass
4) In the different package subclass
5) In the different package non subclass.

46) Explain what access modifiers can be used for methods?

We can use all access modifiers public, private,protected and default for methods.
public : When a method is declared as public it can be accessed
6) In the same class
7) In the same package subclass
8) In the same package nonsubclass
9) In the different package subclass
10) In the different package non subclass.

default : When a method is declared as default, we can access that method in
1) In the same class
2) In the same package subclass
3) In the same package non subclass
We cannot access default access method in
1) Different package subclass
2) Different package non subclass.

protected : When a method is declared as protected it can be accessed
1) With in the same class
2) With in the same package subclass
3) With in the same package non subclass
4) With in different package subclass
It cannot be accessed non subclass in different package.

private : When a method is declared as private it can be accessed only in that class.
It cannot be accessed in
1) Same package subclass
2) Same package non subclass
3) Different package subclass
4) Different package non subclass.

47) Explain what access modifiers can be used for variables?

We can use all access modifiers public, private,protected and default for variables.
public : When a variables is declared as public it can be accessed
1) In the same class

2) In the same package subclass
3) In the same package nonsubclass
4) In the different package subclass
5) In the different package non subclass.

default : When a variables is declared as default, we can access that method in
1) In the same class
2) In the same package subclass
3) In the same package non subclass
We cannot access default access variables in
4) Different package subclass
5) Different package non subclass.

protected : When a variables is declared as protected it can be accessed
1) With in the same class
2) With in the same package subclass
3) With in the same package non subclass
4) With in different package subclass
It cannot be accessed non subclass in different package.

private : When a variables is declared as private it can be accessed only in that class.
It cannot be accessed in
1) Same package subclass
2) Same package non subclass
3) Different package subclass
4) Different package non subclass.

48) What is final access modifier in java?

final access modifier can be used for class, method and variables. The main advantage of final access modifier is security no one can modify our classes, variables and methods. The main disadvantage of final access modifier is we cannot implement oops concepts in java. Ex : Inheritance, polymorphism.
final class : A final class cannot be extended or subclassed. We ar e preventing inheritance by marking a class as final. But we can still access the methods of this class by composition. Ex: String class
final methods: Method overriding is one of the important features in java. But there are situations where we may not want to use this feature. Then we declared method as final which will print overriding. To allow a method from being overridden we use final access modifier for methods.
final variables : If a variable is declared as final ,it behaves like a constant . We cannot modify the value of final variable. Any attempt to modify the final variable results in compilation error. The error is as follows
"final variable cannot be assigned."

49) Explain about abstract classes in java?

Sometimes we may come across a situation where we cannot provide implementation to all the methods in a class. We want to leave the implementation to a class that extends it. In such case we declare a class as abstract.To make a class abstract we use key word abstract. Any class that contains one or more abstract methods is declared as abstract. If we don't declare class as abstract which contains abstract methods we get compile time error. We get the following error.
"The type <class-name> must be an abstract class to define abstract methods."
Signature ; abstract class <class-name>
{
}
For example if we take a vehicle class we cannot provide implementation to it because there may be two wheelers , four wheelers etc. At that moment we make vehicle class abstract. All the common features of vehicles are declared as abstract methods in vehicle class. Any class which extends vehicle will provide its method implementation. It's the responsibility of subclass to provide implementation.
The important features of abstract classes are :
1) Abstract classes cannot be instantiated.
2) An abstract classes contains abstract methods, concrete methods or both.
3) Any class which extends abstract class must override all methods of abstract class.
4) An abstract class can contain either 0 or more abstract methods.

Though we cannot instantiate abstract classes we can create object references . Through superclass references we can point to subclass.

50) Can we create constructor in abstract class ?

We can create constructor in abstract class , it does'nt give any compilation error. But when we cannot instantiate class there is no use in creating a constructor for abstract class.

51) What are abstract methods in java?

An abstract method is the method which does'nt have any body. Abstract method is declared with keyword abstract and semicolon in place of method body.
Signature : public abstract void <method name>();
Ex : public abstract void getDetails();
It is the responsibility of subclass to provide implementation to abstract method defined in abstract class.

Java Exception Handling Interview questions

52) What is an exception in java?

In java exception is an object. Exceptions are created when an abnormal situations are arised in our program. Exceptions can be created by JVM or by our application code. All Exception classes are defined in java.lang. In otherwords we can say Exception as run time error.

53) State some situations where exceptions may arise in java?

1) Accesing an element that does not exist in array.
2) Invalid conversion of number to string and string to number.
(NumberFormatException)
3) Invalid casting of class
(Class cast Exception)
4) Trying to create object for interface or abstract class
(Instantiation Exception)

54) What is Exception handling in java?

Exception handling is a mechanism what to do when some abnormal situation arises in program. When an exception is raised in program it leads to termination of program when it is not handled properly. The significance of exception handling comes here in order not to terminate a program abruptly and to continue with the rest of program normally. This can be done with help of Exception handling.

55) What is an eror in Java?

Error is the subclass of Throwable class in java. When errors are caused by our program we call that as Exception, but some times exceptions are caused due to some environment issues such as running out of memory. In such cases we can't handle the exceptions. Exceptions which cannot be recovered are called as errors in java.
Ex : Out of memory issues.

56) What are advantages of Exception handling in java?

1) Separating normal code from exception handling code to avoid abnormal termination of program.
2) Categorizing in to different types of Exceptions so that rather than handling all exceptions with Exception root class we can handle with specific exceptions. It is recommended to handle exceptions with specific Exception instead of handling with Exception root class.
3) Call stack mechanism : If a method throws an exception and it is not handled immediately, then that exception is propagated or thrown to the caller of that method. This propogation continues till it finds an appropriate exception handler ,if it finds handler it would be handled otherwise program terminates abruptly.

57) In how many ways we can do exception handling in java?

We can handle exceptions in either of the two ways :
1) By specifying try catch block where we can catch the exception.
2) Declaring a method with throws clause .

58) List out five keywords related to Exception handling ?

1) Try
2) Catch
3) throw
4) throws
5) finally

59) Explain try and catch keywords in java?

In try block we define all exception causing code. In java try and catch forms a unit. A catch block catches the exception thrown by preceding try block. Catch block cannot catch an exception thrown by another try block. If there is no exception causing code in our program or exception is not raised in our code jvm ignores the try catch block.

Syntax :

```
try
{
}
Catch(Exception e)
{
}
```

60) Can we have try block without catch block?

Each try block requires atleast one catch block or finally block. A try block without catch or finally will result in compiler error. We can skip either of catch or finally block but not both.

61) Can we have multiple catch block for a try block?

In some cases our code may throw more than one exception. In such case we can specify two or more catch clauses, each catch handling different type of exception. When an exception is thrown jvm checks each catch statement in order and the first one which matches the type of exception is execution and remaining catch blocks are skipped.

Try with multiple catch blocks is highly recommended in java.

If try with multiple catch blocks are present the order of catch blocks is very important and the order should be from child to parent.

62) Explain importance of finally block in java?

Finally block is used for cleaning up of resources such as closing connections, sockets etc. if try block executes with no exceptions then finally is called after try block without executing catch block. If there is exception thrown in try block finally block executes immediately after catch block.

If an exception is thrown,finally block will be executed even if the no catch block handles the exception.

63) Can we have any code between try and catch blocks?

We shouldn't declare any code between try and catch block. Catch block should immediately start after try block.

```
try{
//code
}
System.out.println("one line of code");  // illegal
catch(Exception e){
//
}
```

64) Can we have any code between try and finally blocks?

We shouldn't declare any code between try and finally block. finally block should immediately start after catch block.If there is no catch block it should immediately start after try block.

```
try{
//code
}
System.out.println("one line of code");  // illegal
finally{
//
}
```

65) Can we catch more than one exception in single catch block?

From Java 7, we can catch more than one exception with single catch block. This type of handling reduces the code duplication.

Note : When we catch more than one exception in single catch block , catch parameter is implicity final. We cannot assign any value to catch parameter.

```
Ex : catch(ArrayIndexOutOfBoundsException || ArithmeticException e)
{
```

}
In the above example e is final we cannot assign any value or modify e in catch statement.

66) What are checked Exceptions?

1) All the subclasses of Throwable class except error,Runtime Exception and its subclasses are checked exceptions.
2) Checked exception should be thrown with keyword throws or should be provided try catch block, else the program would not compile. We do get compilation error.
Examples :
1) IOException,
2) SQlException,
3) FileNotFoundException,
4) InvocationTargetException,
5) CloneNotSupportedException
6) ClassNotFoundException
7) InstantiationException

67) What are unchecked exceptions in java?

All subclasses of RuntimeException are called unchecked exceptions. These are unchecked exceptions because compiler does not checks if a method handles or throws exceptions.
Program compiles even if we do not catch the exception or throws the exception.
If an exception occurs in the program,program terminates . It is difficult to handle these exceptions because there may be many places causing exceptions.
Example : 1) Arithmetic Exception
3) ArrayIndexOutOfBoundsException
4) ClassCastException
5) IndexOutOfBoundException
6) NullPointerException
7) NumberFormatException
8) StringIndexOutOfBounds
9) UnsupportedOperationException

68) Explain differences between checked and Unchecked exceptions in java?

Unchecked Exception	Checked Exception
1) All the subclasses of RuntimeException are called unchecked exception.	All subclasses of Throwable class except RuntimeException are called as checked exceptions
2) Unchecked exceptions need not be handled at compile time	Checked Exceptions need to be handled at compile time.
3) These exceptions arise mostly due to coding mistakes in our program.	
4) ArrayIndexOutOfBoundsException, ClassCastException, IndexOutOfBoundException	SqlException, FileNotFoundException,ClassNotFoundException

69) What is default Exception handling in java?

When JVM detects exception causing code, it constructs a new exception handling object by including the following information.
1) Name of Exception
2) Description about the Exception
3) Location of Exception.
After creation of object by JVM it checks whether there is exception handling code or not. If there is exception handling code then exception handles and continues the program. If there is no exception handling code JVM give the responsibility of exception handling to default handler and terminates abruptly.
Default Exception handler displays description of exception,prints the stacktrace and location of exception and terminates the program.
Note : The main disadvantage of this default exception handling is program terminates abruptly.

70) Explain throw keyword in java?

Generally JVM throws the exception and we handle the exceptions by using try catch block. But there are situations where we have to throw userdefined exceptions or runtime exceptions. In such case we use throw keyword to throw exception explicitly.
Syntax : throw throwableInstance;
Throwable instance must be of type throwable or any of its subclasses.
After the throw statement execution stops and subsequent statements are not executed. Once exception object is thrown JVM checks is there any catch block to handle the exception. If not then the next catch statement till it finds the appropriate handler. If appropriate handler is not found ,then default exception handler halts the program and prints the description and location of exception.
In general we use throw keyword for throwing userdefined or customized exception.

71) Can we write any code after throw statement?

After throw statement jvm stop execution and subsequent statements are not executed. If we try to write any statement after throw we do get compile time error saying unreachable code.

72) Explain importance of throws keyword in java?

Throws statement is used at the end of method signature to indicate that an exception of a given type may be thrown from the method.
The main purpose of throws keyword is to delegate responsibility of exception handling to the caller methods, in the case of checked exception.
In the case of unchecked exceptions, it is not required to use throws keyword.
We can use throws keyword only for throwable types otherwise compile time error saying incompatible types.
An error is unchecked , it is not required to handle by try catch or by throws.
Syntax : Class Test{
Public static void main(String args[]) throws IE
{
}
}
Note : The method should throw only checked exceptions and subclasses of checked exceptions.
It is not recommended to specify exception superclasses in the throws class when the actual exceptions thrown in the method are instances of their subclass.

73) Explain the importance of finally over return statement?

finally block is more important than return statement when both are present in a program. For example if there is any return statement present inside try or catch block , and finally block is also present first finally statement will be executed and then return statement will be considered.

74) Explain a situation where finally block will not be executed?

Finally block will not be executed whenever jvm shutdowns. If we use system.exit(0) in try statement finally block if present will not be executed.

75) Can we use catch statement for checked exceptions?

If there is no chance of raising an exception in our code then we can't declare catch block for handling checked exceptions .This raises compile time error if we try to handle checked exceptions when there is no possibility of causing exception.

76) What are user defined exceptions?

To create customized error messages we use userdefined exceptions. We can create user defined exceptions as checked or unchecked exceptions.
We can create user defined exceptions that extend Exception class or subclasses of checked exceptions so that userdefined exception becomes checked.
Userdefined exceptions can extend RuntimeException to create userdefined unchecked exceptions.
Note : It is recommended to keep our customized exception class as unchecked,i.e we need to extend Runtime Exception class but not Excpetion class.

77) Can we rethrow the same exception from catch handler?

Yes we can rethrow the same exception from our catch handler. If we want to rethrow checked exception from a catch block we need to declare that exception.

78) Can we nested try statements in java?

Yes try statements can be nested. We can declare try statements inside the block of another try statement.

79) Explain the importance of throwable class and its methods?

Throwable class is the root class for Exceptions. All exceptions are derived from this throwable class. The two main subclasses of Throwable are Exception and Error. The three methods defined in throwable class are :

1) void printStackTrace() :

This prints the exception information in the following format :

Name of the exception, description followed by stack trace.

2) getMessage()

This method prints only the description of Exception.

3) toString():

It prints the name and description of Exception.

80) Explain when ClassNotFoundException will be raised ?

When JVM tries to load a class by its string name, and couldn't able to find the class classNotFoundException will be thrown. An example for this exception is when class name is misspelled and when we try to load the class by string name hence class cannot be found which raises ClassNotFoundException.

81) Explain when NoClassDefFoundError will be raised ?

This error is thrown when JVM tries to load the class but no definition for that class is found NoClassDefFoundError will occur. The class may exist at compile time but unable to find at runtime. This might be due to misspelled classname at command line, or classpath is not specified properly , or the class file with byte code is no longer available.

Java Interview questions on threads

83) What is process ?

A process is a program in execution.

Every process have their own memory space.Process are heavy weight and requires their own address space. One or more threads make a process.

84) What is thread in java?

Thread is separate path of execution in program.

Threads are

1) Light weight
2) They share the same address space.
3) creating thread is simple when compared to process because creating thread requires less resources when compared to process
4) Threads exists in process. A process have atleast one thread.

85) Difference between process and thread?

Process	Thread
1) Program in execution.	Separate path of execution in program. One or more threads is called as process.
2) Processes are heavy weight	Threads are light weight.
3) Processes require separate address space.	Threads share same address space.
4) Interprocess communication is expensive.	Interthread communication is less expensive compared to processes.
5) Context switching from one process to another is costly.	Context switching between threads is low cost.

86) What is multitasking ?

Multitasking means ***performing more than one activity at a time*** on the computer. Example Using spreadsheet and using calculator at same time.

87) What are different types of multitasking?

There are two different types of multitasking :

1) Process based multitasking
2) Thread based multitasking

Process based multitasking : It allows to ***run two or more programs concurrently***. In process based multitasking a process is the smallest part of code .
Example : Running Ms word and Ms powerpoint at a time.

Thread based multitasking : It allows to ***run parts of a program to run concurrently***.
Example : Formatting the text and printing word document at same time .
Java supports thread based multitasking and provides built in support for multithreading.

88) What are the benefits of multithreaded programming?

Multithreading enables to use idle time of cpu to another thread which results in faster execution of program. In single threaded environment each task has to be completed before proceeding to next task making cpu idle.

89) Explain thread in java?

1) Thread is independent path of execution with in a program.
2) A thread consists of three parts Virtual Cpu, Code and data.
3) At run time threads share code and data i.e they use same address space.
4) Every thread in java is an object of java.lang.Thread class.

