1. Why main method is static? Can we execute a program without main() method? If yes, how?

Yes, we can execute a java program without a main method by using a static block. Static block in Java is a group of statements that gets executed only once when the class is loaded into the memory by Java ClassLoader, It is also known as a static initialization block.

2. What if the static modifier is removed from the signature of the main method?

If you don't add the 'static' modifier in your main method definition, the compilation of the program will go through without any issues but when you'll try to execute it, a "NoSuchMethodError" error will be thrown.

3. Why static method cannot be overridden in Java?

Static methods cannot be overridden because they are not dispatched on the object instance at runtime. The compiler decides which method gets called. Static methods can be overloaded (meaning that you can have the same method name for several methods as long as they have different parameter types)

4. Explain Autoboxing. When do autoboxing and unboxing occur in Java?

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

5. When do you favor abstract class over the interface?

Abstract classes should be used primarily for objects that are closely related, whereas interfaces are best suited for providing a common functionality to unrelated classes. Interfaces are a good choice when we think that the API will not change for a while

6. How are interfaces useful, and how are they abused? Narrate one or two ways you've used them.

7. Why do you need marker interface in Java?

Marker interface is used as a tag to inform a message to the Java compiler so that it can add special behaviour to the class implementing it. ... When a Java class is to be serialized, you should intimate the Java compiler in some way that there is a possibility of serializing this java class

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

Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.

9. Why is string immutable in Java?

Being immutable automatically makes the String thread safe since they won't be changed when accessed from multiple threads. Hence immutable objects, in general, can be shared across multiple threads running simultaneously.

10. Explain the scenarios to choose between String, StringBuilder, and StringBuffer?

1) The String object is immutable in Java but StringBuffer and StringBuilder are mutable objects.

2) StringBuffer is synchronized while StringBuilder is not which makes StringBuilder faster than StringBuffer.

3) Concatenation operator "+" is internally implemented using either StringBuffer or StringBuilder

11. What are the benefits of immutable object? Why do you use them, and in which scenarios should you use them?

Advantages of immutable objects:

An immutable object remains in exactly one state, the state in which it was created. ...

Immutable classes are easier to design, implement, and use than mutable classes.

Immutable objects are good Map keys and Set elements, since these typically do not change once created

12. Can you explain what does substring method do?

Method substring() returns a new string that is a substring of given string. Java String substring() method is used to get the substring of a given string based on the passed indexes.

13. Why compareTo() should be consistent to equals() method in Java?

a.equals(b)=>Compares values for equality. Because this method is defined in the Object class, from which all other classes are derived, it's automatically defined for every class. However, it doesn't perform an intelligent comparison for most classes unless the class overrides it. It has been defined in a meaningful way for most Java core classes. If it's not defined for a (user) class, it behaves the same as ==.It turns out that defining equals() isn't trivial; in fact it's moderately hard to get it right, especially in the case of subclasses

Comparable interface. Compares values and returns an int which tells if the values compare less than, equal, or greater than. If your class objects have a natural order, implement the Comparable<T> interface and define this method.

14. Will two object always be equal when their compareTo() method returns zero?

Now, if Comparable returns Zero, it means two objects are the same by comparison. If two objects are the same by using the equals method, it returns true.

15. How do you avoid NullPointerException while comparing two Strings in Java?

1) Call equals() and equalsIgnoreCase() method on known String literal rather unknown object.

2) Prefer valueOf() over toString() where both return same result.

3) Use of annotation @NotNull and @Nullable.

16. When do Double and BigDecimal give different answers for equals() and compareTo() == 0?

17. What is the purpose of default constructor? When must developer provide constructor explicitly?

The compiler automatically provides a public no-argument constructor for any class without constructors. This is called the default constructor. If we do explicitly declare a constructor of any form, then this automatic insertion by the compiler won't occur.

18. Is it possible to call a constructor from another (within the same class, not from a subclass)? If yes, how?

The compiler automatically provides a public no-argument constructor for any class without constructors. This is called the default constructor. If we do explicitly declare a constructor of any form, then this automatic insertion by the compiler won't occur.

