Top 1000

Java

Interview Question & Answers

Knowledge Powerhouse

Copyright © 2017 Knowledge Powerhouse

All rights reserved.

No part of this book can be copied in any form. The publisher and the author have used good

faith efforts to ensure that the information in this book is correct and accurate. The publisher and the author disclaim all responsibility for errors or omissions. Use of the information in this book is at your own risk.

www.KnowledgePowerhouse.com

DEDICATION

To our readers!

CONTENTS

**Java Basics**

1. **What is the difference between JDK and JRE?**
2. **What is Java Virtual Machine (JVM)?**
3. **What are the different types of memory areas allocated by JVM?**
4. **What is JIT compiler?**
5. **How Java platform is different from other platforms?**
6. **Why people say that Java is 'write once and run anywhere' language?**
7. **How does ClassLoader work in Java?**
8. **Do you think ‘main’ used for main method is a keyword in Java?**
9. **Can we write main method as public void static instead of public static void?**
10. **In Java, if we do not specify any value for local variables, then what will be the default value of the local variables?**
11. **Let say, we run a java class without passing any arguments. What will be the value of String array of arguments in Main method?**
12. **What is the difference between byte and char data types in Java?**

**OOPS**

1. **What are the main principles of Object Oriented Programming?**
2. **What is the difference between Object Oriented Programming language and Object Based Programming language?**
3. **In Java what is the default value of an object reference defined as an**

**instance variable in an Object?**

1. **Why do we need constructor in Java?**
2. **Why do we need default constructor in Java classes?**
3. **What is the value returned by Constructor in Java?**
4. **Can we inherit a Constructor?**
5. **Why constructors cannot be final, static, or abstract in Java? Inheritance**
6. **What is the purpose of ‘this’ keyword in java?**
7. **Explain the concept of Inheritance?**
8. **Which class in Java is superclass of every other class?**
9. **Why Java does not support multiple inheritance?**
10. **In OOPS, what is meant by composition?**
11. **How aggregation and composition are different concepts?**
12. **Why there are no pointers in Java?**
13. **If there are no pointers in Java, then why do we get NullPointerException?**
14. **What is the purpose of ‘super’ keyword in java?**
15. **Is it possible to use this() and super() both in same constructor?**
16. **What is the meaning of object cloning in Java?**
17. **In Java, why do we use static variable?**
18. **Why it is not a good practice to create static variables in Java?**
19. **What is the purpose of static method in Java?**

1. **Why do we mark main method as static in Java?**
2. **In what scenario do we use a static block?**
3. **Is it possible to execute a program without defining a main() method?**
4. **What happens when static modifier is not mentioned in the signature of main method?**
5. **What is the difference between static method and instance method in Java?**
6. **What is the other name of Method Overloading?**
7. **How will you implement method overloading in Java?**
8. **What kinds of argument variations are allowed in Method Overloading?**
9. **Why it is not possible to do method overloading by changing return type of method in java?**
10. **Is it allowed to overload main() method in Java?**
11. **How do we implement method overriding in Java?**
12. **Are we allowed to override a static method in Java?**
13. **Why Java does not allow overriding a static method?**
14. **Is it allowed to override an overloaded method?**
15. **What is the difference between method overloading and method overriding in Java?**
16. **Does Java allow virtual functions?**
17. **What is meant by covariant return type in Java?**

**Polymorphism**

1. **What is Runtime Polymorphism?**
2. **Is it possible to achieve Runtime Polymorphism by data members in Java?**
3. **Explain the difference between static and dynamic binding? Abstraction**
4. **What is Abstraction in Object Oriented programming?**
5. **How is Abstraction different from Encapsulation?**
6. **What is an abstract class in Java?**
7. **Is it allowed to mark a method abstract method without marking the class abstract?**
8. **Is it allowed to mark a method abstract as well as final?**
9. **Can we instantiate an abstract class in Java?**
10. **What is an interface in Java?**
11. **Is it allowed to mark an interface method as static?**
12. **Why an Interface cannot be marked as final in Java?**
13. **What is a marker interface?**
14. **What can we use instead of Marker interface?**
15. **How Annotations are better than Marker Interfaces?**
16. **What is the difference between abstract class and interface in Java?**
17. **Does Java allow us to use private and protected modifiers for variables in interfaces?**
18. **How can we cast to an object reference to an interface reference? Final**

1. **How can you change the value of a final variable in Java?**
2. **Can a class be marked final in Java?**
3. **How can we create a final method in Java?**
4. **How can we prohibit inheritance in Java?**
5. **Why Integer class in final in Java?**
6. **What is a blank final variable in Java?**
7. **How can we initialize a blank final variable?**
8. **Is it allowed to declare main method as final?**
9. **What is the purpose of package in Java?**
10. **What is java.lang package?**
11. **Which is the most important class in Java?**
12. **Is it mandatory to import java.lang package every time?**
13. **Can you import same package or class twice in your class?**
14. **What is a static import in Java?**
15. **What is the difference between import static com.test.Fooclass and import com.test.Fooclass?**
16. **What is Locale in Java?**
17. **How will you use a specific Locale in Java? Serialization**
18. **What is the serialization?**
19. **What is the purpose of serialization?**

1. **What is Deserialization?**
2. **What is Serialization and Deserialization conceptually?**
3. **Why do we mark a data member transient?**
4. **Is it allowed to mark a method as transient?**
5. **How does marking a field as transient makes it possible to serialize an object?**
6. **What is Externalizable interface in Java?**
7. **What is the difference between Serializable and Externalizable interface?**
8. **What is Reflection in Java?**
9. **What are the uses of Reflection in Java?**
10. **How can we access private method of a class from outside the class?**
11. **How can we create an Object dynamically at Runtime in Java? Garbage Collection**
12. **What is Garbage Collection in Java?**
13. **Why Java provides Garbage Collector?**
14. **What is the purpose of gc() in Java?**
15. **How does Garbage Collection work in Java?**
16. **When does an object become eligible for Garbage Collection in Java?**
17. **Why do we use finalize() method in Java?**
18. **What are the different types of References in Java?**
19. **How can we reference an unreferenced object again?**

1. **What kind of process is the Garbage collector thread?**
2. **What is the purpose of the Runtime class?**
3. **How can we invoke an external process in Java?**
4. **What are the uses of Runtime class?**
5. **What is a Nested class?**
6. **How many types of Nested classes are in Java?**
7. **Why do we use Nested Classes?**
8. **What is the difference between a Nested class and an Inner class in Java?**
9. **What is a Nested interface?**
10. **How can we access the non-final local variable, inside a Local Inner class?**
11. **Can an Interface be defined in a Class?**
12. **Do we have to explicitly mark a Nested Interface public static?**
13. **Why do we use Static Nested interface in Java?**
14. **What is the meaning of Immutable in the context of String class in Java?**
15. **Why a String object is considered immutable in java?**
16. **How many objects does following code create?**
17. **How many ways are there in Java to create a String object?**
18. **How many objects does following code create?**

1. **What is String interning?**
2. **Why Java uses String literal concept?**
3. **What is the basic difference between a String and StringBuffer object?**
4. **How will you create an immutable class in Java?**
5. **What is the use of toString() method in java ?**
6. **Arrange the three classes String, StringBuffer and StringBuilder in the order of efficiency for String processing operations?**
7. **What is Exception Handling in Java?**
8. **In Java, what are the differences between a Checked and Unchecked?**
9. **What is the base class for Error and Exception classes in Java?**
10. **What is a finally block in Java?**
11. **What is the use of finally block in Java?**
12. **Can we create a finally block without creating a catch block?**
13. **Do we have to always put a catch block after a try block?**
14. **In what scenarios, a finally block will not be executed?**
15. **Can we re-throw an Exception in Java?**
16. **What is the difference between throw and throws in Java?**
17. **What is the concept of Exception Propagation?**
18. **When we override a method in a Child class, can we throw an additional Exception that is not thrown by the Parent class method?**

**Multi-threading**

**144.How Multi-threading works in Java?**

1. **What are the advantages of Multithreading?**
2. **What are the disadvantages of Multithreading?**
3. **What is a Thread in Java?**
4. **What is a Thread’s priority and how it is used in scheduling?**
5. **What are the differences between Pre-emptive Scheduling Scheduler and Time Slicing Scheduler?**
6. **Is it possible to call run() method instead of start() on a thread in Java?**
7. **How will you make a user thread into daemon thread if it has already started?**
8. **Can we start a thread two times in Java?**
9. **In what scenarios can we interrupt a thread?**
10. **In Java, is it possible to lock an object for exclusive use by a thread?**
11. **How notify() method is different from notifyAll() method? Collections**
12. **What are the differences between the two data structures: a Vector and an ArrayList?**
13. **What are the differences between Collection and Collections in Java?**
14. **In which scenario, LinkedList is better than ArrayList in Java?**
15. **What are the differences between a List and Set collection in Java?**
16. **What are the differences between a HashSet and TreeSet collection in Java?**
17. **In Java, how will you decide when to use a List, Set or a Map**

**collection?**

1. **What are the differences between a HashMap and a Hashtable in Java?**
2. **What are the differences between a HashMap and a TreeMap?**
3. **What are the differences between Comparable and Comparator?**
4. **In Java, what is the purpose of Properties file?**
5. **What is the reason for overriding equals() method?**
6. **How does hashCode() method work in Java?**
7. **Is it a good idea to use Generics in collections?**
8. **What are Wrapper classes in Java?**
9. **What is the purpose of native method in Java?**
10. **What is System class?**
11. **What is System, out and println in System.out.println method call?**
12. **What is the other name of Shallow Copy in Java?**
13. **What is the difference between Shallow Copy and Deep Copy in Java?**
14. **What is a Singleton class?**
15. **What is the difference between Singleton class and Static class? Java Collection**
16. **What is the difference between Collection and Collections Framework in Java?**
17. **What are the main benefits of Collections Framework in Java?**
18. **What is the root interface of Collection hierarchy in Java?**

1. **What are the main differences between Collection and Collections?**
2. **What are the Thread-safe classes in Java Collections framework?**
3. **How will you efficiently remove elements while iterating a Collection?**
4. **How will you convert a List into an array of integers like- int[]?**
5. **How will you convert an array of primitive integers int[] to a List collection?**
6. **How will you run a filter on a Collection?**
7. **How will you convert a List to a Set?**
8. **How will you remove duplicate elements from an ArrayList?**
9. **How can you maintain a Collection with elements in Sorted order?**
10. **What is the difference between Collections.emptyList() and creating new instance of Collection?**
11. **How will you copy elements from a Source List to another list?**
12. **What are the Java Collection classes that implement List interface?**
13. **What are the Java Collection classes that implement Set interface?**
14. **What is the difference between an Iterator and ListIterator in Java?**
15. **What is the difference between Iterator and Enumeration?**
16. **What is the difference between an ArrayList and a LinkedList data structure?**
17. **What is the difference between a Set and a Map in Java?**
18. **What is the use of a Dictionary class?**
19. **What is the default size of load factor in a HashMap collection in Java?**

**199. What is the significance of load factor in a HashMap in Java?**

**200.What are the major differences between a HashSet and a HashMap?**

1. **What are the similarities between a HashSet and a HashMap in Java?**

**202.What is the reason for overriding equals() method?**

1. **How can we synchronize the elements of a List, a Set or a Map?**

**204.What is Hash Collision? How Java handles hash-collision in HashMap?**

**205.What are the Hash Collision resolution techniques?**

**206.What is the difference between Queue and Stack data structures?**

**207.What is an Iterator in Java?**

**208.What is the difference between Iterator and Enumeration in Java?**

**209.What is the design pattern used in the implementation of Enumeration in Java?**

1. **Which methods do we need to override to use an object as key in a HashMap?**
2. **How will you reverse a List in Java?**
3. **How will you convert an array of String objects into a List?**
4. **What is the difference between peek(), poll() and remove() methods of Queue interface in java?**
5. **What is the difference between Array and ArrayList in Java?**
6. **How will you insert, delete and retrieve elements from a HashMap collection in Java?**
7. **What are the main differences between HashMap and ConcurrentHashMap in Java?**

1. **What is the increasing order of performance for following collection classes in Java?**
2. **Why does Map interface not extend Collection interface in Java?**
3. **What are the different ways to iterate elements of a list in Java?**

**220.What is CopyOnWriteArrayList? How it is different from ArrayList in Java?**

1. **How remove() method is implemented in a HashMap?**

**222.What is BlockingQueue in Java Collections?**

1. **How is TreeMap class implemented in Java?**

**224.What is the difference between Fail-fast and Fail-safe iterator in Java?**

**225.How does ConcurrentHashMap work in Java?**

**226.What is the importance of hashCode() and equals() methods?**

**227.What is the contract of hashCode() and equals() methods in Java?**

**228.What is an EnumSet in Java?**

**229.What are the main Concurrent Collection classes in Java?**

**230.How will you convert a Collection to SynchronizedCollection in Java?**

1. **How IdentityHashMap is different from a regular Map in Java?**
2. **What is the main use of IdentityHashMap?**
3. **How can we improve the performance of IdentityHashMap?**

**234.Is IdentityHashMap thread-safe?**

1. **What is a WeakHashMap in Java?**
2. **How can you make a Collection class read Only in Java?**

**237.When is UnsupportedOperationException thrown in Java?**

**238.Let say there is a Customer class. We add objects of Customer class to an ArrayList. How can we sort the Customer objects in ArrayList by using customer firstName attribute of Customer class?**

**239.What is the difference between Synchronized Collection and Concurrent Collection?**

**240.What is the scenario to use ConcurrentHashMap in Java?**

**241. How will you create an empty Map in Java?**

**242.What is the difference between remove() method of Collection and remove() method of Iterator?**

**243.Between an Array and ArrayList, which one is the preferred collection for storing objects?**

**244.Is it possible to replace Hashtable with ConcurrentHashMap in Java?**

**245.How CopyOnWriteArrayList class is different from ArrayList and Vector classes?**

**246.Why ListIterator has add() method but Iterator does not have?**

**247.Why do we sometime get ConcurrentModificationException during iteration?**

**248.How will you convert a Map to a List in Java?**

**249.How can we create a Map with reverse view and lookup in Java?**

**250.How will you create a shallow copy of a Map?**

1. **Why we cannot create a generic array in Java?**

**252.What is a PriorityQueue in Java?**

**253.What are the important points to remember while using Java Collections Framework?**

**254.How can we pass a Collection as an argument to a method and ensure that method will not be able to modify it?**

**255.Can you explain how HashMap works in Java?**

**256.Can you explain how HashSet is implemented in Java?**

**257.What is a NavigableMap in Java?**

**258.What is the difference between descendingKeySet() and descendingMap() methods of NavigableMap?**

**259.What is the advantage of NavigableMap over Map?**

**260.What is the difference between headMap(), tailMap() and subMap() methods of NavigableMap?**

1. **How will you sort objects by Natural order in a Java List?**

**262.How can we get a Stream from a List in Java?**

1. **Can we get a Map from a Stream in Java?**

**264.What are the popular implementations of Deque in Java? Advanced Multi-threading**

**265.What is a Thread in Java?**

**266.What is the priority of a Thread and how it is used in scheduling?**

**267.What is the default priority of a thread in Java?**

**268.What are the three different priorities that can be set on a Thread in Java?**

**269.What is the purpose of join() method in Thread class?**

**270.What is the fundamental difference between wait() and sleep() methods?**

**271. Is it possible to call run() method instead of start() on a thread in**

**Java?**

**272.What is a daemon thread in Java?**

**273.How can we make a regular thread Daemon thread in Java?**

**274.How will you make a user thread into daemon thread if it has already started?**

**275.Can we start a thread two times in Java?**

**276.What is a Shutdown hook in Java?**

**277.What is synchronization in Java?**

**278.What is the purpose of Synchronized block in Java?**

**279.What is static synchronization?**

**280.What is a Deadlock situation?**

**281. What is the meaning of concurrency?**

**282.What is the main difference between process and thread?**

**283.What is a process and thread in the context of Java?**

**284.What is a Scheduler?**

**285.What is the minimum number of Threads in a Java program?**

**286.What are the properties of a Java thread?**

**287.What are the different states of a Thread in Java?**

**288.How will you set the priority of a thread in Java?**

**289.What is the purpose of Thread Groups in Java?**

**290.Why we should not stop a thread by calling its stop() method?**

**291. How will you create a Thread in Java?**

**292.How can we stop a thread in the middle of execution in Java?**

**293.How do you access the current thread in a Java program?**

**294.What is Busy waiting in Multi-threading?**

**295.How can we prevent busy waiting in Java?**

**296.Can we use Thread.sleep() method for real-time processing in Java?**

**297.Can we wake up a thread that has been put to sleep by using Thread.sleep() method?**

**298.What are the two ways to check if a Thread has been interrupted?**

**299.How can we make sure that Parent thread waits for termination of Child thread?**

**300.How will you handle InterruptedException in Java?**

1. **Which intrinsic lock is acquired by a synchronized method in Java?**
2. **Can we mark a constructor as synchronized in Java?**
3. **Can we use primitive values for intrinsic locks?**

**304.Do we have re-entrant property in intrinsic locks?**

1. **What is an atomic operation?**
2. **Can we consider the statement i++ as an atomic operation in Java?**
3. **What are the Atomic operations in Java?**

**308.Can you check if following code is thread-safe?**

**309.What are the minimum requirements for a Deadlock situation in a program?**

1. **How can we prevent a Deadlock?**
2. **How can we detect a Deadlock situation?**

1. **What is a Livelock?**
2. **What is Thread starvation?**
3. **How can a synchronized block cause Thread starvation in Java?**
4. **What is a Race condition?**
5. **What is a Fair lock in multi-threading?**
6. **Which two methods of Object class can be used to implement a Producer Consumer scenario?**
7. **How JVM determines which thread should wake up on notify()?**
8. **Check if following code is thread-safe for retrieving an integer value from a Queue?**

**320.How can we check if a thread has a monitor lock on a given object?**

**321. What is the use of yield() method in Thread class?**

**322.What is an important point to consider while passing an object from one thread to another thread?**

**323.What are the rules for creating Immutable Objects?**

**324.What is the use of ThreadLocal class?**

**325.What are the scenarios suitable for using ThreadLocal class?**

**326.How will you improve the performance of an application by multi-threading?**

**327.What is scalability in a Software program?**

**328.How will you calculate the maximum speed up of an application by using multiple processors?**

**329.What is Lock contention in multi-threading?**

**330.What are the techniques to reduce Lock contention?**

1. **What technique can be used in following code to reduce Lock contention?**

**332.What is Lock splitting technique?**

**333.Which technique is used in ReadWriteLock class for reducing Lock contention?**

**334.What is Lock striping?**

**335.What is a CAS operation?**

**336.Which Java classes use CAS operation?**

1. **Is it always possible to improve performance by object pooling in a multi-threading application?**

**338.How can techniques used for performance improvement in a single thread application may degrade the performance in a multi-threading application?**

**339.What is the relation between Executor and ExecutorService interface?**

**340.What will happen on calling submit() method of an ExecutorService instance whose queue is already full?**

1. **What is a ScheduledExecutorService?**

**342.How will you create a Thread pool in Java?**

**343.What is the main difference between Runnable and Callable interface?**

**344.What are the uses of Future interface in Java?**

**345.What is the difference in concurrency in HashMap and in Hashtable?**

**346.How will you create synchronized instance of List or Map Collection?**

**347.What is a Semaphore in Java?**

**348.What is a CountDownLatch in Java?**

**349.What is the difference between CountDownLatch and CyclicBarrier?**

**350.What are the scenarios suitable for using Fork/Join framework?**

1. **What is the difference between RecursiveTask and RecursiveAction class?**
2. **In Java 8, can we process stream operations with a Thread pool?**
3. **What are the scenarios to use parallel stream in Java 8?**

**354.How Stack and Heap work in Java multi-threading environment?**

1. **How can we take Thread dump in Java?**

**356.Which parameter can be used to control stack size of a thread in Java?**

1. **There are two threads T1 and T2? How will you ensure that these threads run in sequence T1, T2 in Java?**

**358.What are the new features released in Java 8?**

**359.What are the main benefits of new features introduced in Java 8?**

**360.What is a Lambda expression in Java 8?**

1. **What are the three main parts of a Lambda expression in Java?**
2. **What is the data type of a Lambda expression?**
3. **What is the meaning of following lambda expression?**

**364.Why did Oracle release a new version of Java like Java 8?**

1. **What are the advantages of a lambda expression?**
2. **What is a Functional interface in Java 8?**

**367.What is a Single Abstract Method (SAM) interface in Java 8?**

**368.How can we define a Functional interface in Java 8?**

**369.Why do we need Functional interface in Java?**

1. **Is it mandatory to use @FunctionalInterface annotation to define a Functional interface in Java 8?**
2. **What are the differences between Collection and Stream API in Java**

**8?**

**372.What are the main uses of Stream API in Java 8?**

**373.What are the differences between Intermediate and Terminal Operations in Java 8 Streams?**

**374.What is a Spliterator in Java 8?**

**375.What are the differences between Iterator and Spliterator in Java 8?**

**376.What is Type Inference in Java 8?**

**377.Does Java 7 support Type Inference?**

**378.How does Internal Iteration work in Java 8?**

**379.What are the main differences between Internal and External Iterator?**

**380.What are the main advantages of Internal Iterator over External Iterator in Java 8?**

**381. What are the applications in which we should use Internal Iteration?**

**382.What is the main disadvantage of Internal Iteration over External Iteration?**

**383.Can we provide implementation of a method in a Java Interface?**

**384.What is a Default Method in an Interface?**

**385.Why do we need Default method in a Java 8 Interface?**

**386.What is the purpose of a Static method in an Interface in Java 8?**

**387.What are the core ideas behind the Date/Time API of Java 8?**

**388.What are the advantages of new Date and Time API in Java 8 over old Date API?**

**389.What are the main differences between legacy Date/Time API in Java and Date/Time API of Java 8?**

**390.How can we get duration between two dates or time in Java 8?**

1. **What is the new method family introduced in Java 8 for processing of Arrays on multi core machines?**

**392.How does Java 8 solve Diamond problem of Multiple Inheritance?**

**393.What are the differences between Predicate, Supplier and Consumer in Java 8?**

**394.Is it possible to have default method definition in an interface without marking it with default keyword?**

**395.Can we create a class that implements two Interfaces with default methods of same name and signature?**

**396.How Java 8 supports Multiple Inheritance?**

1. **In case we create a class that extends a base class and implements an interface. If both base class and interface have a default method with same name and arguments, then which definition will be picked by JVM?**

**398.If we create same method and define it in a class , in its parent class and in an interface implemented by the class, then definition will be invoked if we access it using the reference of Interface and the object of class?**

**399.Can we access a static method of an interface by using reference of**

**the interface?**

**400.How can you get the name of Parameter in Java by using reflection?**

1. **What is Optional in Java 8?**

**402.What are the uses of Optional?**

**403.Which method in Optional provides the fallback mechanism in case of null value?**

**404.How can we get current time by using Date/Time API of Java 8?**

**405.Is it possible to define a static method in an Interface?**

**406.How can we analyze the dependencies in Java classes and packages?**

**407.What are the new JVM arguments introduced by Java 8?**

**408.What are the popular annotations introduced in Java 8?**

**409.What is a StringJoiner in Java 8?**

1. **What is the type of a Lambda expression in Java 8?**
2. **What is the target type of a lambda expression ?**
3. **What are the main differences between an interface with default method and an abstract class in Java 8?**

**413. Is there any difference between a = a + b and a += b expressions?**

**414.What does the expression 1.0 / 0.0 return? Will there be any compilation error?**

1. **Can we use multiple main methods in multiple classes?**
2. **Does Java allow you to override a private or static method?**
3. **What happens when you put a key object in a HashMap that is already present?**

1. **How can you make sure that N threads can access N resources without deadlock?**
2. **How can you determine if JVM is 32-bit or 64-bit from Java Program?**

**420.What is the right data type to represent Money (like Dollar/Pound) in Java?**

1. **How can you do multiple inheritances in Java?**

**422.Is ++ operation thread-safe in Java?**

**423.How can you access a non-static variable from the static context?**

**424.Let say there is a method that throws NullPointerException in the superclass. Can we override it with a method that throws RuntimeException?**

