GROKKING THE JAVA INTERVIEW

JAVIN PAUL @JAVINPAUL

Table Of Contents

PREFACE

Cracking a Java Interview is not easy and one of the main reason for that is Java is very vast. There are lot of concepts and APIs to master to become a decent Java developer. Many people who are good at general topics like Data Structure and Algorithms, System Design, SQL and Database fail to crack the Java interview because the don't spend time to learn the Core Java concepts and essential APIs and packages like Java Collection Framework, Multithreading, JVM Internals, JDBC, Design Patterns and Object Oriented Programming.

This book aims to fill that gap and introduce you will classical Java interview questions from these topics. By going through these questions and topic you will not only expand your knowledge but also get ready for your Next Java interview. If you are preparing for Java interviews then I highly recommend you go through these questions before your telephonic or face-to-face interviews, you will not only gain confidence and knowledge to answer the question but also learn how to drive Java interview in your favor. This is the single most important tip I can give you as a Java developer. Always, remember, your answers drive interviews, and these questions will show you how to drive Interviewer to your strong areas. All the best for Java interview and if you have any questions or feedback you can always contact me on twitter javinpaul or comment on my blogs Javarevisited and Java67.

OBJECT ORIENTED PROGRAMMING INTERVIEW QUESTIONS

Java is an object-oriented programming language and you will see a lot of object-oriented programming concept questions on Java interviews. The classic questions like the difference between an interface and abstract class are always there but from the last couple of years more sophisticated questions based upon advanced design principles and patterns are also asked to check OOP knowledge of the candidate. Though, Object-oriented programming questions are more popular on Java interviews for 1 to 3 years experienced programmers. It makes sense as well, as these are the programmers who must know the OOP basics like Abstraction, Inheritance, Composition, Class, Object, Interface, Encapsulation, etc.

If you look for Java interview questions for 2 to 4 years experienced programmer, you will find lots of questions based upon OOP fundamentals like Inheritance and Encapsulation but as you gain more experience, you will see questions based on object-oriented analysis and design e.g. code a vending design machine or implement a coffeemaker in Java.

These questions are more difficult and require not only a true understanding of OOP fundamentals but also about SOLID design principles and patterns.

In this section, I am going to share with you some OOPS concept based Java interview questions that I have collected from friends and colleagues and they have seen in various Java interviews on different companies. They are mostly asked at first a few rounds like on screening round or on the telephonic round.

If you are a senior Java developer then you already know answers to this question and I suggest you practice more on object-oriented analysis and design skill i.e. how to do code against a specification. If you are fresher and junior Java developer with 2 to 3 years experience then you must revise these questions, learn if you don't know to do well on your Java Job interviews.

What is method overloading in OOP or Java?

It's one of the oldest OOPS concept questions, I have seen it 10 years ago and still sees it now. When we have multiple methods with the same name but different functionality then it's called method overloading. For example. System.out.println() is overloaded as we have a 6 or 7 println() method each accepting a different type of parameter.

What is the method overriding in OOP or Java?

It's one of the magic of object-oriented programming where the method is chose based upon an object at runtime. In order for method overriding, we need Inheritance and Polymorphism, as we need a method with the same signature in both superclass and subclass. A call to such a method is resolved at runtime depending upon the actual object and not the type o variable.

What is the method of hiding in Java?

When you declare two static methods with same name and signature in both superclass and subclass then they hide each other i.e. a call to the method in the subclass will call the static method declared in that class and a call to the same method is superclass is resolved to the static method declared in the super-class.

Is Java a pure object-oriented language? if not why?

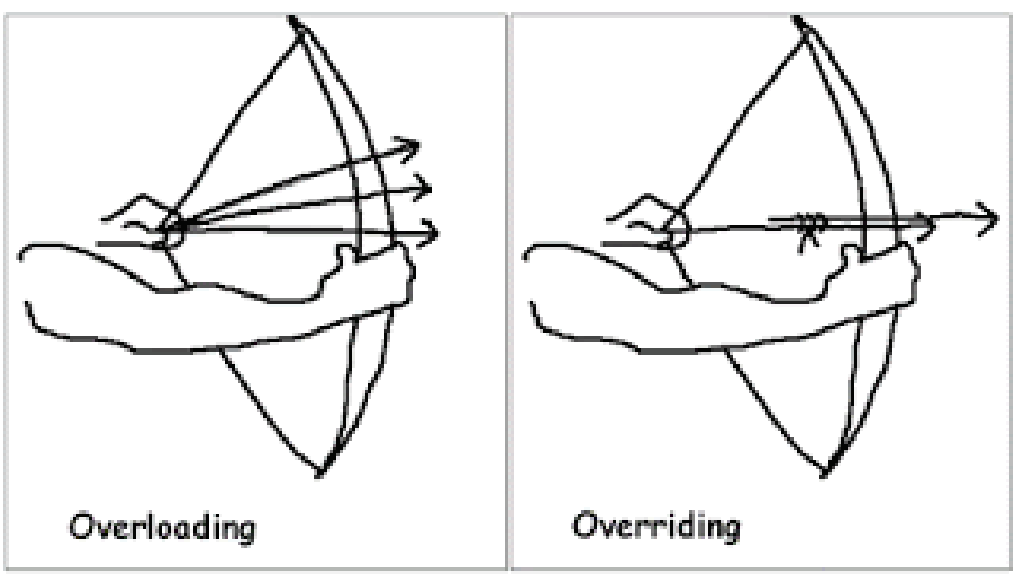
Java is not a pure object-oriented programming language e.g. there are many things you can do without objects e.g. static methods. Also, primitive variables are not objects in Java.

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

One of the most important rules of method overloading in Java is that the method signature should be different i.e. either the number of arguments or the type of arguments. Simply changing the return type of two methods will not result in overloading, instead, the compiler will throw an error. On the other hand, method overriding has more rules e.g. name and return type must be the same, method signature should also be the same, the overloaded method cannot throw a higher exception, etc.

The difference between method overloading and overriding?

Several differences but the most important one is that method overloading is resolved at compile-time and method overriding is resolved at runtime. The compiler only used the class information for method overloading, but it needs to know the object to resolved overridden method calls. This diagram explains the difference quite well, though:



Can we overload a static method in Java?

Yes, you can overload a static method in Java. You can declare as many static methods of the same name as you wish provided all of them have different method signatures.

Can we override the static method in Java?

No, you cannot override a static method because it's not bounded to an object. Instead, static methods belong to a class and resolved at compile time using the type of reference variable. But, Yes, you can declare the same static method in a subclass, that will result in method hiding i.e. if you use the reference variable of type subclass then new method will be called, but if you use the reference variable of superclass than old method will be called.

Can we prevent overriding a method without using the final modifier?

Yes, you can prevent the method overriding in Java without using the final modifier. In fact, there are several ways to accomplish it e.g. you can mark the method private or static, those cannot be overridden.

Can we override a private method in Java?

No, you cannot. Since the private method is only accessible and visible inside the class they are declared, it's not possible to override them in

subclasses. Though, you can override them inside the inner class as they are accessible there.

What is the covariant method overriding in Java?

In the covariant method overriding, the overriding method can return the subclass of the object returned by the original or overridden method. This concept was introduced in Java 1.5 (Tiger) version and it's very helpful in case the original method is returning general type like Object class, because, then by using the covariant method overriding you can return a more suitable object and prevent client-side typecasting. One of the practical use of this concept is when you override the clone() method in Java.

Can we change the return type of method to subclass while overriding?

Yes, you can, but only from Java 5 onward. This feature is known as covariant method overriding and it was introduced in JDK 5 release. This is immensely helpful if the original method return super-class like clone() method return java.lang.Object. By using this, you can directly return the actual type, preventing client-side type-casting of the result.

Can we change the argument list of an overriding method?

No, you cannot. The argument list is part of the method signature and both overriding and overridden

methods must have the same signature.

Can we override a method that throws runtime exception without throws clause?

Yes, there is no restriction on unchecked exceptions while overriding. On the other hand, in the case of checked exception, an overriding exception cannot throw a checked exception which comes higher in type hierarchy e.g. if the original method is throwing IOException than the overriding method cannot throw java.lang.Exception or java.lang.Throwable.

How do you call a superclass version of an overriding method in a subclass?

You can call a superclass version of an overriding method in the subclass by using super keyword. For example to call the toString() method from java.lang.Object class, you can call super.toString().

Can we override a non-static method as static in Java?

Yes, you can override the non-static method in Java, no problem on them but it should not be private or final :)

Can we override the final method in Java?

No, you cannot override a final method in Java, the final keyword with the method is to prevent method

overriding. You use the final when you don't want subclass changing the logic of your method by overriding it due to security reasons. This is why the String class is final in Java. This concept is also used in the template design patterns where the template method is made final to prevent overriding.

Can we have a non-abstract method inside an interface?

From Java 8 onward you can have a non-abstract method inside interface, prior to that it was not allowed as all method was implicitly public abstract. From JDK 8, you can add static and default methods inside an interface.

What is the default method of Java 8?

The default method, also known as the extension method is new types of the method which you can add on the interface now. These method has implementation and intended to be used by default. By using this method, JDK 8 managed to provide common functionality related to lambda expression and stream API without breaking all the clients which implement their interfaces. If you look at Java 8 API documentation you will find several useful default methods on key Java interface like Iterator, Map, etc.

What is an abstract class in Java?

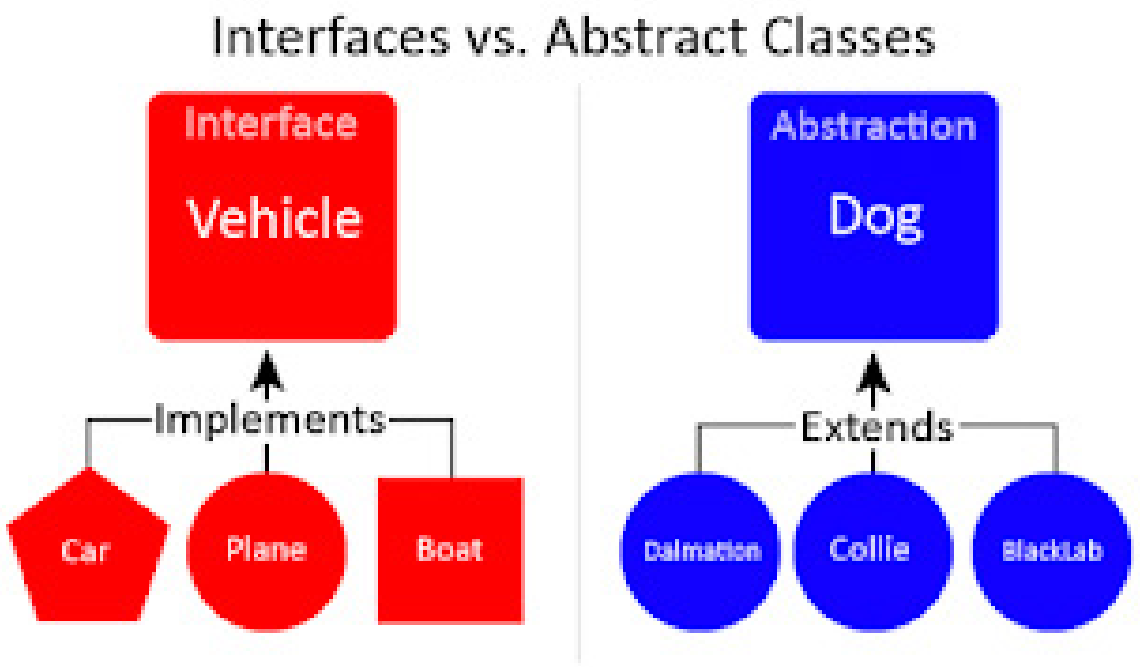
An abstract class is a class that is incomplete. You cannot create an instance of an abstract class in Java. They are provided to define default behavior and ensured that client of that class should adore to those contract which is defined inside the abstract class. In order to use it, you must extend and implement their abstract methods. BTW, in Java, a class can be abstract without specifying any abstract method.

What is an interface in Java? What is the real user of an interface?

Like an abstract class, the interface is also there to specify the contract of an API. It supports the OOP abstraction concept as it defines only abstract behavior. It will tell that your program will give output but how is left to implementors. The real use of the interface to define types to leverage Polymorphism.

The difference between Abstract class and interface?

In Java, the key difference is that abstract class can contain a non-abstract method but the interface cannot, but from Java 8 onward interface can also contain static and default methods that are non-abstract.



Can we make a class abstract without an abstract method?

Yes, just add abstract keyword on the class definition and your class will become abstract.

Can we make a class both final and abstract at the same time?

No, you cannot apply both final and abstract keyword at the class at the same time because they are exactly opposite of each other. A final class in Java cannot be extended and you cannot use an abstract class without extending and make it a concrete class. As per Java specification, the compiler will throw an error if you try to make a class abstract and final at the same time.