19. Why do you use threading? What are some of the drawbacks of threading?

20. Tell me about frameworks you've used to manage threading?

21. How do you manage data that needs to travel across threads?

You should use volatile keyword to keep the variable updated among all threads. Using volatile is yet another way (like synchronized, atomic wrapper) of making class thread safe. Thread safe means that a method or class instance can be used by multiple threads at the same time without any problem.

22. When InvalidMonitorStateException is thrown? Why?

This exception is thrown when you try to call wait()/notify()/notifyAll() any of these methods for an Object from a point in your program where u are NOT having a lock on that object.

23. How to detect a Deadlock in Java?

There is one more method to detect Deadlock in Java, it can be done by running the program in CMD. All we need to do is collect thread dumps and then we have to command to collect, depending upon the operating system. If we are running Java 8 on windows, a command would be jcmd $PID Thread. print

24. How will you take a thread dump in Java? How will you analyze a thread dump?

The task on the left indicates the list of currently running processes. Click on the process for which you want the information, and select the thread tab to check the thread information in real time. Click the Thread Dump button on the top right corner to get the thread dump file.

25. How can you trace whether the application has a thread leak?

If an application has thread leak then with time it will have too many unused threads. ... Give unique and descriptive names to the threads created in application. - Add log entry in all thread at various entry and exit points in threads. Change debugging config levels (debug, info, error etc) and analyze log messages.

26. Is synchronization an overhead? If yes, how?

Any time one task spends waiting for another is considered synchronization overhead. ... This waiting time is called synchronization overhead. For distributed memory programs, MPI profiling tools can help to identify places where there are synchronization issues.

27. What is busy spinning? Why should you use it?

Busy-waiting or spinning is a technique in which a process repeatedly checks to see if a condition is true instead of calling wait or sleep method and releasing CPU. It's often done to avoid losing data in CPU cached which is lost if the thread is paused and resumed in some other core

28. How busy spinning is useful and how can it be implemented in java in a multi-threaded environment?

Busy spinning or busy wait in a multi-threaded environment is a technique where other threads loop continuously waiting for a thread to complete its task and signal them to start.

while(spinningFlag){

System.out.println("Waiting busy spinning");

}

// Reached here means spinningFlag is false… Now thread can start

Impact of Busy Spinning on performance

Busy spinning is wasteful of CPU cycles as thread just keep running in a loop unless the condition given in the loop satisfies. The main thing to note here is thread doesn't relinquish the CPU control as would be the case if wait(), sleep(), yield() methods are used where the thread gives up the CPU.

29. How to avoid busy spinning in Java?

Solution to avoid busy spin : To avoid busy spin we must use wait() and notify() method . Calling Wait() method will release CPU and let thread to wait for notify() signal by other thread . This waiting strategy will avoid wasteful consumption of CPU cycles as in busy spin.

30. What is the use of volatile keyword? When should we use volatile variable in Java?

The volatile modifier is used to let the JVM know that a thread accessing the variable must always merge its own private copy of the variable with the master copy in the memory. Accessing a volatile variable synchronizes all the cached copied of the variables in the main memory.

31. What is the use of transient keyword?

Transient in Java is used to mark the member variable not to be serialized when it is persisted to streams of bytes. This keyword plays an important role to meet security constraints in Java. It ignores the original value of a variable and saves the default value of that variable data type.

32. What is the difference between CyclicBarrier and CountDownLatch in Java?

CyclicBarrier allows a number of threads to wait on each other, whereas CountDownLatch allows one or more threads to wait for a number of tasks to complete.

In short, CyclicBarrier maintains a count of threads whereas CountDownLatch maintains a count of tasks.

33. Give us scenarios when you will use CyclicBarrier.

The key difference is that CountDownLatch separates threads into waiters and arrivers while all threads using a CyclicBarrier perform both roles.