**425.How can you mark an array volatile in Java?**

**426.What is a thread local variable in Java?**

**427.What is the difference between sleep() and wait() methods in Java?**

**428.Can you create an Immutable object that contains a mutable object?**

**429.How can you convert an Array of bytes to String?**

**430.What is difference between CyclicBarrier and CountDownLatch class?**

1. **What is the difference between StringBuffer and StringBuilder?**

**432.Which class contains clone method? Cloneable or Object class?**

**433.How will you take thread dump in Java?**

**434.Can you cast an int variable into a byte variable? What happens if the value of int is larger than byte?**

**435.In Java, can we store a double value in a long variable without explicit**

**casting?**

**436.What will this return 5\*0.1 == 0.5? true or false?**

**437.Out of an int and Integer, which one takes more memory?**

**438.Can we use String in the switch case statement in Java?**

**439.Can we use multiple main methods in same class?**

**440.When creating an abstract class, is it a good idea to call abstract methods inside its constructor?**

**441.How can you do constructor chaining in Java?**

**442.How can we find the memory usage of JVM from Java code?**

**443.What is the difference between x == y and x.equals(y) expressions in Java?**

1. **How can you guarantee that the garbage collection takes place?**

**445.What is the relation between x.hashCode() method and x.equals(y) method of Object class?**

**446.What is a compile time constant in Java?**

**447.Explain the difference between fail-fast and fail-safe iterators?**

1. **You have a character array and a String. Which one is more secure to store sensitive data (like password, date of birth, etc.)?**

**449.Why do you use volatile keyword in Java?**

**450.What is the difference between poll() and remove() methods of Queue in Java?**

**451. Can you catch an exception thrown by another thread in Java?**

**452.How do you decide which type of Inner Class – Static or Non-Static to use in Java?**

**453.What are the different types of Classloaders in Java?**

**454.What are the situations in which you choose HashSet or TreeSet?**

**455.What is the use of method references in Java?**

**456.Do you think Java Enums are more powerful than integer constants?**

**457.Why do we use static initializers in Java?**

**458.Your client is complaining that your code is throwing NoClassDefFoundError or NoSuchMethodError, even though you are able to compile your code without error and method exists in your code. What could be the reason behind this?**

**459.How can you check if a String is a number by using regular expression?**

**460.What is the difference between the expressions String s = "Temporary" and String s = new String("Temporary ")? Which one is better and more efficient?**

1. **In Java, can two equal objects have the different hash code?**

**462.How can we print an Array in Java?**

**463.Is it ok to use random numbers in the implementation of hashcode() method in Java?**

**464.Between two types of dependency injections, constructor injection and setter dependency injection, which one is better?**

**465.What is the difference between DOM and SAX parser in Java?**

**466.Between Enumeration and Iterator, which one has better performance in Java?**

**467.What is the difference between pass by reference and pass by value?**

**468.What are the different ways to sort a collection in Java?**

**469.Why Collection interface doesn’t extend Cloneable and Serializable interfaces?**

**470.What is the difference between a process and a thread in Java?**

1. **What are the benefits of using an unordered array over an ordered array?**

**472.Between HashSet and TreeSet collections in Java, which one is better?**

**473.When does JVM call the finalize() method?**

**474.When would you use Serial Garabage collector or Throughput Garbage collector in Java?**

**475.In Java, if you set an object reference to null, will the Garbage Collector immediately free the memory held by that object?**

**476.How can you make an Object eligible for Garbage collection in Java?**

**477.When do you use Exception or Error in Java? What is the difference between these two?**

**478.What is the advantage of PreparedStatement over Statement class in Java?**

**479.In Java, what is the difference between throw and throws keywords?**

**480.What happens to the Exception object after the exception handling is done?**

1. **How do you find which client machine is sending request to your servlet in Java?**

**482.What is the difference between a Cookie and a Session object in Java?**

**483.Which protocol does Browser and Servlet use to communicate with each other?**

**484.** **What is HTTP Tunneling?**

**485.Why do we use JSP instead of Servlet in Java?**

**486.Is empty ‘.java’ file name a valid source file name in Java?**

**487.How do you implement Servlet Chaining in Java?**

**488.Can you instantiate this class?**

**489.Why Java does not support operator overloading?**

**490.Why String class is Immutable or Final in Java?**

**491. What is the difference between sendRedirect and forward methods?**

**492.How do you fix your Serializable class, if it contains a member that is not serializable?**

**493.What is the use of run time polymorphism in Java?**

**494.What are the rules of method overloading and method overriding in Java?**

**495.What is the difference between a class and an object in Java?**

**496.Can we create an abstract class that extends another abstract class?**

**497.Why do you use Upcasting or Downcasting in Java ?**

**498.What is the reason to organize classes and interfaces in a package in Java?**

**499.What is information hiding in Java?**

**500.Why does Java provide default constructor?**

1. **What is the difference between super and this keywords in Java?**

**502.What is the advantage of using Unicode characters in Java?**

1. **Can you override an overloaded method in Java?**

**504.How can we change the heap size of a JVM?**

**505.Why should you define a default constructor in Java?**

**506.How will you make an Object Immutable in Java?**

**507.How can you prevent SQL Injection in Java Code?**

**508.Which two methods should be always implemented by HashMap key Object?**

**509.Why an Object used as Key in HashMap should be Immutable?**

1. **How can we share an object between multiple threads?**
2. **How can you determine if your program has a deadlock?**

**JSP**

1. **What are the implicit objects in JSP?**
2. **How will you extend JSP code?**
3. **How will you handle runtime exceptions in JSP?**
4. **How will you prevent multiple submits of a page that come by clicking refresh button multiple times?**
5. **How will you implement a thread safe JSP page?**
6. **How will you include a static file in a JSP page?**
7. **What are the lifecycle methods of a JSP?**
8. **What are the advantages of using JSP in web architecture?**

**520.What is the advantage of JSP over Javascript?**

1. **What is the Lifecycle of JSP?**

**522.What is a JSP expression?**

1. **What are the different types of directive tags in JSP?**

**524.What is session attribute in JSP?**

**525.What are the different scopes of a JSP object?**

**526.What is pageContext in JSP?**

**527.What is the use of jsp:useBean in JSP?**

**528.What is difference between include Directive and include Action of JSP?**

**529.How will you use other Java files of your application in JSP code?**

**530.How will you use an existing class and extend it to use in the JSP?**

1. **Why \_jspService method starts with \_ symbol in JSP?**
2. **Why do we use tag library in JSP?**
3. **What is the different type of tag library groups in JSTL?**

**534.How will you pass information from one JSP to another JSP?**

1. **How will you call a stored procedure from JSP?**
2. **Can we override \_jspService() method in JSP?**
3. **What is a directive in JSP?**

**538.How will you implement Session tracking in JSP?**

1. **How do you debug code in JSP?**

**540.How will you implement error page in JSP?**

1. **How will you send XML data from a JSP?**

**542.What happens when we request for a JSP page from web browser?**

**543.How will you implement Auto Refresh of page in JSP?**

**544.What are the important status codes in HTTP?**

**545.What is the meaning of Accept attribute in HTTP header?**

**546.What is the difference between Expression and Scriptlet in JSP?**

**547.How will you delete a Cookie in JSP?**

**548.How will you use a Cookie in JSP?**

**549.What is the main difference between a Session and Cookie in JSP?**

**550.How will you prevent creation of session in JSP?**

**551. What is an output comment in JSP?**

**552.How will you prevent caching of HTML output by web browser in JSP?**

**553.How will you redirect request to another page in browser in JSP code?**

**554.What is the difference between sendRedirect and forward in a JSP?**

**555.What is the use of config implicit object in JSP?**

**556.What is the difference between init-param and context-param?**

**557.What is the purpose of RequestDispatcher?**

**558.How can be read data from a Form in a JSP?**

**559.What is a filter in JSP?**

**560.How can you upload a large file in JSP?**

1. **In which scenario, Container initializes multiple JSP/Servlet objects? Java Design Patterns**

**562.When will you use Strategy Design Pattern in Java?**

1. **What is Observer design pattern?**

**564.What are the examples of Observer design pattern in JDK?**

**565.How Strategy design pattern is different from State design pattern in Java?**

**566.Can you explain Decorator design pattern with an example in Java?**

**567.What is a good scenario for using Composite design Pattern in Java?**

**568.Have you used Singleton design pattern in your Java project?**

**569.What are the main uses of Singleton design pattern in Java project?**

**570.Why java.lang.Runtime is a Singleton in Java?**

1. **What is the way to implement a thread-safe Singleton design pattern in Java?**

**572.What are the examples of Singleton design pattern in JDK?**

**573.What is Template Method design pattern in Java?**

**574.What are the examples of Template method design pattern in JDK?**

**575.Can you tell some examples of Factory Method design pattern implementation in Java?**

**576.What is the benefit we get by using static factory method to create object?**

**577.What are the examples of Builder design pattern in JDK?**

**578.What are the examples of Abstract Factory design pattern in JDK?**

**579.What are the examples of Decorator design pattern in JDK?**

**580.What are the examples of Proxy design pattern in JDK?**

1. **What are the examples of Chain of Responsibility design pattern in JDK?**

**582.What are the main uses of Command design pattern?**

**583.What are the examples of Command design pattern in JDK?**

**584.What are the examples of Interpreter design pattern in JDK?**

**585.What are the examples of Mediator design pattern in JDK?**

**586.What are the examples of Strategy design pattern in JDK?**

**587.What are the examples of Visitor design pattern in JDK?**

**588.How Decorator design pattern is different from Proxy pattern?**

**589.What are the different scenarios to use Setter and Constructor based injection in Dependency Injection (DI) design pattern?**

**590.What are the different scenarios for using Proxy design pattern?**

1. **What is the main difference between Adapter and Proxy design pattern?**

**592.When will you use Adapter design pattern in Java?**

**593.What are the examples of Adapter design pattern in JDK?**

**594.What is the difference between Factory and Abstract Factory design pattern?**

**595.What is Open/closed design principle in Software engineering?**

**596.What is SOLID design principle?**

**597.What is Builder design pattern?**

**598.What are the different categories of Design Patterns used in Object Oriented Design?**

**599.What is the design pattern suitable to access elements of a Collection?**

**600.How can we implement Producer Consumer design pattern in Java?**

1. **What design pattern is suitable to add new features to an existing object?**

**602.Which design pattern can be used when to decouple abstraction from the implementation?**

**603.Which is the design pattern used in Android applications?**

**604.How can we prevent users from creating more than one instance of singleton object by using clone() method?**

**605.What is the use of Interceptor design pattern?**

**606.What are the Architectural patterns that you have used?**

**607.What are the popular uses of Façade design pattern?**

**608.What is the difference between Builder design pattern and Factory design pattern?**

**609.What is Memento design pattern?**

1. **What is an AntiPattern?**
2. **What is a Data Access Object (DAO) design pattern? Spring Questions**
3. **What is Spring framework?**
4. **What are the benefits of Spring framework in software development?**
5. **What are the modules in Core Container of Spring framework?**
6. **What are the modules in Data Access/Integration layer of Spring framework?**
7. **What are the modules in Web layer of Spring framework?**
8. **What is the main use of Core Container module in Spring framework?**
9. **What kind of testing can be done in Spring Test Module?**
10. **What is the use of BeanFactory in Spring framework?**

**620.Which is the most popular implementation of BeanFactory in Spring?**

**621. What is XMLBeanFactory in Spring framework?**

**622.What are the uses of AOP module in Spring framework?**

**623.What are the benefits of JDBC abstraction layer module in Spring framework?**

**624.How does Spring support Object Relational Mapping (ORM) integration?**

**625.How does Web module work in Spring framework?**

**626.What are the main uses of Spring MVC module?**

**627.What is the purpose of Spring configuration file?**

**628.What is the purpose of Spring IoC container?**

**629.What is the main benefit of Inversion of Control (IOC) principle?**

**630.Does IOC containers support Eager Instantiation or Lazy loading of beans?**

**631. What are the benefits of ApplicationContext in Spring?**

**632.How will you implement ApplicationContext in Spring framework?**

1. **Explain the difference between ApplicationContext and BeanFactory in Spring?**

**634.Between ApplicationContext and BeanFactory which one is preferable to use in Spring?**

**635.What are the main components of a typical Spring based application?**

1. **Explain Dependency Injection (DI) concept in Spring framework?**
2. **What are the different roles in Dependency Injection (DI)?**

**638.Spring framework provides what kinds of Dependency Injection mechanism?**

1. **In Spring framework, which Dependency Injection is better? Constructor-based DI or Setter-based DI?**

**640.What are the advantages of Dependency Injection (DI)?**

1. **What are the disadvantages of Dependency Injection (DI)?**

**642.What is a Spring Bean?**

**643.What does the definition of a Spring Bean contain?**

**644.What are the different ways to provide configuration metadata to a Spring Container?**

**645.What are the different scopes of a Bean supported by Spring?**

**646.How will you define the scope of a bean in Spring?**

**647.Is it safe to assume that a Singleton bean is thread safe in Spring Framework?**

**648.What are the design-patterns used in Spring framework?**

**649.What is the lifecycle of a Bean in Spring framework?**

**650.What are the two main groups of methods in a Bean’s lifecycle?**

1. **Can we override main lifecycle methods of a Bean in Spring?**

**652.What are Inner beans in Spring?**

1. **How can we inject a Java Collection in Spring framework?**

**654.What is Bean wiring in Spring?**

**655.What is Autowiring in Spring?**

**656.What are the different modes of Autowiring supported by Spring?**

**657.What are the cases in which Autowiring may not work in Spring framework?**

**658.Is it allowed to inject null or empty String values in Spring?**

**659.What is a Java-based Configuration in Spring?**

**660.What is the purpose of @Configuration annotation?**

1. **What is the difference between Full @Configuration and 'lite' @Beans mode?**

**662.In Spring framework, what is Annotation-based container configuration?**

**663.How will you switch on Annotation based wiring in Spring?**

**664.What is @Autowired annotation?**

**665.What is @Required annotation?**

**666.What are the two ways to enable RequiredAnnotationBeanPostProcessor in Spring?**

**667.What is @Qualifier annotation in Spring?**

**668.How Spring framework makes JDBC coding easier for developers?**

**669.What is the purpose of JdbcTemplate?**

**670.What are the benefits of using Spring DAO?**

**671. What are the different ways to use Hibernate in Spring?**

**672.What types of Object Relational Mapping (ORM) are supported by Spring?**

**673.How will you integrate Spring and Hibernate by using HibernateDaoSupport?**

**674.What are the different types of the Transaction Management supported by Spring framework?**

**675.What are the benefits provided by Spring Framework’s Transaction Management?**

**676.Given a choice between declarative and programmatic Transaction Management, which method will you choose?**

**677.What is Aspect Oriented Programming (AOP)**

**678.What is an Aspect in Spring?**

1. **In Spring AOP, what is the main difference between a Concern and a Cross cutting concern?**

**680.What is a Joinpoint in Spring AOP?**

**681. What is an Advice in Spring AOP?**

**682.What are the different types of Advice in Spring AOP?**

**683.What is a Pointcut in Spring AOP?**

**684.What is an Introduction in Spring AOP?**

**685.What is a Target object in Spring AOP?**

**686.What is a Proxy in Spring AOP?**

**687.What are the different types of AutoProxy creators in Spring?**

**688.What is Weaving in Spring AOP?**

**689.In Spring AOP, Weaving is done at compile time or run time?**

**690.What is XML Schema-based Aspect implementation?**

1. **What is Annotation-based aspect implementation in Spring AOP?**

**692.How does Spring MVC framework work?**

1. **What is DispatcherServlet?**

**694.Can we have more than one DispatcherServlet in Spring MVC?**

**695.What is WebApplicationContext in Spring MVC?**

**696.What is Controller in Spring MVC framework?**

1. **What is @RequestMapping annotation in Spring?**

**698.What are the main features of Spring MVC?**

**699.What is the difference between a Singleton and Prototype bean in Spring?**

**700.How will you decide which scope- Prototype or Singleton to use for a bean in Spring?**

1. **What is the difference between Setter and Constructor based Dependency Injection (DI) in Spring framework?**

**702.What are the drawbacks of Setter based Dependency Injection (DI) in Spring?**

**703.What are the differences between Dependency Injection (DI) and Factory Pattern?**

**704.In Spring framework, what is the difference between FileSystemResource and ClassPathResource?**

**705.Name some popular Spring framework annotations that you use in your project?**

**706.How can you upload a file in Spring MVC Application?**

**707.What are the different types of events provided by Spring framework?**

**708.What is the difference between DispatcherServlet and ContextLoaderListener in Spring?**

**709.How will you handle exceptions in Spring MVC Framework?**

1. **What are the best practices of Spring Framework?**
2. **What is Spring Boot?**

**Hibernate**

**712. What is Hibernate framework?**

1. **What is an Object Relational Mapping (ORM)?**
2. **What is the purpose of Configuration Interface in Hibernate?**
3. **What is Object Relational Impedance Mismatch?**
4. **What are the main problems of Object Relational Impedance Mismatch?**
5. **What are the key characteristics of Hibernate?**
6. **Can you tell us about the core interfaces of Hibernate framework?**
7. **How will you map the columns of a DB table to the properties of a Java class in Hibernate?**

**720.Does Hibernate make it mandatory for a mapping file to have .hbm.xml extension?**

1. **What are the steps for creating a SessionFactory in Hibernate?**

**722.Why do we use POJO in Hibernate?**

1. **What is Hibernate Query Language (HQL)?**

**724.How will you call a stored procedure in Hibernate?**

**725.What is Criteria API in Hibernate?**

1. **Why do we use HibernateTemplate?**
2. **How can you see SQL code generated by Hibernate on console?**

**728.What are the different types of collections supported by Hibernate?**

**729.What is the difference between session.save() and session.saveOrUpdate() methods in Hibernate?**

**730.What are the advantages of Hibernate framework over JDBC?**

**731. How can we get statistics of a SessionFactory in Hibernate?**

**732.What is the Transient state of an object in Hibernate?**

**733.What is the Detached state of an object in Hibernate?**

**734.What is the use of Dirty Checking in Hibernate?**

**735.What is the purpose of Callback interface in Hibernate?**

**736.What are the different ORM levels in Hibernate?**

**737.What are the different ways to configure a Hibernate application?**

**738.What is Query Cache in Hibernate?**

**739.What are the different types of Association mappings supported by Hibernate?**

**740.What are the different types of Unidirectional Association mappings in Hibernate?**

**741. What is Unit of Work design pattern?**

**742.In Hibernate, how can an object go in Detached state?**

**743.How will you order the results returned by a Criteria in Hibernate?**

**744.How does Example criterion work in Hibernate?**

**745.How does Transaction management work in Hibernate?**

**746.How can we mark an entity/collection as immutable in Hibernate?**

**747.What are the different options to retrieve an object from database in Hibernate?**

**748.How can we auto-generate primary key in Hibernate?**

**749.How will you re-attach an object in Detached state in Hibernate?**

**750.What is the first level of cache in Hibernate?**

**751. What are the different second level caches available in Hibernate?**

**752.Which is the default transaction factory in Hibernate?**

**753.What are the options to disable second level cache in Hibernate?**

**754.What are the different fetching strategies in Hibernate?**

**755.What is the difference between Immediate fetching and Lazy collection fetching?**

**756.What is ‘Extra lazy fetching’ in Hibernate?**

**757.How can we check is a collection is initialized or not under Lazy Initialization strategy?**

**758.What are the different strategies for cache mapping in Hibernate?**

**759.What is the difference between a Set and a Bag in Hibernate?**

**760.How can we monitor the performance of Hibernate in an application?**

1. **How can we check if an Object is in Persistent, Detached or Transient state in Hibernate?**

**762.What is ‘the inverse side of association’ in a mapping?**

**763.What is ORM metadata?**

**764.What is the difference between load() and get() method in Hibernate?**

**765.When should we use get() method or load() method in Hibernate?**

**766.What is a derived property in Hibernate?**

**767.How can we use Named Query in Hibernate?**

**768.What are the two locking strategies in Hibernate?**

**769.What is the use of version number in Hibernate?**

**770.What is the use of session.lock() method in Hibernate?**

**771. What inheritance mapping strategies are supported by Hibernate?**

**Maven**

**772.What is Maven?**

**773.What are the main features of Maven?**

**774.What areas of a Project can you manage by using Maven?**

**775.What are the main advantages of Maven?**

**776.Why do we say “Maven uses convention over configuration”?**

**777.What are the responsibilities of a Build tool like Maven?**

**778.What are the differences between Ant and Maven?**

**779.What is MOJO in Maven?**

**780.What is a Repository in Maven?**

1. **What are the different types of repositories in Maven?**

**782.What is a local repository in Maven?**

**783.What is a central repository in Maven?**

**784.What is a Remote repository in Maven?**

**785.Why we should not store jars in CVS or any other version control system instead of Maven repository?**

**786.Can anyone upload JARS or artifacts to Central Repository?**

**787.What is a POM?**

**788.What is Super POM?**

**789.What are the main required elements in POM file?**

**790.What are the phases in Build lifecycle in Maven?**

**791. What command will you use to package your Maven project?**

**792.What is the format of fully qualified artifact name of a Maven project?**

**793.What is an Archetype in Maven?**

**794.What is the command in Maven to generate an Archetype?**

**795.What are the three main build lifecycles of Maven?**

**796.What are the main uses of a Maven plugin?**

**797.How will you find the version of a plugin being used?**

**798.What are the different types of profile in Maven? Where will you define these profiles?**

**799.What are the different setting files in Maven? Where will you find these files?**

**800.What are the main elements we can find in settings.xml?**

1. **How will you check the version of Maven in your system?**

**802.How will you verify if Maven is installed on Windows?**

**803.What is a Maven artifact?**

**804.What are the different dependency scopes in Maven?**

**805.How can we exclude a dependency in Maven?**

**806.How Maven searches for JAR corresponding to a dependency?**

**807.What is a transitive dependency in Maven?**

**808.What are Excluded dependencies in Maven?**

**809.What are Optional dependencies in Maven?**

1. **Where will you find the class files after compiling a Maven project successfully?**

1. **What are the default locations for source, test and build directories in Maven?**
2. **What is the result of jar:jar goal in Maven?**
3. **How can we get the debug or error messages from the execution of Maven?**
4. **What is the difference between a Release version and SNAPSHOT version in Maven?**
5. **How will you run test classes in Maven?**
6. **Sometimes Maven compiles the test classes but doesn't run them? What could be the reason for it?**
7. **How can we skip the running of tests in Maven?**
8. **Can we create our own directory structure for a project in Maven?**
9. **What are the differences between Gradle and Maven?**