90) List Java API that supports threads?

java.lang.Thread : This is one of the way to create a thread. By extending Thread class and overriding run() we can create thread in java.
java.lang.Runnable : Runnable is an interface in java. By implementing runnable interface and overriding run() we can create thread in java.
java.lang.Object : Object class is the super class for all the classes in java. In object class we have three methods wait(), notify(), notifyAll() that supports threads.
java.util.concurrent : This package has classes and interfaces that supports concurrent programming.
Ex : Executor interface, Future task class etc.

91) Explain about main thread in java?

Main thread is the first thread that starts immediately after a program is started.
Main thread is important because :
1) All the child threads spawn from main thread.
2) Main method is the last thread to finish execution.

When JVM calls main method() it starts a new thread. Main() method is temporarily stopped while the new thread starts running.

92) In how many ways we can create threads in java?

We can create threads in java by any of the two ways :
1) By ***extending Thread*** class
2) By ***Implementing Runnable*** interface.

93) Explain creating threads by implementing Runnable class?

This is first and foremost way to create threads . By implementing runnable interface and implementing run() method we can create new thread.
Method signature : public void run()
Run is the starting point for execution for another thread within our program.
Example :

```
public class MyClass implements Runnable {
@Override
public void run() {
// T
}
}
```

94) Explain creating threads by extending Thread class ?

We can create a thread by extending Thread class. The class which extends Thread class must override the run() method.
Example :

```
public class MyClass extends Thread {
@Override
public void run() {
```

```
// Starting point of Execution
}
}
```

95) Which is the best approach for creating thread ?

The best way for creating threads is to implement runnable interface.
When we extend Thread class we can't extend any other class.
When we create thread by implementing runnable interface we can implement Runnable interface. In both ways we have to implement run() method.

96) Explain the importance of thread scheduler in java?

Thread scheduler is part of JVM use to determine which thread to run at this moment when there are multiple threads. Only threads in runnable state are choosen by scheduler.
Thread scheduler first allocates the processor time to the higher priority threads. To allocate microprocessor time in between the threads of the same priority, thread scheduler follows round robin fasion.

97) Explain the life cycle of thread?

A thread can be in any of the five states :
1) New : When the instance of thread is created it will be in New state.
Ex : Thread t= new Thread();
In the above example t is in new state. The thread is created but not in active state to make it active we need to call start() method on it.
2) Runnable state : A thread can be in the runnable state in either of the following two ways :
a) When the start method is invoked or
b) A thread can also be in runnable state after coming back from blocked or sleeping or waiting state.
3) Running state : If thread scheduler allocates cpu time, then the thread will be in running state.
4) Waited /Blocking/Sleeping state:
In this state the thread can be made temporarily inactive for a short period of time. A thread can be in the above state in any of the following ways:
1) The thread waits to acquire lock of an object.
2) The thread waits for another thread to complete.
3) The thread waits for notification of other thread.

5) Dead State : A thread is in dead state when thread's run method execution is complete. It dies automatically when thread's run method execution is completed and the thread object will be garbage collected.

98) Can we restart a dead thread in java?

If we try to restart a dead thread by using start method we will get run time exception since the thread is not alive.

99) Can one thread block the other thread?

No one thread cannot block the other thread in java. It can block the current thread that is running.

100) Can we restart a thread already started in java?

A thread can be started in java using start() method in java. If we call start method second time once it is started it will cause RunTimeException(IllegalThreadStateException). A runnable thread cannot be restarted.

101) What happens if we don't override run method ?

If we don't override run method .Then default implementation of Thread class run() method will be executed and hence the thread will never be in runnable state.

102) Can we overload run() method in java?

We can overload run method but Thread class start method will always cal run method with no arguments. But the overloaded method will not be called by start method we have to explicitly call this start() method.

105) What is a lock or purpose of locks in java?

Lock also called monitor is used to prevent access to a shared resource by multiple threads.
A lock is associated to shared resource. Whenever a thread wants to access a shared resource if must first acquire a lock . If already a lock has been acquired by other it can't access that shared resource. At this

moment the thread has to wait until another thread releases the lock on shared resource. To lock an object we use synchronization in java.
A lock protects section of code allowing only one thread to execute at at a time.

106) In how many ways we can do synchronization in java?

There are two ways to do synchronization in java:
1) Synchronized methods
2) Synchronized blocks
To do synchronization we use synchronize keyword.

107) What are synchronized methods ?

If we want a method of object to be accessed by single thread at a time we declare that method with synchronized keyword.
Signature :

```
public synchronized  void methodName(){}
```

To execute synchronized method first lock has to be acquired on that object. Once synchronized method is called lock will be automatically acquired on that method when no other thread has lock on that method. once lock has been acquired then synchronized method gets executed. Once synchronized method execution completes automatically lock will be released. The prerequisite to execute a synchronized method is to acquire lock before method execution. If there is a lock already acquired by any other thread it waits till the other thread completes.

108) When do we use synchronized methods in java?

If multiple threads tries to access a method where method can manipulate the state of object , in such scenario we can declare a method as synchronized.

109) When a thread is executing synchronized methods , then is it possible to execute other synchronized methods simultaneously by other threads?

No it is not possible to execute synchronized methods by other threads when a thread is inside a synchronized method.

110) When a thread is executing a synchronized method , then is it possible for the same thread to access other synchronized methods of an object ?

Yes it is possible for thread executing a synchronized method to execute another synchronized method of an object.

```
public synchronized  void methodName()

{
}
```

To execute synchronized method first lock has to be acquired on that object. Once synchronized method is called lock will be automatically acquired on that method when no other thread has lock on that method. once lock has been acquired then synchronized method gets executed. Once synchronized method execution completes automatically lock will be released. The prerequisite to execute a synchronized method is to acquire lock before method execution. If there is a lock already acquired by any other thread it waits till the other thread completes.

111) What are synchronized blocks in java?

Synchronizing few lines of code rather than complete method with the help of synchronized keyword are called synchronized blocks.
Signature :

```
Synchronized (object reference){// code}
```

112) When do we use synchronized blocks and advantages of using synchronized blocks?

If very few lines of code requires synchronization then it is recommended to use synchronized blocks. The main advantage of synchronized blocks over synchronized methods is it reduces the waiting time of threads and improves performance of the system.

113) What is class level lock ?

Acquiring lock on the class instance rather than object of the class is called class level lock. The difference between class level lock and object level lock is in class level lock lock is acquired on class .class instance and in object level lock ,lock is acquired on object of class.

114) Can we synchronize static methods in java?

Every class in java has a unique lock associated with it. If a thread wants to execute static synchronize method it need to acquire first class level lock. When a thread was executing static synchronized method no other thread can execute static synchronized method of class since lock is acquired on class.
But it can execute the following methods simultaneously :

1) Normal static methods
2) Normal instance methods
3) synchronize instance methods

Signature :

```
synchronized(Classname.class){}
```

115) Can we use synchronized block for primitives?

Synchronized blocks are applicable only for objects if we try to use synchronized blocks for primitives we get compile time error.

116) What are thread priorities and importance of thread priorities in java?

When there are several threads in waiting, thread priorities determine which thread to run. In java programming language every thread has a priority. A thread inherits priority of its parent thread. By default thread has normal priority of 5. Thread scheduler uses thread priorities to decide when each thread is allowed to run. Thread scheduler runs higher priority threads first.

117) Explain different types of thread priorities ?

Every thread in java has priorities in between 1 to 10. By default priority is 5 (Thread.NORM_PRIORITY). The maximum priority would be 10 and minimum would be 1.Thread class defines the following constants(static final variables) to define properties.
Thread. MIN_PRIORITY = 1;
Thread.NORM_PRIORITY=5;
Thread. MAX_PRIORITY=10;

118) How to change the priority of thread or how to set priority of thread?

Thread class has a set method to set the priority of thread and get method to get the priority of the thread.
Signature : final void setPriority(int value);
The setPriority() method is a request to jvm to set the priority. JVM may or may not oblige the request.
We can get the priority of current thread by using getPriority() method of Thread class.

```
final int getPriority()
{
}
```

119) If two threads have same priority which thread will be executed first ?

We are not guaranteed which thread will be executed first when there are threads with equal priorities in the pool. It depends on thread scheduler to which thread to execute. The scheduler can do any of the following things :

1) It can pick any thread from the pool and run it till it completes.
2) It can give equal opportunity for all the threads by time slicing.

120) What all methods are used to prevent thread execution ?

There are three methods in Thread class which prevents execution of thread.

1) yield()
2) join()
3) sleep()

121) Explain yield() method in thread class ?

Yield() method makes the current running thread to move in to runnable state from running state giving chance to remaining threads of equal priority which are in waiting state. yield() makes current thread to sleep for a specified amount of time.There is no guarantee that moving a current running thread from runnable to running state. It all depends on thread scheduler it doesn't gurantee anything.

Calling yield() method on thread does not have any affect if object has a lock. The thread does'nt lose any lock if it has acquired a lock earlier.

Signature :

```
public static native void yield()
{
_
}
```

122) Is it possible for yielded thread to get chance for its execution again ?

Yield() causes current thread to sleep for specified amount of time giving opportunity for other threads of equal priority to execute. Thread scheduler decides whether it get chance for execution again or not. It all depends on mercy of thread scheduler.

123) Explain the importance of join() method in thread class?

A thread can invoke the join() method on other thread to wait for other thread to complete its execution. Assume we have two threads, t1 and t2 threads . A running thread t1 invokes join() on thread t2 then t1 thread will wait in to waiting state until t2 completes. Once t2 completes the execution, t1 will continue.
join() method throws Interrupted Exception so when ever we use join() method we should handle Interrrupted Exception by throws or by using try catch block.
Signature :

```
public final void join() throws InterruptedException
{

}
```

```
public final synchronized void join(long millis)
throws InterruptedException
{

}
```

```
public final synchronized void join(long millis, int nanos)
throws InterruptedException
{

}
```

124) Explain purpose of sleep() method in java?

sleep() method causes current running thread to sleep for specified amount of time . sleep() method is the minimum amount of the time the current thread sleeps but not the exact amount of time.

Signature :

```
  public static native void sleep(long millis) throws InterruptedException
{

}
```

```

public static void sleep(long millis, int nanos)
throws InterruptedException {


}
```

125) Assume a thread has lock on it, calling sleep() method on that thread will release the lock?

Calling sleep() method on thread which has lock does'nt affect. Lock will not be released though the thread sleeps for a specified amount of time.

126) Can sleep() method causes another thread to sleep?

No sleep() method causes current thread to sleep not any other thread.

127) Explain about interrupt() method of thread class ?

Thread class interrupt() method is used to interrupt current thread or another thread. It doesnot mean the current thread to stop immediately, it is polite way of telling or requesting to continue your present work. That is the reason we may not see the impact of interrupt call immediately.
Initially thread has a boolean property(interrupted status) false. So when we call interrupt() method status would set to true. This causes the current thread to continue its work and does not have impact immediately.
If a thread is in sleeping or waiting status (i.e thread has executed wait () or sleep() method) thread gets interrupted it stops what it is doing and throws an interrupted exception. This is reason we need to handle interrupted exception with throws or try/ catch block.

128) Explain about interthread communication and how it takes place in java?

Usually threads are created to perform different unrelated tasks but there may be situations where they may perform related tasks. Interthread communication in java is done with the help of following three methods :

1) wait()
2) notify()
3) notifyAll()

129) Explain wait(), notify() and notifyAll() methods of object class ?

wait() : wait() method() makes the thread current thread sleeps and releases the lock until some other thread acquires the lock and calls notify().
notify() :notify() method wakes up the thread that called wait on the same object.
notfiyAll() :notifyAll() method wakes up all the threads that are called wait() on the same object. The highest priority threads will run first.
All the above three methods are in object class and are called only in synchronized context.
All the above three methods must handle InterruptedException by using throws clause or by using try catch clause.

130) Explain why wait() , notify() and notifyAll() methods are in Object class rather than in thread class?

First to know why they are in object class we should know what wait(), notify(), notifyAll() methods do. wait() , notify(), notifyAll() methods are object level methods they are called on same object.wait(), notify(), notifyAll() are called on an shared object so to they are kept in object class rather than thread class.

131) Explain IllegalMonitorStateException and when it will be thrown?

IllegalMonitorStateException is thrown when wait(), notify() and notifyAll() are called in non synchronized context. Wait(), notify(),notifyAll() must always be called in synchronized context other wise we get this run time exception.

132) when wait(), notify(), notifyAll() methods are called does it releases the lock or holds the acquired lock?

wait(), notify(), notifyAll() methods are always called in synchronized context. When these methods are called in synchronized context.
So when they enter first in synchronized context thread acquires the lock on current object. When wait(), notify(), notifyAll() methods are called lock is released on that object.

133) Explain which of the following methods releases the lock when yield(), join(),sleep(),wait(),notify(), notifyAll() methods are executed?

Method	Releases lock (Yes or No)
yield()	No
sleep()	No
join()	No
wait()	Yes

Notify()	Yes
notifyAll()	Yes

134) What are thread groups?

Thread Groups are group of threads and other thread groups. It is a way of grouping threads so that actions can be performed on set of threads for easy maintenance and security purposes.
For example we can start and stop all thread groups. We rarely use thread group class. By default all the threads that are created belong to default thread group of the main thread. Every thread belongs to a thread group. Threads that belong to a particular thread group cannot modify threads belonging to another thread group.

135) What are thread local variables ?

Thread local variables are variables associated to a particular thread rather than object. We declare ThreadLocal object as private static variable in a class. Everytime a new thread accesses object by using getter or setter we are accesing copy of object. Whenever a thread calls get or set method of ThreadLocal instance a new copy is associated with particular object.

136) What are daemon threads in java?

Daemon threads are threads which run in background. These are service threads and works for the benefit of other threads. Garbage collector is one of the good example for daemon threads.
By default all threads are non daemon. Daemon nature of a thread can be inherited. If parent thread is daemon , child thread also inherits daemon nature of thread.

137) How to make a non daemon thread as daemon?

By default all threads are non daemon. We can make non daemon nature of thread to daemon by using setDaemon() method. The important point to note here we can call setDaemon() only before start() method is called on it. If we call setDaemon() after start() method an IllegalThreadStateException will be thrown.

138) Can we make main() thread as daemon?

Main thread is always non daemon. We cannot change the non daemon nature of main thread to daemon.

Interview questions on Nested classses and inner classes

139) What are nested classes in java?

Class declared with in another class is defined as nested class.
There are two types of nested classes in java.
1) Static nested class
2) Non static nested class
A static nested class has static keyword declared before class definition.

140) What are inner classes or non static nested classes in java?

Nested classes without any static keyword declaration in class definition are defined as non static nested classes. Generally non static nested classes are referred as inner classes.
There are three types of inner classes in java :
1) Local inner class
2) Member inner class
3) Anonymous inner class

141) Why to use nested classes in java?
(or)
What is the purpose of nested class in java?

1) Grouping of related classes

Classes which are not reusable can be defined as inner class instead of creating inner class.
For example : We have a submit button upon click of submit button we need to execute some code. This code is related only to that class and cannot be reused for other class . Instead of creating a new class we can create inner class

2) To increase encapsulation :

Inner class can access private members of outer class.so by creating getter and setter methods for private variables , outside world can access these variables. But by creating inner class private variables can be accesed only by inner class.

3) Code readable and maintainable :

Rather than creating a new class we can create inner class so that it is easy to maintain.

4) Hiding implementation :
Inner class helps us to hide implementation of class.