Can we overload or override the main method in Java?

No, since main() is a static method, you can only overload it, you cannot override it because the static

method is resolved at compile time without needing object information hence we cannot override the main method in Java.

What is the difference between Polymorphism, Overloading, and Overriding?

This is a slight tricky OOP concept question because Polymorphism is the real concept behind on both Overloading and Overriding. Overloading is compiled time Polymorphism and Overriding are Runtime Polymorphism.

Can an interface extend more than one interface in Java?

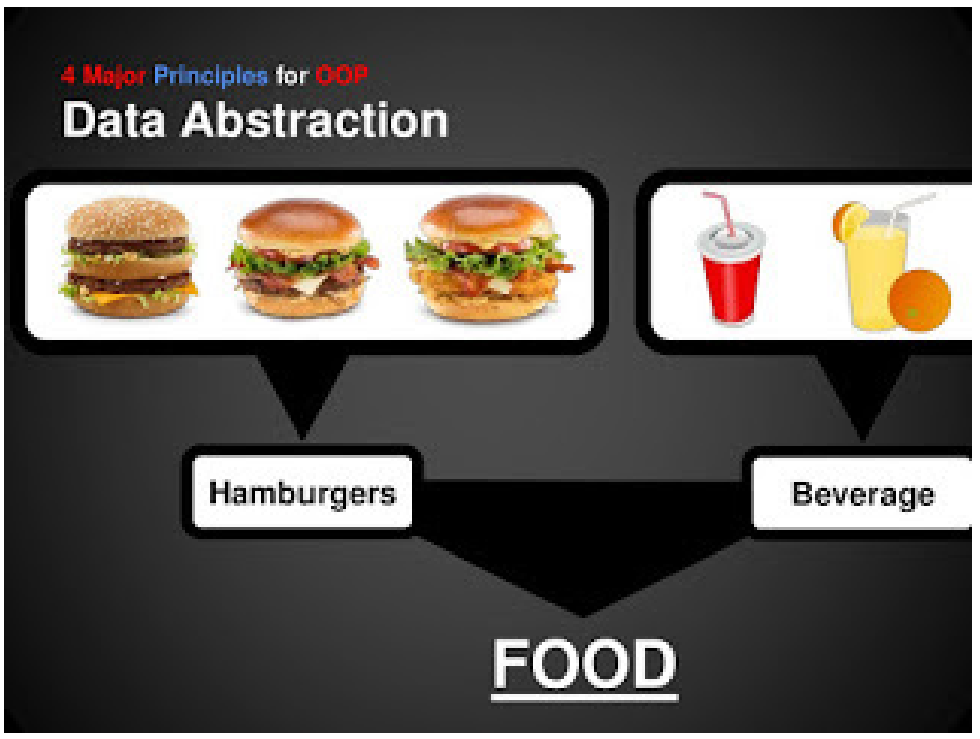
Yes, an interface can extend more than one interface in Java, it's perfectly valid.

Can a class extend more than one class in Java?

No, a class can only extend another class because Java doesn't support multiple inheritances but yes, it can implement multiple interfaces.

What is the difference between abstraction and polymorphism in Java?

Abstraction generalizes the concept and Polymorphism allows you to use different implementation without changing your code. This diagram explains the abstraction quite well, though:



Object-Oriented design principle and pattern Interview Questions

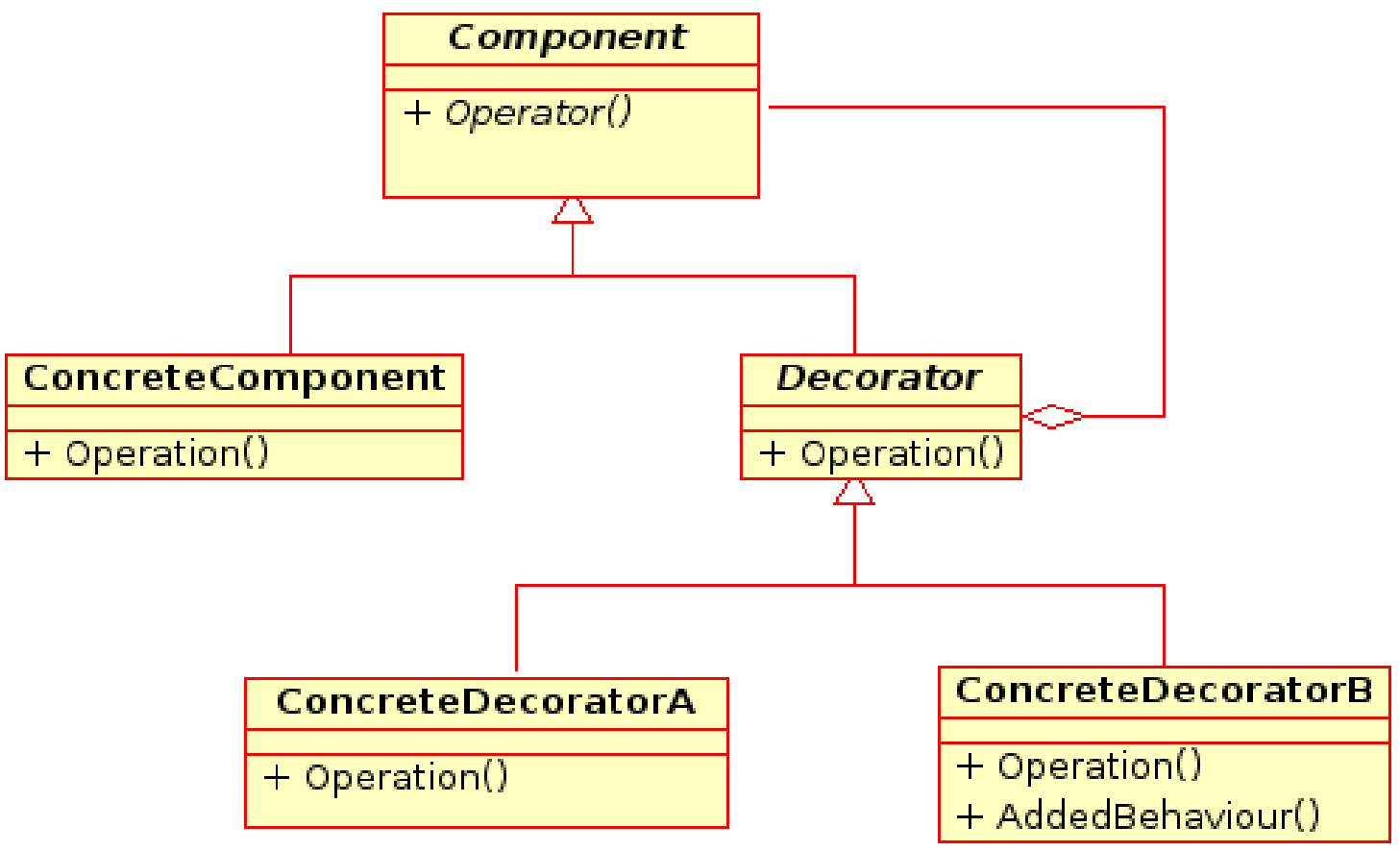
Now let's see some OOPS concept questions based on the SOLID design principles and GOF design patterns that take advantage of the OOPS concept discussed here.

What problem is solved by the Strategy pattern in Java?

Strategy pattern allows you to introduce a new algorithm or new strategy without changing the code which uses that algorithm. For example, the Collections.sort() method which sorts the list of the object uses the Strategy pattern to compare objects. Since every object uses a different comparison strategy you can compare various objects differently without changing the sort method.

Which OOP concept Decorator design Pattern is based upon?

The decorator pattern takes advantage of Composition to provide new features without modifying the original class. A very good to-the-point question for the telephonic round. This is quite clear from the UML diagram of the Decorator pattern, as you can see the Component is associated with a Decorator.



When to use the Singleton design pattern in Java?

When you need just one instance of a class and want that to be globally available then you can use the Singleton pattern. It's not free of cost though because it increases the coupling between classes and makes them hard to test. This is one of the oldest design pattern questions from Java interviews.

What is the difference between State and Strategy Patterns?

Though the structure or class diagram of State and Strategy pattern is the same, their intent is completely different. The state pattern is used to do something specific depending upon state while Strategy allows you to switch between algorithms without changing the code which uses it.

What is the difference between Association, Aggregation, and Composition in OOP?

When an object is related to another object is called association. It has two forms, aggregation, and composition. the former is the loose form of association where the related object can survive individually while later is a stronger form of association where a related object cannot survive individually. For example, the city is an aggregation of people but is the composition of body parts.

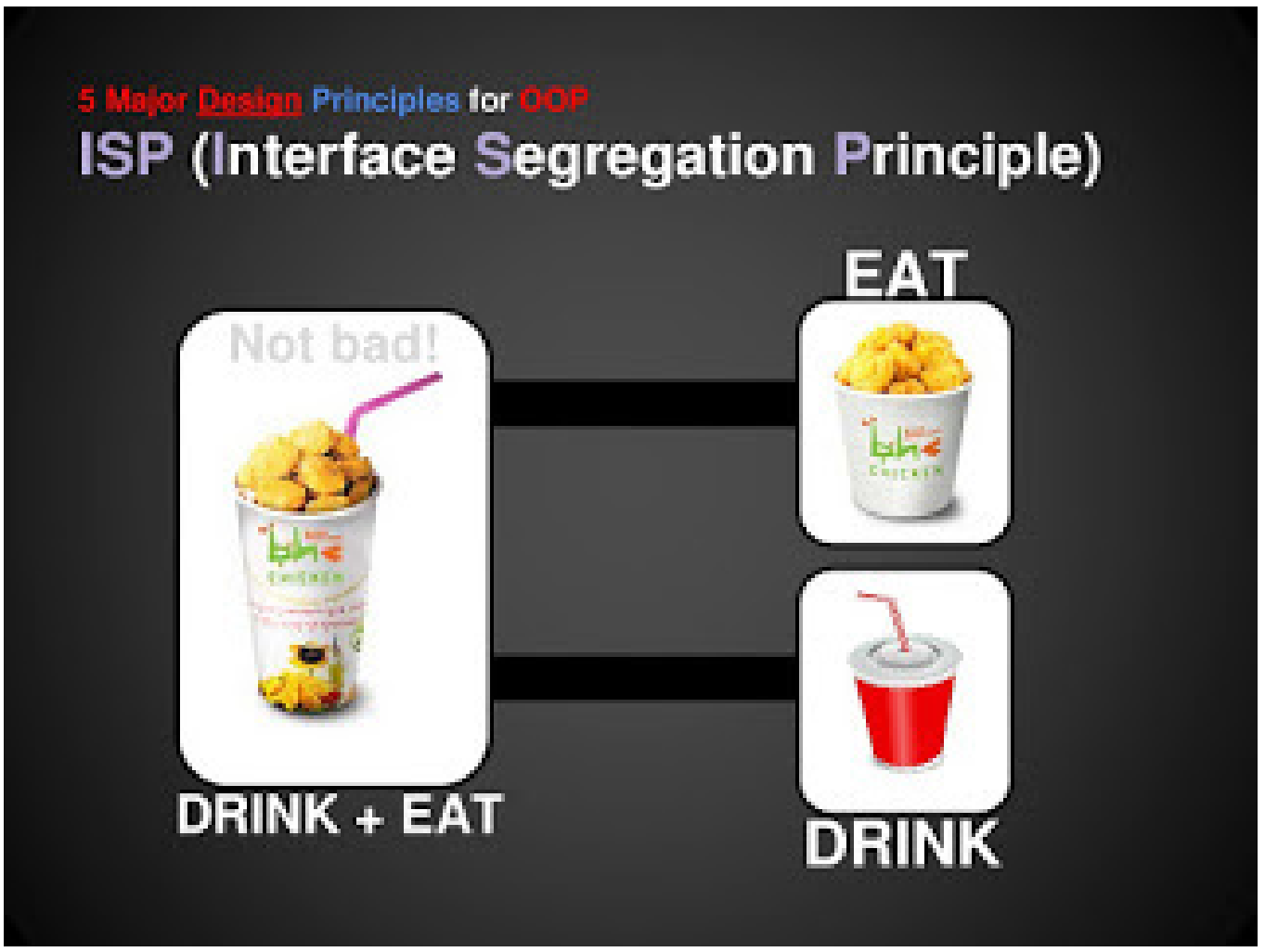
What is the difference between Decorator, Proxy and Adapter pattern in Java?

Again they look similar because their structure or class diagram is very similar but their intent is quite different. The Decorator adds additional functionality without touching the class, Proxy provides access control and Adapter is used to make two incompatible interfaces work together.

What is the 5 objects oriented design principle from SOLID?

SOLID is the term given by Uncle Bob in his classic book, the Clean Code, one of the must-read books for programmers. In SOLID each character stands for one design principle:

- S for Single Responsibility Principle
- O for Open closed design principle
- L for Liskov substitution principle
- I for Interface segregation principle
- D for Dependency inversion principle



What is the difference between Composition and Inheritance in OOP?