**820.What is the difference between Inheritance and Multi-module in Maven?**

1. **What is Build portability in Maven?**

**GIT**

**822.How can we see n most recent commits in GIT?**

**823.How can we know if a branch is already merged into master in GIT?**

**824.What is the purpose of git stash drop?**

**825.What is the HEAD in GIT?**

**826.What is the most popular branching strategy in GIT?**

**827.What is SubGit?**

**828.What is the use of git instaweb?**

**829.What are git hooks?**

**830.What is GIT?**

**831. What is a repository in GIT?**

**832.What are the main benefits of GIT?**

**833.What are the disadvantages of GIT?**

**834.What are the main differences between GIT and SVN?**

**835.How will you start GIT for your project?**

**836.What is git clone in GIT?**

**837.How will you create a repository in GIT?**

**838.What are the different ways to start work in GIT?**

**839.GIT is written in which language?**

**840.What does ‘git pull’ command in GIT do internally?**

1. **What does ‘git push’ command in GIT do internally?**

**842.What is git stash?**

**843.What is the meaning of ‘stage’ in GIT?**

**844.** **What is the purpose of git config command?**

**845.How can we see the configuration settings of GIT installation?**

**846.How will you write a message with commit command in GIT?**

**847.What is stored inside a commit object in GIT?**

**848.How many heads can you create in a GIT repository?**

**849.Why do we create branches in GIT?**

**850.What are the different kinds of branches that can be created in GIT?**

**851. How will you create a new branch in GIT?**

**852.How will you add a new feature to the main branch?**

**853.What is a pull request in GIT?**

**854.What is merge conflict in GIT?**

**855.How can we resolve a merge conflict in GIT?**

**856.What command will you use to delete a branch?**

**857.What command will you use to delete a branch that has unmerged changes?**

**858.What is the alternative command to merging in GIT?**

**859.What is Rebasing in GIT?**

**860.What is the ‘Golden Rule of Rebasing’ in GIT?**

1. **Why do we use Interactive Rebasing in place of Auto Rebasing?**

**862.What is the command for Rebasing in Git?**

**863.What is the main difference between git clone and git remote?**

**864.What is GIT version control?**

**865.What GUI do you use for working on GIT?**

**866.What is the use of git diff command in GIT?**

**867.What is git rerere?**

**868.What are the three most popular version of git diff command?**

**869.What is the use of git status command?**

**870.What is the main difference between git diff and git status?**

1. **What is the use of git rm command in GIT?**

**872.What is the command to apply a stash?**

**873.Why do we use git log command?**

**874.Why do we need git add command in GIT?**

**875.Why do we use git reset command?**

**876.What does a commit object contain?**

**877.How can we convert git log messages to a different format?**

**878.What are the programming languages in which git hooks can be written?**

**879.What is a commit message in GIT?**

**880.How GIT protects the code in a repository?**

1. **How GIT provides flexibility in version control?**

**882.How can we change a commit message in GIT?**

**883.Why is it advisable to create an additional commit instead of amending an existing commit?**

**884.What is a bare repository in GIT?**

**885.How do we put a local repository on GitHub server?**

**886.How will you delete a branch in GIT?**

**887.How can we set up a Git repository to run code sanity checks and UAT tests just before a commit?**

**888.How can we revert a commit that was pushed earlier and is public now?**

**889.In GIT, how will you compress last n commits into a single commit?**

**890.How will you switch from one branch to a new branch in GIT?**

1. **How can we clean unwanted files from our working directory in GIT?**

**892.What is the purpose of git tag command?**

**893.What is cherry-pick in GIT?**

**894.What is shortlog in GIT?**

**895.How can you find the names of files that were changed in a specific commit?**

**896.How can we attach an automated script to run on the event of a new commit by push command?**

**897.What is the difference between pre-receive, update and post-receive hooks in GIT?**

**898.Do we have to store Scripts for GIT hooks within same repository?**

**899.How can we determine the commit that is the source of a bug in GIT?**

**900.How can we see differences between two commits in GIT?**

**901. What are the different ways to identify a commit in GIT?**

**902.When we run git branch <branchname>, how does GIT know the SHA-1 of the last commit?**

**903.What are the different types of Tags you can create in GIT?**

**904.How can we rename a remote repository?**

**905.Some people use git checkout and some use git co for checkout. How is that possible?**

**906.How can we see the last commit on each of our branch in GIT?**

**907.Is origin a special branch in GIT?**

**908.How can we configure GIT to not ask for password every time?**

**909.What are the four major protocols used by GIT for data transfer?**

1. **What is GIT protocol?**
2. **How can we work on a project where we do not have push access?**
3. **What is git grep?**
4. **How can your reorder commits in GIT?**
5. **How will you split a commit into multiple commits?**
6. **What is filter-branch in GIT?**
7. **What are the three main trees maintained by GIT?**
8. **What are the three main steps of working GIT?**
9. **What are ours and theirs merge options in GIT?**
10. **How can we ignore merge conflicts due to Whitespace?**

**920.What is git blame?**

1. **What is a submodule in GIT?**

**AWS**

**922.What do you know about AWS Region?**

**923.What are the important components of IAM?**

**924.What are the important points about AWS IAM?**

**925.What are the important features of Amazon S3?**

**926.What is the scale of durability in Amazon S3?**

**927.What are the Consistency levels supported by Amazon S3?**

**928.What are the different tiers in Amazon S3 storage?**

**929.How will you upload a file greater than 100 megabytes in Amazon S3?**

1. **What happens to an Object when we delete it from Amazon S3?**

**931. What is the use of Amazon Glacier?**

**932.Can we disable versioning on a version-enabled bucket in Amazon S3?**

**933.What are the use cases of Cross Region Replication Amazon S3?**

**934.Can we do Cross Region replication in Amazon S3 without enabling versioning on a bucket?**

**935.What are the different types of actions in Object Lifecycle Management in Amazon S3?**

**936.How do we get higher performance in our application by using Amazon CloudFront?**

**937.What is the mechanism behind Regional Edge Cache in Amazon CloudFront?**

**938.What are the benefits of Streaming content?**

**939.What is Lambda@Edge in AWS?**

**940.What are the different types of events triggered by Amazon CloudFront?**

1. **What is Geo Targeting in Amazon CloudFront?**

**942.What are the main features of Amazon CloudFront?**

**943.What are the security mechanisms available in Amazon S3? Cloud Computing**

**944.What are the benefits of Cloud Computing?**

**945.What is On-demand computing in Cloud Computing?**

**946.What are the different layers of Cloud computing?**

**947.What resources are provided by Infrastructure as a Service (IAAS) provider?**

**948.What is the benefit of Platform as a Service?**

**949.What are the main advantages of PaaS?**

**950.What is the main disadvantage of PaaS?**

1. **What are the different deployment models in Cloud computing?**

**952.What is the difference between Scalability and Elasticity?**

1. **What is Software as a Service?**

**954.What are the different types of Datacenters in Cloud computing?**

**955.Explain the various modes of Software as a Service (SaaS) cloud environment?**

**956.What are the important things to care about in Security in a cloud environment?**

**957.Why do we use API in cloud computing environment?**

**958.What are the different areas of Security Management in cloud?**

**959.What are the main cost factors of cloud based data center?**

**960.How can we measure the cloud-based services?**

**961. How a traditional datacenter is different from a cloud environment?**

**962.How will you optimize availability of your application in a Cloud environment?**

**963.What are the requirements for implementing IaaS strategy in Cloud? DOCKER**

**964.What is Docker?**

**965.What is the difference between Docker image and Docker container?**

**966.How will you remove an image from Docker?**

**967.How is a Docker container different from a hypervisor?**

**968.Can we write compose file in json file instead of yaml?**

**969.Can we run multiple apps on one server with Docker?**

**970.What are the common use cases of Docker?**

1. **What are the main features of Docker-compose?**

**972.What is the most popular use of Docker?**

**973.What is the role of open source development in the popularity of Docker?**

**974.How will you remove all files in current directory? Including the files that are two levels down in a sub-directory.**

**975.What is the difference between the –v and –x options in Bash shell scripts?**

**976.What is a Filter in Unix command?**

**977.What is Kernel in Unix operating system?**

**978.What is a Shell in Unix OS?**

**979.What are the different shells in Unix that you know about?**

**980.What is the first character of the output in ls –l command ?**

1. **What is the difference between Multi-tasking and Multi-user environment?**

**982.What is Command Substitution in Unix?**

**983.What is an Inode in Unix?**

**984.What is the difference between absolute path and relative path in Unix file system?**

**985.What are the main responsibilities of a Unix Shell?**

**986.What is a Shell variable?**

**Microservices**

**987.What is a Microservice?**

**988.What are the benefits of Microservices architecture?**

**989.What is the role of architect in Microservices architecture?**

**990.What is the advantage of Microservices architecture over Service Oriented Architecture (SOA)?**

1. **Is it a good idea to provide a Tailored Service Template for Microservices development in an organization?**

**992.What are the disadvantages of using Shared libraries approach to decompose a monolith application?**

**993.What are the characteristics of a Good Microservice?**

**994.What is Bounded Context?**

**995.What are the points to remember during integration of Microservices?**

**996.Is it a good idea for Microservices to share a common database?**

**997.What is the preferred type of communication between Microservices?**

**Synchronous or Asynchronous?**

**998.What is the difference between Orchestration and Choreography in Microservices architecture?**

**999.What are the issues in using REST over HTTP for Microservices?**

**1000.** **Can we create Microservices as State Machines?**

ACKNOWLEDGMENTS

We thank our readers who constantly send

feedback and reviews to motivate us in creating these useful books with the latest information!

INTRODUCTION

Java is one of the most popular programming language. There is a growing demand for Java Developer jobs in technology companies.

This book contains technical interview questions that an interviewer asks for Java technology and related topics like Spring, Hibernate, Maven, Git, Microservices, AWS etc.

Each question is accompanied with an answer so that you can prepare for job interview in short time.

We have compiled this list after attending dozens of technical interviews in top-notch companies like- Facebook, Oracle, Netflix, Amazon etc.

Once you go through them in the first pass, mark the questions that you could not answer by yourself. Then, in second pass go through only the difficult questions.

After going through this book 2-3 times, you will be well prepared to face a technical interview for a Java Developer position from Software Engineer level to Principal Engineer level.

All the best!!

Java Interview Questions

**Java Basics**

1. **What is the difference between JDK and JRE?**

JDK stands for Java Development Kit. It contains the tools and libraries for development of Java programs. It also contains compilers and debuggers needed to compile Java program,

JRE stands for Java Runtime Environment. This is included in JDK. JRE provides libraries and JVM that is required to run a Java program.

1. **What is Java Virtual Machine (JVM)?**

Java Virtual Machine (JVM) is an abstract machine that executes Java Bytecode. There are different JVM for different hardware and software platforms. So JVM is platform dependent. JVM is responsible for loading, verifying and executing the Bytecode on a platform.

1. **What are the different types of memory areas allocated by JVM?**

In java, JVM allocates memory to different processes, methods and objects. Some of the memory areas allocated by JVM are:

1. ClassLoader: It is a component of JVM used to load class files.
2. Class (Method) Area: It stores per-class structures such as the runtime constant pool, field and method data, and the code for methods.
3. Heap: Heap is created a runtime and it contains the runtime data area in which objects are allocated.
4. Stack: Stack stores local variables and partial results at runtime. It also helps in method invocation and return value. Each thread creates a private JVM stack at the time of thread creation.
5. Program Counter Register: This memory area contains the address of the Java virtual machine instruction that is currently being executed.
6. Native Method Stack: This area is reserved for all the native methods used in the application.

**4. What is JIT compiler?**

Just In Time compiler also known as JIT compiler is used for performance improvement in Java. It is enabled by default. It is compilation done at execution time rather earlier.

Java has popularized the use of JIT compiler by including it in JVM.

1. **How Java platform is different from other platforms?**

Java is a platform independent language. Java compiler converts

Java code in to byte code that can be interpreted by JVM. There are

JVM written for almost all the popular platforms in the world.

Java byte code can run on any supported platform in same way. Where as other languages require libraries compiled for a specific platform to run.

1. **Why people say that Java is 'write once and run anywhere' language?**

You can write Java code on Windows and compile it in Windows platform. The class and jar files that you get from Windows platform can run as it is on Unix environment. So it is a truly platform independent language.

Behind all this portability is Java byte code. Byte code generated by Java compiler can be interpreted by any JVM. So it becomes much easier to write programs in Java and expect those to run on any platform.

Java compiler javac compiles java code and JVM java runs that code.

1. **How does ClassLoader work in Java?**

In Java, ClassLoader is a class that is used to load files in JVM.

ClassLoader loads files from their physical file locations e.g.

Filesystem, Network location etc.

There are three main types of ClassLoaders in Java.

1. Bootstrap ClassLoader: This is the first ClassLoader. It loads classes from rt.jar file.
2. Extension ClassLoader: It loads class files from jre/lib/ext location.
   1. Application ClassLoader: This ClassLoader depends on CLASSPATH to find the location of class files. If you specify your jars in CLASSPATH, then this ClassLoader will load them.

1. **Do you think ‘main’ used for main method is a keyword in Java?**

No, main is just a name of method. There can be multiple methods with same name main in a class file. It is not a keyword in Java.

1. **Can we write main method as public void static instead of public static void?**

No, you cannot write it like this. Any method has to first specify the modifiers and then the return value. The order of modifiers can change.

We can write static public void main() instead of public static void main().

1. **In Java, if we do not specify any value for local variables, then what will be the default value of the local variables?**

Java does not initialize local variables with any default value. So these variables will be just null by default.

1. **Let say, we run a java class without passing any arguments. What will be the value of String array of arguments in Main method?**

By default, the value of String array of arguments is empty in Java.

It is not null.

1. **What is the difference between byte and char data types in Java?**

Both byte and char are numeric data types in Java. They are used to represent numbers in a specific range.

Major difference between them is that a byte can store raw binary data where as a char stores characters or text data.

Usage of char is E.g. char ch = ‘x’;

Byte values range from -128 to 127.

A byte is made of 8 bits. But a char is made of 16 bits. So it is equivalent to 2 bytes.

**OOPS**

1. **What are the main principles of Object Oriented Programming?**

Main principles of Object Oriented Programming (OOPS) are:

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

1. **What is the difference between Object Oriented Programming language and Object Based Programming language?**

Object Oriented Programming languages like Java and C++ follow concepts of OOPS like- Encapsulation, Abstraction, Polymorphism and Inheritance etc.

Object Based Programming languages follow some features of OOPS but they do not provide support for Polymorphism and Inheritance. Egg. JavaScript, VBScript etc.

Object Based Programming languages provide support for Objects and you can build objects from constructor. They languages also support Encapsulation. These are also known as Prototype-oriented languages.

1. **In Java what is the default value of an object reference defined as an instance variable in an Object?**

All the instance variable object references in Java are null.

1. **Why do we need constructor in Java?**

Java is an object-oriented language, in which we create and use objects. A constructor is a piece of code similar to a method. It is used to create an object and set the initial state of the object.

A constructor is a special function that has same name as class name.

Without a constructor, there is no other way to create an object.

By default, Java provides a default constructor for every object. If we overload a constructor then we have to implement default constructor.

1. **Why do we need default constructor in Java classes?**

Default constructor is the no-argument constructor that is automatically generated by Java if no other constructor is defined.

Java specification says that it will provide a default constructor if there is no overloaded constructor in a class. But it does not say anything about the scenario in which we write an overloaded constructor in a class.

We need at least one constructor to create an object, that’s why Java provides a default constructor.

When we have overloaded constructor, then Java assumes that we want some custom treatment in our code. Due to which it does not provide default constructor. But it needs default constructor as per the specification. So it gives error.

1. **What is the value returned by Constructor in Java?**

When we call a constructor in Java, it returns the object created by it. That is how we create new objects in Java.

**19. Can we inherit a Constructor?**

No, Java does not support inheritance of constructor.

1. **Why constructors cannot be final, static, or abstract in Java?**

If we set a method as final it means we do not want any class to override it. But the constructor (as per Java Language Specification) cannot be overridden. So there is no use of marking it final.

If we set a method as abstract it means that it has no body and it should be implemented in a child class. But the constructor is called implicitly when the new keyword is used. Therefore it needs a body.

If we set a method as static it means that it belongs to the class, but not a particular object. The constructor is always called to initialize an object. Therefore, there is no use of marking constructor static.

**Inheritance**

1. **What is the purpose of ‘this’ keyword in java?**

In Java, ‘this’ keyword refers to current instance of the object.

It is useful for differentiating between instance variables and local variables.

It can be used to call constructors. Or it can be used to refer to the instance.

In case of method overriding, this is used for falling the method of current class.

1. **Explain the concept of Inheritance?**

Inheritance is an important concept in Object Oriented Programming. Some objects share certain characteristics and behavior. By using Inheritance, we can put the common behavior and characteristics in a base class which also known as super class. And then all the objects with common behavior inherit from this base class.

It is also represented by IS-A relationship.

Inheritance promotes, code reuse, method overriding and poly-morphism.

1. **Which class in Java is superclass of every other class?**

Java is an object oriented programming language. In Java, Object class is the superclass of every other class.

1. **Why Java does not support multiple inheritance?**

Multiple Inheritance means that a class can inherit behavior from two or more parent classes.

The issue with Multiple Inheritance is that both the parent classes may have different implementation for the same method. So they have different ways of doing the same thing. Now which implementation should the child class choose?

This leads to ambiguity in Multiple Inheritance. This is the main reason for Java not supporting Multiple Inheritance in implementation.

Lets say you have a class TV and another class AtomBomb. Both have method switchOn() but only TV has switchOff() method. If your class inherits from both these classes then you have an issue that you can switchOn() both parents, but switchOff will only switchOff() TV.

But you can implement multiple interfaces in Java.

1. **In OOPS, what is meant by composition?**

Composition is also known as “has-a” relationship. In composition, “has-a” relation relates two classes. E.g. Class Car has a steering wheel.

If a class holds the instance of another class, then it is called composition.

1. **How aggregation and composition are different concepts?**

In OOPS, Aggregation and Composition are the types of association relations. A composition is a strong relationship. If the composite object is destroyed, then all its parts are destroyed. E.g. A Car has a Steering Wheel. If Car object is destroyed, then there is no meaning of Steering Wheel.

In Aggregation, the relationship is weaker than Composition.

E.g. A Library has students. If a Library is destroyed, Students still exist. So Library and Student are related by Aggregation. A Library has Books. If Library is destroyed, the Books are also destroyed. Books of a Library cannot exist without the Library. So Book and Library are related by Composition.

1. **Why there are no pointers in Java?**

In Java there are references instead of pointers. These references point to objects in memory. But there is no direct access to these memory locations. JVM is free to move the objects within VM memory.

The absence of pointers helps Java in managing memory and garbage collection effectively. Also it provides developers with convenience of not getting worried about memory allocation and de-allocation.

1. **If there are no pointers in Java, then why do we get NullPointerException?**

In Java, the pointer equivalent is Object reference. When we use a . it points to object reference. So JVM uses pointers but programmers only see object references.

In case an object reference points to null object, and we try to access a method or member variable on it, then we get NullPointerException.

1. **What is the purpose of ‘super’ keyword in java?**

‘super’ keyword is used in the methods or constructor of a child class. It refers to immediate parent class of an object.

By using ‘super’ we can call a method of parent class from the method of a child class.

We can also call the constructor of a parent class from the constructor of a child class by using ‘super’ keyword.

1. **Is it possible to use this() and super() both in same constructor?**

No, Java does not allow using both super() and this() in same constructor. As per Java specification, super() or this() must be the first statement in a constructor.

1. **What is the meaning of object cloning in Java?**

Object.clone() method is used for creating an exact copy of the object in Java. It acts like a copy constructor. It creates and returns a copy of the object, with the same class and with all the fields having same values as of the original object.

One disadvantage of cloning is that the return type is an Object. It has to be explicitly cast to actual type.

**Static**

1. **In Java, why do we use static variable?**

Whenever we want to have a common property for all objects of a class, we use a class level variable i.e. a static variable.

This variable is loaded in memory only once at the time of class loading. So it saves memory, since it is not defined per object in Java.

1. **Why it is not a good practice to create static variables in Java?**

Static variables are common to all the objects of a class. If a new object is created, there is no need to test the value of static variable. Any code that uses static variable can be in any state. It can be within a new object or at a class level. So the scope of static variable is open ended in a Java class.

If we want tighter control on scope, then variables should be created at the object creation level.

Also defining static variables is not a good practice because they go against the principles of Object Oriented Programming.

1. **What is the purpose of static method in Java?**

Java provides the feature of static method to create behavior at the class level. The static method is common to all the objects of a class. We do not need to create any object of a class to call a static method. So it provides convenience of not creating an object for calling it.

Also a static method can access and modify static data members. This also helps in keeping the behavior as well as state at the class level.

1. **Why do we mark main method as static in Java?**

The main method in Java is marked as static, so that JVM can call it to start the program. If main method is not static, then which constructor will be called by Java process?

As such it is a known as convention to mark main method static in Java. But if we remove the static, then there will be ambiguity. Java process may not know which method of a class to call to start the program.

So this convention helps in Java process to identify the starting code for a program in class that is passed as an argument to java process.

1. **In what scenario do we use a static block?**

At times, there is a class that has static member variables. These variables need some complicated initialization. At this time static block helps as a tool to initialize complex static member variable initialization.

The static block is executed even before the execution of main.

Sometimes, we can also replace static block with a static method of class.

1. **Is it possible to execute a program without defining a main() method?**

No, with Java 7 onwards, you need a main() method to execute a program. In earlier versions of Java, there was a workaround available to use static blocks for execution. But now this gap has been closed.

1. **What happens when static modifier is not mentioned in the signature of main method?**

As per Java specification, main method has to be marked as static.

It needs only one argument that is an array of String.

A program can compile with a non-static method. But on execution it will give NoSuchMethodError.

1. **What is the difference between static method and instance method in Java?**

Often, there is a need to define a behavior for a class that is not dependent on member variables of an object. Such behavior is captured in a static method. If there is a behavior dependent upon the member variables of an object, then we do not mark it static, it remains as instance method.

To call as static method, we do not need to create an object. We just call it with class name. But to call an instance method, we need to create/get an object first.

Instance member variables cannot be accessed by a static method. But an instance method can call both instance variables and static variables.

**Method Overloading and Overriding**

1. **What is the other name of Method Overloading?**

Method Overloading is also known as Static Polymorphism.

1. **How will you implement method overloading in Java?**

In Java, a class can have multiple methods with same name but different arguments. It is called Method Overloading. To implement method overloading we have to create two methods with same name in a class and do one/more of the following:

1. Different number of parameters
2. Different data type of parameters
3. Different sequence of data type of parameters

1. **What kinds of argument variations are allowed in Method Overloading?**

Method Overloading allows two methods with same name to differ in:

1. Number of parameters
2. Data type of parameters
3. Sequence of data type of parameters

1. **Why it is not possible to do method overloading by changing return type of method in java?**

If we change the return type of overloaded methods then it will lead to ambiguous behavior. How will clients know which method will return what type. Due to this different return type are not allowed in overloaded methods.

1. **Is it allowed to overload main() method in Java?**

Yes, Java allows users to create many methods with same name ‘main’. But only public static void main(String[] args) method is used for execution.

1. **How do we implement method overriding in Java?**

To override a method, we just provide a new implementation of a method with same name in subclass. So there will be at least two implementations of the method with same name. One implementation is in parent class. And another implementation is in child class.

1. **Are we allowed to override a static method in Java?**

No. Java does not allow overriding a static method. If you create a static method with same name in subclass, then it is a new method, not an overridden method.

1. **Why Java does not allow overriding a static method?**

To override a method, you need an instance of a class. Static method is not associated with any instance of the class. So the concept of overriding does not apply here.

Therefore, Java does not allow overriding a static method.

1. **Is it allowed to override an overloaded method?**

Yes. You can override an overloaded method in Java.

1. **What is the difference between method overloading and method overriding in Java?**

Differences between method overloading and overriding are:

1. Method overloading is static polymorphism. Method overriding is runtime polymorphism.
2. Method overloading occurs within the same class. Method overriding happens in two classes with hierarchy relationship.
3. Parameters must be different in method overloading. Parameters must be same in method overriding.
4. Method overloading is a compile time concept. Method overriding is a runtime concept.

**50. Does Java allow virtual functions?**

Yes. All instance methods in Java are virtual functions by default. Only class methods and private instance methods are not virtual methods in Java.

1. **What is meant by covariant return type in Java?**

A covariant return type of a method is one that can be replaced by a "narrower" type when the method is overridden in a subclass.

Let say class B is child of class A. There is a get() method in class A as well as class B. get() method of class A can return an instance of A, and get() method of class B return an instance of B. Here class B overrides get() method, but the return type is different.

Before Java 5, any method that overrides the method of parent class would have same return type.

From Java 5 onwards, a child class can override a method of parent class and the child class method can return an object that is child of object return by parent class method.

**Polymorphism**

**52. What is Runtime Polymorphism?**

Runtime Polymorphism or Dynamic Polymorphism is the polymorphism that exists at runtime. In case of method overriding it is not known which method will be called at runtime. Based on the type of object, JVM decides the exact method that should be called.

So at compile time it is not known which method will be called at run time.

1. **Is it possible to achieve Runtime Polymorphism by data members in Java?**

No. We need to create Runtime Polymorphism by implementing methods at two levels of inheritance in Java.

1. **Explain the difference between static and dynamic binding?**

In Static binding references are resolved at compile time. In Dynamic binding references are resolved at Run time.

E.g.

Person p = new Person();

p.walk(); // Java compiler resolves this binding at compile time.

public void walk(Object o){

((Person) o).walk(); // this is dynamic binding.

}

**Abstraction**

1. **What is Abstraction in Object Oriented programming?**

Abstraction is the process of hiding certain implementation details of an object and showing only essential features of the object to outside world.

It is different from Abstract class in Java.

Abstraction process identifies commonalities and hides the complexity of implementation. It helps us in focusing on the interface that we share with the outside world.

1. **How is Abstraction different from Encapsulation?**

Abstraction happens at class level design. It results in hiding the implementation details. Encapsulation is also known as “Information Hiding”. An example of encapsulation is marking the member variables private and providing getter and setter for these member variables.