142) Explain about static nested classes in java?
When a static class is defined inside a enclosing class we define that as nested class. Static nested classes are not inner classes. Static nested classes can be instantiated without instance of outer class.
A static nested doesnot have access to instance variables and non static methods of outer class.

143) How to instantiate static nested classes in java?
We can access static members and static methods of outer class without creating any instance of outer class.
Syntax for instantiating Static nested class :
OuterclassName.StaticNestedClassName ref=new OuterclassName.StaticNestedClassName();

144) Explain about method local inner classes or local inner classes in java?
Nested classes defined inside a method are local inner classes. We can create objects of local inner class only inside method where class is defined. A local inner classes exist only when method is invoked and goes out of scope when method returns.

145) Explain about features of local inner class?
1) Local inner class does not have any access specifier.
2) We cannot use access modifiers static for local inner class. But we can use abstract and final for local inner class.
3) We cannot declare static members inside local inner classes.
4) We can create objects of local inner class only inside method where class is defined.
5) Method local inner classes can only access final variables declared inside a method.
6) Method local inner classes can be defined inside loops(for,while) and blocks such as if etc.

146) Explain about anonymous inner classes in java?
Inner class defined without any class name is called anonymous inner class. Inner class is declared and instantiated using new keyword.The main purpose of anonymous inner classes in java are to provide interface implementation. We use anonymous classes when we need only one instance for a class. We can use all members of enclosing class and final local variables.
When we compile anonymous inner classes compiler creates two files
1) EnclosingName.class
2) EnclsoingName$1.class

147) Explain restrictions for using anonymous inner classes?
1) An anonymous inner class cannot have any constructor because there is no name for class.
2) An anonymous inner class cannot define static methods, fields or classes.
3) We cannot define an interface anonymously.
4) Anonymous inner class can be instantiated only once.

148) Is this valid in java ? can we instantiate interface in java?
```
Runnable r = new Runnable() {
@Override
public void run() {
}
};
```
Runnable is an interface.If we see the above code it looks like we are instantiating Runnable interface. But we are not instantiating interface we are instantiating anonymous inner class which is implementation of Runnable interface.

149) Explain about member inner classes?
Non static class defined with in enclosing class are called member inner class. A member inner class is defined at member level of class. A member inner class can access the members of outer class including private members.
Features of member inner classes :
1) A member inner class can be declared abstract or final.
2) A member inner class can extend class or implement interface.
3) An inner class cannot declare static fields or methods.
4) A member inner class can be declared with public, private, protected or default access.

150) How to instantiate member inner class?

OuterClassName.InnerclassName inner=new OuterClassReference.new InnerClassName();
We cannot instantiate inner class without outer class reference

151) How to do encapsulation in Java?

Make instance variables private.
Define getter and setter methods to access instance variables .

152) What are reference variables in java?

Variables which are used to access objects in java are called reference variables.
Ex : Employee emp=new Employee();
In the above example emp is reference variable.
Reference variable can be of only one type.
A reference variable can point to any number of objects. But if a reference variable is declared final it can't point to other objects.
A reference variable can be declared either to a class type or interface type. If a reference variable is declared with interface type it points to the class that implements the interface.

153) Will the compiler creates a default constructor if I have a parameterized constructor in the class?

No compiler won't create default constructor if there is parameterized constructor in the class. For example if I have a class with no constructors, then compiler will create default constructor.
For Example :

```
public classCar {}
```

In the above Car class there are no constructors so compiler creates a default constructor.

```
public classCar {Car(String name) {

}

}
```

In this example compiler won't create any default constructor because already there is one constructor in the Car class.

154) Can we have a method name same as class name in java?

Yes we can have method name same as class name it won't throw any compilation error but it shows a warning message that method name is same as class name.

155) Can we override constructors in java?

Only methods can be overridden in java. Constructors can't be inherited in java. So there is no point of verriding constructors in java.

156) Can Static methods access instance variables in java?

No.Instance variables can't be accessed in static methods. When we try to access instance variable in static method we get compilation error. The error is as follows:

```
Cannot make a static reference to the non static field name
```

157) How do we access static members in java?

Instance variables and instance methods can be accessed using reference variable . But to access static variables or static methods we use Class name in java.

158) Can we override static methods in java?

Static methods can't be overridden. If we have a static method in superclass and subclass with same signature then we don't say that as overriding. We call that as

159) Difference between object and reference?

Reference and object are both different. Objects are instances of class that resides in heap memory. Objects does'nt have any name so to access objects we use references. There is no alternative way to access objects except through references.

Object cannot be assigned to other object and object cannot be passed as an argument to a method. Reference is a variable which is used to access contents of an object. A reference can be assigned to other reference ,passed to a method.

160) Objects or references which of them gets garbage collected?

Objects get garbage collected not its references.

161) How many times finalize method will be invoked ? who invokes finalize() method in java?

Finalize () method will be called only once on object. Before the object gets garbage collected garbage collector will call finalize() method to free the resources. Finalize() method will be called only when object is eligible for garbage collection.

162) Can we able to pass objects as an arguments in java?

Only references can be passed to a method not an object. We cannot pass the objects to a method. The largest amount of data that can passed as parameters are long or double.

163) Explain wrapper classes in java?

Converting primitives to objects can be done with the help of wrapper classes. Prior to java 1.5 we use Wrapper classes to convert primitives to objects. From java 1.5 we have a new feature autoboxing which is used to convert automatically primitives to objects but in wrapper classes programmer has to take care of converting primitives to objects.
Wrapper classes are immutable in java. Once a value is assigned to it we cannot change the value.

164) Explain different types of wrapper classes in java?

For every primitive in java we have corresponding wrapper class. Here are list of wrapper classes available in java.

Primtive	Wrapper Class
boolean	Boolean
int	Integer
float	Float
char	Character
byte	Byte
long	Long
short	Short

165) Explain about transient variables in java?

To save the state of an object to persistent state we use serialization. If we want a field or variable in the object not to be saved, then we declare that variable or field as transient.

```
Example : public Class Car implements serializable
{
transient int carnumber;
}
```

166) Can we serialize static variables in java?

Static variables cannot be serialized in java.

167) What is type conversion in java?

Assigning a value of one type to variable of other type is called type conversion.

```
Example : int a =10;
long b=a;
```

There are two types of conversion in java:

1) Widening conversion
2) Narrowing conversion

168) Explain about Automatic type conversion in java?

Java automatic type conversion is done if the following conditions are met :

1) When two types are compatible

Ex : int, float
int can be assigned directly to float variable.

2) Destination type is larger than source type.

Ex : int, long

Int can be assigned directly to long .Automatic type conversion takes place if int is assigned to long because long is larger datatype than int.
Widening Conversion comes under Automatic type conversion.

169) Explain about narrowing conversion in java?

When destination type is smaller than source type we use narrowing conversion mechanism in java. Narrowing conversion has to be done manually if destination type is smaller than source type. To do narrowing conversion we use cast. Cast is nothing but explicit type conversion.
Example : long a;
byte b;
b=(byte)a;
Note : casting to be done only on valid types otherwise classcastexception will be thrown.

170) Explain the importance of import keyword in java?

Import keyword is used to import single class or package in to our source file.import statement is declared after package decalaration. We use wild character (*) to import package.
Note : After compilation the compiled code does not contain import statement it will be replaced with fully qualified class names

171) Explain naming conventions for packages ?

Sun defined standard naming conventions for packages.
1) Package names should be in small letters.
2) Package name starts with reverse company domain name (excluding www) followed by department and project name and then the name of package.
Example : com.google.sales.employees

172) What is classpath ?

The path where our .class files are saved is referred as classpath.JVM searches for .class files by using the class path specified. Class path is specified by using CLASSPATH environment variable. CLASSPATH environment variable can contain more than one value. CLASSPATH variable containing more than one value is separated by semicolon.
Example to set class path from command prompt :
set CLASSPATH= C:Program FilesJavajdk1.6.0_25bin;.;
only parent directories need to be added to classpath.Java compiler will look for appropriate packages and classes.

173) What is jar ?

Jar stands for java archive file. Jars are created by using Jar.exe tool. Jar files contains .class files, other resources used in our application and manifest file.Manifest file contains class name with main method.jar contains compressed .class files. Jvm finds these .class files without uncompressing this jar.

174) What is the scope or life time of instance variables ?

When object is instantiated using new operator variables get allocated in the memory.instance variables remain in memory till the instance gets garbage collected

175) Explain the scope or life time of class variables or static variables?

Static variables do not belong to instances of the class. We can access static fields even before instantiating the class. Static variable remain in memory till the life time of application.

176) Explain scope or life time of local variables in java?

Local variables are variables which are defined inside a method. When the method is created local variables gets created in stack memory and this variable gets deleted from memory once the method execution is done.

177) Explain about static imports in java?

From Java 5.0 we can import static variables in to source file. Importing static member to source file is referred as static import. The advantage of static import is we can access static variables without class or interface name.
Syntax : import static packagename.classname.staticvariablename;
Ex : import static com.abc.Employee.eno;
To import all static variables from a class in to our source file we use *.
import static com.abc.Employee.*

178) Can we define static methods inside interface?
We can't declare static methods inside interface. Only instance methods are permitted in interfaces.only public and abstract modifiers are permitted for interface methods. If we try to declare static methods inside interface we get compilation error saying
"Illegal modifier for the interface method Classname.methodName(); only public & abstract are permitted".

179) Define interface in java?
Interface is collection of abstract methods and constants. An interface is also defined as pure or 100 percent abstract class.Interfaces are implicitly abstract whether we define abstract access modifier or not. A class implementing interface overrides all the abstract methods defined in interface. Implements keyword is used to implement interface.

180) What is the purpose of interface?
Interface is a contract . Interface acts like a communication between two objects. When we are defining interface we are defining a contract what our class should do but not how it does. An interface does'nt define what a method does. The power of interface lies when different classes that are unrelated can implement interface. Interfaces are designed to support dynamic method resolution at run time.

181) Explain features of interfaces in java?
1) All the methods defined in interfaces are implicitly abstract even though abstract modifier is not declared.
2) All the methods in interface are public whether they are declared as public or not.
3) variables declared inside interface are by default public, static and final.
4) Interfaces cannot be instantiated.
5) we cannot declare static methods inside interface.
6) ` implements' keyword is used to implement interface.
7) Unlike class, interface can extend any number of interfaces.
8) We can define a class inside interface and the class acts like inner class to interface.
9) An interface can extend a class and implement an interface
10) Multiple inheritance in java is achieved through interfaces.

182) Explain enumeration in java?
Enumeration is a new feature from Java 5.0. Enumeration is set of named constants . We use enum keyword to declare enumeration. The values defined in enumeration are enum constants.Each enum constant declared inside a enum class is by default public , static and final.
Example :
```
package javaexamples;
public enum Days {
SUN,MON,TUE,WED,THU,FRI,SAT;
}
```
SUN,MON,TUE,WED,THU,FRI,SAT are enum constants.

183) Explain restrictions on using enum?
1) Enums cannot extend any other class or enum.
2) We cannot instantiate an enum.
3) We can declare fields and methods in enum class. But these fields and methods should follow the enum constants otherwise we get compilation error.

184) Explain about field hiding in java?
If superclass and subclass have same fields subclass cannot override superclass fields. In this case subclass fields hides the super class fields. If we want to use super class variables in subclass we use super keyword to access super class variables.

185) Explain about Varargs in java?
Beginning with Java 5 has a new feature Varargs which allows methods to have variable number of arguments. It simplifies creation of methods when there are more number of arguments. Earlier to java 5 Varargs are handled by creating method with array of arguments.
Ex : public static void main(String[] args)
A variable length argument is specified using ellispses with type in signature. main method with var args is written as follows:
public static void main(String … args)
If no arguments are passes we get array with size 0.There is no need for null check if no arguments are passed.

186) Explain where variables are created in memory?
When we declare variables variables are created in stack. So when the variable is out of scope those variables get garbage collected.

187) Can we use Switch statement with Strings?
Prior to Java 7 we can use only int values and enum constants in Switch .Starting with Java 7 we can use strings in Switch statement. If we use strings in switch statement prior to Java 7 we will get compile time error "only int and enum constants are permitted".

188) In java how do we copy objects?
In Java we cannot copy two objects but by assigning one reference to other we can copy objects. For example if we have a reference r1 that point to object .so when we declare r2=r1, we are assigning reference r1 to r2 so now r2 points to the same object where r1 points. Any changes done by one reference on an object will reflect to other.

Oops concepts interview questions

189) Explain about procedural programming language or structured programming language and its features?
In traditional programming language to solve a problem we use set of procedures. Once the procedures or functions are determined next they concentrate on storing data.

Features :

1) In this top down approach is followed. First procedures were determined and then concentrate on minute details.
2) Concentrate more on functions and procedure rather than data.
3) In traditional programming language procedures manipulate global data without knowing to other procedures.
4) Very little concentration on minute details

The main drawback of traditional programming languages works well only for small problems. But not suitable for larger problems.

Ex : C language, Pascal

190) Explain about object oriented programming and its features?
Java replaced traditional programming language developed in 1970's. In Object oriented programming everything is made up of object. In this language bottom up approach is followed. Each object communicates with other as opposed to traditional view.

Features :

1) In this bottom approach is followed. First concentrates on minute details like creating objects then concentrates on implementation or solving the problem.
2) Concentrate more on data and give less importance for implementation.
3) Objects communicate with each other

The main advantage of object oriented programming language is works well for larger problems.

191) List out benefits of object oriented programming language?
1) Easy maintenance
2) Code reusability
3) Code extendability
4) Reliable

192) Differences between traditional programming language and object oriented programming language?

Traditional Programming language	Object Oriented Programming Language
A program is divided in to modules and procedures.	A program is divided in to number of objects.
Implementation is done through procedures.	Implementation is done through interfaces.
In traditional programming there is no encapsulation all procedures access data.	In oops encapsulation is done by tightly coupling data and behaviour together in class.
Suitable for small programs or problems	Suitable for large programs and complex problems.

193) Explain oops concepts in detail?
Object oriented programming should support these three features :

1) Inheritance
2) Encapsulation

3) Polymorphism

194) Explain what is encapsulation?

Encapsulation is the process of wrapping of code and behaviour in a single unit called class and preventing from misuse is called encapsulation. Encapsulation exposes only part of object which are safe to exposed and remaining part of object is kept secured.
Encapsulation is supported through access control in java. There are four types of access control specifiers(public,private, protected, default) in java which supports encapsulation.
For example tv manufacturers exposes only buttons not all the thousands of electronic components which it is made up of.

195) What is inheritance ?

Inheritance is one of the important feature of object oriented language. Inheriting is the process of acquiring features of others. For example a child acquires the features of their parents.
In java inheritance is the process of inheriting member of existing classes by extending their functionality. The original class is called base class, parent class or super class. The new class derived from parent is called child class, sub class, and derived class.
We use extends keyword in java to extend a class in java. All java classes extend java.lang.Object since object class is the super class for all classes in java.
When we create a new class by using inheritance 'is-a' relationship is formed.

196) Explain importance of inheritance in java?

Reusability :The major advantage of inheritance is code reuse. We can avoid duplicating code by using inheritance. We can place all common state and behaviour in that class , by extending that class we can
Extendability : We can add new functionality to our application without touching the existing code.
For example if we take Ms word we came across number of versions of msword such as word 2003,2007. Everytime they won't write new code they reuse the existing code and some more features.

197) What is polymorphism in java?