This is another great OOPS concept question because it tests what matters, both of them are very important from a class design perspective. Though both Composition and Inheritance allows you to reuse code, formerly is more flexible than later. Composition allows the class to get an additional feature at runtime, but Inheritance is static. You can not change the feature at runtime by substitution of a new implementation.

That's all about in this list of object-oriented programming or OOPS concept interview questions. We have seen questions from various OOPS concepts like Abstraction, Encapsulation, Inheritance, Composition, Aggregation, Association, Class, Object, and Interface, etc.

We have also seen questions from Object-oriented design principles also known as SOLID principles and GOF design patterns like Strategy, State, and Factory, which are based upon both the object-oriented programming concept and OOP design principle.

But, as I said, if you are a senior Java developer then you focus more on object-oriented analysis and design and learn how to code against a requirement using all your OOP knowledge. You can also read Cracking the Coding Interview, 6th Edition to for more Object Oriented Programming questions.

INTERVIEW QUESTIONS FROM TELEPHONIC ROUND

Here is another section about preparing for Java Interviews, this time we will take a look at 40 core Java questions from telephonic round of Java Programming interviews. Phone interviews are usually the first step to screen candidate after selecting his resume. Since it's easy to call a candidate than to schedule a face-to-face interview, book rooms and arrange for a meeting, telephonic round of interviews are quite popular now days. There were days only one telephonic round of interview was enough but nowadays, it's almost two and three round of phone interview with different team members. Key to success in telephonic interview is to the point and concise answer. Since Interviewer wants to cover lot of things on telephonic round, they appreciate to the point answer instead of blah blah and dragging answer for sake of time. Always remember, its your time to make impression also so make sure you have good phone, fully charged and use voice modulation technique to be as clear as possible. Even if you don't know the answer, think a loud, interviewer appreciate ability to think and some time that proves to be decisive as well.

Here is my list of 40 core Java-based questions which frequently appear on telephonic round of Interview. These questions touch-based of important core java concepts e.g. String, Thread basics, multi-threading, inter thread communication, Java Collection framework, Serialization, Object-oriented programming concepts and Exception handling. If you have faced couple of Java interviews, you will sure have seen some of these questions already.

1) Difference between String, StringBuffer and StringBuilder in Java?

String is immutable while both StringBuffer and StringBuilder is mutable, which means any change e.g. converting String to upper case or trimming white space will produce another instance rather than changing the same instance. On later two, StringBuffer is synchronized while StringBuilder is not, in fact its a ditto replacement of StringBuffer added in Java 1.5.

2) Difference between extends Thread vs implements Runnable in Java?

Difference comes from the fact that you can only extend one class in Java, which means if you extend Thread class you lose your opportunity to extend another class, on the other hand if you implement Runnable, you can still extend another class.

3) Difference between Runnable and Callable interface in Java?

Runnable was the only way to implement a task in Java which can be executed in parallel before JDK 1.5 adds Callable. Just like Runnable, Callable also defines a single call() method but unlike run() it can return values and throw exceptions.

4) Difference between ArrayList and LinkedList in Java?

In short, ArrayList is backed by array in Java, while LinkedList is just collection of nodes, similar to linked list data structure. ArrayList also provides random search if you know the index, while LinkedList only allows sequential search. On other hand, adding and removing element from middle is efficient in LinkedList as compared to ArrayList because it only require to modify links and no other element is rearranged.

5) What is difference between wait and notify in Java?

Both wait and notify methods are used for inter thread communication, where wait is used to pause the thread on a condition and notify is used to send notification to waiting threads. Both must be called from synchronized context e.g. synchronized method or block.

6) Difference between HashMap and Hashtable in Java?

Though both HashMap and Hashtable are based upon hash table data structure, there are subtle difference between them. HashMap is non synchronized while Hashtable is synchronized and because of that HashMap is faster than Hashtable, as there is no cost of synchronization associated with it. One more minor difference is that HashMap allows a null key but Hashtable doesn't.

7) Difference between TreeSet and TreeMap in Java?

Though both are sorted collection, TreeSet is essentially a Set data structure which doesn't allow duplicate and TreeMap is an implementation of Map interface. In reality, TreeSet is implemented via a TreeMap, much like how HashSet is implemented using HashMap.

8) Write a Java program to print Fibonacci series?

Core Java Interview Questions answers from telephonic Interview Fibonacci series is a series of number on which a number is equal to sum of previous two numbers i.e. f(n) = f(n-1) + f(n-2). This program is used to teach recursion to students but you can also solve it without recursion. Check out the solution for both iterative and recursive solution of this problem.

In telephonic interview, this question is not that common but sometime interviewer also wants to check your problem solving skill using such questions.

9) Write a Java program to check if a number is Prime or not?

A number is said prime if it is not divisible by any other number except itself. 1 is not considered prime, so your check must start with 2. Simplest solution of this to check every number until the number itself to see if its divisible or not. When Interviewer will ask you to improve, you can say that checking until square root of the number. If you can further improve the algorithm, you will more impress your interviewer.

10) How to Swap two numbers without using temp variable?

This question is ages old. I have first seen this question way back in 2005 but I am sure its even older than that. Good thing about this problem is that except XOR trick all solution has some flaws, which is used to test whether candidate really knows his stuff or not.

11) How to check if linked list contains loop in Java?

This is another problem solving question which is very popular in telephonic and screening round. This

is a great question to test problem solving skill of candidate, especially if he has not seen this question before. It also has a nice little followup to find the starting of the loop.

12) Write Java program to reverse String without using API?

One more question to test problem solving skill of candidate. You wouldn't expect these kind of question in telephonic round of Java interview but these questions have now become norms. All interviewer is looking it for logic, you don't need to write the code but you should be able to think of solution.

13) Difference between Serializable and Externalizable in Java?

Serializable is a marker interface with no methods defined it but Externalizable interface has two methods defined on it e.g. readExternal() and writeExternal() which allows you to control the serialization process. Serializable uses default serialization process which can be very slow for some application.

14) Difference between transient and volatile in Java?

Transient keyword is used in Serialization while volatile is used in multi-threading. If you want to exclude a variable from serialization process then

mark that variable transient. Similar to static variable, transient variables are not serialized. On the other hand, volatile variables are signal to compiler that multiple threads are interested on this variable and it shouldn't reorder its access. volatile variable also follows happens-before relationship, which means any write happens before any read in volatile variable. You can also make non atomic access of double and long variable atomic using volatile.

15) Difference between abstract class and interface?

From Java 8 onwards difference between abstract class and interface in Java has minimized, now even interface can have implementation in terms of default and static method. BTW, in Java you can still extend just one class but can extend multiple inheritance. Abstract class is used to provide default implementation with just something left to customize, while interface is used heavily in API to define contract of a class.

16) Difference between Association, Composition and Aggregation?

Between association, composition and aggregation, composition is strongest. If part can exists without whole than relationship between two class is known as aggregation but if part cannot exists without whole than relationship between two class is known as

composition. Between Inheritance and composition, later provides more flexible design.

17) What is difference between FileInputStream and FileReader in Java?

Main difference between FileInputStream and FileReader is that former is used to read binary data while later is used to read text data, which means later also consider character encoding while converting bytes to text in Java.

18) How do you convert bytes to character in Java?

Bytes are converted to character or text data using character encoding. When you read binary data from a file or network endpoint, you provide a character encoding to convert those bytes to equivalent character. Incorrect choice of character encoding may alter meaning of message by interpreting it differently.

19) Can we have return statement in finally clause? What will happen?

Yes, you can use return statement in finally block, but it will not prevent finally block from being executed. BTW, if you also used return statement in try block then return value from finally block with override whatever is returned from try block.

20) Can you override static method in Java?

No, you cannot override static method in Java because they are resolved at compile time rather than runtime. Though you can declare and define static method of same name and signature in child class, this will hide the static method from parent class, that's why it is also known as method hiding in Java.

21) Difference between private, public, package and protected in Java?

All four are access modifier in Java but only private, public and protected are modifier keyword. There is no keyword for package access, its default in Java. Which means if you don't specify any access modifier than by default that will be accessible inside the same package. Private variables are only accessible in the class they are declared, protected are accessible inside all classes in same package but only on sub class outside package and public variables e.g. method, class or variables are accessible anywhere. This is highest level of access modifier and provide lowest form of encapsulation.

22) 5 Coding best practices you learned in Java?

If you are developing on a programming language for couple of years, you sure knows lots of best practices, by asking couple of them, Interviewer just checking that you know your trade well. Here are my 5 Java best practices :

- Always name your thread, this will help immensely in debugging.
- Use StringBuilder for string concatenation
- Always specify size of Collection, this will save lot of time spent on resizing
- Always declare variable private and final unless you have good reason.
- Always code on interfaces instead of implementation
- Provide dependency to method instead they get it by themselves, this will make your code unit testable.

23) Write a Program to find maximum and minimum number in array?

This is another coding question test problem solving ability of candidate. Be ready for couple of follow up as well depending upon how you answer this question. Simplest way which comes in mind is to sort the array and then pick the top and bottom element.

24) Write a program to reverse Array in place?

Another problem solving question for Java programmers. Key point here is that you need to reverse the array in place, which means you cannot use additional buffer, one or two variable will be fine. Note you cannot use any library code, you need to create your own logic.

25) Write a program to reverse a number in Java?

This is an interesting program for a very junior programmer, right from the college but can sometime puzzle developer with couple of years of experience as well. Most of these developer does very little coding so they found these kind of questions challenging. Here the trick is to get the last digit of the number by using modulus operator (%) and reducing number in each go by using division operator (/).

26) Write a Program to calculate factorial in Java?

Another beginners coding problem, good for telephonic interview because you can differentiate a guy who can write program to the guy who can't. It's also good to see if developer is familiar with both recursive and iterative algorithm and pros and cons of each. You can also ask lots of follow up e.g. how to improve performance of algorithm? Since factorial of a number is equal to number multiplied by factorial of previous number, you can cache those value instead of recalculating them, this will impress your interviewer a bit.

27) What is difference between calling start() and run() method of Thread?

You might have heard this question before, if calling start() method calls the run() method eventually then why not just call the run() method? Well the difference is, start method also starts a new thread. If you call

the run method directly then it will run on same thread not on different thread, which is what original intention would be.

28) Write a Program to solve Producer Consumer problem in Java?

A very good question to check if candidate can write inter thread communication code or not. If a guy can write producer consumer solution by hand and point out critical section and how to protect, how to communicate with thread then he is good enough to write and maintain your concurrent Java program. This is the very minimum requirement for a Java developer and that's why I love this question, it also has several solution e.g. by using concurrent collections like blocking queue, by using wait and notify and by using other synchronizers of Java 5 e.g. semaphores.

29) How to find middle element of linked list in one pass?

Another simple problem solving question for warm up. In a singly linked list you can only traverse in one direction and if you don't know the size then you cannot write a loop to exactly find out middle element, that is the crux of the problem. One solution is by using two pointers, fast and slow. Slower pointer moves 1 step when faster pointer move to 2 steps, causing slow to point to middle when fast is pointing to end of the list i.e. null.

30) What is equlas() and hashCode() contract in Java? Where does it used?

One of the must ask question in Java telephonic interview. If a guy doesn't know about equals() and hashCode() then he is probably not worth pursuing further because its the core of the Java fundamentals. The key point of contract is that if two objects are equal by equals() method then they must have same hashcode, but unequal object can also have same hashcode, which is the cause of collision on hash table based collection e.g HashMap. When you override equals() you must remember to override hashCode() method to keep the contract valid.

31) Why wait and notify methods are declared in Object class?

This question is more to find out how much experience you really have and what is your thinking towards Java API and its design decision. Similar question is why String is immutable in Java? Well, true answer can only be given by Java designers but you can reason something. For example, wait and notify methods are associated with locks which is owned by object not thread, and that's why it make sense to keep those method on java.lang.Object class.

32) How does HashSet works in Java?

HashSet is internally implemented using HashMap in Java and this is what your interviewer wants to hear. He could then quiz you with some common sense based question e.g. how can you use HashMap because its needs two object key and value? what is the value in case of HashSet? Well, in case of HashSet a dummy object is used as value and key objects are the actual element on Set point of view. Since HashMap doesn't allow duplicate key it also follows contract of set data structure to not allow duplicates.

33) What is difference between synchronize and concurrent Collection in Java?

There was time, before Java 1.5 when you only have synchronized collection if you need them in a multi-threaded Java program. Those classes were plagued with several issue most importantly performance because they lock the whole collection or map whenever a thread reads or writes. To address those issue, Java released couple of Concurrent collection classes e.g. ConcurrentHashMap, CopyOnWriteArrayList and BlockingQueue to provide more scalability and performance.

34) What is difference between Iterator and Enumeration in Java?