**57. What is an abstract class in Java?**

An abstract class in Java has one or more abstract methods. An abstract method is just declared in the abstract class, but it is not implemented.

An abstract class has to be extended in Java and its abstract methods have to be implemented by a child class. Also Java does not allow new instance of Abstract class.

1. **Is it allowed to mark a method abstract method without marking the class abstract?**

No. Java specification says that if there is at least one abstract method in a class, the class has to be marked abstract.

1. **Is it allowed to mark a method abstract as well as final?**

No. It will be contradictory statement to mark a method abstract as well as final.

An abstract method has to be overridden by a child class. And a final method cannot be overridden. Therefore a method can be either abstract or final in Java.

1. **Can we instantiate an abstract class in Java?**

No. We cannot create an instance of an abstract class in Java.

**61. What is an interface in Java?**

An Interface in Java is an abstract type blueprint of a class. It contains the methods that a class must implement. It is like a protocol.

It has method signatures and constant declarations.

1. **Is it allowed to mark an interface method as static?**

Yes, from Java 8 onwards, we can define static and default methods in an interface. Prior to Java 8, it was not allowed.

1. **Why an Interface cannot be marked as final in Java?**

A final method cannot be overridden. But an interface method has to be implemented by another class. So the interface method cannot be marked as final.

**64. What is a marker interface?**

There are interfaces that do not have any data member or methods.

These interfaces are called Marker interface.

E.g. Serializable, Cloneable, Remote etc.

1. **What can we use instead of Marker interface?**

We can use annotations instead of Marker interface.

1. **How Annotations are better than Marker Interfaces?**

Annotations serve the purpose of conveying metadata about the class to its consumers without creating a separate type for it.

Annotations are more powerful than a Marker interface. They allow programmers to pass more sophisticated information to classes that "consume" it.

1. **What is the difference between abstract class and interface in Java?**

Differences between Abstract class and Interface are as follows:

1. An abstract class can have implemented methods with body (non-abstract methods). Interface has only abstract methods. From Java 8 onwards, interface can have static/default methods in implemented form.
2. An abstract class can have instance member variables. An interface cannot have instance variables. It can only have constants.
3. An abstract class can have a constructor. Interface cannot have constructor. It has to be implemented by another class.
4. A class can extend only one abstract class. A class can implement more than one interface.

1. **Does Java allow us to use private and protected modifiers for variables in interfaces?**

No. All the variables in an interface are implicitly public.

1. **How can we cast to an object reference to an interface reference?**

An Object that implements an Interface can be cast to the same Interface. Since An Object implementing an Interface already provides implementation for the methods of that Interface, it is allowed to do so as per the rules of Inheritance.

**Final**

1. **How can you change the value of a final variable in Java?**

Java does not allow changing the value of a final variable. Once the value is set, it cannot be changed.

1. **Can a class be marked final in Java?**

Yes a class can be marked final in Java. Once a class is marked final, it cannot be extended.

1. **How can we create a final method in Java?**

To mark a method, add modifier final to that method. A final method can not be overridden by a child class.

1. **How can we prohibit inheritance in Java?**

If you mark a class final, it cannot be extended. This will prohibit the inheritance of that class in Java.

**74. Why Integer class in final in Java?**

Integer class is a wrapper for int. If it is not marked final, then any other class can extend it and modify the behavior of Integer operations. To avoid this Integer wrapper class is marked as final.

1. **What is a blank final variable in Java?**

When we declare a final variable without giving any initial value, then it is called blank final variable.

1. **How can we initialize a blank final variable?**

A blank final instance variable can be initialized in a constructor.

A blank final static variable can be initialized in the static block of class.

1. **Is it allowed to declare main method as final?**

Yes, we can mark the main method as final.

**Package**

1. **What is the purpose of package in Java?**

A package is used to encapsulate a group of classes, interfaces and sub-packages. Often, it is a hierarchical structure of storing information. It is easier to organize the related classes and sub-packages in this manner.

A Package also provides access protection for classes and interfaces. A package also helps in removing naming collision.

**79. What is java.lang package?**

In Java, java.lang package contains the classes that are fundamental to the design of Java programming language. The most important class in this package is Object class.

It also contains wrapper classes like- Integer, Boolean, Character etc. It provides Math class for mathematical operations.

1. **Which is the most important class in Java?**

It is an open-ended question with many answers. In my view, Object class is the most important class of Java programming language. It is the root of all the classes in Java. It provides some very important and fundamental methods.

1. **Is it mandatory to import java.lang package every time?**

No. By default, JVM loads it internally.

1. **Can you import same package or class twice in your class?**

If we import same package multiple times in a class, compiler includes it only once. So neither JVM nor Compiler gives any error/warning on including a package multiple times.

If you have two classes with same name, then you may get name collision on importing the class erroneously.

JVM internally loads the class only one time.

**83. What is a static import in Java?**

Static import is similar to normal import declaration. Normal import allows us to import classes from packages without using package qualifier. Static import allows us to import static members from a class without using class qualifier.

1. **What is the difference between import static com.test.Fooclass and import com.test.Fooclass?**

First import is a static import and the second import is normal import of a class. First import allows us to import static members of class.

**Internationalization**

**85. What is Locale in Java?**

A Locale object represents a specific geographical, political, or cultural region. It is used to locale-sensitive operations in Java.

It helps is following the local conventions of a country, native or region. These conventions can be for formatting the dates, money, numbers etc.

1. **How will you use a specific Locale in Java?**

To use a specific Locale, we need to load that Locale. We can use ResourceBundle.getBundle("Locale.UK") method to load a Locale.

**Serialization**

**87. What is the serialization?**

Serialization is a process converting an object into a byte array. This byte array represents the class, version and internal state of the object. JVM can use this byte array to transmit/read the object over a network.

1. **What is the purpose of serialization?**

Some of the uses of serialization are:

1. Communication: It is used for transmitting an object over network between two machines.
2. Persistence: We can store the object’s state in a database and retrieve it from database later on.
3. Caching: Serialization can be used for caching to improve performance. We may need 10 minutes to build an object, but it may take just 10 seconds to de-serialize the object.
4. Cross JVM Synchronization: It can be used in same way across multiple JVM that follow different architecture.

**89. What is Deserialization?**

Deserialization is the process of reconstructing the object from the serialized state. It is the reverse process of serialization.

1. **What is Serialization and Deserialization conceptually?**

Serialization is to convert Object data into a stream of bytes

Deserialization is to convert a stream of bytes back into a copy of the original object.

1. **Why do we mark a data member transient?**

Member variables of an object are marked transient to indicate that they should not be serialized.

During serialization process the transient variables are not considered part of the persistent state of an object.

1. **Is it allowed to mark a method as transient?**

No, Java does not allow marking a method as transient. The transient keyword is valid only for member variables.

1. **How does marking a field as transient makes it possible to serialize an object?**

Let say we have a class ABC that implements Serializable interface, but it contains a member variable object of class XYZ that does not implement Serializable interface. Due to this it is not possible to Serialize the class ABC.

To solve this issue, we can mark the member variable XYZ as Transient in class ABC. This will allow us to serialize the class ABC.

1. **What is Externalizable interface in Java?**

Externalizable interface extends Serializable interface in Java. It is used for giving the Class control over saving and restoring the contents of its instances.

A class implements methods writeExternal() and readExternal() to store and restore the object.

1. **What is the difference between Serializable and Externalizable interface?**

Serializable is a marker interface but Externalizable is not a marker interface.

When we implement Serializable interface, the class is serialized automatically by default. We can override writeObject() and readObject()methods to control more complex object Serialization process.

In case of Externalizable, we use readExternal() and writeExternal() methods to give control to class for class's serialization process.

Serializable interface is based on recursive algorithm.

Serializable gives you two options. One option is to provide custom way of serialization, the other default way. In Externalizable, you have to always implement readExternal() and writeExternal() methods.

A public no-arg constructor is needed while using Externalizable interface.

In Serialization, we need to define serialVersionUID. If it is not explicitly defined it will be generated automatically based on all the fields, methods of the class.

**Reflection**

**96. What is Reflection in Java?**

Reflection is Java language's ability to inspect and dynamically call classes, methods, attributes etc. at Runtime. It helps in examining or modifying the Runtime behavior of a class at Runtime.

1. **What are the uses of Reflection in Java?**

Reflection is often used in Testing, Debugging and in Integrated Development Environment (IDE).

Reflection allows you to write programs that do not have to "know" everything at compile time. It makes programs more dynamic, since they can be tied together at runtime.

Many modern frameworks like Spring etc. use Reflection. Some modern languages like Python etc. also use Reflection.

JAVA API for XML Parsing (JAXP) also uses Reflection.

1. **How can we access private**

**method of a class from outside the class?**

We can use Reflection to access private method of a class from outside the class. IN Java, we use getDeclaredMethod() to get instance of a private method. Then we mark this method accessible and finally invoke it.

In following sample code, we are accessing private method message() of class Foo by Reflection.

FileName: Foo.java

public class Foo {

private void message(){System.out.println("hello java"); }

}

FileName: FooMethodCall.java

import java.lang.reflect.Method;

public class FooMethodCall{

public static void main(String[] args)throws Exception{

Class c = Class.forName("Foo");

Object o= c.newInstance();

Method m =c.getDeclaredMethod("message", null);

m.setAccessible(true);

m.invoke(o, null);

}

}

1. **How can we create an Object dynamically at Runtime in Java?**

We can use Reflection to create an Object dynamically at Runtime

in Java. We can use Class.newInstance() or Constructor.newInstance() methods for creating such Objects.

**Garbage Collection**

1. **What is Garbage Collection in Java?**

Java has an internal mechanism called Garbage collection to reclaim the memory of unused projects at run time.

Garbage collection is also known as automatic memory management.

1. **Why Java provides Garbage Collector?**

In Java, there are no pointers. Memory management and allocation is done by JVM. Since memory allocation is automated, after some time JVM may go low on memory. At that time, JVM has to free memory from unused objects. To help with the process of reclaiming memory, Java provides an automated process called Garbage Collector.

1. **What is the purpose of gc() in Java?**

Java provides two methods System.gc() and Runtime.gc() to request the JVM to run the garbage collection. By using these methods, programmers can explicitly send request for Garbage Collection. But JVM process can reject this request and wait for some time before running the GC.

1. **How does Garbage Collection work in Java?**

Java has an automated process called Garbage Collector for Memory Management. It is a daemon in JVM that monitors the memory usage and performs memory cleanup. Once JVM is low on memory, GC process finds the unused objects that are not referenced by other objects. These unused objects are cleaned up by Garbage Collector daemon in JVM.

1. **When does an object become eligible for Garbage Collection in Java?**

An object can be Garbage Collected by JVM, if it is not reachable. There are two cases for deciding eligibility of objects for Garbage Collection:

1. An Object/instance that cannot be reached by a live thread.
2. A set of circularly referenced instances that cannot be reached by any other instance outside that set.

1. **Why do we use finalize() method in Java?**

Java provides finalize() method to perform any cleanup before Garbage Collection. This method is in Object class, and it is invoked by JVM internally. Developers are free to implement this method for any custom cleanup in case of Garbage Collection.

If an Object is not Garbage Collected, then this method may not be called.

This method is never invoked more than once by JVM.

1. **What are the different types of References in Java?**

In Java, there are four types of references:

1. Strong Reference
2. Soft Reference
3. Weak Reference
4. Phantom Reference

1. **How can we reference an unreferenced object again?**

We can provide implementation in finalize() method to reference and unreferenced object. For an unreferenced object, finalize() method is called at the time of Garbage Collection. At this time, Object can pass its reference ‘this’ to finalize() method and revive itself.

1. **What kind of process is the Garbage collector thread?**

Garbage Collection is a Daemon process in JVM. It is an internal process that keep checking Memory usage and cleans up the memory.

1. **What is the purpose of the Runtime class?**

The purpose of the Runtime class is to provide access to the Java Runtime system. This class provides certain important methods like:

1. Runtime.freeMemory() – This method returns the value of free memory in JVM
2. Runtime.maxMemory() - This method returns the value of maximum memory that JVM can use.
3. Runtime.gc() – This method can invoke garbage collection.

1. **How can we invoke an external process in Java?**

Java provides the method Runtime.getRuntime().exec() to invoke an external process from JVM.

1. **What are the uses of Runtime class?**

Runtime class in Java provides following benefits:

1. It allows to read data via key board
2. It can use system properties and environment variables
3. It helps in running non-java programs from within a java application.

**Inner Classes**

**112. What is a Nested class?**

In Java, a Nested class is a class declared inside another class. We can have more than one class declared inside a file.

1. **How many types of Nested classes are in Java?**

Java provides four types of Nested classes:

1. Member inner class
2. Local inner class
3. Anonymous inner class
4. Static nested class

**114. Why do we use Nested Classes?**

There are following reasons for using nested classes:

1. Logical Grouping: We can logically group classes in one place. If one class is useful to only one other class, then we put smaller class within the larger class and keep them in one file. This kind of nesting "helper classes" in a top-level class makes the package more streamlined.
2. Encapsulation: Nested classes increase encapsulation. Let say there are two top-level classes, Foo and Bar. Bar needs access to private members of Foo. We can hide class Bar within class Foo. In this way, private members of Foo can be accessed by class Bar. So class Foo remains encapsulated. Also, class Bar remains hidden from the outside world.
3. Code Clarity: Nested classed make the code more readable and well organized. Only Top-level classes are exposed. The helper classes are kept hidden and closer the code where it is used by a Top-level class.

1. **What is the difference between a Nested class and an Inner class in Java?**

An Inner class in Java is non-static class. It is a type of Nested class that is defined in another class but not qualified with a Static modifier. A Nested class is also a class can be Static Nested class or a non-Static Inner class.

An Inner class has access to other members of the enclosing class, even if they are declared private. A Static Nested class can not access the other members of the enclosing class.

**116. What is a Nested interface?**

A Nested interface is declared inside another interface or a top-level class. By default it is static.

A Nested interface is also known as Static interface.

1. **How can we access the non-final local variable, inside a Local Inner class?**

Java allows a Local Inner class to access only Constant local members. So we have to make the non-final local variable as final constant to access it inside a Local Inner class.

1. **Can an Interface be defined in a Class?**

Yes, we can define a Static Nested interface within a class. Only the enclosing class can access it.

1. **Do we have to explicitly mark a Nested Interface public static?**

A Nested Interface is implicitly public static. So the modifiers public and static are redundant in declaration.

1. **Why do we use Static Nested interface in Java?**

Only the enclosing class can access a Static Nested interface. Consider following code in which interface Xyz is enclosed in class Abc.

public class Abc {

public interface Xyz {

void callback();

}

public static void registerCallback(Xyz xyz) {...}

}

* Client Code Abc.registerCallback(new Abc.Xyz() {

public void callback() {...} });

Any code that cannot access Abc can not access interface Xyz also.

So the purpose of declaring an Inner interface is to restrict its access from outside world.

**String**

1. **What is the meaning of Immutable in the context of String class in Java?**

An Immutable object cannot be modified or changed in Java. String is an Immutable class in Java.

Once a String object is created, it cannot be changed. When we assign the String to a new value, a new object is created.

1. **Why a String object is considered immutable in java?**

Java language uses String for a variety of purposes. For this it has marked String Immutable.

There is a concept of String literal in Java.

Let say there are 2 String variables A and B that reference to a String object “TestData”. All these variables refer to same String literal. If one reference variable A changes the value of the String literal from “TestData” to “RealData”, then it will affect the other variable as well. Due to which String is considered Immutable. In this case, if one variable A changes the value to “RealData”, then a new String literal with “RealData” is created and A will point to new String literal. While B will keep pointing to “TestData”

1. **How many objects does following code create?**

Code:

String s1="HelloWorld";

String s2=" HelloWorld ";

String s3=" HelloWorld ";

The above code creates only one object. Since there is only one String Literal “HelloWorld” created, all the references point to same object.

1. **How many ways are there in Java to create a String object?**

Java provides two ways to create a String object. One is by using String Literal, the other is by using new operator.

1. **How many objects does following code create?**

Code:

String s = new String("HelloWorld");

The above code creates two objects. One object is created in String constant pool and the other is created on the heap in non-pool area.

**126. What is String interning?**

String interning refers to the concept of using only one copy of a distinct String value that is Immutable.

It provides the advantage of making String processing efficient in Time as well as Space complexity. But it introduces extra time in creation of String.

1. **Why Java uses String literal concept?**

Java uses String literal concept to make Java more efficient in memory. If same String already exists in String constant pool, it can be reused. This saves memory usage.

1. **What is the basic difference between a String and StringBuffer object?**

String is an immutable object. Its value cannot change after creation. StringBuffer is a mutable object. We can keep appending or modifying the contents of a StringBuffer in Java.

1. **How will you create an immutable class in Java?**

In Java, we can declare a class final to make it immutable. There are following detailed steps to make it Immutable:

1. Add final modifier to class to prevent it from getting extended
2. Add private modifier to all the fields to prevent direct access
3. Do not provide any setter methods for member variables
4. Add final modifier to all the mutable fields to assign value only once
5. Use Deep Copy to initialize all the fields by a constructor
6. In clone method, return a copy of object instead of the actual object reference

1. **What is the use of toString() method in java ?**

In Java, Object class has toString() method. This method can be used to return the String representation of an Object. When we print an object, Java implicitly calls toString() method.

Java provides a default implementation for toString() method. But we can override this method to return the format that we want to print.

1. **Arrange the three classes String, StringBuffer and StringBuilder in the order of efficiency for String processing operations?**

StringBuilder is the most efficient class. It does not have the overhead of Synchronization. StringBuffer is a Synchronized class. It has better performance than String but it is slower than StringBuilder. String is the slowest for any String processing operations, since it is leads to creation of new String literal with each modification.

So the decreasing order of efficiency is: StringBuilder, StringBuffer, String

**Exception Handling**

1. **What is Exception Handling in Java?**

Java provides Exception Handling mechanism to handle Runtime errors that occur in JVM. There are checked exceptions in a program that we expect to occur in certain situations.

Exception handling mechanism catches these checked exceptions and takes relevant actions.

1. **In Java, what are the differences between a Checked and Unchecked?**

Checked Exceptions extend Throwable class, but they do not extend

RuntimeException or Error classes. UncheckedException extend

RuntimeException class.

Checked Exceptions are checked at compile time in Java. Unchecked Exceptions happen at Runtime, so they are not checked at compile time.

IOException, SQLException etc. are examples of Checked Exceptions. NullPointerException, ArithmeticException etc. are examples of Unchecked Exceptions.

1. **What is the base class for Error and Exception classes in Java?**

Error as well as Exception class is derived from Throwable class in Java.

**135. What is a finally block in Java?**

Java provides a finally block with a try block. This is an optional block. But finally block is always executed after the execution of try block.

1. **What is the use of finally block in Java?**

As per Java specification, a finally block is always executed, whether an error occurs or not, whether an exception is handled or not. It helps in doing the cleanup like- Rollback Transaction, Close Connection, Close a file etc.

1. **Can we create a finally block without creating a catch block?**

Yes. A finally block can follow a try block or catch block. So we can defined a finally block just after a try block.

1. **Do we have to always put a catch block after a try block?**

Java does not enforce the rule to put a catch block after try block.

We can write catch block or finally block after a try block.

Any exception that we want to catch is mentioned in catch block.

1. **In what scenarios, a finally block will not be executed?**

There are two main scenarios in which finally block is not executed:

1. Program exits by calling system.exit() call.
2. A fatal error causes JVM to crash.

1. **Can we re-throw an Exception in Java?**

Yes, Java allows to re-throw an Exception.

1. **What is the difference between throw and throws in Java?**

Java provides throw keyword to throw an exception from a method or a static block. Java provides throws keyword to mention the probable exception thrown by a method in its declaration.

We use throw to explicitly throw an exception. We used throws to declare an exception in method definition.

We cannot propagate checked exceptions with throw only. But checked exceptions can be propagated with throws keyword.

A throw call is followed by an instance. Class or Exception follows a throws keyword.

Call to throw occurs within a method. throws is just used with method signature.

We can throw only one exception at a time. But we can mention as many exceptions in throws clause.

1. **What is the concept of Exception Propagation?**

In Exception Propagation, uncaught exceptions are propagated in the call stack until stack becomes empty. This propagation is called Exception Propagation.

Let say an exception propagates from one method to another method. A() calls B(), which calls C(), which calls D(). And if D() throws an exception, the exception will propagate from D to C to B to A, unless one of the methods catches the exception.

1. **When we override a method in a Child class, can we throw an additional Exception that is not thrown by the Parent class method?**

Yes, Java allows us to throw additional Exception in a child class, but the additional exception should be an unchecked exception (RuntimeException).

**Java Collection**

1. **What is the difference between Collection and Collections Framework in Java?**

In Java, a Collection is an object that contains multiple elements of same type in a single unit. These multiple elements can be accessed through one Collection object.

In Java Collections Framework is a library that provides common architecture for creating, updating and accessing different types of collections. In Collections framework there are common methods that are frequently used by developers for working on a Collection object.

1. **What are the main benefits of Collections Framework in Java?**

Main benefits of Collections Framework in Java are as follows:

1. Reusability: Java Collections Framework provides common classes and utility methods than can be used with different types of collections. This promotes the reusability of the code. A developer does not have to re-invent the wheel by writing the same method again.
2. Quality: Using Java Collection Framework improves the program quality, since the code is already tested and used by thousands of developers.
3. Speed: Most of programmers report that their development speed increased since they can focus on core logic and use the generic collections provided by Java framework.
4. Maintenance: Since most of the Java Collections framework code is open source and API documents is widely available, it is easy to maintain the code written with the help of Java Collections framework. One developer can easily pick the code of previous developer.

1. **What is the root interface of Collection hierarchy in Java?**

The root interface of Collection hierarchy in Java is Collection interface.

But the Collection interface extends Iterable interface. Due to this some people consider Iterable interface as the root interface.

Iterable interface is present in java.lang package but Collection interface is present in java.util package. Oracle Java API docs mention that Collection interface is a member of the Java Collections framework.

Whereas, Iterable interface is not stated as a part of Java Collections framework in Java docs.

Due to this Collection interface is the root of Collections Framework.

1. **What are the main differences between Collection and Collections?**

Main differences between Collection and Collections are as follows:

1. Collection is an interface in Java. But Collections is a class in Java.
2. Collection is a base interface. Collections is a utility class in Java.
3. Collection defines methods that are used for data structures that contain the objects. Collections defines the methods that are used for operations like access, find etc. on a Collection.

1. **What are the Thread-safe classes in Java Collections framework?**

The Thread-safe classes in Java Collections framework are:

Stack



Properties



Vector



Hashtable



BlockingQueue



ConcurrentMap



ConcurrentNavigableMap



1. **How will you efficiently remove elements while iterating a Collection?**

The right way to remove elements from a collection while iterating is by using ListIterator.remove() method.

E.g.

ListIterator<Integer> iter = myList.iterator(); while(iter.hasNext()) { itr.remove();

}

Some developers use following code to remove an element which is incorrect:

Iterator<Integer> iter = myList.iterator(); while(iter.hasNext()) { itr.remove();

}

By doing so we get ConcurrentModificationException.

An iterator is first created to traverse the list. But at the same time the list is changed by remove() method.

In Java, it is not allowed for a thread to modify a collection while another thread is iterating it. ListIterator provides the capability of removing an object during traversal.

1. **How will you convert a List into an array of integers like- int[]?**

We can use ArrayUtils class in Apache Commons Lang library.

Sample code is:

int[]intArray = ArrayUtils.toPrimitive(myList.toArray(new Integer[0]));

If we use List.toArray(), it will convert List to Integer[].

Another option is:

int[] intArray = new int[myList.size()];

for (int i=0; i < myList.size(); i++) {

intArray [i] = myList.get(i);

}

1. **How will you convert an array of primitive integers int[] to a List collection?**

We can use ArrayUtils in Apache Commons Lang library for this purpose.

Sample code is:

List intList = Arrays.asList(ArrayUtils.toObject(intArray));

The other option would be to use a for loop and explicitly adding integers to a List.

Sample code is:

int[]intArray = {10,20,30};

List<Integer> intList = new ArrayList<Integer>(); for (int i: intArray) {

intList.add(i);

}

1. **How will you run a filter on a Collection?**

We can use CollectionUtils of Apache for this purpose. We will have to create a Predicate that will define the condition for our filter. Then we can apply this Predicate in filter() method.