Polymorphism is combination of two greek words which mean many forms. In polymorphism actual type of object involved in method call determines which method to call rather type of reference variable.
59) What is covariant return ?
In java 1.4 and earlier one method can override super class method if both methods have same signature and return types.
From Java 1.5 , a method can override other method if argument types match exactly though return types are different.(Return type must be subtype of other method).
Example : Class A

```
{

A doSomeThing()
{
return new A();
}
}
```

Example : Class B

```
{

B doSomeThing()
{
return new B();
}
}
```

From java 1.5 return type for doSomeThing() in Class B is valid . We get compile time error in 1.4 and earlier.

Collection Framework interview questions

198) What is collections framework ?

A framework is set of classes and interfaces to build a functionality. Java collections framework provides set of interfaces and classes for storing and manipulating collections. Collection framework contains classes and interfaces in java.util package and java.util.concurrent packages.
Advantages or benefits of Collections framework :
1) High performance
2) Using this framework we can create different types of collections

3) We can create our own collection and we can extend a collection.
4) Reduces programming effort.
5) Increases speed and quality : Collections framework provides high performance, implementations of useful data structures and algorithms.

199) What is collection ?

A collection is a container which holds group of objects. Collection provides a way to manage objects easily. Collections manages group of objects as single unit.
Examples include list of strings, integers etc.
Here are few basic operations we do on collections :
1) Adding objects to collection.
2) Removing or deleting objects from collection.
3) Retrieving object from collection.
4) Iterating collection.

200) Difference between collection, Collection and Collections in java?

collection : represent group of objects where objects are stored.
Collection : This is one of the core interface which provides basic functionality for collection.
Collections : Collections contains some utility static methods that operate on collections.

201) Explain about Collection interface in java ?

Collection is the fundamental and root interface in Collections framework. Collection extends Iterable interface and inherits iterator method which returns Iterator object.
Signature :

```
public interface Collection<E> extends Iterable<E> {
}
```

Methods in Collection interface :

boolean add(E e);	Adds an element to the collection. Returns true if element is added.
boolean remove(Object o);	Removes an object from collection if that object is present in collection. Return true if matching object is removed from collection.
boolean addAll(Collection<? extends E> c);	Adds all the elements specified in the collection to this collection.Returns true if all elements are added.
boolean removeAll(Collection<?> c);	Removes all the elements from this collection that are specified in other collection.Returns true if all the elements are removed.
int size();	Returns number of elements in collection.
boolean isEmpty();	Checks whether collection contains elements or not. If no elements are present it returns false.
boolean contains(Object o);	Checks whether specified object is in collection or not. Return true if object is in collection.
Iterator<E> iterator();	Used to iterator over collection. No guarantee on order of elements iterated.
boolean retainAll(Collection<?> c);	Removes all the elements which are not in specified collection. Returns only elements specified in collection removing other elements.
Object[] toArray();	Returns an array of elements in collection.

202) List the interfaces which extends collection interface ?

1) List
2) Set
3) Queue
4) Deque (From Java 6)

203) Explain List interface ?

List interface extends collection interface used to store sequence of elements in collection.
We can even store duplicate elements in list.
We can insert or access elements in list by using index as we do in arrays.
List is an ordered collection.
The main difference between List and non list interface are methods based on position.

Some of the operations we can perform on List :
1) Adding an element at specified index.
2) Removing an element at specified index.
3) To get the index of element
List contains some specific methods apart from Collection interface methods.

204) Explain methods specific to List interface ?

boolean addAll(int index, Collection<? extends E> c);	This method inserts all the elements in specified collection to the list at specified position.
E get(int index);	This method returns an element at specified position in the list.
E set(int index, E element);	This method replaces the element at specified position in the list with the specified element.
void add(int index, E element);	This method inserts the specified element with the index specified.
E remove(int index);	This method removes the element at specified index and returns the element removed.
int indexOf(Object o);	indexOf() method returns the index of last occurrence of specified element. If there is no element in the list it removes the element.
ListIterator<E> listIterator();	Returns a list iterator of elements in list.
List<E> subList(int fromIndex, int toIndex);	This method returns list of elements between indexes specified.

205) List implementations of List Interface ?

1) ArrayList
2) Vector
3) LinkedList

206) Explain about ArrayList ?

ArrayList is an ordered collection which extends AbstractList and implements List interface.
We use ArrayList mainly when we need faster access and fast iteration of elements in list.
We can insert nulls in to arraylist.
Arraylist is nothing but a growable array.

```
public class ArrayList<E> extends AbstractList<E>    implements List<E>,
RandomAccess, Cloneable, java.io.Serializable{}
```

From java 1.4 ArrayList implements RandomAccess interface which is a marker interface which supports fast and random access.

Advantages :

1) Faster and easier access.
2) Used for Random access of elements.

Drawbacks :

1) We cannot insert or delete elements from middle of list.

207) Difference between Array and ArrayList ?

Arrays are used to store primitives or objects of same type or variables that are subclasses of same type.
ArrayList : It is an ordered collection which grows dynamically.
In list we can insert nulls values and list allows duplicate elements.

ARRAY	ARRAY LIST
1) While creating array we have to know the size.	1) But it is not required to know size while creating ArrayList, because arraylist grows dynamically.
2) To put an element in to array we use the following syntax :String array[] = **new**String[5];array[1] = "java";We must know specific location to insert an element in to	2) We can add element to arraylist with following syntax :List<String> stringList = new ArrayList<String>();stringList.add("java");

array. If we try to put element in index which is out of range we get ArrayIndexOutOfBounds Exception	
3) Arrays are static	3) ArrayList is dynamic
4) We can store objects and primitives	4) We can store only primitives prior to 1.5 . From 1.5 we can store even objects also.
5) We have to manually write logic for inserting and removing elements.	5) Just a method call would add or remove elements from list.
6) Arrays are faster	6) Arraylist is slower.
	7) Arraylist is implemented using arrays.

208) What is vector?

Vector is similar to arraylist used for random access.
Vector is a dynamic array like arraylist.
vector size increases or decreases when elements are added and removed .
Vector is synchronized .
vector and Hashtable are the only collections since 1.0.
Rest of the collections are added from 2.0.

```
public class Vector<E>extends AbstractList<E>implements List<E>,
RandomAccess, Cloneable, java.io.Serializable
```

209) Difference between arraylist and vector ?

Both ArrayList and vector grows dynamically. The differences between arraylist and vector are :
1) Arraylist is not synchronized and vector is synchronized.
2) Vector is legacy collection introduced in 1.0 and Arraylist introduced in java 2.0.
Performance wise it is recommended to use arraylist rather than vector because by default vector is synchronized which reduces performance if only one thread accesses it.

210) Define Linked List and its features with signature ?

Linked list is used for storing a collection of objects that allows efficient addition and removal of elements in the middle of the collection.
The main drawback with arrays is if we want to insert an element in the middle of the list we need to move each element to next position and insert the element. Similarly with remove if we want to remove an element we need to remove the element and move the list of elements.
But with linked list we can insert and delete in the middle of the list efficiently by just updating the neighbouring node reference.
Linked list class is in java.util package.
Linked List class extends class extends AbstractSequentialList and I mplements List, Deque, Cloneable and Serializable.

```
Signature :public class LinkedList<E>    extends
AbstractSequentialList<E>
   implements List<E>, Deque<E>, Cloneable, java.io.Serializable
{
}
```

Important methods specific to LinkedList class :
1) public E getFirst() :
getFirst() will returns the first element in the list.

2) public E getLast() :
getLast() returns the last element in the list.

3) public E removeFirst() :
removeFirst() method removes the first element in the list.

4) public E removeLast() :

removeLast() method removes the last element in the list.

5) public void addFirst(E e)
Inserts the element at beginning of the list.

6) public void addLast(E e) :
Inserts the element at end of the list.

211) Define Iterator and methods in Iterator?

If we want to iterate through all the elements in collection we use Iterator. Iterator is a standard way to access elements one by one in collection. Iterator is an object associated with collection used to loop through the collection.
Steps for accessing elements in Iterator :
1) Obtain Iterator object by calling iterator() method on collection.
Ex : ArrayList <String> al=new ArrayList<String>();
Iterator itr=al.iterator();
2) Call hasNext() method on iterator object in loop as long as hasNext() returns true.
Ex : while(itr.hasNext())
{
}
3) Get each element by calling next() inside the loop.

```
while(itr.hasNext())
{
String str=itr.next();
}
```

Methods in iterator :

Method	Description
boolean hasNext();	This method returns true if there is next element.hasNext() points to position before first lement.If there are any elements if will return true.
E next();	Returns the next element in the iteration. . If there are no elements in the Iteration NoSuchElementException is thrown. next() will move the pointer to next position and returns the element.
void remove();	Removes the element.

Note : If we call next() on last element it will throw java.util.NoSuchElementException. So before calling next() first we should call hasNext() whether it has elements or not. If there is next element we can call next() so that we can avoid exception.

212) In which order the Iterator iterates over collection?

The order in which Iterator will iterate the collection depends on the traversal order of collection.
For example : for list traversal order will be sequential, and for set the order cannot be determined, and for sorted sorted set will sort the elements in sorted order.
So it all depends on the collection in which order iterator iterates.

212) Explain ListIterator and methods in ListIterator?

List Iterator is similar to Iterator but ListIterator is bidirectional.
We can traverse through the collection in either forward or backward direction.
List Iterator extends Iterator and all the methods in Iterator will be there in ListIterator too with some additional methods .
List Iterator doesn't have current element .Position of List Iterator lies between two elements i.e previous element and next element.
Features of ListIterator :
1) Traversal of List in either direction.
2) Modification of its elements.
3) Access to elements position.

Signature :

```
public interface ListIterator<E> extends Iterator<E> {
}
```

ListIterator methods :

Method	Description

Void add(E obj)	Inserts element in to the list infront of the element returned by call to next() and after the element returned by call to next().
boolean hasNext();	Returns true if there are more elements in the list instead of throwing exception if there are no elements.
E next();	Returns the next element . NoSuchElementException is thrown if there is no next element.
_boolean hasPrevious();	Returns true if there are elements when iterating list in reverse direction.
E previous();	Returns the previous element in the list.
int nextIndex();	Returns the index of the element returned by next() method. If there are no elements it returns the size of the list.
int previousIndex();	Returns the index of the element returned by previous() method. If there are no elements it returns the size of the list. Returns -1 if the iterator is at beginning of list.
void remove();	Removes the element that was returned by calling next() or previous(). An Illegal state Exception will be thrown if remove() is called before next() or previous().
void set(E e);	This method replaces an element in the list with the specified element.

213) Explain about Sets ?

A set is a collection which does not allow duplicates. Set internally implements equals() method which doesn't allow duplicates.Adding an duplicate element to a set would be ignored .Set interface is implemented in java.util.set package.Set interface does not have any additional methods . It has only collection methods. A set can contain atmost one null value.
ArrayList is an ordered collection.In arraylists order remains same in which they are inserted. But coming to set it is an unordered collection.

```
public interface Set<E> extends Collection<E> {
}
```

Important operations that can be performed on set :

1) Adding an element to set.
2) Removing an element from set.
3) Check if an element exist in set.
4) Iterating through set.

214) Implementations of Set interface ?

1) HashSet
2) Linked HashSet
3) TreeSet

215) Explain HashSet and its features ?

Hashset implements set interface and extends AbstractSet. Features of Hashset are :

1) It does not allow duplicates.
2) It does not gurantee ordering of elements.
3) It is unsorted and unordered set.
4) Performance wise it is recommended to use hashset when compared to other sets because it internally uses hashing mechanism.
5) Allows insertion of nulls.

Note : For efficiency whenever objects are added to HashSet it need to implement the hashCode() method.

```
public class HashSet<E> extends AbstractSet<E>
implements Set<E>, Cloneable, java.io.Serializable
{
}
```

216) Explain Tree Set and its features?

TreeSet implements navigableSet interface and extends Abstract set.It creates collection that uses tree for storage.
Features of Treeset are :

1) It does not allow duplicates.
2) When we retrieve the elements in treeset we will get elements in sorted order.

public class TreeSet<E> **extends** AbstractSet<E>
implements NavigableSet<E>, Cloneable, java.io.Serializable
{

217) When do we use HashSet over TreeSet?

If we want to search for an element in collection and does not want any sorting order we go for HashSet.
82) When do we use TreeSet over HashSet?
TreeSet is preferred
1) if elements are to be maintained in sorting order.
2) Fast insertion and retrieval of elements.

218) What is Linked HashSet and its features?

LinkedHashSet extends HashSet and implements Set interface.
public class LinkedHashSet<E>
extends HashSet<E>
implements Set<E>, Cloneable, java.io.Serializable {
}
Linked HashSet is similar to HashSet but in linked HashSet we maintain order but in HashSet we don't maintain order. Maintaining order means elements will be retrieved in order which they are inserted.

219) Explain about Map interface in java?

A map is an association of key-value pairs. Both keys and values in map are objects.
Features of map :
1) Maps cannot have duplicate keys but can have duplicate value objects.

220) What is linked hashmap and its features?

LinkedHashMap extends HashMap and implements Map.lLinked hashmap gurantees order of elements . Elements are retrieved in same order they are inserted.Linked HashMap uses internally double linked lists to keep insertion order.
The differences between Hashmap and linked hashmap is
1) LinkedHashMap maintains the insertion order while HashMap doesnot maintain order.
2) HashMap if faster for insertion and deletion of elements when compared to linked hashmap. Linked hashmap is preferred only for faster iteration of elements.
public class LinkedHashMap<K,V>
extends HashMap<K,V>
implements Map<K,V>
{
}

221) What is SortedMap interface?

SortedMap extends Map interface.Sorted Map ***maintains sorted order of keys*** in a map.
By default sorted map ***maintains natural ordering*** if we want custom order we can specify using comparator.
public interface SortedMap<K,V> extends Map<K,V> {
}

222) What is Hashtable and explain features of Hashtable?

Hashtable was available before collection framework.
When collection framework was started Hashtable extends Dictionary class and Map interface.
Hashtable offers a convenient way of ***storing key/ value pairs***.
Hashtable ***does not allow nulls either keys or values***.
Hashtable is ***synchronized***.

223) Difference between HashMap and Hashtable?

Difference	HashMap	Hashtable
Synronization	HashMap is ***not synchronized***.	Hashtable is ***synchronized***.
Nulls	HashMap allows atmost ***one null key and any number of null values***.	Hashtable ***does not allow null values***.
Performance	Since HashMap is not synchronized its performance is ***faster than*** Hashtable.	Performance is ***slower*** when compared to HashMap.
Introduction	HashMap introduced starting from	Hashtable is even before collection

	collection framework.	framework.

224) Difference between arraylist and linkedlist?

Difference	Arraylist	LinkedList
Access	Implements RandomAccess interface we can ***search randomly*** all the elements in the list.	It extends Abstract sequential List interface which provides ***sequential access*** to elements.
Searching and retrieval of elements	***Searching and retrieval*** of elements is ***fast*** since arraylist provides random access.	***Searching and retrieval*** of elements is ***slow*** because of sequential access to elements.
Addition and removal of elements	***Adding and removal of elements in random positions is slow***.For example if we want to add element to middle of the list we have to move the elements in the list and then we need to insert the element. Similarly for removing the element we need to follow the same thing.	***Adding and removal of elements in random positions is fast*** because there is no need of resizing the array just by updating the node structures with new addresses.

225) Difference between Comparator and Comparable in java?