Main difference is that Iterator was introduced in place of Enumeration. It also allows you to remove

elements from collection while traversing which was not possible with Enumeration. The methods of Iterator e.g. hasNext() and next() are also more concise then corresponding methods in Enumeration e.g. hasMoreElements(). You should always use Iterator in your Java code as Enumeration may get deprecated and removed in future Java release.

35) What is difference between Overloading and Overriding in Java?

Another frequently asked question from telephonic round of Java interviews. Though both overloading and overriding are related with methods of same names but they have different characteristics e.g.overloaded methods must have different method signature than original method but overridden method must have same signature. Also, overloaded methods are resolved at compiled time while overridden methods are resolved at runtime.

36) Difference between static and dynamic binding in Java?

This is usually asked as follow-up of previous question, static binding is related to overloaded method and dynamic binding is related to overridden method. Method like private, final and static are resolved using static binding at compile time but virtual methods which can be overridden are resolved using dynamic binding at runtime.

37) Difference between Comparator and Comparable in Java?

This is one more basic concept, I expect every Java candidate to know. You will deal with them on every Java project. Several core classes in Java e.g. String, Integer implements Comparable to define their natural sorting order and if you define a value class or a domain object then you should also implement Comparable and define natural ordering of your object. Main difference between these two is that, you could create multiple Comparator to define multiple sorting order based upon different attribute of object. Also, In order to implement Comparable you must have access of the class or code, but you can use Comparator without having source code of class, all you need is the JAR file of particular object. That's why Comparator is very powerful to implement custom sorting order and from Java 8 you can do it even more elegantly, as seen here.

38) How do you sort ArrayList in descending order?

You can use Collections.sort() method with reverse Comparator, which can sort elements in the reverse order of their natural order e.g.

```
List<String> listOfString = Arrays.asList("London",
"Tokyo", "NewYork"); Collections.sort(listOfString,
Collections.reverseOrder()); System.out.
println(listOfString); //[Tokyo, NewYork, London]
```

39) What is the difference between PATH and CLASSPATH in Java?

PATH is an environment variable which points to Java binary which is used to run Java programs. CLASSPATH is another environment variable which points to Java class files or JAR files. If a class is not found in CLASSPATH then Java throws ClassNotFoundException.

40) What is the difference between Checked and Unchecked Exception in Java?

Checked exception ensures that handling of the exception is provided and its verified by compiler also, while for throwing unchecked exception no special provision is needed e.g. throws clause. A method can throw unchecked exceptions without any throw clause.

That's all about 40 Core Java Interview Questions from the telephonic round. Don't take a phone interview lightly, its your first chance to impress your prospective employer. Given that you are not seeing your interviewer and just communicating with your voice, it's a little bit different than a face-to-face interview.

So always be calm, relaxed and answer questions to the point and precisely, speak slowly and make sure you are not in a place where your sound echoes like on a close staircase. Having a good phone, head set and

being on good reception area is also very important. I have seen it many times where candidate lost on interview because not able to hear it properly or not able to understand question before answering them. I know some people take telephonic interviews on those secret places but if you do make sure your phone connectivity is enough.

MULTITHREADING AND CONCURRENCY INTERVIEW QUESTIONS

You go to any Java interview, senior or junior, experience or freshers, you are bound to see a couple of questions from the thread, concurrency, and multi-threading. In fact, this built-in concurrency support is one of the strongest points of Java programming language and helped it to gain popularity among the enterprise world and programmers equally. Most of the lucrative Java developer position demands excellent core Java multi-threading skills and experience in developing, debugging, and tuning high-performance low latency concurrent Java applications. This is the reason, it is one of the most sought after skills in Java interviews. The multithreading and concurrency are also hard to master the concept and only good developers with solid experience can effectively deal with concurrency issues.

In a typical Java interview, the Interviewer slowly starts from basic concepts of Thread by asking questions like, why you need threads, how to create threads, which one is a better way to create threads e.g. by extending thread class or implementing Runnable and then slowly goes into Concurrency issues, challenges faced during the development of concurrent Java

applications, Java memory model, higher-order concurrency utilities introduced in JDK 1.5, principles and design patterns of concurrent Java applications, classical multi-threading problems e.g. producer-consumer, dining philosopher, reader-writer or simply bounded buffer problems.

Since its also not enough just to know the basics of threading, you must know how to deal with concurrency problems like deadlock, race conditions, memory inconsistency, and various thread safety-related issues. These skills are thoroughly get tested by presenting various multi-threading and concurrency problems.

Many Java developers are used to only look and read interview questions before going for the interview, which is not bad but you should not be too far away. Also collecting questions and going through the same exercise is too much time consuming, that's why I have created this list of the top 50 Java multi-threading and concurrency related questions, collected from various interviews. I am only going to add new and recent interview questions as and when I am going to discover them.

Though you need good knowledge and solid experience to do well on Java interviews focused on advanced multithreading and concurrency skill, I strongly recommend Java programmers to read Effective Java and Java Concurrency in Practice twice before going to an interview. They do not only help

you to answer questions better but also help you to present your idea clearly.

And, if you are serious about mastering Java multi-threading and concurrency then I also suggest you take a look at the Java Multithreading, Concurrency, and Performance Optimization course by Michael Pogrebinsy on Udemy. It's an advanced course to become an expert in Multithreading, concurrency, and Parallel programming in Java with a strong emphasis on high performance

By the way, I have not provided answers to some questions here, Why? because I expect most of Java developers to know the answers to this question and if not, also answers are widely available by using Google.

Here is our list of top questions from Java thread, concurrency, and multi-threading. You can use this list to prepare well for your Java interview.

1) What is Thread in Java?

The thread is an independent path of execution. It's a way to take advantage of multiple CPU available in a machine. By employing multiple threads you can speed up CPU bound tasks. For example, if one thread takes 100 milliseconds to do a job, you can use 10 threads to reduce that task into 10 milliseconds. Java provides excellent support for multithreading at the language level, and it's also one of the strong selling points.

2) What is the difference between Thread and Process in Java?

The thread is a subset of Process, in other words, one process can contain multiple threads. Two processes runs on different memory spaces, but all threads share the same memory space. Don't confuse this with stack memory, which is different for the different threads and used to store local data to that thread.

3) How do you implement Thread in Java?

At the language level, there are two ways to implement Thread in Java. An instance of java.lang.Thread represents a thread but it needs a task to execute, which is an instance of interface java.lang.Runnable. Since Thread class itself implement Runnable, you can override run() method either by extending Thread class or just implementing Runnable interface.

4) When to use Runnable vs Thread in Java?

This is a follow-up of previous multi-threading interview question. As we know we can implement thread either by extending Thread class or implementing Runnable interface, the question arise, which one is better and when to use one? This question will be easy to answer if you know that Java programming language doesn't support multiple inheritances of class, but it allows you to

implement multiple interfaces. Which means, it's better to implement Runnable then extends Thread if you also want to extend another class e.g. Canvas or CommandListener.

6) What is the difference between start() and run() method of Thread class?

One of trick Java question from early days, but still good enough to differentiate between shallow understanding of Java threading model start() method is used to start newly created thread, while start() internally calls run() method, there is difference calling run() method directly. When you invoke run() as normal method, its called in the same thread, no new thread is started, which is the case when you call start() method.

7) What is the difference between Runnable and Callable in Java?

Both Runnable and Callable represent task which is intended to be executed in a separate thread. Runnable is there from JDK 1.0 while Callable was added on JDK 1.5. Main difference between these two is that Callable's call() method can return value and throw Exception, which was not possible with Runnable's run() method. Callable return Future object, which can hold the result of computation.

8) What is the difference between CyclicBarrier and CountDownLatch in Java?

Though both CyclicBarrier and CountDownLatch wait for number of threads on one or more events, the main difference between them is that you can not re-use CountDownLatch once count reaches to zero, but you can reuse same CyclicBarrier even after barrier is broken.

9) What is Java Memory model?

Java Memory model is set of rules and guidelines which allows Java programs to behave deterministically across multiple memory architecture, CPU, and operating system. It's particularly important in case of multi-threading. Java Memory Model provides some guarantee on which changes made by one thread should be visible to others, one of them is happens-before relationship. This relationship defines several rules which allows programmers to anticipate and reason behaviour of concurrent Java programs. For example, happens-before relationship guarantees :

- Each action in a thread happens-before every action in that thread that comes later in the program order, this is known as program order rule.
- An unlock on a monitor lock happens-before every subsequent lock on that same monitor lock, also known as Monitor lock rule.
- A write to a volatile field happens-before every

subsequent read of that same field, known as Volatile variable rule.

- A call to Thread.start on a thread happens-before any other thread detects that thread has terminated, either by successfully return from Thread.join() or by Thread.isAlive() returning false, also known as Thread start rule.
- A thread calling interrupt() on another thread happens-before the interrupted thread detects the interrupt(either by having InterruptedException thrown, or invoking isInterrupted or interrupted), popularly known as Thread Interruption rule.
- The end of a constructor for an object happens-before the start of the finalizer for that object, known as Finalizer rule.
- If A happens-before B, and B happens-before C, then A happens-before C, which means happens-before guarantees Transitivity.

I strongly suggest reading Chapter 16 of Java Concurrency in Practice to understand Java Memory model in more detail.

10) What is volatile variable in Java?

Volatile is a special modifier, which can only be used with instance variables. In concurrent Java programs, changes made by multiple threads on instance variables is not visible to other in absence of any synchronizers like synchronized keyword or locks. Volatile variable guarantees that a write will happen

before any subsequent read: as stated: “volatile variable rule” in previous question.

11) What is thread-safety? is Vector a thread-safe class?

Thread-safety is a property of an object or code which guarantees that if executed or used by multiple threads in any manner e.g. read vs write it will behave as expected. For example, a thread-safe counter object will not miss any count if same instance of that counter is shared among multiple threads. Apparently, you can also divide collection classes in two category, thread-safe and non-thread-safe. Vector is indeed a thread-safe class and it achieves thread-safety by synchronizing methods which modify state of Vector, on the other hand, its counterpart ArrayList is not thread-safe.

12) What is race condition in Java? Given one example?

Race condition are cause of some subtle programming bugs when Java programs are exposed to concurrent execution environment. As the name suggests, a race condition occurs due to race between multiple threads, if a thread which is supposed to execute first lost the race and executed second, behaviour of code changes, which surface as non-deterministic bugs. This is one of the hardest bugs to find and re-produce because of random nature of racing between threads.

13) How to stop a thread in Java?

I always said that Java provides rich APIs for everything but ironically Java doesn't provide a sure shot way of stopping thread. There was some control methods in JDK 1.0 e.g. stop(), suspend() and resume() which was deprecated in later releases due to potential deadlock threats, from then Java API designers has not made any effort to provide a consistent, thread-safe and elegant way to stop threads. Programmers mainly rely on the fact that thread stops automatically as soon as they finish execution of run() or call() method. To manually stop, programmers either take advantage of volatile boolean variable and check in every iteration if run method has loops or interrupt threads to abruptly cancel tasks.

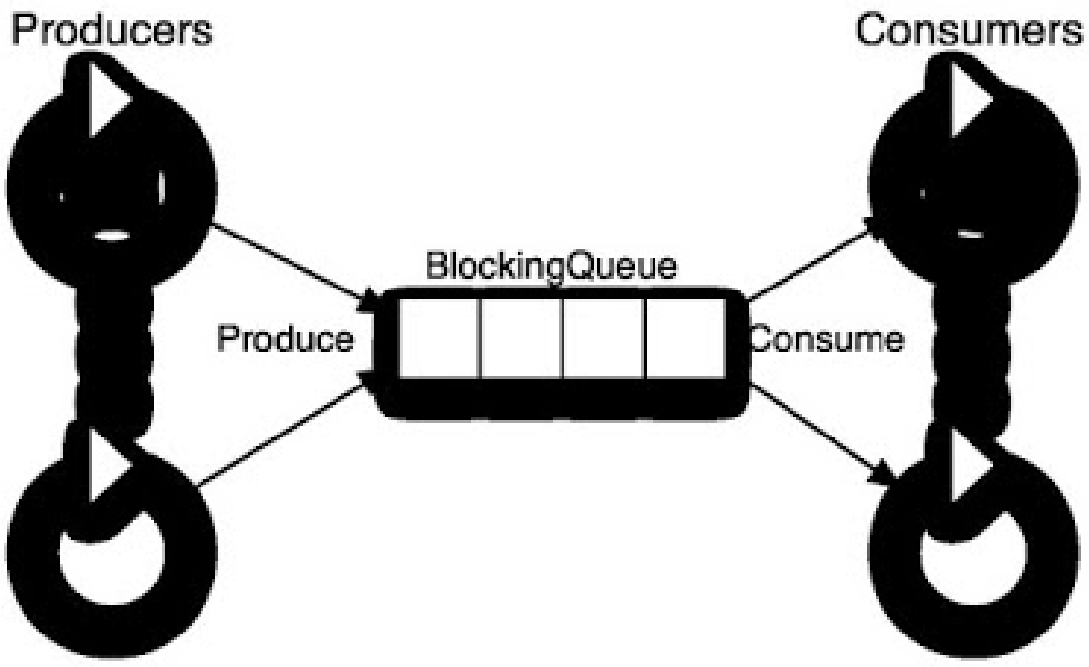
14) What happens when an Exception occurs in a thread?