With a latch, the waiters wait for the last arriving thread to arrive, but those arriving threads don't do any waiting themselves.

With a barrier, all threads arrive and then wait for the last to arrive.

Your latch example implies that all ten people must wait to lift the stone together. This is not the case. A better real world example would be an exam prompter who waits patiently for each student to hand in their test. Students don't wait once they complete their exams and are free to leave. Once the last student hands in the exam (or the time limit expires), the prompter stops waiting and leaves with the tests.

34. Are the wait, notify, and notifyAll methods in Object class?

35. Why isn't thread class final? Why should one extend thread?

Another reason that Thread is not final is that in the early days of Java, overriding run() was considered to be a good design pattern. (I guess, in the days before anonymous classes, it was thought to be "neater" to subclass Thread than to create a free-standing class that implements Runnable .)

36. Why thread communication methods wait(), notify(), and notifyAll() are in Object class?

37. Why wait(), notify() and notifyAll() methods have to be called from synchronized method or block?

38. Why are thread sleep() and yield() methods static?

So since the only thread worth calling yield on is the current thread, they make the method static so you won't waste time trying to call yield on some other thread. This is because whenever you are calling these methods, those are applied on the same thread that is running

39. How can we achieve thread safety in Java?

Java supports multithreading out of the box. This means that by running bytecode concurrently in separate worker threads, the JVM is capable of improving application performance.

Although multithreading is a powerful feature, it comes at a price. In multithreaded environments, we need to write implementations in a thread-safe way. This means that different threads can access the same resources without exposing erroneous behavior or producing unpredictable results. This programming methodology is known as “thread-safety”.

40. What is BlockingQueue? How can we implement Producer-Consumer problem using Blocking Queue?

Java provides a built-in blocking queue data structure in java.util.concurrent package. It was added on JDK with multiple concurrent utilities e.g. CountDownLatch, CyclicBarrier, and Callable and Future classes.

The java.util.concurrent.BlockingQueue is an interface and comes with two ready-made implementations then ArrayLinkedBlockingQueue and LinkedBlockingQueue. As the name suggests, one is backed by an array while other is backed by linked list.

41. What happens if an exception is thrown in finally block? Is the remaining finally executed or not?

Your finally block will not be completed beyond the point where the exception is thrown. If the finally block was executing during the handling of an earlier exception then that first exception is lost

42. Provide an example how inheritance can break encapsulation?

43. Explain how cloning of mutable object works.

The clone() is a tricky method from java.lang.Object class, which is used to create a copy of an Object in Java. The intention of the clone() method is simple, to provide a cloning mechanism, but somehow it's implementation became tricky and has been widely criticized from a long time. Anyway, we will not go to classic debate of clone in Java, at least for now; instead, we will try to learn how clone method works in Java. To be fair, understating cloning mechanism in Java is not easy and even experienced Java programmer fail to explain how cloning of mutable object works, or a difference between deep and shallow copy in Java.

44. Explain scenarios when we use shallow copy and when go for deep copy.

To create the deep copy of an object, you have to override clone method. Shallow copy is preferred if an object has only primitive fields. Deep copy is preferred if an object has references to other objects as fields

45. Why is object clone() method available only to classes that implement Cloneable interface?

Object's clone() method probably just checks it using: this instanceof Cloneable. The reason that the clone() method is defined in the Object class, is because some 'magic' is needed to actually make a clone. First of all, a new object has to be created without the use of a constructor.

46. Which objects should be called immutable?

Immutable objects are simply objects whose state (the object's data) cannot change after construction.

47. Why should you make an object immutable?

Immutable objects are thread-safe so you will not have any synchronization issues. Immutable objects are good Map keys and Set elements, since these typically do not change once created. Immutability makes it easier to parallelize your program as there are no conflicts among objects.

48. If we have a class with all static members, is it immutable?

Yes, Immutable means that once the constructor for an object has completed execution that instance can't be altered.