Sno	Comparator	Comparable
1.	Defined in ***java.util package***	Defined in ***java.lang package.***
2.	Comparator interface is used when ***we want to compare two different instances***	Comparable is used ***to compare itself with other instance***.
3.	Comparator is used when we want ***custom sorting***.Ex : If we take employee class sorting by employeeId is natural sorting.	Comparable is used for ***natural sorting*** of objects.Ex : If we take employee class sorting by ename and age we can say as custom sorting.
4.	Should override int compare(T o1, T o2) method which takes two instances.	Should override public int compareTo(T o) method which takes one instance.
5.	For sorting objects we use collections.sort(list,new Comparator);	For sorting objects we use collections.sort(list);

226) What is concurrent hashmap and its features ?

Concurrent HashMap is implemented in *java.util.concurrent* package.
Concurrent HashMap extends Abstract Map and implements concurrent Map.
Concurrent HashMap is used in multi threaded environment.
]It is similar to Hashtable and synchronized version of hashmap but with minor differences.
Concurrent HashMap *does not allow null keys and values*.

227) Difference between Concurrent HashMap and Hashtable and collections.synchronizedHashMap?

Locking Mechansim :ConcurrentHashMap uses completely different hashing mechanism called *lock striping* which offers better concurrency and scalability.
The main advantage of this mechanism is better concurrency instead of synchronizing every method by using common lock which allows only one thread to access at a time, *it allows better concurrency by allowing multiple threads to access*.
ConcurrentModificationException :ConcurrentHashMap provides iterators which doesnot throw concurrent modification exception which allows only one thread to access iterator, *while synchronized map may throw concurrent modification exception*.

228) Explain copyOnWriteArrayList and when do we use copyOnWriteArrayList?

copyOnWriteArrayList is used in multithreaded environment. If we want to iterate over arraylist ,but the arraylist is updated by other threads to prevent concurrent modification exception we have two solutions :
1) First one is we need to synchronize the list manually by using collections.synchronized(list) and iterate over the list in synchronized block to avoid concurrent modification exception.
2) The second one is to use copyOnWriteArrayList which takes care of concurrency.

The advantage of using copyOnWriteArrayList is no need to synchronize list explicitly. So when we use copyOnWriteArrayList when a thread modifies the list while the other thread was iterating it does not modify original list but creates a copy of list with modified contents so that the iterator won't know the modifications made to original list.

229) Explain about fail fast iterators in java?

When iterator iterates over collection, collection should not be modified except by that iterator. Modification means collection cannot be modified by thread when other thread is iterating, if such modification happens a concurrent modification exception will be thrown.Such kind of iterators are fail fast iterators.
Ex : ArrayList,HashSet,HashMap. Almost all the iterators implemented in collections framework are fail fast.

230) Explain about fail safe iterators in java?

Fail safe iterators are iterators which does not throw concurrent modification exception, when one thread modifies collection and other thread in the process of iterating the collection.
It does not throw concurrent modification exception because when other thread was iterating it does not modify original list but creates a copy of list with modified contents so that the iterator won't know the modifications made to original list.
Ex : copyOnWriteArrayList

Core java Serialization interview questions

231) What is serialization in java?

Serialization is the process of converting an object in to bytes, so that it can be transmitted over the network,or stored in a flat file and can be recreated later. Serialized object is an object represented as sequence of bytes that includes objects data, object type, and the types of data stored in the object.

232) What is the main purpose of serialization in java?

The main uses of serialization are :
1) Persistence:
We can write data to a file or database and can be used later by deserializing it.
2) Communication :
To pass an object over network by making remote procedure call.
3) Copying :
We can create duplicates of original object by using byte array.
4) To distribute objects across different JVMs.

233) What are alternatives to java serialization?

XML based data transfer
JSON based data transfer.
XML based data transfer : We can use JIBX or JAXB where we can marshall our object's data to xml and transfer data and then unmarshall and convert to object.
JSON based transfer : We can use json to transfer data.

234) Explain about serializable interface in java?

To implement serialization in java there is an interface defined in java.io package called serializable interface. Java.io.Serializable interface is an marker interface which doesnot contain any any methods. A class implements Serializable lets the JVM know that the instances of the class can be serialized.
Syntax:

```
public interface Serializable {
}
```

235) How to make object serializable in java?

1) Our class must implement serializable interface.If our object contains other objects those class must also implement serializable interface.
2) We use ObjectOutputStream which extends OutputStream used to write objects to a stream.
3) We use ObjectInputStream which extends InputStream used to read objects from stream

236) What is serial version UID and its importance in java?

Serial version unique identifier is a 64 bit long value .This 64 bit long value is a hash code of the class name,super interfaces and member. Suid is a unique id no two classes will have same suid. Whenever an object is serialized suid value will also serialize with it.
When an object is read using ObjectInputStream, the suid is also read. If the loaded class suid does not match with suid read from object stream, readObject throws an InvalidClassException.

237) What happens if we don't define serial version UID ?

If we don't define serial version UID JVM will create one suid for us. But it is recommended to have suid rather than JVM creating because at run time JVM has to compute the hashcode of all the properties of class. This process makes serialization low. We can't serialize static fields one exception to this is suid where suid gets serialized along with the object.
Ex :private static final long serialVersionUID = -5885568094444284875L;

238) Can we serialize static variables in java?

We can't serialize static variables in java. The reason being static variable are class variables that belongs to a class not to object, but serialization mechanism saves only the object state not the class state.

239) When we serialize an object does the serialization mechanism saves its references too?

When we serialize an object even the object it refers must implement serializable then the reference objects also get serialized. If we don't make reference objects serializable then we get NotSerializableException.

240) If we don't want some of the fields not to serialize How to do that?

If we don't want to serialize some fields during serialization we declare those variables as transient. During deserialization transient variables are initialized with default values for primitives and null for object references.

More questions and answers

http://becomejavasenior.com/blog/2015/07/01/327-interview-questions-java-developer/

65+ Most Asked

JAVA

Interview Questions

& Answers

Java Interview Questions for Freshers & Beginners

1. What is Java? Explain its meaning and definition.

It is one of the most popular programming languages today, with its role in the development of web applications, mobile apps, software, gaming systems, as well as server-side technologies.

Java programming has been here for over 20 years now and has been nothing less than a boon for programmers and developers. Famous as an object-oriented programming language, Java is not only multi-purpose, but also a secure, high-performing, and trusted coding language. This is the reason behind its usage in even enterprise-grade software development.

2. The syntax of Java is based on which programming language?

C and C++ programming are the base of the Java syntax.

3. How is Java platform independent?

Among the list of Java interview questions, you can be asked why Java is called platform independent. You must be ready with the correct answer.

The meaning of platform-independent is that you can write a Java program on one machine and execute it on other machines or platforms. It has become possible because of the bytecode, and VMs can handle this code accordingly. So, there will be no issues related to the hardware when it comes to running the code.

4. When was Java developed?

Java was developed in the year 1991.

5. Who developed the Java programming language?

Java was developed by James Gosling. He was a computer scientist based in Canada, and is famous as the founder of Java programming. When he invented Java, he was working at Sun Microsystems, which Oracle later acquired.

6. What does ‘write once run anywhere’ mean in Java?

Write once, run anywhere, or WORA in Java means that it is a coding language where you write a program for some purpose only once and then use it or run it across multiple operating systems. For instance, you can write a program and run it on Windows, macOS, Android, Linux, etc.

“Java’s write once, run anywhere” term was first initiated by Sun Microsystems, where the founder of this language used to work. This characteristic makes Java a portable programming language.

7. What is Java programming used for? Explain its primary applications.

There is a wide range of use cases of Java programming language. Below are its main applications:

- Mobile App Development

Despite the introduction of Kotlin, Java is still used as a reliable programming language for Android app development. This coding language has the software development kits (SDKs) and libraries that are required to develop mobile apps.

- Chatbot Development

Another use of Java is in chatbot development. Smart chatbots that use natural language processing (NLP) can be built using this programming language.

- Development of Games

One of the most important applications of Java is in building games or gaming apps. Some of the world-famous games like Minecraft, Spiral Knights, SimCity, Saints Row 2, Asphalt 3, FIFA 11, Wakfu, Tokyo City Nights and many more are built on Java.

- Cloud Computing

The write once, use anywhere characteristic of Java makes it a highly applicable language for cloud applications as well. Plenty of cloud platforms rely on this programming for a decentralized experience.

- Big Data

Big data platforms or tools heavily depend on Java, and it is considered a language on which the future of big data relies. This is because of its features that enable faster processing of large sets of data.

- Enterprise-grade Web Apps

Enterprise-level apps that are mission-critical are developed using Java programming. Even top brands like Wipro, Google, Infosys, and HCL use it to develop enterprise apps. It is because of its high performance it enables and supports a wide range of server-side technologies.

Some of the most popular web apps built on Java include LinkedIn, IRCTC, and AliExpress.

- Internet of Things (IoT)

In IoT technology, sensors and hardware devices process the data. These things are mostly run using programs written in Java programming language.

- Artificial Intelligence (AI)

As one of the most suitable programming languages for artificial intelligence (AI) projects, Java can be used to develop intelligent solutions. For instance, it is great for building search algorithms, neural networks, ML-based services, deep learning applications, etc.

8. What is Java Virtual Machine (JVM)?

JVM in Java, as the name suggests, is a virtual machine that plays a crucial part in the execution of source code. It works as an abstraction layer between the runtime environment and the hardware.

9. What is Java Runtime Environment (JRE)?

JRE in Java is simply an environment that allows developers or programmers to run Java-based apps on operating systems. You can say that it facilitates the interaction between OS and the program.

Java JRE provides several resources to programmers, such as libraries, JVM, Java Plug-in, Web Start, etc. It is available to download on Windows, Linux, macOS, and Oracle Solaris.

10. What is Java SE (Standard Edition)?

Java Standard Edition, abbreviated as Java SE, is a computing platform on which programmers and developers build and deploy Java-based projects. This platform comes with plenty of Java libraries and APIs, including java.util, java.net, java.math, java.io, and many more.

11. What are operators in Java?

Java operators are simply the symbols used to perform a wide range of different operations. Every operator has its specific operation or functionality.

For instance, you can use the **+** operator for the addition of two values, the **-** operator for subtraction, **the *** operator for multiplication, and the **/** operator for division.

12. What are the different types of Java operators?

You can classify the operators in Java into five categories, as mentioned below:

- Arithmetic operators

These are used for mathematical calculations, or arithmetic operations, to be precise.

+	Addition
-	Subtraction
*	Multiplication
/	Division

%	Modulo

- Unary operators

These operators play a vital role in performing operations related to increasing or decreasing the values.

Unary minus (-)	To make a value negative
Unary plus (+)	Generally not used as values are positive by default
Increment (++)	To increase the value by 1
Decrement (--)	To decrease the value by 1
Inverting (!)	To inverse the value

- Assignment operators

These operators, as the name suggests, are used to assign values to variables in a Java program.

Operator	**Presentation**	**Meaning**
=	X = Y	X = Y
+=	X += Y	X = X + Y
-=	X -= Y	X = X = Y
*=	X *= Y	X = X * Y
%=	X %=	X = X % Y

- Relational or comparison operators

If you want to see the relations between values, then the relational operators in Java are used. For instance, you can check whether the given values are equal to each other, greater/lesser than each other, greater than equal to, or lesser than equal to each other.

If the boolean values don't meet the relational operator criteria, then accordingly, it returns *true* or *false*.

Operator	**Meaning**
==	Is equal to
!=	Is not equal to

>	Is greater than
<	Is lesser than
>=	Is greater than or equal to
<=	Is lesser than or equal to

- Logical operators

The digital electronics field heavily depends on AND and OR gate operations. The same applies in Java for decision-making based on logical AND and OR operators.

Operator	**Meaning**
&&	Logical AND
||	Logical OR
!	Logical NOT

13. What is JIT compiler in Java?

Java JIT compiler or Just-In-Time compiler is one of the key parts of Java Runtime Environment (JRE). Its role is the compilation of bytecode to native machine code when it's in runtime. As a result, the overall performance of the Java apps is optimized.

14. What is a Java class?

A class in Java is a template based on some logic using which we can create multiple objects that follow the same logic as the class. The role of a Java class is to define the data types and methods of the objects.

In simple terms, you can call the class the main category, which includes several items called objects. Here, the objects under a class will have similar properties or characteristics.

For example, if you visit Amazon and browse the “Mobiles” category, it will show you smartphones of all brands and types. But all these smartphones will have similar properties like a camera, some RAM, the ability to make calls, download apps, send messages, etc.

So, here, the ‘Mobiles’ is a class, and all the smartphones are objects.

15. What is a package in Java?

A package in Java is like a folder that holds classes, interfaces, and sub-packages. Here, the point to be noted is that everything in this folder or Java package has some similarity or relation based on their functionality.

The aim of using a Java package is to better organize the workloads, avoid conflicts in names, and control access.

In Java, there are two types of packages:

a. Built-in packages
b. User-defined packages

The built-in packages include java.lang, java.util, java.io, and java.net, which can be used from Java API. Apart from these, you can also build your own packages, which are called user-defined packages.

16. What are keywords in Java?

Keywords in Java programming are actually some predefined words in syntax that a programmer can't use in the form of classes, methods, identifiers, or variables. These are also known as reserved words in Java.

17. How many keywords are there in Java?

There are over 50 keywords in Java. Here is the full list in alphabetical order:

1. abstract
2. assert
3. boolean
4. break
5. byte
6. case
7. catch
8. char
9. class
10. continue
11. const
12. default
13. do
14. double
15. else
16. enum
17. exports
18. extends
19. final
20. finally
21. float
22. for
23. goto
24. if
25. implements
26. import

27. instanceof
28. int
29. interface
30. long
31. module
32. native
33. new
34. package
35. private
36. protected
37. public
38. requires
39. return
40. short
41. static
42. strictfp
43. super
44. switch
45. synchronized
46. this
47. throw
48. throws
49. transient
50. try
51. var
52. void
53. volatile
54. while

18. What are the key features of Java?

It is one of the basic Java interview questions for freshers, and sometimes for experienced professionals as well.

Here are the top 10 features of Java that you must know:

- Simple, Clean, Easy to Learn

One of the best things about Java is that it is easy to learn and understand, even for beginners. Its syntax is simple as it is based on basic languages like C++. The code written in Java is also clean and easy to run.

- Object-oriented Programming Language

Java is completely based on objects, hence it is called an object-oriented programming language.

- Java is Both Compiled and Interpreted

A programming language is generally compiled or it is interpreted. Only rare languages exhibit both things. Java is one such coding language that has the features of both compilation and interpretation.

- Java is Platform Independent

This is one of the top features of Java programming. The meaning of platform-independent here is that you can write a Java program on one machine and execute it on other machines or platforms. It has become possible because of the BYTE code.

- Portability

The portability in Java is the result of having features like platform independence and architecture neutrality. Programmers can run Java code on a wide range of virtual machines and hardware because its bytecode can be converted accordingly.

- Robust Programming Language

The abilities of Java like garbage collection and exception handling make it a solid programming language.

- Highly Secure

Security plays a crucial part whether you are developing a basic app or a business-critical solution. On that front, Java is considered as the most secure language as it helps in writing code that is free from viruses and other security threats. That is the reason behind its application in enterprise-grade app development.

- Java Multithreading Features

The multithreading feature helps in writing code that can perform multiple tasks simultaneously. Moreover, the thread tasks consume less processing power and memory.

- Easy Interpretation

Regardless of the computer architecture, Java programs can be run and interpreted on any type of machine. You can call it architecture-neutral language.

- High Performance

Instead of being an interpreted language, Java offers faster performance because of its just-in-time compiler.

19. What is object in Java?

As you might already know that Java is all about classes and objects. But what is an object? This is a crucial topic that you should include in your list of Java interview questions and answers for freshers. Let's know its meaning below.