This is one of the good tricky Java question I have seen in interviews. In simple words, If not caught thread will die, if an uncaught exception handler is registered then it will get a call back. Thread. UncaughtExceptionHandler is an interface, defined as nested interface for handlers invoked when a Thread abruptly terminates due to an uncaught exception. 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.getUncaughtExceptionHandler() and will invoke the handler's uncaughtException() method,

passing the thread and the exception as arguments.

15) How do you share data between two thread in Java?

You can share data between threads by using shared object, or concurrent data structure like BlockingQueue. See this tutorial to learn inter-thread communication in Java. It implements Producer consumer pattern using wait and notify methods, which involves sharing objects between two threads.



16) What is the difference between notify and notifyAll in Java?

This is another tricky questions from core Java interviews, since multiple threads can wait on single monitor lock, Java API designer provides method to inform only one of them or all of them, once waiting condition changes, but they provide half implementation. There notify() method doesn't provide

any way to choose a particular thread, that's why its only useful when you know that there is only one thread is waiting. On the other hand, notifyAll() sends notification to all threads and allows them to compete for locks, which ensures that at-least one thread will proceed further.

17) Why wait, notify and notifyAll are not inside thread class?

This is a design related question, which checks what candidate thinks about existing system or does he ever thought of something which is so common but looks in-appropriate at first. In order to answer this question, you have to give some reasons why it make sense for these three method to be in Object class, and why not on Thread class. One reason which is obvious is that Java provides lock at object level not at thread level. Every object has lock, which is acquired by thread. Now if thread needs to wait for certain lock it make sense to call wait() on that object rather than on that thread. Had wait() method declared on Thread class, it was not clear that for which lock thread was waiting. In short, since wait, notify and notifyAll operate at lock level, it make sense to defined it on object class because lock belongs to object.

18) What is ThreadLocal variable in Java?

ThreadLocal variables are special kind of variable available to Java programmer. Just like instance variable is per instance, ThreadLocal variable is

per thread. It's a nice way to achieve thread-safety of expensive-to-create objects, for example you can make SimpleDateFormat thread-safe using ThreadLocal. Since that class is expensive, its not good to use it in local scope, which requires separate instance on each invocation. By providing each thread their own copy, you shoot two birds with one arrow. First, you reduce number of instance of expensive object by reusing fixed number of instances, and Second, you achieve thread-safety without paying cost of synchronization or immutability. Another good example of thread local variable is ThreadLocalRandom class, which reduces number of instances of expensive-to-create Random object in multi-threading environment.

19) What is FutureTask in Java?

FutureTask represents a cancellable asynchronous computation in concurrent Java application. This class provides a base implementation of Future, with methods to start and cancel a computation, query to see if the computation is complete, and retrieve the result of the computation. The result can only be retrieved when the computation has completed; the get methods will block if the computation has not yet completed. A FutureTask object can be used to wrap a Callable or Runnable object. Since FutureTask also implements Runnable, it can be submitted to an Executor for execution.

20) What is the difference between the interrupted() and isInterrupted() method in Java?

Main difference between interrupted() and isInterrupted() is that former clears the interrupt status while later does not. The interrupt mechanism in Java multi-threading is implemented using an internal flag known as the interrupt status. Interrupting a thread by calling Thread.interrupt() sets this flag. When interrupted thread checks for an interrupt by invoking the static method Thread. interrupted(), interrupt status is cleared. The non-static isInterrupted() method, which is used by one thread to query the interrupt status of another, does not change the interrupt status flag. By convention, any method that exits by throwing an InterruptedException clears interrupt status when it does so. However, it's always possible that interrupt status will immediately be set again, by another thread invoking interrupt

21) Why wait and notify method are called from synchronized block?

Main reason for calling wait and notify method from either synchronized block or method is that it made mandatory by Java API. If you don't call them from synchronized context, your code will throw IllegalMonitorStateException. A more subtle reason is to avoid the race condition between wait and notify calls.

22) Why should you check condition for waiting in a loop?

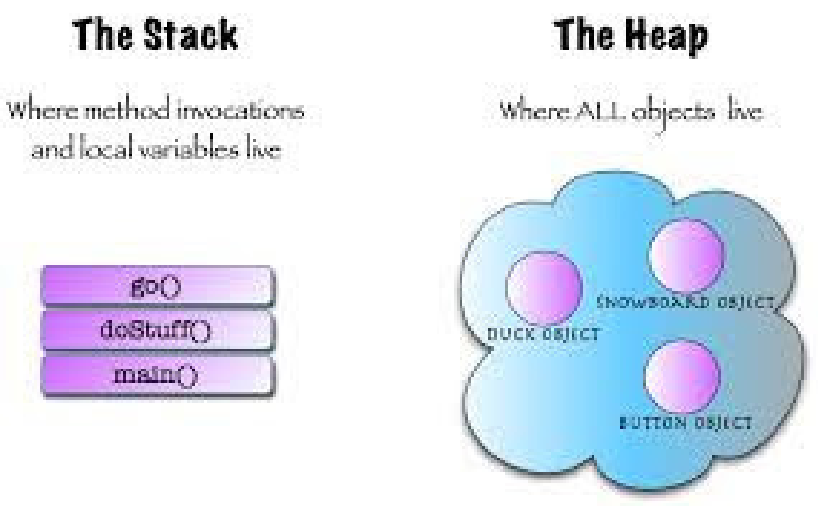
Its possible for a waiting thread to receive false alerts and spurious wake up calls, if it doesn't check the waiting condition in loop, it will simply exit even if condition is not met. As such, when a waiting thread wakes up, it cannot assume that the state it was waiting for is still valid. It may have been valid in the past, but the state may have been changed after the notify() method was called and before the waiting thread woke up. That's why it always better to call wait() method from loop, you can even create template for calling wait and notify in Eclipse. To learn more about this question, I would recommend you to read Effective Java items on thread and synchronization.

23) What is the difference between synchronized and concurrent collection in Java?

Though both synchronized and concurrent collection provides thread-safe collection suitable for multi-threaded and concurrent access, later is more scalable than former. Before Java 1.5, Java programmers only had synchronized collection which becomes source of contention if multiple thread access them concurrently, which hampers scalability of system. Java 5 introduced concurrent collections like ConcurrentHashMap, which not only provides thread-safety but also improves scalability by using modern techniques like lock stripping and partitioning internal table.

24) What is the difference between Stack and Heap in Java?

Why does someone this question as part of multi-threading and concurrency? because Stack is a memory area which is closely associated with threads. To answer this question, both stack and heap are specific memories in Java application. Each thread has their own stack, which is used to store local variables, method parameters and call stack. Variable stored in one Thread's stack is not visible to other. On another hand, the heap is a common memory area which is shared by all threads. Objects whether local or at any level is created inside heap. To improve performance thread tends to cache values from heap into their stack, which can create problems if that variable is modified by more than one thread, this is where volatile variables come into the picture. volatile suggest threads read the value of variable always from main memory.



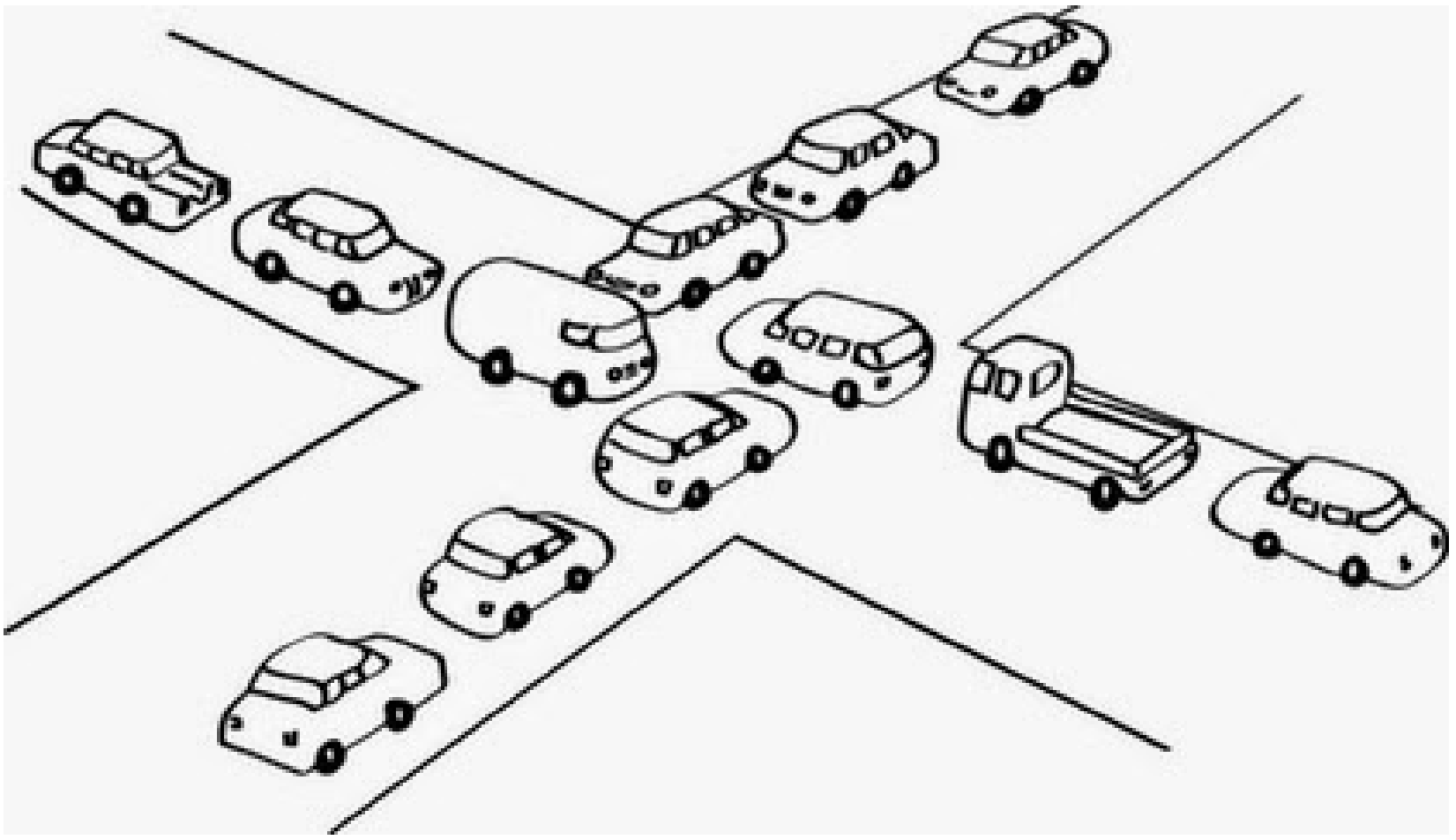
25) What is thread pool? Why should you thread pool in Java?

Creating thread is expensive in terms of time and resource. If you create thread at time of request processing it will slow down your response time, also there is only a limited number of threads a process can create. To avoid both of these issues, a pool of thread is created when application starts-up and threads are reused for request processing. This pool of thread is known as “thread pool” and threads are known as worker thread. From JDK 1.5 release, Java API provides Executor framework, which allows you to create different types of thread pools e.g. single thread pool, which process one task at a time, fixed thread pool (a pool of fixed number of threads) or cached thread pool (an expandable thread pool suitable for applications with many short lived tasks).

26) Write code to solve Producer Consumer problem in Java?

Most of the threading problem you solved in the real world are of the category of Producer consumer pattern, where one thread is producing task and another thread is consuming that. You must know how to do inter thread communication to solve this problem. At the lowest level, you can use wait and notify to solve this problem, and at a high level, you can leverage Semaphore or BlockingQueue to implement Producer consumer pattern.

27) How do you avoid deadlock in Java? Write Code?



Deadlock is a condition in which two threads wait for each other to take action which allows them to move further. It's a serious issue because when it happen your program hangs and doesn't do the task it is intended for. In order for deadlock to happen, following four conditions must be true:

- **Mutual Exclusion** : At least one resource must be held in a non-shareable mode. Only one process can use the resource at any given instant of time.
- **Hold and Wait**: A process is currently holding, at least, one resource and requesting additional resources which are being held by other processes.
- **No Pre-emption**: The operating system must not de-allocate resources once they have been allocated;

they must be released by the holding process voluntarily.
- **Circular Wait**: A process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource.