Sample code is:

In this example we filter any names that are less than 5 characters long.

List<String> namesList = asList( "Red", "Blue", "Green" );

List<String> shortNamesList = new ArrayList<String>(); shortNamesList.addAll( namesList );

CollectionUtils.filter( shortNamesList, new Predicate(){ public boolean evaluate( Object input ) {

return ((String) input).length() < 5;

}

} );

We can also use Google Guava library for this.

In Java 8, we can use Predicate to filter a Collection through Stream.

1. **How will you convert a List to a**

**Set?**

There are two ways to convert a List to a Set in Java.

Option 1: Use HashSet

Set<Integer> mySet = new HashSet<Integer>(myList);

In this case we put a list into a HashSet. Internally hashCode() method is used to identify duplicate elements.

Option 2: Use TreeSet

In this case we use our own comparator to find duplicate objects.

Set<Integer> mySet = new TreeSet<Integer>(myComparator); mySet.addAll(myList);

1. **How will you remove duplicate elements from an ArrayList?**

The trick in this question is to use a collection that does not allow duplicate elements. So we use a Set for this purpose.

Option 1: Use Set

If ordering of elements is not important then we just put the elements of ArrayList in a HashSet and then add them back to the ArrayList.

Sample Code is:

ArrayList myList = // ArrayList with duplicate elements

Set<Integer> mySet = new HashSet<Integer>(myList);

myList.clear();

myList.addAll(mySet);

Option 2: Use LinkedHashSet

If ordering of elements is important then we put the elements of ArrayList in a LinkedHashSet and then add them back to the ArrayList.

Sample Code is:

ArrayList myList = // ArrayList with duplicate elements

Set<Integer> mySet = new LinkedHashSet<Integer>(myList);

myList.clear();

myList.addAll(mySet);

1. **How can you maintain a Collection with elements in Sorted order?**

In Java, there are many ways to maintain a Collection with elements in sorted order.

Some collections like TreeSet store elements in the natural ordering. In case of natural ordering we have to implement Comparable interface for comparing the elements.

We can also maintain custom ordering by providing a custom Comparator to a Collection.

Another option is to use the utility method Collections.sort() to sort a List. This sorting gives nlog(n) order of performance. But if we have to use this method multiple times then it will be costly on performance.

Another option is to use a PriorityQueue that provides an ordered queue. The main difference between PriorityQueue and Collections.sort() is that PriorityQueue maintains a queue in Order all the time, but we can only retrieve head element from queue. We cannot access the elements of PriorityQueue in Random order.

We can use TreeSet to maintain sorted order of elements in collection if there are no duplicate elements in collection.

1. **What are the differences between the two data structures: a Vector and an ArrayList?**

An ArrayList is a newer class than a Vector. A Vector is considered a legacy class in Java. The differences are:

1. Synchronization: Vector is synchronized, but the ArrayList is not synchronized. So an ArrayList has faster operations than a Vector.
2. Data Growth: Internally both an ArrayList and Vector use an array to store data. When an ArrayList is almost full it increases its size by 50% of the array size. Whereas a Vector increases it by doubling the underlying array size.

1. **What are the differences between Collection and Collections in Java?**

Main differences between Collection and Collections are:

1. Type: Collection is an interface in Java. Collections is a class.
2. Features: Collection interface provides basic features of data structure to List, Set and Queue interfaces. Collections is a utility class to sort and synchronize collection elements. It has polymorphic algorithms to operate on collections.
3. Method Type: Most of the methods in Collection are at instance level. Collections class has mainly static methods that can work on an instance of Collection.

1. **In which scenario, LinkedList is better than ArrayList in Java?**

ArrayList is more popular than LinkedList in Java due to its ease of use and random access to elements feature.

But LinkedList is better in the scenario when we do not need random access to elements or there are a lot of insertion, deletion of elements.

1. **What are the differences between a List and Set collection in Java?**

Main differences between a List and a Set are:

1. Order: List collection is an ordered sequence of elements. A Set is just a distinct collection of elements that is unordered.
2. Positional Access: When we use a List, we can specify where exactly we want to insert an element. In a Set there is no order, so we can insert element anywhere without worrying about order.
3. Duplicate: In a List we can store duplicate elements. A Set can hold only unique elements.

1. **What are the differences between a HashSet and TreeSet collection in Java?**

Main differences between a HashSet and TreeSet are:

1. Ordering: In a HashSet elements are stored in a random order. In a TreeSet, elements are stored according to natural ordering.
2. Null Value Element: We can store null value object in a HashSet. A TreeSet does not allow to add a null value object.
3. Performance: HashSet performs basic operations like add(), remove(), contains(), size() etc in a constant size time. A TreeSet performs these operations at the order of log(n) time.
4. Speed: A HashSet is better than a TreeSet in performance for most of operations like add(), remove(), contains(), size() etc .
5. Internal Structure: a HashMap in Java internally backs a HashSet. A NavigableMap backs a TreeSet internally.
6. Features: A TreeSet has more features compared to a HashSet. It has methods like pollFirst(), pollLast(), first(), last(), ceiling(), lower() etc.
7. Element Comparison: A HashSet uses equals() method for comparison. A TreeSet uses compareTo() method for

comparison to maintain ordering of elements.

1. **In Java, how will you decide when to use a List, Set or a Map collection?**
2. If we want a Collection that does not store duplicate values, then we use a Set based collection.
3. If we want to frequently access elements operations based on an index value then we use a List based collection. E.g. ArrayList
4. If we want to maintain the insertion order of elements in a collection then we use a List based collection.
5. For fast search operation based on a key, value pair, we use a HashMap based collection.
6. If we want to maintain the elements in a sorted order, then we use a TreeSet based collection.

1. **What are the differences between a HashMap and a Hashtable in Java?**

Main differences between a HashMap and a Hashtable are:

1. Synchronization: HashMap is not a synchronized collection. If it is used in multi-thread environment, it may not provide thread safety. A Hashtable is a synchronized collection. Not more than one thread can access a Hashtable at a given moment of time. The thread that works on Hashtable acquires a lock on it and it makes other threads wait till its work is completed.
2. Null values: A HashMap allows only one null key and any number of null values. A Hashtable does not allow null keys and null values.
3. Ordering: A HashMap implementation by LinkedHashMap maintains the insertion order of elements. A TreeMap sorts the mappings based on the ascending order of keys. On the other hand, a Hashtable does not provide guarantee of any kind of order of elements. It does not maintain the mappings of key values in any specific order.
4. Legacy: Hashtable was not the initial part of collection framework in Java. It has been made a collection framework member, after being retrofitted to implement the Map interface. A HashMap implements Map interface and is a part of collection framework since the beginning.
5. Iterator: The Iterator of HashMap is a fail-fast and it throws ConcurrentModificationException if any other Thread modifies the map by inserting or removing any element except iterator’s own remove() method.

Enumerator of the Hashtable is not fail-fast.

1. **What are the differences between a HashMap and a TreeMap?**

Main differences between a HashMap and a TreeMap in Java are:

1. Order: A HashMap does not maintain any order of its keys. In a HashMap there is no guarantee that the element inserted first will be retrieved first.
2. In a TreeMap elements are stored according to natural ordering of elements. A TreeMap uses compareTo() method to store elements in a natural order.
3. Internal Implementation: A HashMap uses Hashing internally. A TreeMap internally uses Red-Black tree implementation.
4. Parent Interfaces: A HashMap implements Map interface. TreeMap implements NavigableMap interface.
5. Null values: A HashMap can store one null key and multiple null values. A TreeMap can not contain null key but it may contain multiple null values.
6. Performance: A HashMap gives constant time performance for operations like get() and put(). A TreeMap gives order of log(n) time performance for get() and put() methods.
7. Comparison: A HashMap uses equals() method to compare keys. A TreeMap uses compareTo() method for maintaining natural ordering.
8. Features: A TreeMap has more features than a HashMap. It has methods like pollFirstEntry() , pollLastEntry() , tailMap() , firstKey() , lastKey() etc. that are not provided by a HashMap.

1. **What are the differences between Comparable and Comparator?**

Main differences between Comparable and Comparator are:

1. Type: Comparable<T> is an interface in Java where T is the type of objects that this object may be compared to.
2. Comparator<T> is also an interface where T is the type of objects that may be compared by this comparator.
3. Sorting: In Comparable, we can only create one sort sequence. In Comparator we can create multiple sort sequences.
4. Method Used: Comparator<T> interface in Java has method public int compare (Object o1, Object o2) that returns a negative integer, zero, or a positive integer when the object o1 is less than, equal to, or greater than the object o2. A Comparable<T> interface has method public int compareTo(Object o) that returns a negative integer, zero, or a positive integer when this object is less than, equal to, or greater than the object o.
5. Objects for Comparison: The Comparator compares two objects given to it as input. Comparable interface compares "this" reference with the object given as input.
6. Package location: Comparable interface in Java is defined in java.lang package. Comparator interface in Java is defined in java.util package.

1. **In Java, what is the purpose of Properties file?**

A Properties file in Java is a list of key-value pairs that can be parsed by java.util.Properties class.

Generally a Properties file has extension .properties e.g.

myapp.properties.

Properties files are used for many purposes in all kinds of Java applications. Some of the uses are to store configuration, initial data, application options etc.

When we change the value of a key in a properties file, there is no need to recompile the Java application. So it provides benefit of changing values at runtime.

1. **What is the reason for overriding equals() method?**

The equals() method in Object class is used to check whether two objects are same or not. If we want a custom implementation we can override this method.

For example, a Person class has first name, last name and age. If we want two Person objects to be equal based on name and age, then we can override equals() method to compare the first name, last name and age of Person objects.

Generally in HashMap implementation, if we want to use an object as key, then we override equals() method.

1. **How does hashCode() method work in Java?**

Object class in Java has hashCode() method. This method returns a hash code value, which is an integer.

The hashCode() is a native method and its implementation is not pure Java.

Java doesn't generate hashCode(). However, Object generates a HashCode based on the memory address of the instance of the object.

If two objects are same then their hashCode() is also same.

1. **Is it a good idea to use Generics in collections?**

Yes. A collection is a group of elements put together in an order or based on a property. Often the type of element can vary. But the properties and behavior of a Collection remains same. Therefore it is good to create a Collection with Generics so that it is type-safe and it can be used with wide variety of elements.

1. **What is the difference between Collections.emptyList() and creating new instance of Collection?**

In both the approaches, we get an empty list. But Collections.emptyList() returns an Immutable list. We cannot add new elements to an Immutable empty list.

Collections.emptyList() works like Singleton pattern. It does not create a new instance of List. It reuses an existing empty list instance.

Therefore, Collections.emptylist() gives better performance if we need to get an emptyList multiple times.

1. **How will you copy elements from a Source List to another list?**

There are two options to copy a Source List to another list.

Option 1: Use ArrayList constructor

ArrayList<Integer> newList = new ArrayList<Integer>(sourceList);

Option 2: Use Collection.copy()

To use Collections.copy() destination list should be of same or larger size than source list.

ArrayList<Integer> newList = new ArrayList<Integer>

(sourceList.size());

Collections.copy(newList, sourceList);

Collections.copy() does not reallocate the capacity of destination List if it does not have enough space to contain all elements of source List. It throws IndexOutOfBoundsException.

The benefit of Collection.copy() is that it guarantees that the copy will happen in linear time. It is also good for the scenario when we want to reuse an array instead of allocating more memory in the constructor of ArrayList.

One limitation of Collections.copy() is that it can accept only List as source and destination parameters.

1. **What are the Java Collection classes that implement List interface?**

Java classes that implement List interface are:

AbstractList



AbstractSequentialList



ArrayList



AttributeList



CopyOnWriteArrayList



LinkedList



RoleList



RoleUnresolvedList



Stack



Vector



1. **What are the Java Collection classes that implement Set interface?**

Java classes that implement Set interface are:

AbstractSet



ConcurrentSkipListSet



CopyOnWriteArraySet



EnumSet



HashSet



JobStateReasons



LinkedHashSet



TreeSet



1. **What is the difference between an Iterator and ListIterator in Java?**

Iterator and ListIterator are two interfaces in Java to traverse data structures. The differences between these two are:

1. ListIterator can be used to traverse only a List. But Iterator can be used to traverse List, Set, and Queue etc.
2. An Iterator traverses the elements in one direction only. It just goes. ListIterator can traverse the elements in two directions i.e. backward as well as forward directions.
3. Iterator cannot provide us index of an element in the Data Structure. ListIterator provides us methods like nextIndex() and previousIndex() to get the index of an element during traversal.
4. Iterator does not allow us to add an element to collection

while traversing it. It throws ConcurrentModificationException. ListIterator allows use to add an element at any point of time while traversing a list.

1. An existing element’s value cannot be replaced by using Iterator. ListIterator provides the method set(e) to replace the value of last element returned by next() or previous() methods.

1. **What is the difference between Iterator and Enumeration?**

Both Iterator and Enumeration are interfaces in Java to access Data Structures. The main differences between these are:

1. Enumeration is an older interface. Iterator is a newer interface.
2. Enumeration can only traverse legacy collections. Iterator can traverse both legacy as well as newer collections.
3. Enumeration does not provide remove() method. So we cannot remove any element during traversal. Iterator provides remove() method.
4. Iterator is a fail-fast interface, it gives ConcurrentModificationException if any thread tries to modify an element in the collection being iterated. Enumeration is not fail-fast.
5. Method names in Iterator are shorter than in an Enumeration.

1. **What is the difference between an ArrayList and a LinkedList data structure?**

Main differences between ArrayList and LinkedList data structures are:

1. **Data Structure**: An ArrayList is an indexed baseddynamic array. A LinkedList is a Doubly Linked List data structure.
2. **Insertion**: It is easier to insert new elements in aLinkedList, since there is no need to resize an array. Insertion in ArrayList is O(n), since it may require resizing of array and copying its contents to new array.
3. **Remove elements**: LinkedList has better performance inremoval of elements than ArrayList.
4. **Memory Usage**: LinkedList uses more memory thanArrayList, since it has to maintain links for next and previous nodes as well.
5. **Access**: LinkedList is slower in accessing an element,since we have to traverse the list one by one to access the right location.

1. **What is the difference between a Set and a Map in Java?**

Main differences between a Set and a Map in Java are:

1. **Duplicate Elements**: A Set does not allow insertingduplicate elements. A Map does not allow using duplicate keys, but it allows inserting duplicate values for unique keys.
2. **Null values**: A Set allows inserting maximum one nullvalue. In a Map we can have single null key at most and any number of null values.
3. **Ordering**: A Set does not maintain any order of elements.Some of sub-classes of a Set can sort the elements in an order like LinkedHashSet. A Map does not maintain any order of its elements. Some of its sub-classes like TreeMap store elements of the map in ascending order of keys.

1. **What is the use of a Dictionary class?**

The Dictionary class in Java is used to store key-value pairs. Any non-null object can be used for key or value. But we cannot insert a null key or null object in Dictionary.

Dictionary class is deprecated now. So it should not be used in newer implementations.

1. **What is the default size of load factor in a HashMap collection in Java?**

Default value of load factor in a HashMap is 0.75.

1. **What is the significance of load factor in a HashMap in Java?**

A HashMap in Java has default initial capacity 16 and the load factor is 0.75f (i.e. 75% of current map size). The load factor of a HashMap is the level at which its capacity should be doubled.

For example, in a HashMap of capacity 16 and load factor .75. The capacity will become 32 when the HashMap is 75% full. Therefore, after storing the 12th key– value pair (16 \* .75 = 12) into HashMap, its capacity becomes 32.

1. **What are the major differences between a HashSet and a HashMap?**

The main difference between a HashSet and a HashMap are:

1. **Base class**: A HashSet class implements the Set interface.Whereas a HashMap class implements the Map interface.
2. **Storage**: A HashSet is used to store distinct objects. AHashMap is used for storing key & value pairs, so that these can be retrieved by key later on.
3. **Duplicate Elements**: A HashSet does not allow storingduplicate elements. A HashMap also does not allow duplicate keys. But we can store duplicate values in a HashMap.
4. **Null Elements**: In a HashSet we can store a single nullvalue. In a HashMap we can store single null key, but any number of null values.
5. **Element Type**: A HashSet contains only values of objectsas its elements. Whereas a HashMap contains entries(key value pairs).
6. **Iteration**: By using an Iterator we can iterate a HashSet.But a HashMap has to be converted into Set for iteration.

1. **What are the similarities between a HashSet and a HashMap in Java?**

As the name suggests, HashSet and HashMap are Hashing based collections. Similarities between HashSet and HashMap are:

1. **Thread Safety**: Both HashMap and HashSet are notsynchronized collections. Therefore they are not good for thread-safe operations. To make these thread-safe we need to explicitly use synchronized versions.
2. **Order of Elements**: None of these classes guarantee theorder of elements. These are unordered collections.
3. **Internal Implementation**: A HashMap backs up a HashSetinternally. So HashSet uses a HashMap for performing its operations.
4. **Performance**: Both of these collections provide constanttime performance for basic operations such as insertion and removal of elements.

1. **What is the reason for overriding equals() method?**

The equals() method in Object class is used to check whether two objects are same or not. If we want a custom implementation we can override this method.

For example, a Person class has first name, last name and age. If we want two Person objects to be equal based on name and age, then we can override equals() method to compare the first name, last name and age of Person objects.

Generally in HashMap implementation, if we want to use an object as key, then we override equals() method.

1. **How can we synchronize the elements of a List, a Set or a Map?**

Sometimes we need to make collections Thread-safe for use in Multi-threading environment. In Java, Collections class provides useful static methods to make a List, Set or Map as synchronized collections. Some of these methods are:

static <T> Collection<T> synchronizedCollection(Collection<T> c)

Returns a synchronized (thread-safe) collection backed by the specified collection.

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

Returns a synchronized (thread-safe) list backed by the specified list.

static <K,V> Map<K,V>synchronizedMap(Map<K,V> m)

Returns a synchronized (thread-safe) map backed by the specified map.

static <T> Set<T> synchronizedSet(Set<T> s)

Returns a synchronized (thread-safe) set backed by the specified set.

static <K,V> SortedMap<K,V> synchronizedSortedMap(SortedMap<K,V> m)

Returns a synchronized (thread-safe) sorted map backed by the specified sorted map.

static <T> SortedSet<T> synchronizedSortedSet(SortedSet<T> s) Returns a synchronized (thread-safe) sorted set backed by the specified sorted set.

1. **What is Hash Collision? How Java handles hash-collision in HashMap?**

In a Hashing scenario, at times two different objects may have same HashCode but they may not be equal. Therefore, Java will face issue while storing the two different objects with same HashCode in a HashMap. This kind of situation is Hash Collision.

There are different techniques of resolving or avoiding Hash Collision. But in HashMap, Java simply replaces the Object at old Key with new Object in case of Hash Collision.

1. **What are the Hash Collision resolution techniques?**

To resolve a Hash Collision we can use one of the following techniques:

Separate Chaining with Linked List



Separate Chaining with List Head Cells



Open Addressing with Coalesced Hashing



Open Addressing with Cuckoo Hashing



Hopscotch Hashing



Robinhood Hashing



1. **What is the difference between Queue and Stack data structures?**

Queue is a FIFO data structure. FIFO stands for First In First Out. It means the element added first will be removed first from the queue. A real world example of Queue is a line for buying tickets at a station. The person entering first in the Queue is served first.

Stack is a LIFO data structure. LIFO stands for Last In First Out.

The element that is added last is removed first from the collection.

In a Stack elements are added or removed from the top of stack.

A real world example of Stack is back button in browser. We can go back one by one only and it works in the reverse order of adding

webpages to history .

**187. What is an Iterator in Java?**

Iterator is an interface in Java to access the elements in a collection.

It is in java.util package.

It provides methods to iterate over a Collection class in Java.

Iterator interface in Java is based on Iterator design pattern. By using an Iterator one can traverse a container of objects and can also access the objects in the container. A container of objects is a Collection class in Java.

1. **What is the difference between Iterator and Enumeration in Java?**

Main differences between Iterator and Enumeration in Java are:

1. **Version**: Enumeration interface is in Java since JDK 1.0.Iterator interface was introduced in Java 1.2.
2. **remove() method**: The main difference betweenEnumeration and Iterator interface is remove() method. Enumeration can just traverse a Collection object. If we use Enumeration, we cannot do any modifications to a Collection while traversing the collection. Iterator interface provides remove() method to remove an element while traversing the Collection. There is not remove() method in Enumeration interface.
3. **Method names**: Names of methods in Iterator interface arehasNext(), next(), remove(). Names of methods in

Enumeration interface are hasMoreElements(), nextElement().

1. **Legacy Interface**: Enumeration is considered as a legacyinterface. It is used to traverse legacy classes like Vector, Stack and HashTable. Iterator is a newer interface that is used to traverse almost all of the classes in Java Collections framework.
2. **Fail-fast vs. Fail-safe**: Iterator is based on fail-fastprinciple. It throws ConcurrentModificationException if a collection is modified during iteration over that collection. An Enumeration is based on fail-safe principle. It doesn’t throw any exception if a collection is modified during traversal.
3. **Safety**: Since Iterator is fail-fast and does not allowmodification of a collection by other threads, it is considered safer than Enumeration.

1. **What is the design pattern used in the implementation of Enumeration in Java?**

Enumeration is based on Iterator design pattern. Iterator design pattern provides a common interface with methods to traverse the collection of objects. It hides the underlying implementation details of the collection.

1. **Which methods do we need to override to use an object as key in a HashMap?**

If we want to use an object as a key in a HashMap in Java, then we have to make sure that it has the implementation of equals() and hashCode() methods.

1. **How will you reverse a List in Java?**

In Collections class, Java provides a method reverse(List list) that can be used to reverse a List.

E.g.

Collections.reverse(myList);

1. **How will you convert an array of String objects into a List?**

Java provides Arrays class in java.util package. Arrays class has a method asList() that accepts an Array as input and returns a List as output.

public static <T> List<T> asList(T... a)

String[] myArray = {"George" , "Jack" , "Ryan"}; List myList = Arrays.asList(myArray);

1. **What is the difference between peek(), poll() and remove() methods of Queue interface in java?**

In a Java Queue, poll() and remove() methods can be used for removing the head object of Queue. The main difference arises in the case when Queue is empty().

If Queue is empty then poll() method returns null value. If Queue is empty then remove() method throws NoSuchElementException.

In a Java Queue, peek() method retrieves the head of Queue but it does not remove it. If queue is empty then peek() method returns null value.

1. **What is the difference between Array and ArrayList in Java?**

The main differences between Array and ArrayList in Java are:

1. **Size**: Array in Java is fixed in size. We cannot change thesize of array after creating it. ArrayList is dynamic in size. When we add elements to an ArrayList, its capacity increases automatically.
2. **Performance**: In Java Array and ArrayList give differentperformance for different operations.
3. **add() or get():** Adding an element to or retrieving anelement from an array or ArrayList object has similar performance. These are constant time operations.
4. **resize():** Automatic resize of ArrayList slows down theperformance. ArrayList is internally backed by an Array. In resize() a temporary array is used to copy elements from old array to new array.
5. **Primitives**: Array can contain both primitive data types aswell as objects. But ArrayList cannot contain primitive data types. It contains only objects.
6. **Iterator**: In an ArrayList we use an Iterator object totraverse the elements. We use for loop for iterating elements in an array.
7. **Type Safety**: Java helps in ensuring Type Safety ofelements in an ArrayList by using Generics. An Array can

contain objects of same type of class. If we try to store a different data type object in an Array then it throws ArrayStoreException.

1. **Length**: Size of ArrayList can be obtained by using size()method. Every array object has length variable that is same as the length/size of the array.
2. **Adding elements**: In an ArrayList we can use add()method to add objects. In an Array assignment operator is used for adding elements.
3. Multi-dimension: An Array can be multi-dimensional. An ArrayList is always of single dimension.

1. **How will you insert, delete and retrieve elements from a HashMap collection in Java?**

We use following methods to insert, delete and retrieve elements in

* HashMap.
  1. **Retrieve**: We use get() method to retrieve elements from aHashMap.

Value get(Object key)

* 1. **Insert**: We use put() method to insert a key value pair in aHashMap.

Value put(Key k, Value v)

* 1. **Delete**: We use remove() method to delete key-value pairfrom the HashMap.