An object is an instance of a class in Java. In fact, it is created from a class only using the 'new' keyword. Every object has its identity, behavior, and state, the way things in the real world also have these three things.

20. What is difference between Java and JavaScript?

There are several differences between Java and JavaScript. Whether you are a fresher or an experienced professional, this is among the top Java interview questions for you. Below, we have curated a tabular comparison of Java vs JavaScript so that it becomes easier for you to understand the main differences.

Java	**JavaScript**
Object-oriented programming language	Object-based scripting language
Can be used for complicated tasks and processes	Can't be used for complicated tasks
Needs code compilation	Text-based code
Independent language	Needs to be used with HTML
Strongly typed programming language. Need to declare variables before using them in the program.	Loosely typed language. No issues whether data types are declared or not
It's statically-type	It's dynamically-typed
High memory consumption	Low memory consumption
Saved as byte code	Saved as source code
For concurrency, it uses threads	For concurrency, it uses events
.java extension used to save programs	.js extension used to save programs
Supports multithreading	Doesn't support multithreading
Objects are based on class	Objects are based on prototype
Need JDK or Java Development Kit to run the code	Need text editor to run the code
Primarily used for backend development	Can use for both front-end and back-end

21. Which Java class is considered a superclass of all other classes?

The object class is considered the superclass of all the remaining classes.

22. Is it possible for a class to extend itself?

No. It's not possible.

23. What is difference between Java and C++ programming?

Another important Java interview questions for freshers and experienced developers can be about the differences between Java and C++ programming languages. To help you understand it easily, we have created the following comparison between the both:

Java	**C++**
Platform independent	Platform dependent
Uses compiler and interpreter both	Compiler only
Garbage controller to automate memory management	Manual memory management
Support for comments	Doesn't support comments
Doesn't support goto statement	Supports goto statement
Developed by James Gosling	Developed by Bjarne Stroustrup
Used for various types of development purposes, like web apps, Windows apps, etc.	System programming is the primary use.
Supports procedural and object-oriented programming both	Supports object-oriented programming only
Limited number of libraries	Large number of libraries
Write once, run anywhere	Write once, compile anywhere

24. Explain the difference between JDK, JRE, and JRM.

Java interview questions about the difference between Java JDK, JRE, and JRM are very common. Here is the tabular comparison to help you find the right answer.

JDK	**JRE**	**JVM**
Java Development Kit	Java Runtime Environment	Java Virtual Machine
An SDK required to build Java-based apps. It comes with several tools like debugger, compiler, and more.	A software package that comes with class libraries. Used for running Java projects.	A virtual machine that makes Java a platform-independent language.
Platform-dependent.	Platform-dependent.	Platform-independent.
Mostly used in code execution during development.	Used for providing an environment where the code execution can be done.	Used to define the execution and supporting JRE.
JDK = JRE + Development	JRE = JVM + Libraries	JVM = Support JRE to load,

Tools		verify, and execute code
Comes with various tools related to debugging, monitoring, and overall development.	Comes with class libraries and supporting files.	Doesn't include any tools or a library.

25. Is it possible to assign a superclass to a subclass in Java?

Not. It's not possible.

26. How to print text in Java?

The println() and print() methods are used to print a text in Java.

- println() example

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hi There!");
System.out.println("Welcome to WsCube Tech!");
System.out.println("Let's know top Java Interview Questions and Answers!");
  }
}
```

Output:
Hi There!
Welcome to WsCube Tech!
Let's know the top Java Interview Questions and Answers!

- print() example

```
public class Main {
  public static void main(String[] args) {
    System.out.print("Hi There! ");
System.out.print("I can now learn the core interview questions on Java.");
  }
}
```

Output:
Hi There! I can now learn the core interview questions on Java.

Intermediate Java Interview Questions for Experienced (2-5 Years)

1. What is multithreading in Java?

When a Java program is divided into multiple small parts, and these parts are executed parallelly and run simultaneously, this process is called multithreading.

The role of Java multithreading is to create lightweight programs or threads so that processing power can be used in an optimum manner.

2. Which class is used for multithreading in Java?

The Java Thread class is used to execute multithreading. It enables the creation of small threads which can run in a concurrent manner.

3. What is Java applet?

An applet in Java is used for setting up dynamic content on a web page. You can call it a program that is added to the page, which then runs in a browser and shows the dynamic content to the end user.

Some of the primary benefits of a Java applet include lower response time at the front end, secure code, and the applet program working great on popular operating systems, like Windows, Linux, and macOS.

4. What is garbage collection in Java?

It is an important concept that you must know while appearing for the interview. This is one of the most asked Java garbage collection interview questions.

The meaning of garbage collection in Java programming is that it automates the memory management for programs running on the Java virtual machine.

So, when you create and run programs, heaps of memory are allocated for memory consumption. In the long run, there will be several objects of the Java program that won't be required. What the garbage collection does is remove the unused objects automatically. As a result, it optimizes memory.

5. What if you use Java keywords as a variable or identifier?

In case a program contains keywords as a variable, class, etc., then it will show a compile-time error.

6. What is inheritance in Java?

If you have been searching for Java OOPS interview questions and answers, then this is going to be a top question. It is because inheritance is a crucial concept in Java related to object-oriented programming (OOP).

In simple terms, Java inheritance means creating classes that inherit features of some other classes. So, if you want to build a class that has some relationship with any other classes, then you need to use the inheritance method.

As a result, the new class that you create will have the features of the inherited class. It is a good mechanism to use to set up a hierarchy between classes.

7. What is polymorphism in Java?

This is yet another important concept in the list of top Java interview questions and answers related to OOPS.

The general meaning of polymorphism is “the condition of occurring in several different forms.” So, polymorphism in Java can be defined as the ability of a class to take different forms or to provide different functionalities.

For example, you can use this method to show a single message in multiple forms, based on the set parameters.

8. What is encapsulation in Java?

If you have been in the field of programming for some time now, then one of the core Java interview questions for 3 years experience will be about encapsulation. Let’s understand its meaning here.

Java encapsulation is a process that allows you to integrate the data variables and code and store them as a single thing. It is like two different capsules of medicine are mixed together to create a new single capsule. This is the logic behind calling it encapsulation.

An important thing to know here is that after the encapsulation of variables of a class, It won’t be possible for other classes to access these variables.

9. What is serialization in Java?

Java interview questions related to serialization are so common to be asked when you appear for a job interview.

When there is a need for transferring an object code in Java from one JVM to another, then the serialization process is used. Here, what it does is convert the object code to a stream so that it can be transferred to another JVM over the internet or network.

Once the stream is received on another JVM, it then goes through the deserialization process so that it can be converted back to the object code and brought into use.

10. What is Java JDBC?

JDBC stands for Java Database Connectivity.

It is an API to establish and manage a connection to the database right from within the Java program. Using JDBC, the developers or programmers can connect to not only one but multiple databases.

You can say that it works like a communication channel between the program and the database. Developers can use it to connect to any database if the relevant drivers are in use.

Since it is not a basic topic, it is an important concept for people looking for core Java interview questions for experienced.

11. What is Java enum?

Enum in Java stands for enumeration. It is a data type that comes with a set of pre-defined constant values. These values are separated by a comma. The enum concept was brought to this programming as part of Java 5. For declaring the enums, the *enum* keyword is used.

12. What is constructor overloading in Java?

Another concept to keep in your list of Java interview questions and answers for experienced developers is constructor overloading.

The role of Java constructors is to define the state of an object. When there are various constructors of a single class to be defined, it is called constructor overloading. As a result, a class becomes capable of possessing multiple constructors.

13. What is copy constructor in Java?

It is a constructor that is used when you need to make a copy of an object of an existing object of the same class. Here, the new copy of the object wouldn't impact the existing object.

14. Which keyword in Java is used to inherit a class?

We should use the *extends* keyword for this purpose.

15. What are the top benefits of inheritance in Java?

Following are the main benefits of Java inheritance:

- Code reusability
- Method overriding
- Ability to achieve runtime polymorphism
- Optimize duplicate code
- Improve the redundancy of the app
- Code flexibility so that it can be changed easily

16. What are the different memory areas assigned by JVM?

There are five types of memory areas that a JVM can allocate:

- Class(Method) Area
- Heap memory
- Stack memory
- Program Counter Register
- Native Method Stack

17. What are access modifiers in Java?

As a programming enthusiast, you should know about the access modifiers while preparing for the Java interview questions and answers.

As the name suggests, the access modifiers in Java are used to manage the access level for classes, variables, methods, constructors, etc. The access can be changed or specified using these access modifiers.

Access modifiers are of four types:

- Public
- Private
- Default
- Protected

18. How many types of inheritance are there in Java?

There are five types of Java inheritance:

- Single-level inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Multiple Inheritance
- Hybrid Inheritance

19. Can you restrict an object from inheriting its subclass? If yes, how?

Yes. It is possible if we declare the member or object private. In such a case, the subclass can't access the private members directly.

20. How can we remove the duplicate elements from a list of numbers if Java 8 is being used?

First, we need to apply the stream so those duplicate elements will be found and then make a new collection by applying Collections.toSet() method.

Advanced Java Interview Questions and Answers for Experienced (5-10 Years)

1. What is the difference between heap memory and stack memory in Java?

Two types of memory are used in the Java Virtual Machine (JVM). One is heap memory, and another is stack memory.

The primary difference between the two is that heap memory's role is to store objects, whereas stack memory stores local variables and the order of method execution. The following tabular comparison shows all the key differences between the both. While preparing for Java interview questions and answers, ensure to understand this concept well.

Heap Memory	**Stack Memory**
Used to save JRE classes and objects	Used to save methods, variables, and reference variables
Memory size is larger	Memory size is small
It takes more time to access or allocate heap memory	It takes less time to access or allocate stack memory
No fixed format or order	LIFO (Last In First Out) order
Allows changes to the allocated memory	Doesn't allow changes to the allocated memory
-Xmx and -Xms are used to increase/decrease memory size	-Xss is used to increase memory size
Memory allocation or deallocation is done manually	Memory allocation or deallocation is done using compiler
Shared memory for all threads	Dedicated memory for every object
Higher cost	Lower cost

2. Which are the best Java compilers?

If you are an experienced developer, then you must know about the top Java compilers. Because this is going to be one of the top interview questions on Java for experienced professionals.

Here is the list of best compilers for Java programming:

- Eclipse
- NetBeans

- Xcode
- AndroidStudio
- Tabnine
- Codota
- Codenvy
- JDeveloper
- jGrasp
- IntelliJ IDEA
- BlueJ
- MyEclipse
- Slickedit
- JBoss Forge
- JEdit

3. What is the difference between equals() method and equality (==) operator in Java?

There are a number of key differences between the equals method and the equality operator in Java. The primary difference is that one is a method, and another is an operator.

Such tricky concepts are usually asked when you have some experience in this field. So, you need to study the core Java interview questions and answers for experienced professionals really well.

For this question, we have created a tabular comparison to help you understand the differences between the equals method and equality operator in Java.

Equals() Method	Equality Operator (==)
It is a method	It is an operator
Its role is for comparing the content of an object	Its role is for comparing the reference values and objects
It can be overridden	Can't be overridden
Can't be used with primitives	Can be used with primitives

4. Can you inherit static members to a subclass?

No. It can't be done.

5. Can you override the final method in Java?

No. It can't be overridden.

6. How to declare an infinite loop in Java?

You must be well-prepared for such Java programming interview questions. There are three ways to declare an infinite loop in Java.

1. While loop

Syntax

```
while(condition){
        //code
}
```

2. For Loop

Syntax

```
for(initialization;condition;updation){
        //code
}
```

3. Do-While Loop

Syntax