The easiest way to avoid deadlock is to prevent Circular wait, and this can be done by acquiring locks in a particular order and releasing them in reverse order so that a thread can only proceed to acquire a lock if it held the other one.

28) What is the difference between livelock and deadlock in Java?

This question is extension of previous interview question. A livelock is similar to a deadlock, except that the states of the threads or processes involved in the livelock constantly change with regard to one another, without any one progressing further. Livelock is a special case of resource starvation. A real-world example of livelock occurs when two people meet in a narrow corridor, and each tries to be polite by moving aside to let the other pass, but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time. In short, the main difference between livelock and deadlock is that in former state of process change but no progress is made.

29) How do you check if a Thread holds a lock or not?

I didn't even know that you can check if a Thread already holds lock before this question hits me in a telephonic round of Java interview. There is a method called holdsLock() on java.lang.Thread, it returns true if and only if the current thread holds the monitor lock on the specified object.

30) How do you take thread dump in Java?

There are multiple ways to take thread dump of Java process depending upon operating system. When you take thread dump, JVM dumps state of all threads in log files or standard error console. In windows you can use Ctrl + Break key combination to take thread dump, on Linux you can use kill -3 command for same. You can also use a tool called jstack for taking thread dump, it operate on process id, which can be found using another tool called jps.

31) Which JVM parameter is used to control stack size of a thread?

This is the simple one, -Xss parameter is used to control stack size of Thread in Java. You can see this list of JVM options to learn more about this parameter.

32) What is the difference between synchronized and ReentrantLock in Java?

There were days when the only way to provide mutual exclusion in Java was via synchronized keyword, but it has several shortcomings e.g. you can not extend lock beyond a method or block boundary, you can not give up trying for a lock etc. Java 5 solves this problem by providing more sophisticated control via Lock interface. ReentrantLock is a common implementation of Lock interface and provides re-entrant mutual exclusion Lock with the same basic behavior and semantics as the implicit monitor lock accessed using synchronized methods and statements, but with extended capabilities.

33) There are three threads T1, T2, and T3? How do you ensure sequence T1, T2, T3 in Java?

Sequencing in multi-threading can be achieved by different means but you can simply use the join() method of thread class to start a thread when another one has finished its execution. To ensure three threads execute you need to start the last one first e.g. T3 and then call join methods in reverse order e.g. T3 calls T2. join and T2 calls T1.join, these ways T1 will finish first and T3 will finish last.

34) What does yield method of Thread class do?

Yield method is one way to request current thread to relinquish CPU so that other thread can get a

chance to execute. Yield is a static method and only guarantees that current thread will relinquish the CPU but doesn't say anything about which other thread will get CPU. Its possible for the same thread to get CPU back and start its execution again.

35) What is the concurrency level of ConcurrentHashMap in Java?

ConcurrentHashMap achieves it's scalability and thread-safety by partitioning actual map into a number of sections. This partitioning is achieved using concurrency level. Its optional parameter of ConcurrentHashMap constructor and it's default value is 16. The table is internally partitioned to try to permit the indicated number of concurrent updates without contention.

36) What is Semaphore in Java?

Semaphore in Java is a new kind of synchronizer. It's a counting semaphore. Conceptually, a semaphore maintains a set of permits. Each acquire() blocks if necessary until a permit is available, and then takes it. Each release() adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the Semaphore just keeps a count of the number available and acts accordingly. Semaphore is used to protect an expensive resource which is available in fixed number e.g. database connection in the pool. See this section to learn more about counting Semaphore in Java.

37) What happens if you submit a task when the queue of the thread pool is already filled?

This is another tricky question on my list. Many programmers will think that it will block until a task is cleared but its true. ThreadPoolExecutor's submit() method throws RejectedExecutionException if the task cannot be scheduled for execution.

38) What is the difference between the submit() and execute() method thread pool in Java?

Both methods are ways to submit a task to thread pools but there is a slight difference between them. execute(Runnable command) is defined in Executor interface and executes given task in future, but more importantly, it does not return anything. Its return type is void. On other hand submit() is an overloaded method, it can take either Runnable or Callable task and can return Future object which can hold the pending result of computation. This method is defined on ExecutorService interface, which extends Executor interface, and every other thread pool class e.g. ThreadPoolExecutor or ScheduledThreadPoolExecutor gets these methods.

39) What is blocking method in Java?

A blocking method is a method which blocks until the task is done, for example, accept() method of ServerSocket blocks until a client is connected. here blocking means control will not return to the caller

until the task is finished. On the other hand, there is an asynchronous or non-blocking method which returns even before the task is finished.

40) Is Swing thread-safe? What do you mean by Swing thread-safe?

You can simply this question as No, Swing is not thread-safe, but you have to explain what you mean by that even if the interviewer doesn't ask about it. When we say swing is not thread-safe we usually refer its component, which can not be modified in multiple threads. All update to GUI components has to be done on AWT thread, and Swing provides synchronous and asynchronous callback methods to schedule such updates.

41) What is the difference between invokeAndWait and invokeLater in Java?

These are two methods Swing API provides Java developers for updating GUI components from threads other than Event dispatcher thread. InvokeAndWait() synchronously update GUI component, for example, a progress bar, once progress is made, the bar should also be updated to reflect that change. If progress is tracked in a different thread, it has to call invokeAndWait() to schedule an update of that component by Event dispatcher thread. On another hand, invokeLater() is an asynchronous call to update components.

42) Which method of Swing API are thread-safe in Java?

This question is again related to swing and thread-safety though components are not thread-safe there is a certain method which can be safely called from multiple threads. I know about repaint(), and revalidate() being thread-safe but there are other methods on different swing components e.g. setText() method of JTextComponent, insert() and append() method of JTextArea class.

43) How to create an Immutable object in Java?

This question might not look related to multi-threading and concurrency, but it is. Immutability helps to simplify already complex concurrent code in Java. Since immutable object can be shared without any synchronization its very dear to Java developers. Core value object, which is meant to be shared among thread should be immutable for performance and simplicity. Unfortunately there is no @Immutable annotation in Java, which can make your object immutable, hard work must be done by Java developers. You need to keep basics like initializing state in constructor, no setter methods, no leaking of reference, keeping separate copy of mutable object to create Immutable object.

44) What is ReadWriteLock in Java?

In general, read write lock is the result of lock stripping technique to improve the performance of concurrent applications. In Java, ReadWriteLock is an interface which was added in Java 5 release. A 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. If you want you can implement this interface with your own set of rules, otherwise you can use ReentrantReadWriteLock, which comes along with JDK and supports a maximum of 65535 recursive write locks and 65535 read locks.

45) What is busy spin in multi-threading?

Busy spin is a technique which concurrent programmers employ to make a thread wait on certain condition. Unlike traditional methods e.g. wait(), sleep() or yield() which all involves relinquishing CPU control, this method does not relinquish CPU, instead it the just runs empty loop. Why would someone do that? to preserve CPU caches. In a multi-core system, it's possible for a paused thread to resume on a different core, which means rebuilding cache again. To avoid cost of rebuilding cache, programmer prefer to wait for much smaller time doing busy spin.

46) What is the difference between the volatile and atomic variable in Java?

This is an interesting question for Java programmer, at first, volatile and atomic variable look very similar, but they are different. Volatile variable provides you happens-before guarantee that a write will happen before any subsequent write, it doesn't guarantee atomicity. For example count++ operation will not become atomic just by declaring count variable as volatile. On the other hand AtomicInteger class provides atomic method to perform such compound operation atomically e.g. getAndIncrement() is atomic replacement of increment operator. It can be used to atomically increment current value by one. Similarly you have atomic version for other data type and reference variable as well.

47) What happens if a thread throws an Exception inside synchronized block?

This is one more tricky question for average Java programmer, if he can bring the fact about whether lock is released or not is a key indicator of his understanding. To answer this question, no matter how you exist synchronized block, either normally by finishing execution or abruptly by throwing exception, thread releases the lock it acquired while entering that synchronized block. This is actually one of the reasons I like synchronized block over lock interface, which requires explicit attention to release lock, generally this is achieved by releasing the lock in a finally block.

48) What is double checked locking of Singleton?

This is one of the very popular question on Java interviews, and despite its popularity, chances of candidate answering this question satisfactory is only 50%. Half of the time, they failed to write code for double checked locking and half of the time they failed how it was broken and fixed on Java 1.5. This is actually an old way of creating thread-safe singleton, which tries to optimize performance by only locking when Singleton instance is created first time, but because of complexity and the fact it was broken for JDK 1.4, I personally don't like it. Anyway, even if you not prefer this approach its good to know from interview point of view.

49) How to create thread-safe Singleton in Java?

This question is actually follow-up of the previous question. If you say you don't like double checked locking then Interviewer is bound to ask about alternative ways of creating thread-safe Singleton class. There are actually man, you can take advantage of class loading and static variable initialization feature of JVM to create instance of Singleton, or you can leverage powerful enumeration type in Java to create Singleton.

50) List down 3 multi-threading best practice you follow?

This is my favorite question because I believe that you must follow certain best practices while writing concurrent code which helps in performance, debugging and maintenance. Following are three best practices, I think an average Java programmer should follow:

- **Always give meaningful name to your thread**

This goes a long way to find a bug or trace an execution in concurrent code. OrderProcessor, QuoteProcessor or TradeProcessor is much better than Thread-1. Thread-2 and Thread-3. The name should say about task done by that thread. All major framework and even JDK follow this best practice.

- **Avoid locking or Reduce scope of Synchronization**

Locking is costly and context switching is even costlier. Try to avoid synchronization and locking as much as possible and at a bare minimum, you should reduce critical section. That's why I prefer synchronized block over synchronized method because it gives you absolute control on the scope of locking.

- **Prefer Synchronizers over wait and notify**

Synchronizers like CountDownLatch, Semaphore,

CyclicBarrier or Exchanger simplifies coding. It's very difficult to implement complex control flow right using wait and notify. Secondly, these classes are written and maintained by best in business and there is good chance that they are optimized or replaced by better performance code in subsequent JDK releases. By using higher level synchronization utilities, you automatically get all these benefits.

- **Prefer Concurrent Collection over Synchronized Collection**

This is another simple best practice which is easy to follow but reap good benefits. Concurrent collection are more scalable than their synchronized counterpart, that's why its better to use them while writing concurrent code. So next time if you need map, think about ConcurrentHashMap before thinking Hashtable.

51) How do you force to start a Thread in Java?

This question is like how do you force garbage collection in Java, there is no way though you can make a request using System.gc() but it's not guaranteed. On Java multi-threading there is absolutely no way to force start a thread, this is controlled by thread scheduler and Java exposes no API to control thread schedule. This is still a random bit in Java.

52) What is the fork-join framework in Java?

The fork-join framework, introduced in JDK 7 is a powerful tool available to Java developer to take advantage of multiple processors of modern day servers. It is designed for work that can be broken into smaller pieces recursively. The goal is to use all the available processing power to enhance the performance of your application. One significant advantage of The fork/join framework is that it uses a work-stealing algorithm. Worker threads that run out of things to do can steal tasks from other threads that are still busy.

53) What is the difference between calling wait() and sleep() method in Java multi-threading?

Though both wait and sleep introduce some form of pause in Java application, they are the tool for different needs. Wait method is used for inter-thread communication, it relinquishes lock if waiting for a condition is true and wait for notification when due to an action of another thread waiting condition becomes false. On the other hand sleep() method is just to relinquish CPU or stop execution of current thread for specified time duration. Calling sleep method doesn't release the lock held by the current thread.

You can use this list to not only to prepare for your core Java and programming interviews but also to check your knowledge about basics of threads, multi-threading, concurrency, design patterns and threading

issues like race conditions, deadlock and thread safety problems.

This master list is equally useful to Java developers of all levels of experience. You can read through this list even if you have 2 to 3 years of working experience as a junior developer or 5 to 6 years as a senior developer. It's even useful for freshers and beginners to expand their knowledge.

COLLECTION FRAMEWORK INTERVIEW QUESTIONS

Interview questions from Collection package or framework are most common in any Core Java Interview yet a tricky one. Together Collection and multithreading make any Java interview tough to crack and having a good understanding of Collection and threads will help you to excel in Java interview. I thought about writing interview questions on Java collection framework and important classes like ArrayList, HashMap, Hashtable, and newly added concurrent collections e.g. ConcurrentHashMap when I first wrote 10 multi-threading Interview questions but somehow this section got delayed. Though I have shared several questions individually in between.

I think the main reason for this delay could me spending a lot of time for researching and preparing the mega list of core Java questions from last 5 years. In this section, we will see a mix of some beginners and advanced Java Collection interviews and their answers which have been asked in various Core Java interviews.

These Java Collection framework interview questions have been collected from various friends and colleagues and Answers of these interview questions can also be found by Google. BTW, if you are preparing for Java interview then you can also take help from **Java Programming Interview Exposed**, an excellent resource to do well in Java interviews.