49. Tell me about different reference types available in Java, e.g. WeakReference, SoftReference or PhantomReference? Why should you use them?

Interview Questions on Collections:

1. Where will you use ArrayList and where will you use LinkedList?

ArrayList provides constant time for search operation, so it is better to use ArrayList if searching is more frequent operation than add and remove operation. The LinkedList provides constant time for add and remove operations. So it is better to use LinkedList for manipulation.

2. What is HashMap? Why do you use it?

HashMap in Java works on hashing principles. It is a data structure which allows us to store object and retrieve it in constant time O(1) provided we know the key. In hashing, hash functions are used to link key and value in HashMap

3. How does get () method of HashMap work in Java?

get() method is used to get the value by its Key. It will not fetch the value if you don't know the Key. When get(K Key) method is called, it calculates the hash code of the Key.

4. When you are writing equals() method, which other method or methods you need to override?

hashcode()

5. What will be the problem if you don't override hashcode() method?

If you don't override hashcode() then the default implementation in Object class will be used by collections. This implementation gives different values for different objects, even if they are equal according to the equals() method

6. Where have you written equals() and hashCode in any of your projects?

almost all of Java programmer are exposed to this, you can point out value objects, Hibernate entities from your domain, where you have overridden equals and hashCode. Always gives examples from your domain and from your project, rather than a trivial example from a test program, because if Interviewer is asking this question, it means he is interested in examples from your domain.

7. Is it possible for equals() to return false, even if contents of two Objects are same?

In java the method public boolean equals(Object obj) is inherited from the Object.class. Since all Java objects inherit (eventually) from Object, they all inherit that method as well. However, the implementation of the method as defined in the Object class is that the equals method will return if and only if the two objects being compared are the same instance.

public class WrappedString {

private final String str = "hello";

}

public void foo() {

WrappedString ws1 = new WrappedString();

WrappedString ws2 = new WrappedString();

System.out.println(ws1.equals(ws2));

}

The output of the above code snippet will be false since ws1 will only be equal to itself (e.g. other references to the same instance since equals is not overridden).

8. What will happen if two different objects have the same hashcode?

When two unequal objects have the same hash value, this causes a collision in the hash table, because both objects want to be in the same slot (sometimes called a bucket). The hash algorithm must resolve such collisions

9. How will you retrieve value object if two keys have the same hashcode?

f two key have the same hashCode (which doesn't mean they are identical), they will be stored in the same linked list inside the HashMap (assuming you are asking about HashMap s), and the value to be returned will be determined be comparing all the keys that have the same hashCode with the requested key (using equals())

10. What happens on HashMap in Java if the size of the HashMap exceeds the given threshold defined by load factor ?

When the number of entries in the hash table exceeds the product of the load factor and the current capacity, the hash table is rehashed (that is, internal data structures are rebuilt) so that the hash table has approximately twice the number of buckets

11. What is ReadWrite Lock? Does ConcurrentHashMapusesReadWrite Lock?

ReadWriteLock is an advanced thread lock mechanism. It allows multiple threads to read a certain resource, but only one to write it, at a time

Part of the map called Segment (internal data structure) is only getting locked while adding or updating the map. So ConcurrentHashMap allows concurrent threads to read the value without locking at all.

12. State implementation of ReadWriteLock with different policies.

ReadWriteLock maintains a pair of associated locks,one for read-only operations and one for writing.The read lock may be held simultaneously by multiple reader threads,so long as there are no writers. The write lock is exclusive.

Reader threads can read shared data simultaneously. A read operation does not block other read operations. This is the case when you execute an SQL SELECT statement. But write operation is exclusive. This means all readers and other writers are blocked when a writer thread holds the lock for modifing shared data.