```
do{
        //code
}while(condition);
```


7. What are the roles of final, finally, and finalize keywords in Java?

There are 50+ keywords in Java, and three similar-sounding keywords from them are final, finally, and finalize. Let's understand the differences between them with the following comparison. You must know it because it can be one of those core Java interview questions for experienced.

final	**finally**	**finalize**
Its role is to execute limitations or restrictions on classes, methods, or variables	Its role is in exception handling. finally keyword runs the crucial code no matter whether the exception occurs or not.	Used for processing clean up during garbage collection.
Can be applied to classes, methods, and variables	Can be applied to exception handling cases	Can be applied to objects
After declaring the final keyword, it can't be updated.	Whether an exception occurs or it does not, the finally keyword will run the crucial code.	The cleaning of objects during garbage collection is done using the finalize keyword.

This keyword is applied only when it is called.	It gets applied once the execution of the try-catch block is done.	It applies at the time of object cleaning.

8. When to use the super keyword in Java?

The role of the super keyword in Java is to refer to the adjacent parent class object. It is generally used with variables, methods, and constructors. In addition to being a reference keyword for parent class objects, it can also be used to trigger the parent class methods and constructors.

9. What is a ClassLoader in Java?

A Java ClassLoader is used to load the classes in JRE in a dynamic manner. It is an important component in the Runtime Environment that loads the class into the memory part of the JRE.

It is because of ClassLoaders that the JRE doesn't have to have information about the files loaded to it.

10. What are the different types of ClassLoaders in Java?

There are three ClassLoader types in Java, as defined below:

1. **BootStrap ClassLoader**

Used for loading the important classes and internal classes of Java Development Kit (JDK). This ClassLoader runs only when it is called by the Java Virtual Machine (JVM).

2. **Extension ClassLoader**

Used for loading classes from the extensions directory of the JDK. It is a child of the BootStrap ClassLoader.

3. **System ClassLoader**

Also called Application ClassLoader, it is used for loading the classes from the environment variable CLASSPATH. It is a child of the Extension ClassLoader.

11. Is it possible to access the members of a subclass if you create a superclass' object?

No. It is not possible to access the subclass members. Only superclass members will be accessible.

12. How to define a functional interface in Java?

We can use the @Functionalinterface annotation in the Java 8 to define a functional interface.

13. How Lambda expressions and functional interface are interrelated?

We can call the functional interface a large platform that comes with numerous expressions. The lambda expressions are one such part of this interface. This is the interrelation between the two.

14. What methods are used in Java 8 to define a number in functional interface?

Generally, the static method and default method are used when a number is defined in a functional interface.

15. What are the things to know and guidelines related to functional interface in Java 8?

Programmers need to follow these guidelines:

- Only a single method should be used to define the interface
- You can't define multiple abstracts
- Utilize @Functionalinterface annotation in order to define a functional interface
- To define a number, you can use whatever method you want to
- In case you override the method of java.lang.object class, it won't be counted as an abstract method.

16. What are the different types of functional interfaces in Java 8?

These are the main types of functional interface:

- Consumer
- Predicate
- Supplier
- Function (UnaryOperator and BinaryOperator)

17. State the biggest difference between Map and FlatMap.

The primary difference between the two is that Map wraps the return value in the ordinal type, whereas FlatMap doesn't do it.

18. What is the primary benefit for which one should use Metaspace over PermGen?

There is one big reason to go for Metaspace instead of PermGen. This reason is that the size of PermGen is fixed. As a result, it can't increase in a dynamic manner. On the other hand, the Metaspace does not have any limitations in terms of size. Its size can increase dynamically.

19. What is the difference between composition and aggregation in Java?

Both composition and aggregation are associations in Java. The former is considered a strong association, while the latter is considered a weak association. Let's understand the differences between them with the below tabular comparison:

Aggregation	**Composition**
Weak	Strong
There is a relationship between classes	A class belongs to another class
Interrelated classes can be independent	Classes are dependent on each other.
As the classes can be independent, it is great for reusing the code	As the classes are not independent, code reusability becomes difficult

List of Java 8 Interview Questions

Here are some of the most common interview questions on Java 8:

1. What are the new features in Java SE 8?
2. What are some of the main benefits of using Java 8?
3. Define optional class.
4. What is a functional interface in Java 8?
5. Define MetaSpace.
6. What is the meaning of the String::ValueOf expression?
7. Explain the concept of streams in Java 8.
8. What is Nashorn in Java 8?
9. How is MetaSpace different from PermGen?
10. What do you mean by method reference?
11. Explain intermediate and terminal operations.
12. Which are the most used terminal operations?
13. Is it possible for a functional interface to inherit another interface?
14. What is the difference between findFirst() and findAny()?
15. Which are the key components of a Java stream?
16. Which functional interfaces come pre-defined in Java 8?
17. What does type interface mean?
18. State the syntax of a lambda expression.
19. What is the difference between collection and stream?
20. Explain the role of JJS in Java 8.

Java Interview FAQs

1. **What is Java interview questions?**

The Java interview questions mean the concepts or things that are very likely to be asked to a candidate when he/she goes for the job interview. These are appropriate for candidates applying for jobs as:

- Java Developer
- Java Programmer
- Senior Java Developer
- Java Web Developer
- Java Android Developer
- Java EE Developer
- Java Engineer
- Java Technical Lead

2. **Can you share some Java OOPS interview questions?**

Following OOPS concepts in Java interview questions can be asked to you:

- What is the meaning of OOPS in programming?
- What is the role of OOPS in Java?
- What are the primary features of OOPS?
- What are the pros and cons of OOPS?
- What is encapsulation?
- What is polymorphism?
- What is abstraction?
- Is Java a pure object-oriented programming language? If not, why?
- What is the difference between a class and an object?
- What are manipulators?
- What is the difference between constructor and method?
- What is a destructor?
- What are the different types of inheritance?
- What is the difference between OOP and procedural programming?
- Explain the difference between error and exception.
- Which are the most popular object-oriented programming languages?
- What is the difference between runtime polymorphism and compile-time polymorphism?
- Is it mandatory that objects will always be created from a class?
- What is a subclass?
- What is a superclass?
- What is static polymorphism?
- What is dynamic polymorphism?
- What is the difference between overriding and overloading?
- What is garbage collection?

3. **What are the top Java spring boot interview questions?**

Here is the list of questions related to spring boot:

- What is spring boot in Java?
- Why use spring boot? What are its benefits?
- Which are the main components of spring boot?
- What is spring intializer?
- Explain the differences between @Controller and @RestController?
- What do you understand by dependency injection?
- How to define properties in spring boot?
- What do you mean by starter dependency?
- What is the role of @SpringBootApplication?
- Why do you use @ComponentScan?
- Explain start dependencies.
- Explain the differences between GetMapping and RequestMapping?
- What do you mean by Spring Boot CLI?
- Tell me about some of the most used CLI commands?

4. What are the most asked Java microservices interview questions?

Here is the list of questions related to microservices that can be asked in your Java interview:

- What do you understand by microservices architecture?
- Which are the top microservices tools?
- Why do we use microservices?
- What is the role of reports and dashboards in microservices?
- Explain the difference between monolithic architecture and microservices?
- What is monolithic architecture?
- Tell about the key features of microservices.
- What is cohesion?
- What do you understand by coupling?
- What is the meaning of domain-driven design?
- What is bounded context?
- Explain which tests are used in microservices?
- Why is PACT used in microservices?

5. What are some Java interview questions for Selenium Tester?

As a Selenium Automation Tester, you can expect the following interview questions on Java:

- What is data hiding in Java?
- Explain the concept of encapsulation?
- What do you understand by a tightly encapsulated class?
- Tell me about the getter and setter methods in Java?
- What does the Is-A relationship mean in Java?
- Explain the limitations of Java in Selenium testing?
- What is Java method overloading?
- Tell me about the same-origin policy and the way of handling it?
- How to implement inheritance in Java?
- Does Java support multiple inheritances using class? If not, why?
- Can you make use of super() and this() in a single constructor?

6. **Which are the top Java concurrency interview questions?**

The interviewer may ask you the following concurrency questions in Java:

- What is concurrency?
- What do you mean by the executors framework?
- Explain the atomic operation?
- Explain the lifecycle of a thread?
- How to set up the environment for Java concurrency?
- What is an atomic operation in Java?
- Which atomic classes are used in the API of Java concurrency?
- What is an executor class?
- Explain the concept of lock interface in concurrency API?
- What is BlockingQueue in Java concurrency?
- What do you understand by FutureTask class?

7. **What are the frequently asked Java collections interview questions?**

Here is the list of interview questions on the Java collections concept:

- What is Java collection?
- Explain the differences between collection and array in Java?
- Explain the difference between ArrayList and LinkedList?
- What is the difference between enumeration and iterator in Java?
- Tell me about the collection framework hierarchy?
- What is a priority queue?
- Explain the differences between HashSet and TreeSet in Java?
- State the differences between HashSet and HashMap?
- Is it possible to add a null element to HashSet?
- What are fail-fast and fail-safe iterators?
- Explain the differences between ListIterator and Iterator?

8. **Which are the primary Java thread interview questions?**

Below is the list of common questions related to threading and multithreading in Java programming:

- Explain the concept of multithreading in Java?
- What benefits does multithreading offer?
- What are the different states of a thread lifecycle?
- How to create a Java thread?
- How to implement a thread in Java?
- Tell about the concept of thread priority.
- How do the threads in Java interact with each other?
- Explain ThreadLocal in Java.
- What are some ways to get thread safety?
- What is the reason behind a sleep() thread being static?
- What is a daemon thread, and how to create it?

- What is deadlock?
- Explain the concept of a thread pool and how can we create a thread pool.

9. Which are the top Java string interview questions?

Following is the list of interview questions on Java string concept:

- Define the string in Java?
- What are the ways for string declaration in Java?
- What is the role of the string intern() method?
- Explain the differences between String and StringBuffer?
- When saving passwords in Java, developers choose a character array over a string. Why do they do so?
- What are the reasons behind string being immutable?
- State the differences between StringBuilder and StringBuffer in Java.
- How can you compare two strings in Java?
- Explain the role of the substring() method in Java?
- Can you check whether a string is empty or not? If yes, how?
- What are the ways for converting a string to a byte array?
- How to find the longest palindrome in a Java string?
- What is a string pool in Java?

10. What are the most asked Java inheritance interview questions?

Recruiters or interviewers often ask these questions related to inheritance in Java:

- Explain Java inheritance.
- What is the purpose of using inheritance?
- Define the Is-A relationship in Java?
- How to implement inheritance in Java?
- How to create the subclass of a class?
- What are the main benefits of inheritance?
- State the difference between inheritance and multi-level inheritance?
- Explain hybrid inheritance.
- What are the different types of inheritance?
- Does Java support multi inheritance through class? If not, why?

11. What are the top Java 8 interview questions for 10 years experience?

These tough Java interview questions can be asked to an experienced candidate:

- What is new in Java 8 compared to previous versions?
- What is a functional interface in Java 8?
- Explain the differences between a functional interface and a SAM interface.
- How to define a functional interface?
- What guidelines need to be followed for the functional interface?
- How to define a number in a functional interface?
- How are lambda expressions and functional interfaces interrelated?
- What are the main types of functional interfaces in Java 8?

- State the differences between Map and FlatMap.
- What are the benefits of using Metaspace compared to PerGen?

12. What are the Java technical lead interview questions?

In case you are applying for the role of Java Technical Lead, then expect some tricky and technical Java interview questions, as mentioned below:

- Can you debug a Java program while it's running? How?
- What is an asynchronous event?
- Which tools are best to use for testing Java code?
- Which tools are the best to probe Java memory leaks?
- Explain the decorator design pattern in Java.
- What are the key Java 8 features that can make the lives of programmers easier?
- What is the role of LDAP servers?
- Have you developed enterprise software or applications using Java? Share the name and your experience with it.
- What is an LDAP server and its uses?
- Explain the Spring MVC flow.
- What are RESTful web services in Java?
- What are some good ways to avoid a database deadlock?
- Explain the concepts of Spring security authentication and authorization.
- What is digest authentication in Spring security?
- What is SecurityContext in Spring security?
- State the use of AbstractSecurityInterceptor in spring security.
- What is a 2-way SSL, and why is it required?
- How to implement 2-way SSL using spring boot?
- Explain the design patterns in microservices architecture.

13. What are the most asked Java interview questions for 2 years experience?

For developers or programmers working for a couple of years now, below are the top Java interview questions for 2 years experience:

- Is it possible to override a static method? If yes, how?
- Which Java class is considered the base class?
- State the difference between HashMap and HashSet.
- Why Java strings are immutable?
- What is a ClassPath in Java?
- Explain the differences between StringBuffer and StringBuilder.
- What is Java multithreading?
- What is an applet in Java?
- Explain garbage collection in Java.
- What is inheritance in Java?
- What is polymorphism in Java?
- Explain Java serialization.
- When should we use the transient variable?
- Is it possible to call the start method twice? If yes, how?
- How to make a class immutable in Java?

- What is a Java copy constructor?
- How the sorting of custom objects is done in Java?
- Explain the marker interface in Java.
- Explain the differences between LinkedList and ArrayList.

Master Java Programming

With WsCube Tech's Online Course

This comprehensive course Includes:

- 25 Hours of Video Sessions
- Professional Certification
- Training by Expert Python Programmer
- 200+ Top-rated MCQs and Quizzes
- Interview Preparation
- Accessible on mobile and desktop both

~~*₹4999*~~ *80% OFF*

₹999

View Details Full Curriculum

A Brief About WsCube Tech

(India's Most Trusted IT Training Institute)

WsCube Tech is a leading IT training institute and software development company based in Jodhpur. Since 2011, WsCube Tech has trained 1,50,000+ students and offered internship opportunities to 3,500+ candidates.

In addition, WsCube Tech is one of the fastest-growing tech-based YouTube channels in India, with a community of 1.8+ Million subscribers and 20+ Million monthly views.

With an expert team of trainers in various technological fields, the institute has helped thousands of students across India and other Asian countries to acquire new skills and explore high-paying career opportunities.

Our learners are working at top brands, enterprises, and unicorns across India and globally.

Live Online Certification Courses Offered by WsCube Tech

Rigorous regular training with interactive live classes

- ☐ Ethical Hacking Certification
- ☐ Penetration Testing Certification
- ☐ Web Development Certification
- ☐ Content Writing Certification
- ☐ SEO Certification
- ☐ Digital Marketing Certification
- ☐ Data Science Certification
- ☐ Python Certification
- ☐ WordPress Certification
- ☐ Flutter Certification
- ☐ Android App Development Certification

Students We've Trained Work at Renowned Companies, Startups, and Unicorns

Let's connect on social media and around the web:

- Website: https://www.wscubetech.com/
- YouTube: https://www.youtube.com/c/wscubetechjodhpur
- Facebook: https://www.facebook.com/wscubetech.india
- LinkedIn: https://www.linkedin.com/company/wscubetech
- Instagram: https://www.instagram.com/wscubetechindia/
- Twitter: https://twitter.com/wscubetechindia

100 Most Asked Java Interview QnA

Made By:

Yadneyesh (Curious Coder)
CodWithCurious.com

1. What is Java?

Java is a high-level, object-oriented programming language that is widely used for developing a variety of applications, including web, desktop, and mobile applications.

2. What is the difference between Java and JavaScript?

Java and JavaScript are two different programming languages with different purposes. Java is used for building applications, while JavaScript is primarily used for adding interactivity to web pages.

3. What is the main principle of Java programming?

Java follows the principle of "write once, run anywhere" (WORA), which means that Java code can be compiled into bytecode and executed on any platform that has a Java Virtual Machine (JVM).

4. What are the main features of Java?

Some of the main features of Java include platform independence, object-oriented programming, automatic memory management (garbage collection), and strong type checking.

5. What is a class in Java?

In Java, a class is a blueprint or template for creating objects. It defines the properties (attributes) and behaviors (methods) that objects of that class can have.

6. What is an object in Java?

An object in Java is an instance of a class. It represents a specific entity or item that can have its own set of attributes and behaviors defined by its class.

7. What is a method in Java?

A method in Java is a block of code that performs a specific task. It can be called or invoked to execute its defined functionality.

8. What is the difference between a class and an object?

A class is a blueprint or template, while an object is an instance of that class. A class defines the structure and behavior of objects, while objects represent specific instances of the class.

9. What is inheritance in Java?

Inheritance is a mechanism in Java where a class can inherit properties and behaviors from another class. It allows for code reuse and creating a hierarchical relationship between classes.

10. What are the types of inheritance in Java?

Java supports single inheritance, where a class can inherit from only one superclass, and multiple inheritance through interfaces, where a class can implement multiple interfaces.

11. What is polymorphism in Java?

Polymorphism is the ability of an object to take on many forms. In Java, it allows objects of different classes to be treated as objects of a common superclass, enabling code flexibility and reusability.

12. What are the access modifiers in Java?

Java provides four access modifiers: public, private, protected, and default (no modifier). They control the visibility and accessibility of classes, methods, and variables.

13. What is encapsulation in Java?

Encapsulation is the process of hiding internal details and providing a public interface to interact with an object. It helps in achieving data abstraction and protects data from

unauthorized access.

14. What is a constructor in Java?

A constructor in Java is a special method that is used to initialize objects of a class. It is called automatically when an object is created and has the same name as the class.

15. What is the difference between a constructor and a method?

A constructor is a special method used for object initialization and is called automatically when an object is created. A method, on the other hand, is a block of code that performs a specific task and needs to be called explicitly.

16. What is the Java Virtual Machine (JVM)?

The JVM is a crucial part of the Java platform. It is responsible for executing Java bytecode and provides a runtime environment in which Java programs can run on any hardware or operating system.

17. What is the Java Development Kit (JDK)?

The JDK is a software development kit provided by Oracle, which includes the necessary tools and libraries to develop, compile, and run Java programs. It consists of the JVM, compiler, and other utilities.

18. What is the difference between the JDK and the JRE?

The JDK (Java Development Kit) is a software development kit that includes the tools needed to develop Java applications, while the JRE (Java Runtime Environment) is a runtime environment required to run Java applications.

19. What is a package in Java?

A package in Java is a way of organizing related classes and interfaces. It provides a namespace and helps in avoiding naming conflicts.

20. What is the difference between an abstract class and an interface?