Java Collection interview questions answersNow let's start with interview questions on collections. Since collection is made of various data structures e.g. Map, Set and List and there various implementation, mostly interviewer checks whether interviewee is familiar with basics of these collections or not and whether he knows when to use Map, Set or List. Based on Role for which interview is going on questions starts with beginner's level or more advanced level. Normally 2 to 3 years experience counted as beginners while over 5 years comes under advanced category, we will see questions from both categories.

1. How does HashMap work in Java?

This is Classical Java Collection interview questions which I have also discussed in my earlier section how does HashMap works in Java. This collection interview questions is mostly asked during AVP Role interviews on Investment-Banks and has a lot of follow-up questions based on the response of interviewee e.g.

Why HashMap keys need to be immutable, what is race conditions on HashMap and how HashMap resize in Java.

2. What is the difference between poll() and remove() method of Queue interface?

Though both poll() and remove() method from Queue is used to remove the object and returns the head of the queue, there is a subtle difference between them. If Queue is empty() then a call to remove() method will throw Exception, while a call to poll() method returns null. By the way, exactly which element is removed from the queue depends upon queue's ordering policy and varies between different implementation, for example, PriorityQueue keeps the lowest element as per Comparator or Comparable at head position.

3. What is the difference between fail-fast and fail-safe Iterators?

This is relatively new collection interview questions and can become trick if you hear the term fail-fast and fail-safe first time. Fail-fast Iterators throws ConcurrentModificationException when one Thread is iterating over collection object and other thread structurally modify Collection either by adding, removing or modifying objects on underlying collection. They are called fail-fast because they try to immediately throw Exception when they encounter failure. On the other hand fail-safe Iterators works on copy of collection instead of original collection.

4. How do you remove an entry from a Collection? and subsequently what is the difference between the remove() method of Collection and remove() method of Iterator, which one you will use while removing elements during iteration?

Collection interface defines remove(Object obj) method to remove objects from Collection. List interface adds another method remove(int index), which is used to remove object at specific index. You can use any of these method to remove an entry from Collection, while not iterating. Things change, when you iterate. Suppose you are traversing a List and removing only certain elements based on logic, then you need to use Iterator's remove() method. This method removes current element from Iterator's perspective. If you use Collection's or List's remove() method during iteration then your code will throw ConcurrentModificationException. That's why it's advised to use Iterator remove() method to remove objects from Collection.

5. What is the difference between Synchronized Collection and Concurrent Collection?

Java 5 has added several new Concurrent Collection classes e.g. ConcurrentHashMap, CopyOnWriteArrayList, BlockingQueue etc, which has made Interview questions on Java Collection even

trickier. Java Also provided a way to get Synchronized copy of collection e.g. ArrayList, HashMap by using Collections.synchronizedMap() Utility function.One Significant difference is that Concurrent Collections has better performance than synchronized Collection because they lock only a portion of Map to achieve concurrency and Synchronization.

6. What is the difference between Iterator and Enumeration?

This is a beginner level collection interview questions and mostly asked during interviews of Junior Java developer up to experience of 2 to 3 years Iterator duplicate functionality of Enumeration with one addition of remove() method and both provide navigation functionally on objects of Collection.Another difference is that Iterator is more safe than Enumeration and doesn't allow another thread to modify collection object during iteration except remove() method and throws ConcurrentModificaitonException. See Iterator vs Enumeration in Java for more differences.

7. How does HashSet is implemented in Java, How does it use Hashing?

This is a tricky question in Java because for hashing you need both key and value and there is no key for the store it in a bucket, then how exactly HashSet store element internally. Well, HashSet is built on

top of HashMap. If you look at source code of java.util.HashSet class, you will find that that it uses a HashMap with same values for all keys, as shown below:

```
private transient HashMap map;

// Dummy value to associate with an Object in the
backing Map
private static final Object PRESENT = new Object();

When you call add() method of HashSet, it put entry
in HashMap :

public boolean add(E e) {
  return map.put(e, PRESENT)==null;
}
```

Since keys are unique in a HashMap, it provides uniqueness guarantee of Set interface.

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

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

9. The difference between HashMap and Hashtable?

This is another Classical Java Collection interview asked on beginner's level and most of Java developer has a predefined answer for this interview questions e.g. HashMap is not synchronized while Hashtable is not or hashmap is faster than hash table etc. What could go wrong is that if he placed another follow-up question like how hashMap works in Java or can you replace Hashtable with ConcurrentHashMap etc.

10. When do you use ConcurrentHashMap in Java?

This is another advanced level collection interview questions in Java which normally asked to check whether the interviewer is familiar with optimization done on ConcurrentHashMap or not. ConcurrentHashMap is better suited for situation where you have multiple readers and one Writer or fewer writers since Map gets locked only during the write operation. If you have an equal number of reader and writer than ConcurrentHashMap will perform in the line of Hashtable or synchronized HashMap.

11. What is the difference between Set and List in Java?

Another classical Java Collection interviews popular on telephonic round or the first round of interview. Most of Java programmer knows that Set doesn't allowed

duplicate while List does and List maintains insertion order while Set doesn't. What is key here is to show the interviewer that you can decide which collection is more suited based on requirements.

12. How do you Sort objects on the collection?

This Collection interview question serves two purpose it not only test an important programming concept Sorting but also utility class like Collections which provide several methods for creating synchronized collection and sorting. Sorting is implemented using Comparable and Comparator in Java and when you call Collections.sort() it gets sorted based on the natural order specified in compareTo() method while Collections.sort(Comparator) will sort objects based on compare() method of Comparator.

13. What is the difference between Vector and ArrayList?

One more beginner level collection interview questions, this is still very popular and mostly asked in the telephonic round. ArrayList in Java is one of the most used Collection class and the most interviewers asked questions on ArrayList. See Difference between Vector and ArrayList for the answer to this interview question.

14. What is the difference between HashMap and HashSet?

This collection interview questions is asked in conjunction with HashMap vs Hashtable. HashSet implements java.util.Set interface and that's why only contains unique elements, while HashMap allows duplicate values. In fact, HashSet is actually implemented on top of java.util.HashMap. If you look internal implementation of java.util.HashSet, you will find that it adds element as key on internal map with same values.

15) What is NavigableMap in Java? What is a benefit over Map?

NavigableMap Map was added in Java 1.6, it adds navigation capability to Map data structure. It provides methods like lowerKey() to get keys which is less than specified key, floorKey() to return keys which is less than or equal to specified key, ceilingKey() to get keys which is greater than or equal to specified key and higherKey() to return keys which is greater specified key from a Map. It also provide similar methods to get entries e.g. lowerEntry(), floorEntry(), ceilingEntry() and higherEntry(). Apart from navigation methods, it also provides utilities to create sub-Map e.g. creating a Map from entries of an exsiting Map like tailMap, headMap and subMap. headMap() method returns a NavigableMap whose keys are less than specified, tailMap() returns a NavigableMap whose keys are

greater than the specified and subMap() gives a NavigableMap between a range, specified by toKey to fromKey.

16) Which one you will prefer between Array and ArrayList for Storing object and why?

Though ArrayList is also backed up by array, it offers some usability advantage over array in Java. Array is fixed length data structure, once created you can not change it's length. On the other hand, ArrayList is dynamic, it automatically allocate a new array and copies content of old array, when it resize. Another reason of using ArrayList over Array is support of Generics. Array doesn't support Generics, and if you store an Integer object on a String array, you will only going to know about it at runtime, when it throws ArrayStoreException. On the other hand, if you use ArrayList, compiler and IDE will catch those error on the spot. So if you know size in advance and you don't need re-sizing than use array, otherwise use ArrayList.

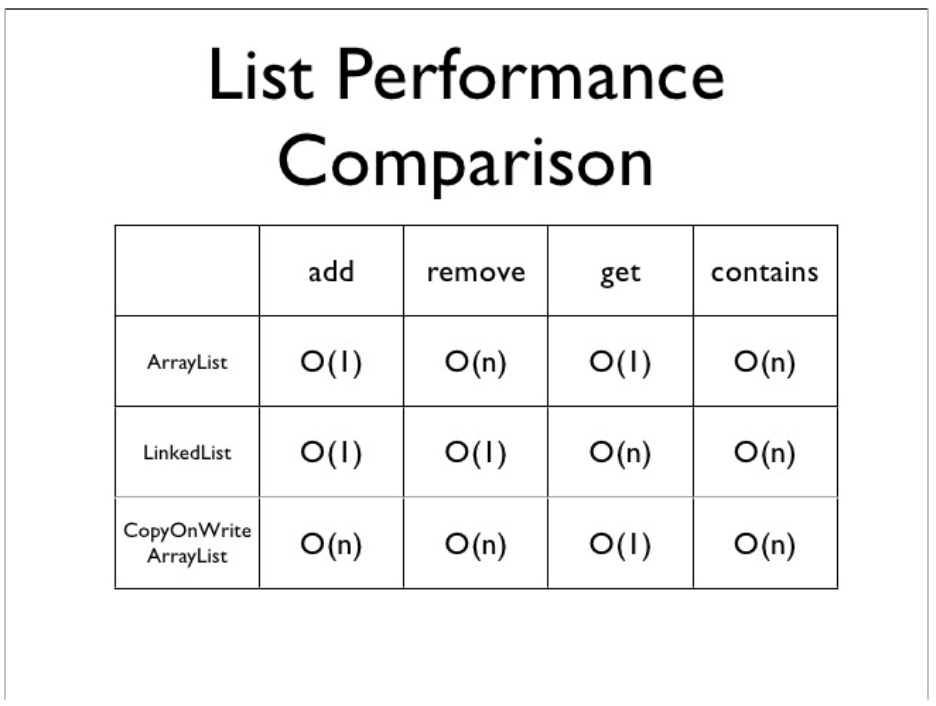
17) Can we replace Hashtable with ConcurrentHashMap?

Answer 3: Yes we can replace Hashtable with ConcurrentHashMap and that's what suggested in Java documentation of ConcurrentHashMap. but you need to be careful with code which relies on locking behavior of Hashtable. Since Hashtable locks whole Map instead of a portion of Map, compound operations like if(Hashtable.get(key) == null) put(key, value) works

in Hashtable but not in concurrentHashMap. instead of this use putIfAbsent() method of ConcurrentHashMap

18) What is CopyOnWriteArrayList, how it is different than ArrayList and Vector?

Answer: CopyOnWriteArrayList is new List implementation introduced in Java 1.5 which provides better concurrent access than Synchronized List. better concurrency is achieved by Copying ArrayList over each write and replace with original instead of locking. Also CopyOnWriteArrayList doesn't throw any ConcurrentModification Exception. Its different than ArrayList because its thread-safe and ArrayList is not thread-safe and it's different than Vector in terms of Concurrency. CopyOnWriteArrayList provides better Concurrency by reducing contention among readers and writers. Here is a nice table which compares performance of three of popular List implementation ArrayList, LinkedList and CopyOnWriteArrayList in Java:

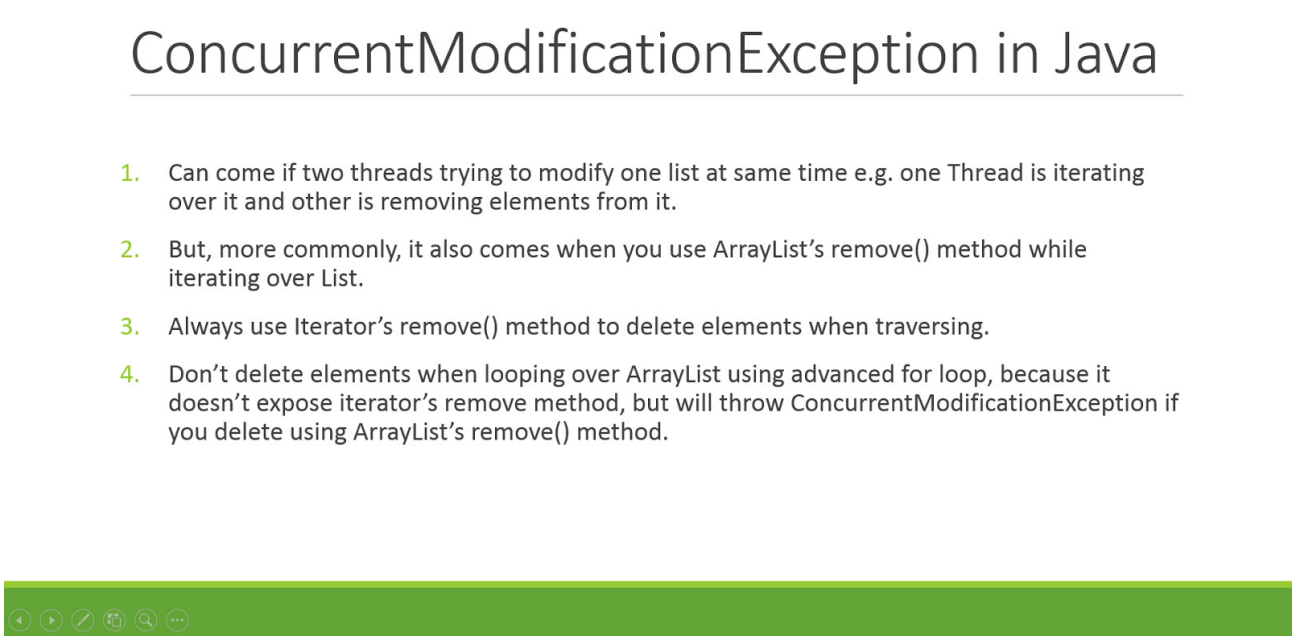


19) Why ListIterator has added() method but Iterator doesn't or Why to add() method is declared in ListIterator and not on Iterator.