Interview Questions on J2EE

1. How do you prevent SQL Injection in Java Code? (java EE)

Before substitute into query, we need to do the validation. for remove ir escaped the special character like single quote, key words like select, Union…

Use Prepared Statement with placeholder

PreparedStatement preparedStatement=conn.prepareStatement('SELECT \* FROM usercheck where username=?') ;

preparedStatement.setString(1, user);

that setXXX() method do all the validation and escaping the special charcter

Now if use same blind sql injection like

sdfssd’ or ‘1’=’1 then

1

SELECT \* FROM User where userId='sdfssd\' or \'1\'=\'1'

Here all special character are escaped When we use JPA kind of ORM tools like Hibernate, EclipseLink, TopLink that time also may be sqlinjection is possible.

To prevent the SQL injection we need to use NamedQuery instead of normal Query. Because NamedQuery internally used PreparedStement but normal query used norma Stement in java.

Normal Query in JPA

String q='SELECT r FROM User r where r.userId=''+user+''';

Query query=em.createQuery(q);

List users=query.getResultList();

So don’t use normal query, use Named query like this

Query query=em.createNamedQuery('User.findByUserId');

query.setParameter('userId', user);

List users=query.getResultList();

2. Are you familiar with Session Fixation attack or Cross Site Scripting attack and how to resolve them?

3. Is it good to have instance/static variables in a servlet? Why not? Then where do you store "state"? What & where is a "state" for a (web) application?

we know that there will only be a single instance of the Servlet, and that an instance variable will act like a static variable. Therefore, rather than requiring people to know about the single instance (since many people do not) by making the variable static rather than instance, it removes any confusion in the usage. Therefore the intent of the variable is clearer and less likely to be misunderstood.

4. What happens if I started creating/closing DB connections in "JSP"?

5. What are the ways to handle JSP exceptions? try-catch? Hmmm..is there anything else?

6. Explain the various access modifiers used in Java. I have had lots of people struggle with this, especially default access.

7. If you could change one thing about the Java language or platform what would it be?

8. What is the general contract when overriding equals?

9. Is better option prefer lists or arrays?

10. What are the generally accepted naming conventions?

11. How serialization works?

12. How to implement Comparable?

13. What are the advantages of using JDBC's Prepared Statements?

14. What happens when an MDB encounters an exception that it cannot handle

15. What are some ways for the client to obtain a reference to the DataSource from the app server?

16. What is the difference between SOAP-based web services and REST-based web services?

17. What exactly is REST?

18. What is the purpose of each of the HTTP verbs?

19. What is decoupling? Why are loosely-coupled classes desirable? What are some drawbacks?

20. What is cohesion? Why are highly cohesive classes desirable? What are some drawbacks?

21. Describe polymorphism. What is the importance of contracts between interfaces and concrete types? Why is polymorphic code desirable? What are some drawbacks?

22. If stuck in a situation where accessing a Hibernate/JPA POJO contents leads to exceptions, how would one resolve it?

23. How are dependent libraries packaged into an archive? Or Visibility of various classes in a bundled archive.

24. How could a simple web service be provided?

Interview Questions on Design Pattern:

1. Which design patterns have you used? Explain how you used it with proper example and why you choose a particular pattern?

2. Can you name a few design patterns used in standard JDK library?

3. Which design pattern you will use to shield your code from a third party library, which will likely to be replaced in another couple of years?

4. Do you know about Open Closed Design Principle or Liskov Substitution Principle?

5. What is Singleton design pattern in Java? Write code for thread-safe singleton in Java.

6. When doesn't Singleton remain Singleton in Java?

7. What is lazy and early loading of Singleton and how will you implement it?

8. What is double checked locking in Singleton?

9. How do you prevent for creating another instance of Singleton using reflection?

10. How do you prevent for creating another instance of Singleton during serialization?

11. When to choose Singleton over Static Class?

12. Can you replace Singleton with Static Class in Java? Describe to me a design pattern you used in a past solution and why that approach was appropriate.

13. When using the Observer pattern, what approaches can be used by the subject so that its observers can be more efficiently updated?