An abstract class can have both abstract and non-abstract methods and can be extended by other classes, while an interface only contains abstract method declarations and can be implemented by classes.

21. What is a static method in Java?

A static method in Java is a method that belongs to the class rather than an instance of the class. It can be called without creating an object of the class.

22. What is the keyword "final" used for in Java?

The "final" keyword in Java can be used to declare a variable, a method, or a class. A final variable cannot be changed, a final method cannot be overridden, and a final class cannot be inherited.

23. What is method overloading in Java?

Method overloading is the ability to define multiple methods with the same name but different parameters in the same class. The appropriate method is called based on the arguments passed.

24. What is method overriding in Java?

Method overriding is the ability to provide a different implementation of a method in a subclass that is already defined in its superclass. It allows for the execution of the overridden method instead of the superclass method.

25. What is the difference between method overloading and method overriding?

Method overloading involves defining multiple methods with the same name but different parameters in the same class, while method overriding involves providing a different implementation of a method in a subclass that is already defined in its superclass.

26. What is the "this" keyword in Java?

The "this" keyword in Java refers to the current instance of a class. It can be used to access instance variables, call instance methods, or invoke constructors.

27. What is a static variable in Java?

A static variable in Java is a variable that belongs to the class rather than an instance of the class. It is shared among all instances of the class.

28. What is the purpose of the "final" keyword in method parameters?

The "final" keyword in method parameters is used to make the parameter value unchangeable within the method. It ensures that the parameter cannot be reassigned or modified.

29. What is the purpose of the "static" keyword in Java?

The "static" keyword in Java is used to declare variables, methods, and nested classes that belong to the class itself, rather than instances of the class. It allows accessing them without creating an object of the class.

30. What is the difference between "==" and ".equals()" in Java?

The "==" operator in Java is used to compare the equality of object references, while the ".equals()" method is used to compare the equality of object values. The ".equals()" method can be overridden to provide custom equality comparison.

31. What is the purpose of the "super" keyword in Java?

The "super" keyword in Java is used to refer to the superclass of a class. It can be used to access superclass members, invoke superclass constructors, or differentiate between superclass and subclass members with the same name.

32. What is a thread in Java?

A thread in Java is a lightweight unit of execution within a program. It allows concurrent execution of multiple tasks or activities, enabling better utilization of system resources.

33. How do you create and start a thread in Java?

To create and start a thread in Java, you can either extend the "Thread" class and override the "run()" method, or implement the "Runnable" interface and pass it to a new "Thread" object. Then call the "start()" method on the thread object to begin execution.

34. What is synchronization in Java?

Synchronization in Java is a technique used to control the access and execution of multiple threads to ensure that only one thread can access a shared resource or code block at a time.

35. What is the difference between the "synchronized" block and the "synchronized" method?

A "synchronized" block in Java allows a specific block of code to be synchronized, ensuring that only one thread can execute it at a time. A "synchronized" method applies synchronization to the entire method, making it mutually exclusive for all threads.

36. What is the purpose of the "volatile" keyword in Java?

The "volatile" keyword in Java is used to indicate that a variable's value may be modified by multiple threads. It ensures that any read or write operation on the variable is directly performed on the main memory, rather than relying on CPU caches.

37. What is an exception in Java?

An exception in Java is an event that occurs during the execution of a program, which disrupts the normal flow of instructions. It represents an error condition or an exceptional circumstance.

38. What is the difference between checked and unchecked exceptions?

Checked exceptions are checked at compile-time, and the programmer is required to handle or declare them using the "throws" keyword. Unchecked exceptions, on the other hand, are not checked at compile-time, and the programmer is not obligated to handle or declare them.

39. How do you handle exceptions in Java?

Exceptions in Java can be handled using try-catch blocks. The code that may throw an exception is placed inside the try block, and if an exception occurs, it is caught and handled in the catch block.

40. What is the purpose of the "finally" block in exception handling?

The "finally" block in Java is used to define a block of code that will be executed regardless of whether an exception occurs or not. It is often used to release resources or perform cleanup operations.

41. What is the difference between the "throw" and "throws" keywords in Java?

The "throw" keyword in Java is used to manually throw an exception, while the "throws" keyword is used in method declarations to specify that the method may throw certain types of exceptions.

42. What is the difference between checked exceptions and runtime exceptions?

Checked exceptions are checked at compile-time and must be handled or declared, while runtime exceptions (unchecked exceptions) are not required to be handled or declared.

43. What is the Java API?

The Java API (Application Programming Interface) is a collection of classes, interfaces, and other resources provided by the Java Development Kit (JDK). It provides a set of predefined classes and methods for building Java applications.

44. What is the difference between an ArrayList and a LinkedList?

An ArrayList is implemented as a resizable array, allowing fast random access but slower insertion and removal of elements. A LinkedList is implemented as a doubly-linked list, allowing fast insertion and removal but slower random access.

45. What is the difference between a HashSet and a TreeSet?

A HashSet in Java stores elements in no particular order, using a hash table for fast access but does not maintain any specific order. A TreeSet stores elements in sorted order and allows for efficient retrieval of elements based on their natural ordering or a custom comparator.

46. What is the difference between the "equals()" method and the "hashCode()" method?

The "equals()" method is used to compare the equality of objects based on their values, while the "hashCode()" method is used to calculate a unique hash code value for an object, typically used for efficient retrieval in hash-based data structures like HashMaps.

47. What is the difference between a shallow copy and a deep copy?

A shallow copy creates a new object that shares the same references as the original object, while a deep copy creates a new object and recursively copies all the referenced objects as well, resulting in separate copies.

48. What is a lambda expression in Java?

A lambda expression in Java is an anonymous function that can be used to simplify the syntax of functional interfaces. It allows for more concise and readable code, especially when working with functional programming constructs.

49. What is functional programming in Java?

Functional programming in Java is a programming paradigm that emphasizes writing programs using pure functions and immutable data. It involves treating functions as first-class citizens and utilizing higher-order functions and lambda expressions.

50. What are the Java 8 features for functional programming?

Java 8 introduced several features to support functional programming, including lambda expressions, functional interfaces, the Stream API for working with collections, and default methods in interfaces.

51. What is the difference between an interface and an abstract class?

An interface in Java can only declare method signatures and constants but cannot provide implementations, while an abstract class can have both method declarations and concrete implementations. A class can implement multiple interfaces but can inherit from only one abstract class.

52. What is the purpose of the "default" keyword in interface methods?

The "default" keyword in Java interfaces is used to define a default implementation for a method. It allows adding new methods to existing interfaces without breaking the implementations of classes that implement those interfaces.

53. What is the difference between a BufferedReader and a Scanner?

A BufferedReader in Java reads text from a character stream with efficient buffering, while a Scanner can parse different types of data from various sources such as files, strings, or standard input.

54. What is the purpose of the "StringBuilder" class in Java?

The "StringBuilder" class in Java is used to create and manipulate mutable sequences of characters. It is more efficient than concatenating strings using the "+" operator, as it avoids unnecessary object creations.

55. What is the difference between the "Comparable" and "Comparator" interfaces?

The "Comparable" interface is used to define a natural ordering for a class by implementing the "compareTo()" method. The "Comparator" interface, on the other hand, provides a way to define custom ordering by implementing the "compare()" method and is independent of the class being compared.

56. What is the purpose of the "assert" keyword in Java?

The "assert" keyword in Java is used to perform assertions, which are checks placed in the code to verify specific conditions. It is primarily used during development and testing to catch potential bugs or invalid assumptions.

57. What is the difference between a local variable and an instance variable?

A local variable in Java is declared inside a method or a block and has a limited scope within that method or block. An instance variable, also known as a member variable, is declared within a class but outside any method and is accessible to all methods of the class.

58. What is the purpose of the "transient" keyword in Java?

The "transient" keyword in Java is used to indicate that a variable should not be serialized during object serialization. When an object is deserialized, transient variables are set to their default values.

59. What is the purpose of the "static" block in Java?

The "static" block in Java is used to initialize static variables or perform one-time initialization tasks for a class. It is executed when the class is loaded into memory, before any objects of that class are created.

60. What is the purpose of the "strictfp" keyword in Java?

The "strictfp" keyword in Java is used to ensure strict adherence to the IEEE 754 standard for floating-point calculations. It ensures consistent results across different platforms by disabling some optimizations that can affect precision.

61. What is the difference between a public class and a default (package-private) class?

A public class in Java can be accessed from any other class, regardless of the package they belong to. A default class, also known as a package-private class, is only accessible within the same package and cannot be accessed from outside the package.

62. What is the purpose of the "enum" keyword in Java?

The "enum" keyword in Java is used to define an enumeration, which is a special type that represents a fixed set of constants. It allows for more structured and type-safe representation of predefined values.

63. What is the purpose of the "break" and "continue" statements in Java?

The "break" statement in Java is used to terminate the execution of a loop or switch statement and resume execution after the loop or switch block. The "continue" statement is used to skip the current iteration of a loop and move to the next iteration.

64. What is the purpose of the "try-with-resources" statement in Java?

The "try-with-resources" statement in Java is used to automatically close resources that implement the "AutoCloseable" interface. It ensures that resources, such as file streams or database connections, are properly closed, even if an exception occurs.

65. What is the purpose of the "instanceof" operator in Java?

The "instanceof" operator in Java is used to check whether an object is an instance of a specific class or implements a specific interface. It returns a boolean value indicating the result of the check.

66. What is the difference between the pre-increment and post-increment operators?

The pre-increment operator (++i) in Java increments the value of a variable and returns the incremented value, while the post-increment operator (i++) increments the value of a variable but returns the original value before the increment.

67. What is the difference between the pre-decrement and post-decrement operators?

The pre-decrement operator (--i) in Java decrements the value of a variable and returns the decremented value, while the post-decrement operator (i--) decrements the value of a variable but returns the original value before the decrement.

68. What is the purpose of the "Math" class in Java?

The "Math" class in Java provides various methods for performing common mathematical operations, such as square roots, trigonometric functions, exponential calculations, rounding, and more.

69. What is the purpose of the "StringBuffer" class in Java?

The "StringBuffer" class in Java is used to create and manipulate mutable sequences of characters, similar to the "StringBuilder" class. However, "StringBuffer" is synchronized and thread-safe, making it suitable for multi-threaded environments.

70. What is the purpose of the "Math.random()" method in Java?

The "Math.random()" method in Java returns a random double value between 0.0 (inclusive) and 1.0 (exclusive). It is often used to generate random numbers or simulate random behavior.

71. What is the purpose of the "Character" class in Java?

The "Character" class in Java provides methods for working with individual characters, such as checking for character types (letters, digits, whitespace), converting case, and performing character-based operations.

72. What is the purpose of the "Integer" class in Java?

The "Integer" class in Java is a wrapper class that provides methods for working with integer values, such as converting strings to integers, performing arithmetic operations, and converting integers to different representations (binary, hexadecimal).

73. What is the purpose of the "Double" class in Java?

The "Double" class in Java is a wrapper class that provides methods for working with double-precision floating-point values. It offers functionality for parsing strings, performing arithmetic operations, and converting doubles to different representations (binary, hexadecimal).

74. What is the purpose of the "System" class in Java?

The "System" class in Java provides access to system resources and allows interaction with the system environment. It contains methods for standard input/output, error output, current time, copying arrays, and more.

75. What is the purpose of the "File" class in Java?

The "File" class in Java is used to represent and manipulate file and directory paths. It provides methods for creating, deleting, renaming, and querying file properties such as size, last modified date, and permissions.

76. What is the purpose of the "FileNotFoundException" in Java?

The "FileNotFoundException" in Java is an exception that is thrown when an attempt to access a file that does not exist or cannot be found is made. It is typically caught and handled to handle file-related errors.

77. What is the purpose of the "NullPointerException" in Java?

The "NullPointerException" in Java is an exception that is thrown when a null reference is accessed and used where an object reference is expected. It indicates a programming error and should be handled or prevented to avoid unexpected crashes.

78. What is the purpose of the "ArrayIndexOutOfBoundsException" in Java?

The "ArrayIndexOutOfBoundsException" in Java is an exception that is thrown when an invalid index is used to access an array. It indicates that the index is either negative or exceeds the array's bounds.

79. What is the purpose of the "ArithmeticException" in Java?

The "ArithmeticException" in Java is an exception that is thrown when an arithmetic operation produces an illegal or undefined result. It typically occurs when dividing by zero or performing unsupported mathematical operations.

80. What is the purpose of the "NumberFormatException" in Java?

The "NumberFormatException" in Java is an exception that is thrown when a string cannot be parsed into a numeric value of the expected format. It occurs when attempting to convert a string to an integer, float, or double, but the string does not represent a valid number.

81. What is the purpose of the "StringBuilder" class in Java?

The "StringBuilder" class in Java is used to create and manipulate mutable sequences of characters. It provides methods for appending, inserting, deleting, and modifying

character sequences efficiently.

82. What is the purpose of the "HashSet" class in Java?

The "HashSet" class in Java is an implementation of the Set interface that stores unique elements in no particular order. It provides constant-time performance for basic operations like adding, removing, and checking for the presence of elements.

83. What is the purpose of the "HashMap" class in Java?

The "HashMap" class in Java is an implementation of the Map interface that stores key-value pairs. It provides fast retrieval and insertion of elements based on their keys and allows for efficient mapping and lookup operations.

84. What is the purpose of the "LinkedList" class in Java?

The "LinkedList" class in Java is an implementation of the List interface that uses a doubly-linked list to store elements. It provides efficient insertion and removal of elements at both ends of the list but slower random access.

85. What is the purpose of the "Comparator" interface in Java?

The "Comparator" interface in Java is used to define custom ordering of objects. It provides a way to compare objects based on specific criteria other than their natural ordering defined by the "Comparable" interface.

86. What is the purpose of the "Comparable" interface in Java?

The "Comparable" interface in Java is used to define the natural ordering of objects of a class. It provides a method, "compareTo()", that allows objects to be compared and sorted based on their natural order.

87. What is the purpose of the "super" keyword in Java?

The "super" keyword in Java is used to refer to the superclass of a class or to call the superclass's constructor, methods, or variables. It is primarily used to differentiate between superclass and subclass members with the same name.

88. What is the purpose of the "this" keyword in Java?

The "this" keyword in Java is used to refer to the current instance of a class. It is primarily used to differentiate between instance variables and parameters or to invoke other constructors within a class.

89. What is the purpose of the "final" keyword in Java?

The "final" keyword in Java is used to define constants, make variables unchangeable, or prevent method overriding or class inheritance. It ensures that the value of a variable or the implementation of a method or class cannot be modified.

90. What is the purpose of the "static" keyword in Java?

The "static" keyword in Java is used to define class-level variables and methods that are shared among all instances of a class. It allows accessing variables or methods without creating an instance of the class.

91. What is the purpose of the "abstract" keyword in Java?

The "abstract" keyword in Java is used to define abstract classes or methods. An abstract class cannot be instantiated and serves as a base class for subclasses. An abstract method does not have an implementation and must be overridden in a subclass.

92. What is the purpose of the "interface" keyword in Java?

The "interface" keyword in Java is used to define interfaces, which declare methods that implementing classes must provide. It allows for multiple inheritance by implementing multiple interfaces and enables the concept of polymorphism.

93. What is the purpose of the "package" keyword in Java?

The "package" keyword in Java is used to define a package, which is a way to organize related classes and interfaces. It provides a hierarchical structure and helps prevent naming conflicts between classes.

94. What is the purpose of the "import" keyword in Java?

The "import" keyword in Java is used to import classes, interfaces, or packages into a source file. It allows using classes from other packages without specifying their fully qualified names.

95. What is the purpose of the "throw" keyword in Java?

The "throw" keyword in Java is used to manually throw an exception. It is typically used when a program encounters an error or exceptional situation that cannot be handled, and the control should be transferred to an exception handler.

96. What is the purpose of the "throws" keyword in Java?

The "throws" keyword in Java is used in method declarations to specify that a method may throw certain types of exceptions. It allows the caller of the method to handle the exception or propagate it further.

97. What is the purpose of the "try-catch-finally" block in Java?

The "try-catch-finally" block in Java is used to handle exceptions. The "try" block contains the code that may throw an exception, the "catch" block catches and handles the exception, and the "finally" block contains cleanup code that is executed regardless of whether an exception occurs or not.

98. What is the purpose of the "instanceof" operator in Java?

The "instanceof" operator in Java is used to check the type of an object at runtime. It returns a boolean value indicating whether an object is an instance of a particular class or implements a specific interface.

99. What is the purpose of the "break" statement in Java?

The "break" statement in Java is used to terminate the execution of a loop or switch statement. It allows exiting a loop prematurely or skipping the remaining cases in a switch statement.

100. What is the purpose of the "continue" statement in Java?

The "continue" statement in Java is used to skip the current iteration of a loop and continue with the next iteration. It allows skipping certain iterations based on specific conditions without exiting the loop entirely.