Value remove(Object key)

1. **What are the main differences between HashMap and ConcurrentHashMap in Java?**

Main differences between HashMap and ConcurrentHashMap are:

1. **Synchronization**: A HashMap is not synchronized. But aConcurrentHashMap is a synchronized object.
2. **Null Key**: A HashMap can have one null key and anynumber of null values. A ConcurrentHashMap cannot have null keys or null values.
3. **Multi-threading**: A ConcurrentHashMap works well in amulti-threading environment.

1. **What is the increasing order of performance for following collection classes in Java?**

The increasing order of performance is:

Hashtable



Collections.SynchronizedMap



ConcurrentHashMap



HashMap



Hashtable has the worst performance and HashMap has the best performance.

1. **Why does Map interface not extend Collection interface in Java?**

A Map is a collection objects. But Map interface is not compatible with Collection interface in Java.

A Map requires key as well as a value. So it requires two parameters to add an element to a HashMap.

But Collection interface provides add(Object o) method with only one parameter.

Map collection has to provide methods like valueSet, keySet etc. These methods are specific to Map collection. Where as methods in Collection interface can be reused by a List, Set, Queue etc.

1. **What are the different ways to iterate elements of a list in Java?**

There are mainly two ways to iterate the elements of list in Java:

1. **Iterator**: We can get an Iterator for list and use it to iteratethe objects of the list.
2. **For-each loop**: We can use for-each loop to traverse allthe elements of a list.

1. **What is CopyOnWriteArrayList? How it is different from ArrayList in Java?**

CopyOnWriteArrayList was introduced in Java 5 version. It is a thread-safe collection. It is similar to an ArrayList.

In CopyOnWriteArrayList, all mutative operations (add, set etc.) are implemented by making a fresh copy of the underlying array.

Iterator of CopyOnWriteArrayList is guaranteed to not throw ConcurrentModificationException. But Iterator also does not reflect any additions, removals that happened to list after the Iterator was created.

All elements including null are permitted in CopyOnWriteArrayList.

1. **How remove() method is implemented in a HashMap?**

Remove() method in HashMap uses logic similar to the one used in get() method. First we locate the correct bucket in HashMap for an entry. Then within that bucket we remove the element e. It is similar to removing a node from a single-linked list.

If e is the first element in the bucket we set the corresponding element of Hash to e.next. Else we set the next field of the element just before e to e.next.

1. **What is BlockingQueue in Java Collections?**

BlockingQueue was introduced in Java 1.5. It extends Queue interface in Java.

BlockingQueue supports operations that wait for the queue to become non-empty when retrieving an element. Also it supports the operations that wait for space to become available in the queue while storing an element.

Some of the features of BlockingQueue are:

It does not accept null elements.



Its main use is in producer-consumer problems.



BlockingQueue implementation is thread-safe.



It can be used in inter-thread communications.



It does not support any kind of "close" or "shutdown"



operation to indicate that no more items will be added.

1. **How is TreeMap class implemented in Java?**

Internally, a TreeMap class in Java uses Red-Black tree.

It is a NavigableMap. The map sorts the keys in natural order or it can use a Comparator supplied at the creation time.

The implementation of TreeMap is not synchronized in Java.

1. **What is the difference between Fail-fast and Fail-safe iterator in Java?**

Differences between Fail-fast and Fail-safe iterators are as follows:

Fail-fast iterator throws ConcurrentModificationException. But Fail-safe iterator does not throw this exception.

Fail-fast iterator does not clone the original collection. Fail-safe iterator creates a copy of the original collection of objects.

A Fail-fast iterator tries to immediately throw Exception when it encounters failure. A Fail-safe Iterator works on a copy of collection instead of original collection.

1. **How does ConcurrentHashMap work in Java?**

ConcurrentHashMap extends AbstractMap in Java. It was introduced in Java 1.5. It provides concurrency in a collection based on a HashMap.

All methods are thread-safe in ConcurrentHashMap.

Internally there is a Hashtable backing a ConcurrentHashMap. This Hashtable supports the concurrent methods for retrieval of data as well as updates on ConcurrentHashMap.

It has same functional specification as a Hashtable.

It also supports a set of sequential and bulk operations. These operations accept parallelismThreshold argument.

1. **What is the importance of hashCode() and equals() methods?**

In a HashMap collection it is very important for a key object to implement hashCode() method and equals() method. If hashCode() method returns same hashcode for all key objects then the hash collision will be high in HashMap. Also with same hashcode, we will get same equals method that will make our HashMap inefficient.

The problem arises when HashMap treats both outputs same instead of different. It will overwrite the most recent key-value pair with the previous key-value pair.

So it is important to implement hashCode() and equals() methods correctly for an efficient HashMap collection.

1. **What is the contract of hashCode() and equals() methods in Java?**

Contract of hashCode() and equals() methods is as follows in Java:

If object1.equals(object2), then object1.hashCode() == object2.hashCode() should always be true. It means if two objects are equal then their hashCode should be same.

If object1.hashCode() == object2.hashCode() is true, it does not guarantee that object1.equals(object2). It means if two objects have same hashCode, then can still have different values so that may not be equal objects.

1. **What is an EnumSet in Java?**

Set: EnumSet is a specialized implementation of Set.

1. **Use**: It is mainly used with enum types.
2. **Single enum type**: All the elements in an EnumSet mustcome from a single enum type when the set is created.
3. **Bit vector**: Internally, EnumSet is represented as bitvector.
4. **Iterator**: The iterator of EnumSet traverses the elements intheir natural order. (It is the order in which the enum constants are declared).
5. **Null**: In an EnumSet, null elements are not permitted. If wetry to insert a null element it throws NullPointerException.
6. **Thread-safe**: EnumSet is not a synchronized collection.For use in multi-threading scenarios, EnumSet should be synchronized.
7. **Bit flags**: EnumSet is a very good alternative to int based“bit flags” implementation.

1. **What are the main Concurrent Collection classes in Java?**

Java 1.5 has provided new package java.util.concurrent. This package contains thread-safe collection classed. These collection classes can be modified while iterating. The iterator of these classes is fail-safe.

Main Concurrent Collection classes in Java 8 are:

ArrayBlockingQueue



CopyOnWriteArrayList



CopyOnWriteArraySet



ConcurrentHashMap



ConcurrentLinkedDeque



ConcurrentLinkedQueue



LinkedBlockingQueue



LinkedBlockingDeque



PriorityBlockingQueue



1. **How will you convert a Collection to SynchronizedCollection in Java?**

Java provides an easy method in java.utils.Collections class to create a ThreadSafe collection from a regular collection.

We can use the method synchronizedCollection() for this purpose.

For any class of type T we can use following method:

static <T> Collection<T> synchronizedCollection(Collection<T> c)

1. **How IdentityHashMap is different from a regular Map in Java?**

IndentityHashMap in Java implements Map interface. But it is not a general purpose implementation. It violates the general contract of Map interface by a different implementation of equals() method.

In an IdentityHashMap, two keys k1 and k2 are equal if and only if (k1==k2). (In a normal Map implementation (like HashMap) two keys k1 and k2 are considered equal if and only if (k1==null ? k2==null : k1.equals(k2)).)

It implements the Map interface with a hash table, using reference-equality in place of object-equality when comparing keys (and values).

1. **What is the main use of IdentityHashMap?**

Main uses of IdentityHashMap are:

1. Topology Preservation: The typical use of IdentityHashMap class is topology-preserving object graph transformations, such as serialization or deep-copying. In such a scenario, a program must maintain a "node table" to keep track of all the object references that have already been processed.

1. The node table should not considered distinct objects as equal even if they happen to be equal.
2. Proxy objects: Another use of this class is to maintain proxy objects. A debugging program has to maintain a proxy object for each object in the program being debugged.

1. **How can we improve the performance of IdentityHashMap?**

IdentityHashMap class has one tuning parameter for performance improvement: expectedMaxSize.

This parameter is the maximum number of key-value mappings that the map is expected to hold.

We can use this parameter is used to determine the number of buckets initially in the hash table. The precise relationship between the expected maximum size and the number of buckets is unspecified.

If the number of key-value mappings exceeds the expected maximum size, the number of buckets is increased.

Increasing the number of buckets is also known as rehashing. Rehashing may be fairly expensive. So it is better to create identity hash maps with a sufficiently large expected maximum size.

But iteration over a Map collection requires time proportional to the number of buckets in the hash table. So iteration may take extra time due to large number of buckets.

Therefore the value of expectedMaxSize should be set in consideration with both of these aspects.

1. **Is IdentityHashMap thread-**

**safe?**

The implementation of IdentityHashMap is not thread-safe, since its methods are not synchronized.

The iterators returned by the iterator method of IdentityHashMap are fail-fast. But the fail-fast behavior of an iterator cannot be guaranteed.

Since the Iterator is fail-fast, it throws ConcurrentModificationException.

1. **What is a WeakHashMap in Java?**

WeakHashMap is a class similar to IdentityHashMap.

Internally, it is represented by a Hashtable.

It is not a synchronized class. We can make a WeakHashMap thread safe by using Collections.synchronizedMap() method.

An entry in WeakHashMap is automatically removed when it is no longer in ordinary use.

The presence of a mapping for a given key does not prevent the key from being discarded by the garbage collector.

WeakHashMap also permits null keys and null values.

1. **How can you make a Collection class read Only in Java?**

In Java, there are useful methods to make a Collection class read Only. We can make the Collection read Only by using one of the following methods:

Collections.unmodifiableMap(Map m)



Collections.unmodifiableList(List l)



Collections.unmodifiableSet(Set s)



Collections.unmodifiableCollection(Collection c)



1. **When is UnsupportedOperationException thrown in Java?**

In a Java collection UnsupportedOperationException is thrown when the requested operation is not supported by the collection.

It is an unchecked exception that is thrown on optional operations.

If there is an optional add() or remove() methods in a read only collection, then this exception can be thrown.

1. **Let say there is a Customer class. We add objects of Customer class to an ArrayList. How can we sort the Customer objects in ArrayList by using customer firstName attribute of Customer class?**

There are two ways to handle this scenario. We can use these options:

Comparable: Implement the Comparable interface for Customer class and compare customer objects by firstName attribute.

Comparator: Implement Comparator for comparing two Customer objects on the basis of firstName attribute. Then use this comparator object in sort method of Collections class.

1. **What is the difference between Synchronized Collection and Concurrent Collection?**

In Java 1.5 many Concurrent collection classes were added in SDK.

These are ConcurrentHashMap, CopyOnWriteArrayList, BlockingQueue etc.

Java also provides utility methods to get a synchronized copy of collection like ArrayList, HashMap etc. by using

Collections.synchronizedList(), Collections.synchronizedMap() methods.

The main difference is in performance. Concurrent collection classes have better performance than synchronized collection classes because they lock only a portion of the class to achieve concurrency and thread-safety.

1. **What is the scenario to use ConcurrentHashMap in Java?**

ConcurrentHashMap is more suited for scenarios where we have multiple reader threads and one writer thread. In this case map is locked only during the write operation.

If we have an equal number of reader and writer threads then ConcurrentHashMap performance is similar to a Hashtable or a synchronized HashMap.

1. **How will you create an empty Map in Java?**

There are two ways to create an empty Map in Java.

1. **Immutable**: If we want an immutable empty Map, we canuse following code:

myMap = Collections.emptyMap();

1. **Any map**: For all other scenarios, we can use followingcode by using new method:

myMap = new HashMap();

1. **What is the difference between remove() method of Collection and remove() method of Iterator?**

In Collection interface remove(Object o) method is used to remove objects from a Collection.

List interface also provides remove(int index) method to remove an object at a specific index.

These methods are used to remove an entry from Collection, while no thread is iterating over it.

When we are iterating over a Collection, then we have to remove() method of Iterator. This method removes current element from Iterator’s point of view. If we use remove(0 method of Collection or List, then we will get ConcurrentModificationException.

Therefore, it is recommended to use remove() method of Iterator during the traversal of a Collection by an Iterator.

1. **Between an Array and ArrayList, which one is the preferred collection for storing objects?**

An ArrayList is backed up by array internally. There are many usability advantages of using an ArrayList over an array in Java.

Array has a fixed length at the time of creation. Once it is created we cannot change its length.

ArrayList is dynamic in size. Once it reaches a threshold, it automatically allocates a new array and copies contents of old array to new array.

Also ArrayList provides support of Generics. But Array does not support Generics.

E.g. If we store an Integer object in a String array at Runtime it will throw ArrayStoreException. Whereas, if we use ArrayList then as compile time we will get the error. This helps in preventing errors from happening at runtime.

If we know the size in advance and do not need re-sizing the collection then Array should be used in place of an ArrayList.

1. **Is it possible to replace Hashtable with ConcurrentHashMap in Java?**

Yes, a ConcurrentHashMap can be replaced with Hashtable in Java.

But it requires careful observation, since locking behavior of Hashtable is different than that of ConcurrentHashmap.

A Hashtable locks whole Map instead of a portion of Map. Compound operations like if(Hashtable.get(key) == null) put(key, value) work in Hashtable but not in ConcurrentHashMap.

In a ConcurrentHashMap we use putIfAbsent() method for such a scenario.

1. **How CopyOnWriteArrayList class is different from ArrayList and Vector classes?**

CopyOnWriteArrayList was introduced in Java 1.5. It implements List interface.

It provides better concurrent access methods than a Synchronized List.

In CopyOnWriteList, concurrency is achieved by copying ArrayList over each write and replace with original instead of locking.

CopyOnWriteArrayList also does not throw any ConcurrentModification Exception during Iteration.

It is a thread-safe list.

It is different from a Vector in terms of Concurrency. CopyOnWriteArrayList provides better Concurrency by reducing contention among readers and writers.

1. **Why ListIterator has add() method but Iterator does not have?**

ListIterator can iterate in the both directions of a Collection. It maintains two pointer for previous and next element. In ListIterator we can use add() method to add an element into the list immediately before the element returned by next() method.

So a subsequent call to next() method will not be affected. And the call to previous() method will return the newly added element.

In Iterator we can only traverse in one direction. So there is no purpose of add() method there.

1. **Why do we sometime get ConcurrentModificationException during iteration?**

When we remove an object by using remove() method of a Collection or List while an Iterator thread is traversing it, we get ConcurrentModificationException. If an Iterator detects any

structural change in Collection it can throw ConcurrentModificationException.

1. **How will you convert a Map to a List in Java?**

In Java, a Map has three collection sets:

key set

value set

key-value set

Each of these Sets can be converted to List by using a constructor.

Sample code is as follows:

List keyList = new ArrayList(map.keySet()); List valueList = new ArrayList(map.values()); List entryList = new ArrayList(map.entrySet());

1. **How can we create a Map with reverse view and lookup in Java?**

In a Map we can lookup for a value by using a distinct key. In a Map with reverse view and lookup, even the values are distinct. So there is one to one mapping between keys and values and vice version.

If we enable this constraint on a Map then we can look up a key by its value. Such data structure is called bi-directional map.

There is no built data structure similar to reverse lookup Map in JDK.

But Apache Common Collections and Guava libraries provide implementation of bidirectional map. It is called BidiMap and BiMap. Both of these data structure enforce the constraint of one to one mapping between keys and values.

1. **How will you create a shallow copy of a Map?**

In Java, most implementations of Map interface provide a constructor to create copy of another map. But the copy method is not synchronized.

Therefore, when a thread is copying the map, another thread can modify it.

To prevent such a scenario, we should use Collections.synchronizedMap() method to first create a thread-safe map.

Another way of to create a shallow copy is by using clone() method.

But it is not considered as a recommended approach.

1. **Why we cannot create a generic array in Java?**

Java does not allow creation of array with generics as elements.

In Java an array has to know the type information of its elements at runtime.

This information is used at runtime to throw ArrayStoreException if data type of an element to be inserted does not match the type of Array.

In case of Generics, the type information of a collection is erased at runtime by Type Erasure. Due to this array cannot use generics as elements.

1. **What is a PriorityQueue in**

**Java?**

A PriorityQueue is data structure based on Queue. Unlike Queue, the elements on PriorityQueue are not returned in FIFO order.

A PriorityQueue maintains the natural order of its elements or it uses a Comparator provided at initialization.

It is an unbounded queue based on a priority heap.

PriorityQueue does not allow null values. We cannot add any object that does not provide natural ordering to PriorityQueue.

PriorityQueue in Java is not thread-safe.

It gives O(log n) time for enqueing and dequeing operations.

1. **What are the important points to remember while using Java Collections Framework?**

Some of the important points to remember while using Java Collections Framework are:

1. **Interfaces**: For Collections, we should write code withgeneric interfaces instead of concrete implementation. Due to this we maintain the flexibility of changing the implementation at a later point of time.
2. **Generics**: We should use Generics for type-safety and toavoid ClassCastException at runtime.
3. **Collections**: It is recommended to use Collections utilityclass for algorithms and various other common methods for Collections.
4. **Right Type**: We have to choose the right type of Javacollection based on our need. If size is fixed, we can use Array over ArrayList. If we do not want duplicate elements we use Set.

If we need the ability to iterate the elements of a Map in the order of insertion then we use a TreeMap.

1. **Initial Size**: In some collection classes we can specify theinitial size/capacity. Therefore we should have an estimate of number of elements in a Collection before deciding the right collection type. We can use it to avoid rehashing or resizing.
2. **Map**: We should use immutable classes provided by Javaas key elements in a Map.

1. **How can we pass a Collection as an argument to a method and ensure that method will not be able to modify it?**

To ensure that a method is not able to modify a Collection passed as an argument, we have to make the Collection read only.

We can make a read only collection by using Collections.unmodifiableCollection(Collection c) method.

This will make sure that any operation to change the collection will throw UnsupportedOperationException.

1. **Can you explain how HashMap works in Java?**

In Java, a HashMap works on the concept of hashing.

A HashMap in Java stores both key and value objects, in a bucket. It is stored as an Entry object that implements Map.Entry interface.

The key object used in a HashMap has to provide implementation for hashCode() and equals() methods.

When put() method is used to store a key-value pair, the HashMap implementation calls hashCode() method on Key object to calculate a hash that is used to find a bucket where Entry object will be stored.

When get() method is used to retrieve a value stored against a key object, we first calculate a hash of Key object. Then we use this hash to find the bucket in which that particular key is stored.

Once Key object’s location is found, it may happen that more than one Key is stored in same location. So now we use equals() method to find the exact Key object. Once the exact Key object is found we use it to get Value object.

1. **Can you explain how HashSet is implemented in Java?**

Internally, a HashSet uses a HashMap to store the elements and to maintain the uniqueness of elements.

When we create a HashSet object, a corresponding HashMap object is also created.

When we insert an element in HashSet, it inserts it into corresponding HashMap.

1. **What is a NavigableMap in**

**Java?**

As the name suggests, NavigableMap provides the capability to navigate the keys of a Map in Java. A NavigableMap extends SortedMap interface.

Some of the interesting methods of a NavigableMap are descendingKeySet(), descendingMap(), headMap() and tailMap().

1. **What is the difference between descendingKeySet() and descendingMap() methods of NavigableMap?**

The descendingKeySet() method of NavigableMap returns a NavigableSet in which the elements are stored in reversed order as compared to the original key set.

The returned view is internally represented by the original KeySet of NavigableMap. Therefore any changes to the descending set also get reflected in the original set.

But it is not recommended to remove elements directly from the key set. We should use the Map.remove() method.

The descendingMap() method of NavigableMap returns a NavigableMap which is an inverse view of the original Map. The order of the elements in this view are in reverse order of the elements in original map. Any changes to this view are also reflected in the original map.

1. **What is the advantage of NavigableMap over Map?**

The main advantage of NavigableMap over Map is the Navigation capability.

It provides the capabilities of a Map, SortedMap and navigation in one collection.

It even returns the closest matches for given search targets.

Methods like lowerEntry, floorEntry, ceilingEntry, and higherEntry return Map.Entry objects associated with keys respectively less than, less than or equal, greater than or equal, and greater than a given key.

Methods like lowerKey, floorKey, ceilingKey, and higherKey return only the associated keys. All of these methods are designed for locating, not traversing entries.

1. **What is the difference between headMap(), tailMap() and subMap() methods of NavigableMap?**

The headMap() method returns a view of the original NavigableMap that contains the elements that are less than a given element.

NavigableMap original = new TreeMap(); original.put("1", "1"); original.put("2", "2"); original.put("3", "3");

//this headmap1 will contain elements "1" and "2"

SortedMap headmap1 = original.headMap("3");

//this headmap2 will contain elements "1", "2", and "3" because "inclusive"=true

NavigableMap headmap2 = original.headMap("3", true);

The tailMap() method works similar to headMap() method, but it returns all elements that are higher than the given input element.

The subMap() method accepts two parameters demarcating the boundaries of the view map to return.

All the three methods return a subset of the original map in a view form.

1. **How will you sort objects by Natural order in a Java List?**

We can use Collections.sort method to sort the elements of a List in natural order. To use this method, we have to make sure that element objects implement compareTo() method.

We can also use a Comparator to define the natural ordering for elements of a List. Then we can use this Custom Comparator in sort method of Collections class.

1. **How can we get a Stream from a List in Java?**

From Java 8 onwards it is a very easy to get a Stream from a List. We can just use stream() method to get a stream from a list of elements.

1. **Can we get a Map from a Stream in Java?**

Yes, we can create a Map from the elements of a Stream. We can use map() method to get a Map.

E.g. items.stream()

.map( item -> item.toLowerCase() )

In this example we are creating a map with each item object mapped to its LowerCase equivalent.

This is also used in Map-Reduce implementation on a Stream.

1. **What are the popular**

**implementations of Deque in Java?**

The two most popular implementation of Deque interface in Java are:

1. **ArrayDeque**: It is a resizable array implementation ofDeque. The capacity of ArrayDeque can increase based on the need of the program. It is not thread safe implementation. Also the iterator on ArrayDeque is fail-fast.
2. **LinkedList**: This is another popular implementation ofDeque interface in Java. It is also not synchronized, so it is not thread-safe. It mainly provides functionality of a doubly linked list.

**Java Tricky Questions**

1. **Is there any difference between a = a + b and a += b expressions?**

When we add two integral variables e.g. variables of type byte, short, or int in Java, then they are first promoted to int type, and then addition happens.

The += operator implicitly casts the result of addition into the type of variable used to hold the result.

What happens when you put return statement or System.exit () on try or catch block? Will finally block execute?

It is a popular tricky Java interview question. Most of the programmers think that no matter what the finally block will always execute. This question challenges that concept by putting a return statement in the try or catch block or calling System.exit() from try or catch block.

You can answer by saying that finally block executes even if we put a return statement in the try block or catch block. But finally block does not execute if you call System.exit() from try or catch block.

1. **What does the expression 1.0 / 0.0 return? Will there be any compilation error?**

Double class is the source of many tricky interview questions. You may know about the double primitive type and Double class. But while doing floating point arithmetic some people don't pay enough attention to Double.INFINITY, NaN, and -0.0. There are rules that govern the floating point arithmetic calculations involving Double.

The answer to this question is that 1.0 / 0.0 will compile successfully. And it will not throw ArithmeticException. It will just return Double.INFINITY.

.

1. **Can we use multiple main methods in multiple classes?**

Yes. When we start an application in Java, we just mention the class name to be run to java command. The JVM looks for the main method only in the class whose name is passed to java command. Therefore, there is no conflict amongst the multiple classes having main method.

1. **Does Java allow you to override a private or static method?**

The question is tricky but the answer is very simple. You cannot override a private or static method in Java. If we create a similar method with same return type and same method arguments in child class, then it will hide the superclass method. This is known as method hiding.

Also, you cannot override a private method in sub class because Private method is not visible even in a subclass. Therefore, what you can do is to create another private method with the same name in the child class.

So in both the cases, it is not method overriding. It is either method hiding or a new method.

1. **What happens when you put a key object in a HashMap that is already present?**

In a HashMap there are buckets in which objects are stored. Key objects with same HashCode go to same bucket.

If you put the same key again in a HashMap, then it will replace the old mapping because HashMap doesn't allow duplicate keys. The same key will have same HashCode as previous key object. Due to same HashCode, it will be stored at the same position in the bucket.

1. **How can you make sure that N threads can access N resources without deadlock?**

This question checks your knowledge of writing multi-threading code. If you have experience with deadlock and race conditions, you can easily answer this.

The answer is that by resource ordering you can prevent deadlock. If in our program we always acquire resources in a particular order and release resources in the reverse order, then we can prevent the deadlock.