14. When and why would you favor the Decorator pattern over inheritance?

15. How does the synchronized keyword make a method thread safe?

16. Name the creational design pattern that is used to build complex objects step by step.

17. What is the role of director in builder pattern implementation?

18. Write an example to show how builder pattern can be applied.

Interview Questions on Hibernate

1. Why is it important to provide a default constructor in Java? What happens if your class doesn't have a no argument constructor?

2. Explain hibernate architecture. (Dig more on this)

3. What are the core interfaces of Hibernate?

4. What role does the Session/SessionFactory interface play in Hibernate?

5. What is the difference between get and load method? When to use them?

6. How do you map Java Objects with Database tables?

7. Give an overview of Hibernate and ORM.

8. How do you load objects into the session?

9. What does the session do with the objects while in the session?

10. What is the difference between getting a persistent object from the session and querying for persistent objects?

11. When is it better to use plain SQL instead of ORM?

12. Why do we need ORM?

Interview Questions on Spring

1. What are types of IoC containers? Explain them.

2. Why is dependency injection useful?

3. Give an overview of how Spring Dependency Injection container works?

4. Give an example of BeanFactory implementation.

5. What are the common implementations of the ApplicationContext?

6. What is the difference between Bean Factory and ApplicationContext?

7. What is Annotation-based container configuration?

8. How do you turn on annotation wiring?

9. Explain the @Controller annotation.

10. Explain @RequestMapping annotation.

11. What are the differences between Spring MVC and Spring Web Flow?

12. Tell us about the different types of Listener related events?

Interview Questions on JVM:

1. What are the different class loaders used by JVM?

2. Is it possible to load a class by two ClassLoader?

3. How many types of memory areas are allocated by JVM?

4. How JVM Performs Thread Synchronizations?

5. How to Profile Heap Usage?

6. How do you identify minor and major garbage collection in Java?

7. What is the difference between ParNew and DefNew Young Generation Garbage collector?

8. If Object A has reference to Object B and Object B has reference to Object A, apart from that there is no live reference to either object A or B, are they eligible for Garbage collection?

9. Does Garbage collection occur in permanent generation space in JVM?

10. How can you control size of PermGen space?

11. What are the best approaches to deal with OutOfMemeryError?

12. How do you monitor garbage collection activities?

13. Provide a Garbage collection output shown as below and ask following questions based on GC output:

eg.

[GC

[ParNew: 1512K->64K(1512K), 0.0635032 secs]

15604K->13569K(600345K), 0.0636056 secs]

[Times: user=0.03 sys=0.00, real=0.06 secs]

• Is this output of Major Collection or Minor Collection?

• Which young Generation Garbage collector is used?

• What is the size of Young Generation, Old Generation and total Heap Size?

• How much memory is freed from Garbage collection?

• How much time is taken for Garbage collection?

• What is current Occupancy of Young Generation?

1. Should one Pool Object to help Garbage Collector? Should one call System.gc() periodically?

2. Why Java based GUI intensive program has Performance Issues?

3. There is a garbage collector alright, but then is memory leak totally absent in Java applications? If not, how so?

Interview Questions on JDBC

1. What are database warnings, and how can I handle database warnings in JDBC?

2. Connected vs Disconnected RowSet, which one should I use and when?

3. Why do we need a JdbcRowSet like wrapper around ResultSet?

4. How JDBC work with REF CURSOR returned by the stored procedure and retrieve results?

5. What is “dirty read” in JDBC? Which isolation level prevents dirty read?

6. How to invoke Oracle Stored Procedure with Database Objects as IN/OUT?

7. When do we get java.sql.SQLException: No suitable driver found?

8. What are JDBC Best Practices?

Interview Questions for Senior Java Developer:

1. What has been your experience with X, how did you use that?

2. What challenges you had with X and how did you overcome it?

3. Tell me some difficult problems you have solved and please tie that with the technologies you've listed on your resume.

4. Provide a hypothetical architecture arrangement, and ask for insights on design considerations.

5. Ask for an overview of processes when implementing a particular system in such a way your interviewee may demonstrate the "senior" part of the developer position.