So a thread waiting for same resource can not get into deadlock while the other thread is trying to get it and holding the resource required by first thread. If both of them release the resources in right order, one of them can acquire it to finish the work.

1. **How can you determine if JVM is 32-bit or 64-bit from Java Program?**

We can find JVM bit size 32 bit or 64 bit by running java command from the command prompt.

Or we can get it from Java program.

Sun has a Java System property to determine the bit size of the

JVM: 32 or 64:

sun.arch.data.model=32 // 32 bit JVM

sun.arch.data.model=64 // 64 bit JVM

We can use System.getProperty("sun.arch.data.model") to determine if it is 32/64 bit from Java program.

1. **What is the right data type to represent Money (like Dollar/Pound) in Java?**

To represent money you need decimal points in the numbers like $1.99.

BigDecimal class provides good methods to represent Money. Using BigDecimal, we can do the calculation with decimal points and correct rounding. But using BigDecimal is a little bit high on memory usage.

We can also use double with predefined precision. But calculation on double can give erroneous results.

1. **How can you do multiple inheritances in Java?**

This is a question to trick people coming from C++ and Scala background to Java. There are many Object Oriented languages that support multiple inheritances. But Java is not one of them.

Answer of this question can be that, Java does support multiple inheritances of by allowing an interface to extend other interfaces. You can implement more than one interface. But you cannot extend multiple classes. So Java doesn't support multiple inheritances of implementation.

But in Java 8, the default method breaks the rule of multiple inheritances behavior.

1. **Is ++ operation thread-safe in**

**Java?**

No, ++ operator is not a thread safe operation. It involves multiple instructions like- reading a value, incrementing it and storing it back into memory. These instructions can overlap between multiple threads. So it can cause issues in multi-threading.

1. **How can you access a non-static variable from the static context?**

We cannot access a non-static variable from the static context in Java. If you write a code like that, then you will get compile time error. It is one of the most common problems for beginner Java programmers, when they try to access instance variable inside the main method in a class.

Since main method is static in Java, and instance variables are non-static, we cannot access instance variable inside main. The solution is to create an instance of the object and then access the instance variables.

1. **Let say there is a method that throws NullPointerException in the superclass. Can we override it with a method that throws RuntimeException?**

This question is checking your understanding of the concepts of method overloading and overriding in Java.

We can throw superclass of RuntimeException in an overridden method, but we cannot do the same if it is a checked Exception.

1. **How can you mark an array volatile in Java?**

If you know multi-threading well then you can easily answer it.

We can mark an array volatile in Java. But it makes only the reference to array volatile, not the whole array.

If one thread changes the reference variable to point to another array, then it will provide a volatile guarantee. But if multiple threads are changing individual array elements, they won't be having same reference due to the reference itself being volatile.

1. **What is a thread local variable in Java?**

Thread-local variable is a variable restricted to a specific thread. It is like thread's own copy of variable that is not shared among multiple threads.

Java provides ThreadLocal class to support thread-local variables. To achieve thread-safety, you can use it. To avoid any memory leak, it is always good to remove a thread-local variable, once its work is done.

1. **What is the difference between sleep() and wait() methods in Java?**

In Java, we use these methods to pause currently running thread.

There is a simple difference between these.

sleep() is actually meant for short pause because it doesn't release lock.

wait() is meant for conditional wait and it can release a lock that can be acquired by another thread to change the condition on which it is waiting.

1. **Can you create an Immutable object that contains a mutable object?**

In Java, it is possible to create an Immutable object that contains a mutable object.

We should not share the reference of the mutable object, since it is inside an immutable object. Instead, we can return a copy of it to other methods.

1. **How can you convert an Array of bytes to String?**

You can convert an Array of bytes to String object by using the String constructor that accepts byte[]. We need to make sure that right character encoding is used. Else we may get different results after conversion.

1. **What is difference between CyclicBarrier and CountDownLatch class?**

CyclicBarrier and CountDownLatch classes were introduced from Java 5.

We can reuse CyclicBarrier even if it is broken, but we cannot reuse CountDownLatch in Java.

1. **What is the difference between StringBuffer and StringBuilder?**

StringBuilder was introduced in Java 5. The main difference between both of them is that StringBuffer methods e.g. length(), capacity(), append() are synchronized. But corresponding methods in StringBuilder are not synchronized.

Due to this difference, concatenation of String using StringBuilder is faster than StringBuffer. Now it is considered bad practice to use StringBuffer, because, in most of the scenarios, we perform string concatenation in the same thread.

1. **Which class contains clone method? Cloneable or Object class?**

It is a very basic trick question. clone() method is defined in Object class. Cloneable is a marker interface that doesn't contain any method.

1. **How will you take thread dump in Java?**

There are platform specific commands to take thread dump in Java.

In Linux/Unix, just use kill -3 PID, where PID is the process id of Java process. It will give the thread dump of Java process.

In Windows, press Ctrl + Break. This will instruct JVM to print thread dump in standard out or err. It can also go to console or log file depending upon your application configuration.

1. **Can you cast an int variable into a byte variable? What happens if the value of int is larger than byte?**

An int is 32 bit in Java. But a byte is just 8 bit in Java. We can cast an int to byte. But we will lose higher 24 bits of int while casting. Because a byte can hold only first 8 bits of int. Remaining 24 bits (32-8 = 24) will be lost.

1. **In Java, can we store a double value in a long variable without explicit casting?**

No, we cannot store a double value into a long variable without casting it to long. The range of double is more than that of long. So we need to type cast.

To answer this question, just remember which one is bigger between double and long in Java.

1. **What will this return 5\*0.1 == 0.5? true or false?**

The answer is false because floating point numbers can not be represented exactly in Java, so 5\*0.1 is not same as 0.5.

1. **Out of an int and Integer, which one takes more memory?**

An Integer object takes more memory than an int in Java. An Integer is an object and it stores meta-data overhead about the object. An int is a primitive type so its takes less memory and there is no meta-data overhead.

1. **Can we use String in the switch case statement in Java?**

Yes. From Java 7 onwards, String can be used in switch case statement. This gives convenience to programmer. But internally hash code of String is used for the switch statement.

1. **Can we use multiple main methods in same class?**

Yes. You can have multiple methods with name main in the same class. But there should be only one main method with the signature public static void main(String[] args). JVM looks for main with this signature only. Other methods with name main in same class are just ignored.

1. **When creating an abstract class, is it a good idea to call abstract methods inside its constructor?**

No, we should avoid calling abstract methods in the constructor of an abstract class. Because, it can restrict how these abstract methods can be implemented by child classes.

Many IDE give “Overridable method call in constructor” warning for such implementation.

This is a problem of object initialization order. The superclass constructor will run before the child class constructor. It means child class is not yet initialized. But due to presence of overridden method in superclass, the overridden method of subclass is called when the subclass is not fully initialized.

1. **How can you do constructor chaining in Java?**

When we call one constructor from another constructor of the same class, then it is known as constructor chaining in Java. When you have multiple overloaded constructors in a class, you can do constructor chaining.

1. **How can we find the memory usage of JVM from Java code?**

We can use memory management related methods provided in java.lang.Runtime class to get the free memory, total memory and maximum heap memory in Java.

By using these methods, you can find out how much of the heap is used and how much heap space still remains.

Runtime.freeMemory() returns amount of free memory in bytes.

Runtime.totalMemory() returns total memory in bytes.

Runtime.maxMemory() returns maximum memory in bytes.

1. **What is the difference between x == y and x.equals(y) expressions in Java?**

The x == y expression does object reference matching if both a and b are an object and only returns true if both are pointing to the same object in the heap space.

The x.equals(y) expression is used for logical mapping and it is expected from an object to override this method to provide logical equality.

Eg. A Book object may be logically equal to another copy of same Book, but it is a different object which will be false while doing x == y.

1. **How can you guarantee that the garbage collection takes place?**

No. We cannot guarantee the garbage collection in Java. Java documentation explicitly says that GarbageCollection is not guaranteed.

You can call System.gc() to request garbage collection, however, that's what it is - a request. It is upto GC's discretion to run.

1. **What is the relation between x.hashCode() method and x.equals(y) method of Object class?**

x.hashCode() method returns an int hash value corresponding to an object instance.

It is used in hashCode based collection classes like Hashtable, HashMap, LinkedHashMap etc.

hashCode() method is also related to equals() method.

As per Java specification, two objects which are equal to each other using equals() method must have same hash code.

Therefore, two objects with same hashCode may or may not be equal to each other. But two equal objects should have same hash code.

1. **What is a compile time constant in Java?**

A compile time constant is public static final variable. The public modifier is optional here. At compile time, they are replaced with actual values because compiler knows their value up-front and it also knows that it cannot be changed during run-time. So they are constants.

1. **Explain the difference between fail-fast and fail-safe iterators?**

The main difference between fail-fast and fail-safe iterators is whether or not the collection can be modified while it is being iterated.

Fail-safe iterators allow modification of collection in an iteration task. But fail-fast iterators do not allow any modification to collection during iteration.

During iteration, fail-fast iterators fail as soon as they realize that the collection has been modified. Modification can be addition, removal or update of a member. And it will throw a ConcurrentModificationException.

Eg. ArrayList, HashSet, and HashMap are fail-fast.

Fail-safe iterators operate on a copy of the collection. Therefore they do not throw an exception if the collection is modified during iteration.

Eg. ConcurrentHashMap, CopyOnWriteArrayList are fail-safe.

1. **You have a character array and a String. Which one is more secure to store sensitive data (like password, date of birth, etc.)?**

Short answer is, it is safe to store sensitive information in character array.

In Java, String is immutable and it is stored in the String pool. Once a String is created, it stays in the pool in memory until it is garbage collected. You have no control on garbage collection. Therefore, anyone having access to a memory dump can potentially extract the sensitive data and use it.

Whereas, if you use a mutable object like a character array, to store the value, you can set it to blank once you are done with it. Once it is made blank it cannot be used by anyone else.

1. **Why do you use volatile keyword in Java?**

The volatile keyword guarantees global ordering on reads and writes to a variable. This implies that every thread accessing a volatile field will read the variable’s current value instead of using a cached value.

By marking the variable volatile, the value of a variable is never cached thread-locally. All reads and writes will go straight to main memory of Java.

1. **What is the difference between poll() and remove() methods of Queue in Java?**

It is a basic question to know the understanding of Queue data structure. Both poll() and remove() methods remove and return the head of the Queue.

When Queue is empty, poll() method fails and it returns null, but remove() method fails and throws Exception.

1. **Can you catch an exception thrown by another thread in Java?**

Yes, it can be done by using Thread.UncaughtExceptionHandler.

Java Documentation says “When a thread is about to terminate due to an uncaught exception the Java Virtual Machine will query the

thread for its UncaughtExceptionHandler using[Thread.getUncaughtExceptionH](https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html#getUncaughtExceptionHandler()) will invoke the handler's uncaughtException method, passing the thread and the exception as arguments.”

1. **How do you decide which type of Inner Class – Static or Non-Static to use in Java?**

An inner class has full access to the fields and methods of the enclosing class. This is convenient for event handlers, but comes at a cost. Every instance of an inner class retains and requires a reference to its enclosing class.

Due to this cost, there are many situations where static nested classes are preferred over inner classes. When instances of the nested class outlive instances of the enclosing class, the nested class should be static to prevent memory leaks.

At times, due to their “hidden” reference to enclosing class, Inner classes are harder to construct via reflection.

1. **What are the different types of Classloaders in Java?**

Java Classloader is the part of the Java Runtime Environment (JRE) that loads classes on demand into Java Virtual Machine (JVM).

When the JVM is started, three types of class loaders are used:

1. Bootstrap Classloader: It loads core java API file rt.jar classes from folder.
2. Extension Classloader: It loads jar files from lib/ext folder.
3. System/Application Classloader: It loads jar files from path specified in the CLASSPATH environment variable.

Classes may be loaded from the local file system, a remote file system, or even the web.

1. **What are the situations in which you choose HashSet or TreeSet?**

HashSet is better than TressSet in almost every way. It gives O(1) for add(), remove() and contains() operations. Whereas, TressSet gives O(log(N)) for these operations.

Still, TreeSet is useful when you wish to maintain order over the inserted elements or query for a range of elements within the set.

We should use TreeSet when we want to maintain order. Or when there are enough read operations to offset the increased cost of write operations.

1. **What is the use of method references in Java?**

Java 8 has introduced Method references. It allows constructors and methods to be used as lambdas.

The main uses of Method reference are to improve code organization, clarity and terseness.

1. **Do you think Java Enums are more powerful than integer constants?**

Yes. Java Enums provide many features that integer constants cannot. Enums can be considered as final classes with a fixed number of instances. Enums can implement interfaces but cannot extend another class.

While implementing the strategy pattern, we can use this feature of Enums. Especially, when the number of strategies is fixed.

You can also attach meta-data to enum values in Java. Also enum values are typesafe, where as integer constants are not.

You can also define custom behavior in enum values.

1. **Why do we use static initializers in Java?**

In Java, a static initializer can run code during the initial loading of a class and it guarantees that this code will only run once. Also the static code will finish running before a class can be accessed in any way.

Initializing static members from constructors is more work. You have to make sure that every constructor does this. You need to maintain a flag to mark the static work when it is done. You may have to think about synchronization or races conditions for work in static block not initialized from static context.

1. **Your client is complaining that your code is throwing NoClassDefFoundError or NoSuchMethodError, even though you are able to compile your code without error and method exists in your code. What could be the reason behind this?**

Sometimes we upgrade our libraries even with same method name. But we forget to let the client know about the new version. Due this different in version, we get NoClassDefFoundError or NoSuchMethodError at runtime when one library was not compatible with such an upgrade.

Java build tools and IDEs can also produce dependency reports that tell you which libraries depend on that JAR. Mostly, identifying and upgrading the library that depends on the older JAR resolve the issue.

1. **How can you check if a String is a number by using regular expression?**

Regex is a powerful tool for matching patterns and searching patterns.

A numeric String can only contain digits i.e. 0 to 9. It can also contain + and - sign at start of the String. We can create a regular expression for these two rules. One simple example is as follows:

Pattern pattern = Pattern.compile(".\*\\D.\*");

1. **What is the difference between the expressions String s = "Temporary" and String s = new String("Temporary ")? Which one is better and more efficient?**

In general, String s = " Temporary " is more efficient to use than String s = new String("Temporary ").

In case of String s = " Temporary ", a String with the value “Temporary” is created in String pool. If another String with the same value is created (e.g., String s2 = " Temporary "), it will reference the same object in the String pool.

But, when you use String s = new String("Temporary "), Java creates a String with the value “Temporary” in the String pool. Also, that String object is then passed to the constructor of the String Object i.e. new String("Temporary "). And this call creates another String object (not in the String pool) with that value.

Therefore, each such call creates an additional String object. E.g. String s2 = new String("Temporary ") creates an extra String object, rather than just reusing the same String object from the String pool.

So String s = “Temporary” is always an efficient way.

1. **In Java, can two equal objects have the different hash code?**

No. It is not possible for two equal objects to have different hashcode. But two objects with same hashcode may or may not be equal.

1. **How can we print an Array in**

**Java?**

We can print an array by using methods of Arrays class. We can either use Arrays.toString() method or we can use Arrays.deepToString() method.

Since array doesn't implement toString() method by itself, just passing an array to System.out.println() will not print its contents. But we can use Arrays.toString() to print each element of an array.

1. **Is it ok to use random numbers in the implementation of hashcode() method in Java?**

No. The hashcode of an object should be always same. If you use random number in hashcode() method, then you may get a different value of hashcode for same object. This will break the hashcode contract.

1. **Between two types of**

**dependency injections, constructor injection and setter dependency injection, which one is better?**

Constructor injection guarantees that a class will be initialized with all its dependencies during creation. But setter injection provides flexibility to set an optional dependency.

If we are using an XML file to describe dependencies, the setter injection is more readable.

In general, it is a good practice to use constructor injection for mandatory dependencies and use setter injection for optional dependencies.

1. **What is the difference between DOM and SAX parser in Java?**

In Java, Document Object Model (DOM) parser loads the whole XML into memory and creates a tree based on DOM model. This helps it in quickly locating the nodes, and making a change in the structure of XML.

On the other hand, Simple API for XML (SAX) parser is an event based parser. It doesn't load the whole XML into memory. Due to this reason DOM is faster than SAX but require more memory and is not suitable to parse large XML files.

1. **Between Enumeration and Iterator, which one has better performance in Java?**

Enumeration interface is a read-only interface. It has better performance than Iterator. It is almost twice as fast as compared to an Iterator. It also uses very less memory. Also Enumeration does not have remove() method.

On the other hand, Iterator interface is safer than Enumeration, since it can check whether a collection is modified or not during iteration. If a collection is altered while an Iterator is iterating, then it throws ConcurrentModificationException.

1. **What is the difference between pass by reference and pass by value?**

Whenever an object is passed by value, it means that a copy of the object is passed. Even if changes are made to that object, it doesn’t affect the original value.

Whenever an object is passed by reference, it means that the actual object is not passed, rather a reference of the object is passed. Therefore, any changes made by an external method, are also reflected in the actual object and its reference.

1. **What are the different ways to sort a collection in Java?**

The most popular way to sort a collection in Java is by calling Collections.sort() method. You can provide your custom Comparator to sort() method for sorting the data in your custom way.

The other way is to use a Sorted collection like TreeSet or TreeMap that stores the information in a sorted order and then you can convert it to a List.

1. **Why Collection interface doesn’t extend Cloneable and Serializable interfaces?**

Collection interface just 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 may allow duplicate keys, while other collections may not.

A lot of collection implementations have clone method. But many do not. It is not worthwhile to include it in all, since Collection is an abstract representation. What matters is the concrete implementation.

Cloning and serialization come into picture while doing concrete implementation. Therefore, the concrete implementations of collections should decide how they can be cloned or serialized.

1. **What is the difference between a process and a thread in Java?**

A process is simply an execution of a program.

A Thread is a single execution sequence within a process.

A process may contain multiple threads. A Thread is also called as a lightweight process.

1. **What are the benefits of using an unordered array over an ordered array?**

In an ordered array the search time has time complexity of O(log n).

Whereas, in an unordered array, search time complexity is O (n).

In an ordered array, the insert operation has a time complexity of O(n). Whereas, the insertion operation for an unordered array takes constant time of O(1).

Therefore, when we have more writes than reads, it is preferable to use an unordered array.

1. **Between HashSet and TreeSet collections in Java, which one is better?**

A HashSet is Implemented using a HashTable. Therefore, its elements are stored in a random order. The add(), remove(), and contains() methods of a HashSet have constant time complexity O(1).

A TreeSet is implemented using a tree data structure. The elements in a TreeSet are sorted in a natural order. Therefore, add(), remove(), and contains() methods have time complexity of O(logn).

So from performance perspective, HashSet has better performance than TreeSet. But if you want to store elements in a natural sorting order, then TreeSet is a better collection.

1. **When does JVM call the finalize() method?**

JVM instructs the Garbage Collector to call the finalize method, just before releasing an object from the memory. A programmer can implement finalize() method to explicitly release the resources held by the object. This will help in better memory management and avoid any memory leaks.

1. **When would you use Serial Garabage collector or Throughput Garbage collector in Java?**

The Serial Garbage collector is used for small applications that require heap memory upto 100 MB.

The Throughput Garbage collector is used in medium to large size Java applications.

1. **In Java, if you set an object reference to null, will the Garbage Collector immediately free the memory held by that object?**

No. JVM decides to run the Garbage Collector whenever it is low on memory. When Garbage Collector runs, it looks for objects that are available for garbage collection and then frees the memory associated with this object.

So just setting an Object reference null makes it eligible for Garbage Collection, but it does not immediately free the memory.

1. **How can you make an Object eligible for Garbage collection in Java?**

To make an Object eligible for Garbage collection, just make sure that it is unreachable to the program in which it is currently defined

* created / used. You can set the object reference to null and make sure no other object refers it. Once the object cannot be reached, Garbage Collection can clean it during the next run.

1. **When do you use Exception or Error in Java? What is the difference between these two?**

Throwable class is the superclass of Exception and Error classes in Java.

When you want to catch the exceptional conditions that your program can create or encounter, then use the Exception class or subclass of Exception.

When you come across situations that are unexpected then use Error class in Java. Also recovering from Error is not possible in most of cases. So it is better to terminate the program.

1. **What is the advantage of PreparedStatement over Statement class in Java?**

PreparedStatements are precompiled statements for database queries. Due to this their performance is much better. Also, we can reuse PreparedStatement objects with different input values to the same query.

Where as, Statement class does not provide these features.

1. **In Java, what is the difference between throw and throws keywords?**

When we want to raise an exception in our code, we use the throw keyword with the name of the exception to be raised.

Where as, throws keyword is used in method declaration. Throws keyword tells us the Exception that can be thrown by this method. Any caller of this method should be prepared to expect this Exception.

Another minor difference is that throw is used only with one exception, but throws can be used with comma-separated list of multiple exceptions.

1. **What happens to the Exception object after the exception handling is done?**

Once the exception handling is complete, the Exception object is not reachable. Then it is garbage collected in the next run of Garbage Collector.

1. **How do you find which client machine is sending request to your servlet in Java?**

We can use the ServletRequest class to find the IP address or host name of the client machine.

There are methods getRemoteAddr() to get the IP address of the client machine and getRemoteHost() to get the host name of the client machine.

1. **What is the difference between a Cookie and a Session object in Java?**

Both Cookie and Session are used during communication between Client and Server. The Client can disable a Cookie. Due to which the Web server cannot send a cookie. But a client cannot disable a session. So a Session always works irrespective of any setting at the client side.

Also a Session can store any Java object. But the Cookie can only store small information in a String object.

1. **Which protocol does Browser and Servlet use to communicate with each other?**

HTTP protocol. The Browser and Servlet communicate with each other by using the HTTP protocol.

1. **What is HTTP Tunneling?**

There are many network communication protocols on the Internet. But HTTP is the most popular among them. HTTP Tunneling is a technique in which HTTP or HTTPS protocol encapsulated the communication done by any other type of protocol. The masking of other protocol requests as HTTP requests is known as HTTP Tunneling.

1. **Why do we use JSP instead of Servlet in Java?**

Since JSP pages are dynamically compiled into servlets, the programmers can easily make updates to the presentation layer code.

For better performance, JSP pages can be pre-compiled.

Also JSP pages provide flexibility to combine static templates like HTML or XML snippets.

In addition, programmers can make logic changes at the class level, without editing the JSP pages that use the class logic.

1. **Is empty ‘.java’ file name a valid source file name in Java?**

Yes. You can create a class and store it in a file with name .java. You can try it yourself, by creating, compiling and running such a file. It will run correctly.

1. **How do you implement Servlet Chaining in Java?**

To implement, Servlet Chaining, there has to be more than one servlet. The output of one servlet has to be sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. In this way, a chain of servlets is formed to complete a task.

The last servlet in the chain will be responsible for sending final response to client.

1. **Can you instantiate this class?**

public class A

{

A a = new A();

}

No, this class cannot be instantiated, since it will result in recursively calling its constructor.

1. **Why Java does not support operator overloading?**

Java supports Method overloading but does not support operator overloading. It would make the design more complex by adding operator loading. Also it will make more complex compiler.

One more reason is that, it will reduce the performance of JVM by operator overloading, since JCM has to do extra work to find the real meaning of overloaded operators at run time.

1. **Why String class is Immutable or Final in Java?**

Since String objects are cached in a String pool, it makes sense to make the String immutable. The cached String literals are shared between multiple clients. And there is a possibility that one client's action may affect another client’s access to String pool.

String is also used as a parameter in many Java classes. Eg. You can pass hostname, port number as String while opening a network connection. If any one can modify your copy of the String, it can change the hostname. Due to this reason, it makes sense to make String final as soon as it is created.

1. **What is the difference between sendRedirect and forward methods?**

When you use sendRedirect method, it creates a new request. When you use the forward method, it just forwards a request to a new target.

In case of sendRedirect, the previous request scope objects are not available, because it creates a new request.

In case of forward method, the previous request scope objects are available after forwarding.

Also the sendRedirect method is considered slower than the forward method.

1. **How do you fix your Serializable class, if it contains a member that is not serializable?**

If you want to make a class Serializable, but find that this class contains members that are not Serializable, then you have to mark those members as transient. This will ensure that this member is not persisted to a stream of bytes during Serialization.

Therefore, Transient keyword of Java comes to help in this scenario.

1. **What is the use of run time polymorphism in Java?**

During the run time the behavior of an Object can change based on its run time state. Due to this run time polymorphism is introduced in Java. If you override a method in a child class, then you are providing run time polymorphism. Nothing will happen at the compile time. But at the run time, JVM decides which method will be called based on the class of the Object.

1. **What are the rules of method overloading and method overriding in Java?**

When we want to overload a method, we need to make sure that the method name remains same. But method signature can vary in the number or datatype of arguments or in the order of arguments.

When we want to override a method, we ensure that the method is not throwing checked exceptions that are new or higher than those declared by the overridden method. Also we make sure that the method name, arguments and return type remain the same.

Also we cannot override Static and Final methods in Java.

1. **What is the difference between a class and an object in Java?**

A Class is a template or a blue print of an Object to be created. An Object is an instance of a Class. A Class defines the methods and member variables. But an Object populates the values of the member variables.

Therefore a class is a blueprint that you use to create objects. An object is an instance of a class – it is a concrete 'thing' that you made using a specific class.

Most of the OOPS concepts are valid only when an Object is created.

1. **Can we create an abstract class that extends another abstract class?**

Yes. An abstract class can extend another abstract class. It does not need to define the methods of parent abstract class. Only the last non-abstract class has to define the abstract methods of a parent abstract class.

1. **Why do you use Upcasting or Downcasting in Java ?**

When we want to cast a Sub class to Super class, we use Upcasting.

It is also known as widening. Upcasting is always allowed in Java.

When we want to cast a Super class to Sub class, we use Downcasting. It is also known as narrowing.

At times, Downcasting can throw the ClassCastException if it fails the type check.

1. **What is the reason to organize classes and interfaces in a package in Java?**

As the name suggests, a package contains a collection of classes. It helps in setting the category of a file. Like- whether it is a Data Access Object (DAO) or an API.

It helps in preventing the collision of Name space.

Also we can introduce access restriction by using package and the right modifiers on a class and its methods.

1. **What is information hiding in**

**Java?**

Information hiding is OOPS concept. In Java you can use encapsulation to do Information hiding. An object can use the access modifiers like-public, private, protected to hide its internal details from another object. This helps in decoupling the internal logic of an object from outside world.

By using Information hiding, an object can change its internal implementation without impacting the outside calling client’s code.

1. **Why does Java provide default constructor?**

In Java all the interaction takes place between Object instances. To create an Object instance, JVM needs a constructor. Java does not enforce the rule on a programmer to define a default constructor for every class.

Whenever an object has to be created and programmer has not provided a constructor, Java uses default constructor to create the object. Default constructor also initializes member variables with their default values.

1. **What is the difference between super and this keywords in Java?**

We use super keyword to access the methods of the super class from child class.

We use this keyword to access methods of the same class.

1. **What is the advantage of using Unicode characters in Java?**

Unicode characters have much larger number of characters in the specification.

They also contain Asian and non-western European characters.

Most of the modern technologies, websites and browsers support these Unicode characters.

1. **Can you override an overloaded method in Java?**

Yes. Java allows to override an overloaded method, if that method is not a static or final method.

1. **How can we change the heap size of a JVM?**

Java provides the command line parameters to set the heap size for JVM.

You can specify the values in –Xms and –Xmx parameters. These parameters stand for initial and maximum heap size of JVM.

1. **Why should you define a default constructor in Java?**

In general, Java provides a default constructor with each class. But there are certain cases when we want to define our own version of default constructor.

When we want to construct an object with default values, we create our default constructor.

At times, we can mark the default constructor private. So that any other class cannot create an instance of our class. This technique is generally used in Singleton design pattern.

1. **How will you make an Object Immutable in Java?**

To make an object immutable follow these two rules. One, do not use any setter methods that can change the fields of your class. Two, make the fields final. By following these rules, the member variables cannot be changed after initialization. This will ensure that member variables of an Object do not change. And thus the Object will be considered Immutable.

1. **How can you prevent SQL Injection in Java Code?**

In Java, you can use PreparedStatement to prevent SQL injection. In a PreparedStatement you can pass the precompiled SQL queries with pre-defined parameters. This helps in checking the type of parameters to SQL queries. So it protects your code from SQL injection attacks.

1. **Which two methods should be always implemented by HashMap key Object?**

Any object that we want to use as key for HashMap or in any other hash based collection data structure e.g. Hashtable, or ConcurrentHashMap must implement equals() and hashCode() method.

1. **Why an Object used as Key in HashMap should be Immutable?**

The Key object should be immutable so that hashCode() method always return the same value for that object.

The Hashcode returned by hashCode() method depends on values of member variables of an object. If an object is mutable, then the member variables can change. Once the member variables change, the Hashcode changes. If the same object returns different hash code at different times, then it is not reliable to be used in the HashMap.

Let say, when you insert the object, the Hashcode is X, the HashMap will store it in bucket X. But when you search for it the Hashcode is Y, then HashMap will look for the object in bucket Y. So you are not getting what you stored.

To solve this, a key object should be immutable.

Although, the compiler does not enforce this rule, a good programmer always remembers this rule.

1. **How can we share an object between multiple threads?**

There are many ways to share same object between multiple threads. You can use a BlockingQueue to pass an object from one thread to another thread.

You can also use Exchanger class for this purpose. An Exchanger is a bidirectional form of a SynchronousQueue in Java. You can use it to swap the objects as well.

1. **How can you determine if your program has a deadlock?**

If we suspect that our application is stuck due to a Deadlock, then we just take a thread dump by using the command specific to environment in which your application is running. Eg. In Linux you can use command kill -3.

In case of deadlock, you will see in thread dump the current status and stack trace of threads in the JVM, and one or more of them will be stuck with message deadlock.

Also you can do this programmatically by using the ThreadMXBean class that ships with the JDK.

If you don't need programmatic detection you can do this via JConsole. On the thread tab there is a "detect deadlock" button.

**Mixed Questions**

**1. What are Wrapper classes in Java?**

Java has concept of Wrapper classes to allow primitive types to be accessed as objects. Primitive types like boolean, int, double, float etc. have corresponding Wrappers classes – Boolean, Integer, Double, Float etc.

Many of these Wrapper classes are in java.lang package.

Java 5.0 has launched the concept of Autoboxing and Unboxing in Java for Wrapper classes.

E.g.

public class WrapperTest{

public static void main(String args[]){

//Converting int into Integer

int count=50;

Integer i=Integer.valueOf(count);//converting int into Integer

Integer j=a;//autoboxing, now compiler will write Integer.valueOf(count) internally

System.out.println(count+" "+i+" "+j);

}}

1. **What is the purpose of native method in Java?**

The native keyword is used for applying to a method to indicate that the method is implemented in native code using JNI(Java Native Interface).

Therefore, native methods allow Java Developer to directly access platform specific APIs.

Often, native methods are linked to native library.

**3. What is System class?**

System.class is a final class provided by java.lang package. It contains several useful class fields and methods.

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

1. **What is System, out and println in System.out.println method call?**

System is a final class provided by java.lang package.

out refers to PrintStream class and a static member of System class.

println is a method of PrintStream class.

1. **What is the other name of Shallow Copy in Java?**

Object Cloning. A Shallow Copy just copies the values of references in a Class.

1. **What is the difference between Shallow Copy and Deep Copy in Java?**

A Shallow copy just copies the values of the references in the class.

A Deep copy copies the values of the objects as well.

**7. What is a Singleton class?**

A Singleton class in Java has maximum one instance of the class present in JVM, all the time. The constructor of this class is written in such a way that it never creates more than one object of same class.

1. **What is the difference between Singleton class and Static class?**

A static class in Java has only static methods. It is a container of functions. It is created based on procedural programming design.

Singleton class is a pattern in Object Oriented Design. A Singleton class has only one instance of an object in JVM. This pattern is implemented in such a way that there is always only one instance of that class present in JVM.

**JSP**

1. **What are the implicit objects in JSP?**

JSP has following implicit objects:

1. Request
2. Response
3. Application
4. Exception
5. Page
6. Config
7. Session

**10. How will you extend JSP code?**

We can extend JSP code by using Tag libraries and Custom actions.

1. **How will you handle runtime exceptions in JSP?**

We use Errorpage attribute in JSP to catch runtime exceptions. This attribute forwards user request to the error page automatically.

1. **How will you prevent multiple submits of a page that come by clicking refresh button multiple times?**

We can use Post Redirect Get (PRG) pattern to solve the issue of multiple submission of same data. It works as follows:

First time when a user submits a form to server by POST or GET method, then we update the state in application database.

Then we send a redirect response to send reply to client.

Then we load a view by using GET command. There is no data is sent in this. Since this a new JSP page, it is safe from multiple submits. The code that processes the request is idempotent. So it does not do same action twice for same request.

1. **How will you implement a thread safe JSP page?**

We can use SingleThreadModel Interface to implement a thread safe JSP page.

We can also add <%@page isThreadSafe=”false” %> directive in JSP page to make it thread safe.

1. **How will you include a static file in a JSP page?**

We can use include directive of JSP to include a Static page in JSP. In this approach, we use translation phase to include a static page. We have to specify the URL of the resource to be included as file attribute in this directive.

E.g. <%@ include file="footer.html" %>

1. **What are the lifecycle methods of a JSP?**

A JSP has following lifecycle methods:

1. **jspInit**(): This method is invoked when the JSP is calledfor the first time. We can do initial setup for servicing a request in this method.
2. \_**jspService**(): This method is used to serve every request of the JSP.
3. **jspDestroy**(): Once we remove a JSP from the container,we call this method. It is used for cleanup of resources like Database connections etc.

1. **What are the advantages of using JSP in web architecture?**

We get following advantages by using JSP in web architecture:

1. **Performance**: JSP provides very good performance due totheir design of using same code to service multiple requests.
2. **Fast**: Since JSP is pre-compiled, server can serve thepages very fast.
3. **Extendable**: JSP is based on Java Servlets. This helps inextending JSP architecture with other Java technologies like JDBC, JMS, JNDI etc.
4. **Design**: It is easier to design user interface with JSP, sinceit is very close to HTML. UI designers can create a JSP with mock data and developers can later provide implementation of dynamic data.

1. **What is the advantage of JSP over Javascript?**

In JSP we can write Java code seamlessly. It allows for writing code that can interact with the rest of the application.

Javascript code is mostly executed at client side. This limits the tasks that can be done in Javascript code. We cannot connect to database server from Javascript at the client side.

**18. What is the Lifecycle of JSP?**

JSP has following lifecycle stages:

1. **Compilation**: When a request is made for a JSP, thecorresponding JSP is converted into Servlet and compiled. If there is already a compiled form of JSP and there is not change in JSP page since last compilation, this stage does not do anything.
2. **Initialization**: In this stage, jspInit() method is called toinitialize any data or code that will be later used multiple times in \_jspService() method.
3. **Service**: In this stage, with each request to JSP,\_jspService() method is called to service the request. This is the core logic of JSP that generates response for request.
4. **Destroy**: In this stage, JSP is removed from thecontainer/server. Just before removal, this stage performs the cleanup of any resources held by JSP.

**19. What is a JSP expression?**

A JSP expression is an element of a JSP page that is used to evaluate a Java expression and convert into a String. This String is replaced into the locations wherever the expression occurs in JSP page.

E.g. <%= expression =%>

1. **What are the different types of directive tags in JSP?**

JSP has following directive tags:

1. **Page**: This directive is used for page related attributes. Itcan be put anywhere in the JSP page. But by convention we put it on the top of the page.

E.g.

<%@ page attribute="value" %>

1. **Taglib**: We can create custom tags in JSP and use these bytaglib directive in a JSP page.

E.g.

<%@ taglib uri=“abc.html” prefix=“tag\_prefix” >

1. **Include**: We use include directive to read a file and mergeits content with the JSP page. This is done during compilation stage.

<%@ include file="relative url" >

**21. What is session attribute in JSP?**

Session attribute in JSP is used for HTTP session mechanism. If we do not want to use HTTP session in JSP, then we set this attribute to false. If it is set to true, we can use built in session object in JSP.

1. **What are the different scopes of a JSP object?**

A JSP object, implicit or explicit, can have one of the following scopes:

1. **Page**: In this scope, the object is accessible from the pagewhere it was created. Important point here is that when a user refreshes the page, the objects of this scope also get created again.
2. **Request**: In request scope, the object is accessible to theHTTP request that created this object.
3. **Session**: In this scope, the object is available throughoutthe same HTTP session.
4. **Application**: This is the widest scope. The object isavailable throughout the application in which JSP was created.

**23. What is pageContext in JSP?**

In JSP, pageContext is an implicit object. This is used for storing and accessing all the page scope objects of JSP.

It is an instance of the PageContext class from javax.servlet.jsp package.

1. **What is the use of jsp:useBean in JSP?**

We use jsp:useBean to invoke the methods of a Java Bean class. The Java Bean class has some data and setter/getters to access the data.

With this tag, container will try to locate the bean. If bean is not already loaded then it will create an instance of a bean and load it. Later this bean can be used in expressions or JSP code.

1. **What is difference between include Directive and include Action of JSP?**

Some of the main differences between include Directive and include Action are as follows:

1. Include directive is called at translation phase to include content in JSP. Include Action is executed during runtime of JSP.
2. It is not possible to pass parameters to include directive. Include action can accept parameters by jsp:param tag.
3. Include directive is just copying of content from another file to JSP code and then it goes through compilation. Include action will dynamically process the resource being called and then include it in the JSP page.

1. **How will you use other Java files of your application in JSP code?**

We can use import tag to import a Java file in JSP code. Once a file is imported, it can be used by JSP code. It is a very convenient method to use Java classes in JSP code.

For better organization of Java code, we should create a package of classes that we are planning to use in JSP code.

1. **How will you use an existing class and extend it to use in the JSP?**

We can use extends attribute in include tag to use an existing class and extend it in the current JSP.

E.g.

<%@ include page extends=“parent\_class” %>

1. **Why \_jspService method starts with \_ symbol in JSP?**

All the code that we write in a JSP goes into \_jspService method during translation phase. We cannot override this method. Where as other lifecycle methods jspInit() and jspDestroy() can be overridden.

It appears that container uses \_ symbol to distinguish the method that cannot be overridden by client code.

**29. Why do we use tag library in JSP?**

At times we want to create a UI framework with custom tags. In such a scenario, taglib is a very good feature of JSP. With taglib we can create tags that can provide custom features.

Taglib is also a nice way to communicate with UI designers who can use custom tags in the html without going into the details of how the code is implemented.

Another benefit of taglib is reusability of the code. This promotes writing code only once and using is multiple times.

1. **What is the different type of tag library groups in JSTL?**

JSTL stands for JavaServer Pages Standard Tag Library. In JSTL, we have a collection of JSP tags that can be used in different scenarios. There are following main groups of tags in JSTL:

1. Core tags
2. SQL tags
3. Formatting tags
4. XML tags
5. JSTL Functions

1. **How will you pass information from one JSP to another JSP?**

We can pass information from one JSP to another by using implicit objects. If different JSP are called in same session, we can use session object to pass information from one JSP to another.

If we want to pass information from one JSP to another JSP included in the main JSP, then we can use jsp:param to pass this information.

1. **How will you call a stored procedure from JSP?**

JSP allows running Java code from a .jsp file. We can call a stored procedure by using JDBC code.

We can call a CallableStatement from JSP code to invoke a stored procedure.

If we are using Spring framework, then we can use JdbcTemplate class to invoke stored procedure from a JSP.

1. **Can we override \_jspService() method in JSP?**

No, JSP specification does not allow overriding of \_jspService method in JSP. We can override other methods like jspInit() and jspDestroy().

**34. What is a directive in JSP?**

JSP directive is a mechanism to pass message to JSP container. JSP directive does not produce an output to the page. But it communicates with JSP container.

E.g. <%@include ..%> directive is used for telling JSP container to include the content of another file during translation of JSP.

There can be zero or more attributes in a directive to pass additional information to JSP container.

Some of the important directives in JSP are: page, include and taglib.

1. **How will you implement Session tracking in JSP?**

We can use different mechanisms to implement Session tracking

JSP. Some these mechanisms are as follows:

1. **Cookies**: We can use cookie to set session information andpass it to web client. In subsequent requests we can use the information in cookie to track session.
2. **Hidden Form Field**: We can send session id in a hiddenfield in HTML form. By using this we can track session.
3. **Session object**: We can use the built in session object totrack session in JSP.
4. **URL Rewriting**: We can also add session id at the end ofa URL.

Like- [www.abcserver.com?sessionid=1234](http://www.abcserver.com?sessionid=1234)

**36. How do you debug code in JSP?**

In simplest form we can write logger statements or System.out.println() statements to write messages to log files. When we call a JSP, the log messages get written to logs. With useful information getting logged we can easily debug the code.

Another option in debugging is to link JSP container with an IDE. Once we link IDE debugger to JSP Engine, we can use standard operations of debugging like breakpoint, step through etc.

1. **How will you implement error page in JSP?**

To implement an error-handling page in JSP, we first create a JSP with error page handling information. In most of the cases we gracefully handle error by giving a user-friendly message like “Sorry! There is system error. Please try again by refreshing page.”

In this error page, we show user-friendly message to user, but we also log important information like stack trace to our application log file.

We have to add parameter isErrorPage=true in page directive of this page. This tells to JSP container that this is our error page.

<%@page isErrorPage=”true” %>

Now we can use this error page in other JSP where we want to handle error. In case of an error or exception, these JSP will direct it to errorPage.

<% page errorPage=”ErrorPage.jsp” %>

1. **How will you send XML data from a JSP?**

In general, JSP is used to pass HTML data to web browser. If we want to send data in XML format, we can easily do it by setting contentType=”text/xml” in page directive.

E.g. <%@page contentType=”text/xml” %>

1. **What happens when we request for a JSP page from web browser?**

When a user calls JSP page from web browser, the request first comes to web server. Web server checks for .jsp extension of page and passes the request to JSP container like Tomcat.

The JSP container checks whether it has precompiled JSP class or not. If this is the first time this JSP is called, then JSP container will translate JSP into a servlet and compiles it.

After compiling, JSP code if loaded in memory and JSP container will call jspInit() method and \_jspService() methods.

The \_jspService() method will create the output that will be sent by JSP container to client browser.

1. **How will you implement Auto Refresh of page in JSP?**

We can use setIntHeader() method to set the refresh frequency with which we want to auto-refresh a JSP page.

We can send key “Refresh” with the time in seconds for auto refresh of the JSP page.

E.g. response.setIntHeader(“Refresh”,10)

1. **What are the important status codes in HTTP?**

Every HTTP request comes back with a status code from the server.

The important status codes in HTTP are as follows:

1. 200: It means the request is successful.
2. 400: It means the request was bad.
3. 401: It means request was not authorized.
4. 404: It means the resource requested was not found.
5. 503: It means the service is not available.

1. **What is the meaning of Accept attribute in HTTP header?**

In HTTP header, Accept attribute is used to specify the MIME types that a HTTP client or browser can handle. MIME type is the identifier for specifying the type of file/data that we are planning to pass over the internet.

1. **What is the difference between Expression and Scriptlet in JSP?**

We use Expression in a JSP to return a value and display it at a specific location. It is generally used for dynamically print information like- time, counter etc in a HTML code.

Scriptlet is for writing Java code in a JSP. We can define variable, methods etc in a Scriptlet. A Scriptlet can handle much more complex code and can be also reused.

1. **How will you delete a Cookie in JSP?**

We can use following options to delete a Cookie in JSP:

1. **setMaxAge**(): we can set the maximum age of a cookie.After this time period, Cookie will expire and will be deleted.
2. **Header**: We can also set the expiry time in header ofresponse. Respone.setHeader(). This will also expire the cookie after specified time period.

**45. How will you use a Cookie in JSP?**

We can use a Cookie in JSP by performing following steps:

First we create a Cookie object. We set the name and value of the cookie to be created.

We set the expiry time of the Cookie by setting the maximum age.

We can use setMaxAge() method for this.

Finally, we can send the cookie in a HTTP Response by sending it in HTTP header. In this way cookie goes to client browser and gets stored there till the maximum age is not achieved.

Once a Cookie is set in the client browser, we can call getCookies() method to get the list of all the cookies set in Client. We iterate through the list of all the cookies and get the value of the cookie that was set in earlier request.

In this way we can use Cookie to set some information at client side and retrieve its value.

1. **What is the main difference between a Session and Cookie in JSP?**

A Session is always stored at the Server side. In JSP, session is a built-in object in JSP container.

A Cookie is always stored at the client side.

We can use both the methods for Session tracking. But Cookie method needs permission from user for storing cookie at the client location.

1. **How will you prevent creation of session in JSP?**

We can simply set the session attribute as false in page directive to prevent creation of session object.

E.g. <% @page session=”false” %>

1. **What is an output comment in JSP?**

We can write output in JSP in such a way that it becomes a comment in HTML code. This comment will not be visible in the web browser. But when we view page source to see HTML, we can see output comment.

An HTML comment is of following format:

<!-- comment -->

If we output comment in above format, it will be visible to client.

1. **How will you prevent caching of HTML output by web browser in JSP?**

We can use set the header in response object for Cache-Control to specify no caching.

Sample code is as follows:

response.setHeader(“Cache-Control”, “no-store”); response.setDateHeader(“Expires”,”0”);

1. **How will you redirect request to another page in browser in JSP code?**

We can use sendRedirect() method in JSP to redirect the request to another location or page.

In this case the request will not come back to server. It will redirect in the browser itself.

Sample code is as follows:

<% response.sendRedirect(URL); %>

1. **What is the difference between sendRedirect and forward in a JSP?**

Both forward and sendRedirect are mechanisms of sending a client to another page. The main difference between these two are as follows:

1. In forward, the processing takes place at server side. In case of sendRedirect() the processing takes place the client side.
2. In forward, the request is transferred to another resource within same server. In case of sendRedirect the request can be transferred to resource on some other server.
3. In forward only one request call is consumed. In case of sendRedirect two request response calls are created and consumed.
4. The forward is declared in RequestDispatcher interface.

Where as sendRedirect is declared in HttpServletResponse object.

1. **What is the use of config implicit object in JSP?**

In JSP, config object is of type ServletConfig. This object is created by Servlet Container for each JSP page. It is used for setting initialization parameters for a specific JSP page.

1. **What is the difference between init-param and context-param?**

We can specify both init-param and context-param in web.xml file.

We use init-param to specify the parameters that are specific to a servlet or jsp. This information is confined to the scope of that JSP.

We use context-param to specify the parameters for overall application scope. This information does not change easily. It can be used by all the JSP/Servlet in that Container.

1. **What is the purpose of RequestDispatcher?**

We use RequestDispatcher interface to forward requests to other resources like HTML, JSP etc.

It can also be used to include the content of another page in a JSP.

It has two methods: forward and include.

We have to first get the RequestDispatcher object from the container and then we can call include or forward method on this object.

1. **How can be read data from a Form in a JSP?**

There is a built-in request object in a JSP that provides methods to read Form data. Some of the methods are as follows::

1. **getParameterNames():** This method returns the list of allthe parameters in the Form.
2. **getParameter():** We call this method to get the value ofparameter set in the Form. It returns null if the parameter is not found.
3. **getParameterValues():** If a Parameter is mentioned

multiple times in a Form, we use request.getParameterValues() method to get all the values. This method returns an array of String values.

1. **getParameterMap():** This method returns the map of allthe Parameters in Form.

**56. What is a filter in JSP?**

We can define filters in JSP to intercept requests from a client or to change response from a server.

Filter is a Java class that is defined in the deployment descriptor of web.xml of an application. The JSP container reads filter from web.xml and applies a filter as per the URL pattern associated with the filter.

JSP Engine loads all the filters in when we start the server.

1. **How can you upload a large file in JSP?**

To upload a file by JSP we can use <input type=”file”> in the Form data being passed from HTML.

If the file is very large in size, we can set enctype=multipart/form-data.

We have to use POST method in the Form to send a file.

Once the request is received, we can implement the logic to read mulitpart data in doPost() method of JSP. There are methods in JSP framework to read large files via this method.

1. **In which scenario, Container initializes multiple JSP/Servlet objects?**

To initialize multiple JSP objects, we have to specify same Servlet object multiple times in web.xml.

This indicates to JSP container to initialize separate JSP/Servlet object for each element. Each of the Servlet instance will have its own ServletConfig object and parameters.