ListIterator has added() method because of its ability to traverse or iterate in both direction of the collection. it maintains two pointers in terms of previous and next call and in a position to add a new element without affecting current iteration.

20) When does ConcurrentModificationException occur on iteration?

When you remove object using Collection's or List's remove method e.g. remove(Object element) or remove(int index), instead of Iterator's remove() method than ConcurrentModificationException occurs. As per Iterator's contract, if it detect any structural change in Collection e.g. adding or removing of the element, once Iterator begins, it can throw ConcurrentModificationException. Here are some tips to avoid ConcurrentModification in Java.



21) Difference between Set, List and Map Collection classes?

java.util.Set, java.util.List and java.util.Map defines three of most popular data structure support in Java. Set provides uniqueness guarantee i.e.g you can not store duplicate elements on it, but it's not ordered. On the other hand List is an ordered Collection and also allowes duplicates. Map is based on hashing and stores key and value in an Object called entry. It provides O(1) performance to get object, if you know keys, if there is no collision. Popular impelmentation of Set is HashSet, of List is ArrayList and LinkedList, and of Map are HashMap, Hashtable and ConcurrentHashMap. Another key difference between Set, List and Map are that Map doesn't implement Collection interface, while other two does. For a more detailed answer, see Set vs List vs Map in Java

Collection	Duplicates	Order	Positional Access	Performance	Implementation	Real Scenario Usage
List	Yes	Ordered	Yes	?	ArrayList, LinkedList	?
Set	No	Depends on Implementation	No	?	HashSet, LinkedHashSet, TreeSet	?
Map	No duplicate keys	?	?	?	HashMap, LinkedHashMap, TreeMap	?

22) What is BlockingQueue, how it is different than other collection classes?

BlockingQueue is a Queue implementation available in java.util.concurrent package. It's one of the concurrent Collection class added on Java 1.5, main difference between BlockingQueue and other collection classes is that apart from storage, it also provides flow control. It can be used in inter-thread communication and also provides built-in thread-safety by using happens-before guarantee. You can use BlockingQueue to solve Producer Consumer problem, which is what is needed in most of concurrent applications.

Few more questions for practice, try to find answers to these question by yourself:

23) How does LinkedList is implemented in Java, is it a Singly or Doubly linked list?
Hint: LinkedList in Java is a doubly linked list.

24) How do you iterator over Synchronized HashMap, do you need to lock iteration and why?

25) What is Deque? when do you use it?

SERIALIZATION INTERVIEW QUESTIONS

There is no doubt that Java is vast and there are some Java topics which many Java developers rarely explore. Serialization is one of them which is rarely used in practice but quite popular during Java interviews. It's also one of the difficult topics to master and that's why I am going to share some frequently asked Java Serialization interview questions in this section. I don't know why people ask so many questions from Selrization if not everyone uses it but I have always seen some questions from Java realization. These are also the toughest and confusing questions to answer and more likely cannot be answered by an average Java developer.

I still remember when I was asked about the readObject() and writeObject() method on one of the Java interviews on a big investment bank during the early days of my career. That was the first time I heard about those methods. I am sure, I wasn't alone there were many Java developers who haven't heard about

those two methods during the early stage of their career.

I have come a long way since then and learned a lot about Serialization in Java and seen them in the real world. To be honest with you, it is a very complex feature of Java and if you have to touch a code that uses Serialization, you will more likely break it than fix it.

I was given a task to add a new field in one of our domain class Order.java which implemented Serializable. I added the field only to see NotSerlizableException in our log file then I realized that any field you are going to add must implement either Serializable or Externalizable interface if it's not transient or static.

There are many such fine details which you can only know by hard reading or real practical experience. Though, one book which helped me a lot was **Effective Java**. It has some of the best material converting this topic and I strongly suggest every Java developer read this book and all the items pertaining to Serialization in Java.

Without wasting any more of your time, here are some Java questions from the Serialization topic, you need more questions to prepare well though.

1. What is a Serializable interface? What is the purpose of it?

Serializable is a marker interface and it indicates that the object of any class which implements a Serializable interface can be stored on the disc using JVM's default serialization mechanisms or transferred via the network.

2. What is the difference between the Serializable and Externalizable interface?

Both interfaces allow your object to be sterilized or converted to aq binary format which can be saved into disc or transfers over the network but key difference between them is that Serializable uses default serialization mechanism of JVM and Externalizable provide hooks and callbacks to do things before and after realization.

3. What is a transient variable? What is the purpose of it?

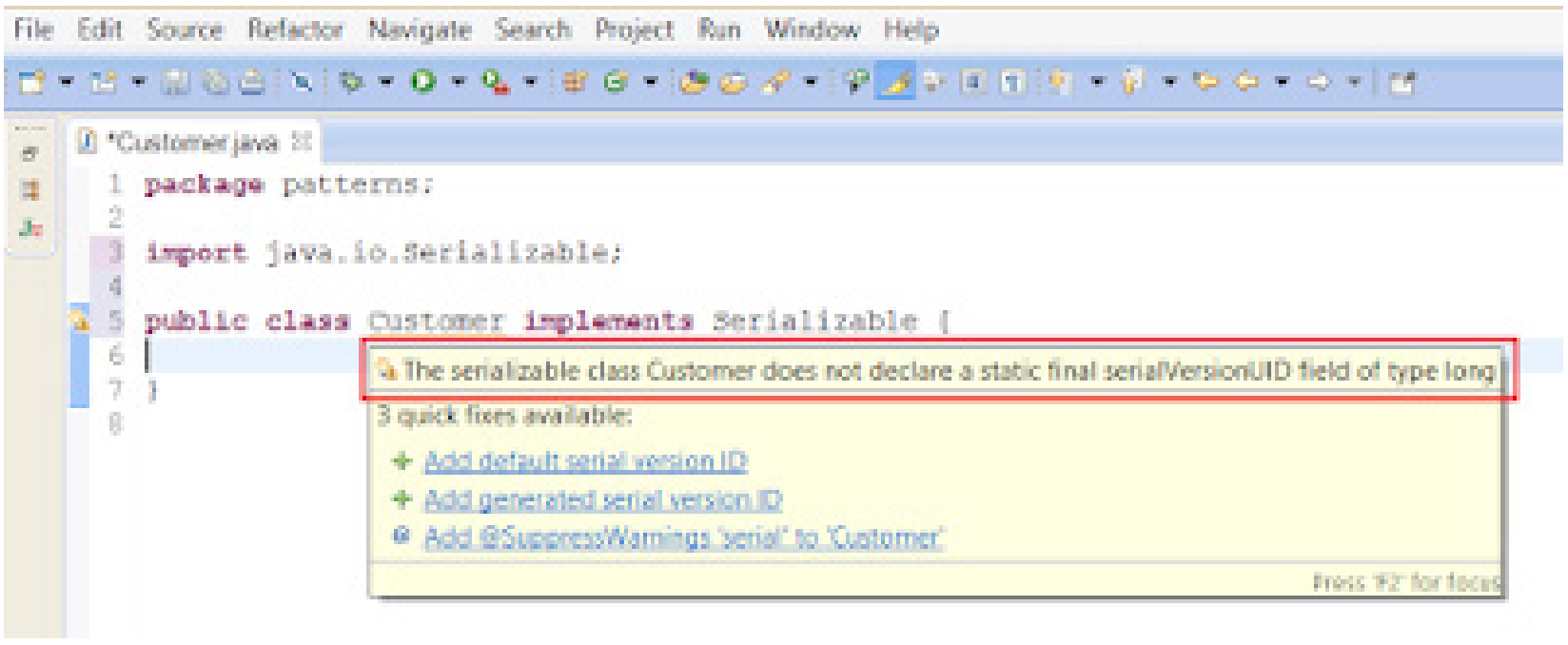
Transient variables are not saved during the Serialization process. As the name suggests they are transient and not really part of the object's state. You can use a transient variable to exclude certain fields from being serialized like a field that is not serializable should be either marked transient or static.

4. What is SerialVersionUID in Java? Why it's important?

SerialVersionUID is an identifier generated during the Serialization process if not defined in the class. The danger of not specifying explicitly is that the default logic goes generate this id depends upon class metadata like a number of fields, which means if you add new fields or remove olds fields the it will be different.

This means, you may not be able to restore the old serialized instances with the new version of your code. When you define SerialVersionUID as public static final long constant then the default serialization mechanism doesn't generate it, hence it becomes independent of class metadata.

As I have said, a good knowledge of this topic is very important for Java programmers and if you want to learn more, I strongly suggest you join a good Java course like Java In-Depth: Become a Complete Java Engineer! on Udemy.



5. How many methods we have in the Serializable interface?

None, java.io.Serializable is a marker interface, it doesn't have any method.

6. What is a marker interface in Java?

An interface without any method. Such an interface is just an indication to the compiler, JVM, or some tools for some special processing. They are similar to annotation and hence obsolete once annotation was introduced in Java 5.

7. Can you add a field in a Serializable class which doesn't implement Serializable interface?

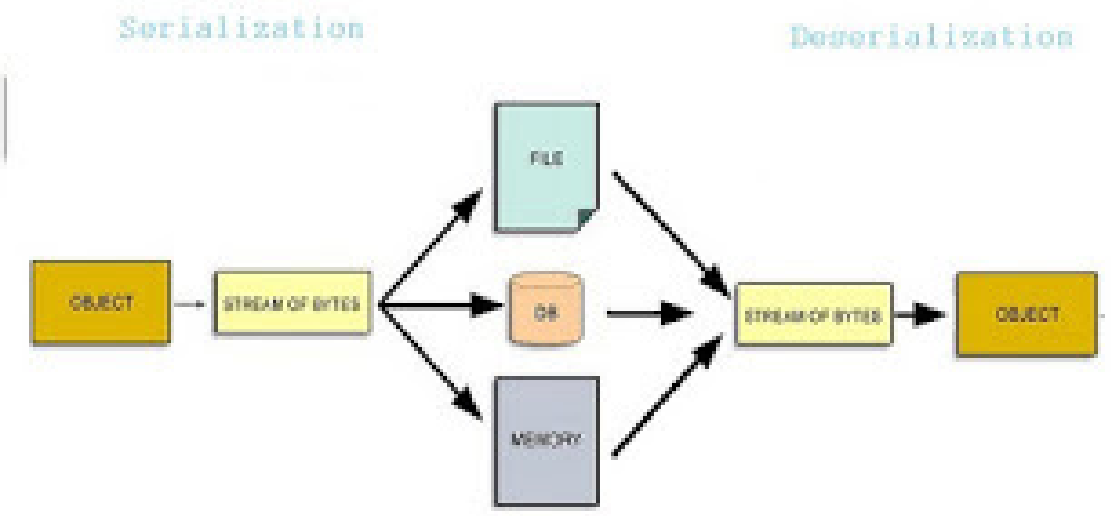
No, that will throw NotSerializableException, if you absolutely want to add it then mark it static or transient, but the state of that object will not be saved when the object will be serialized.

8. How do you serialize an object in Java?

In order to serialize object make sure it implements java.io.Serializable and then use ObjectOutputStream. writeObject() method to save the object into a disk.

9. How Serialization of an object works in Java?

When you serialize an object, first its superclass is serialized and then the subclass. If any field doesn't implement the Serializable interface (only reference type) then it will throw NotSerializableException. Though, if you want to learn more about Serialization in Java then I also suggest you go through a comprehensive Java course like The Complete Java Masterclass on Udemy.



10. What about static variables? Are they Serialized?

No, static variables are not serialized. Since static is a class level variable i.e. it has the same value for each instance. This can create an issue if your serialized object id dependent upon the value of a static variable because when you deserialize the object, it will get the value for static variable what other instances will have, this may not be the same. So, don't make your serialized object depends upon the static variable.

11. What is the difference between the transient and volatile variables in Java?

They are completely different, volatile is used in multi-threading to instruct the compiler that variable will be accessed by multiple threads and provide ordering and visibility guarantee, while the transient variable is used to exclude fields from being persisted as part of serialization.

12. Can you make a subclass NotSerializable if the Superclass is Serializable? Is it mandatory for a subclass to implement the Serializable interface?

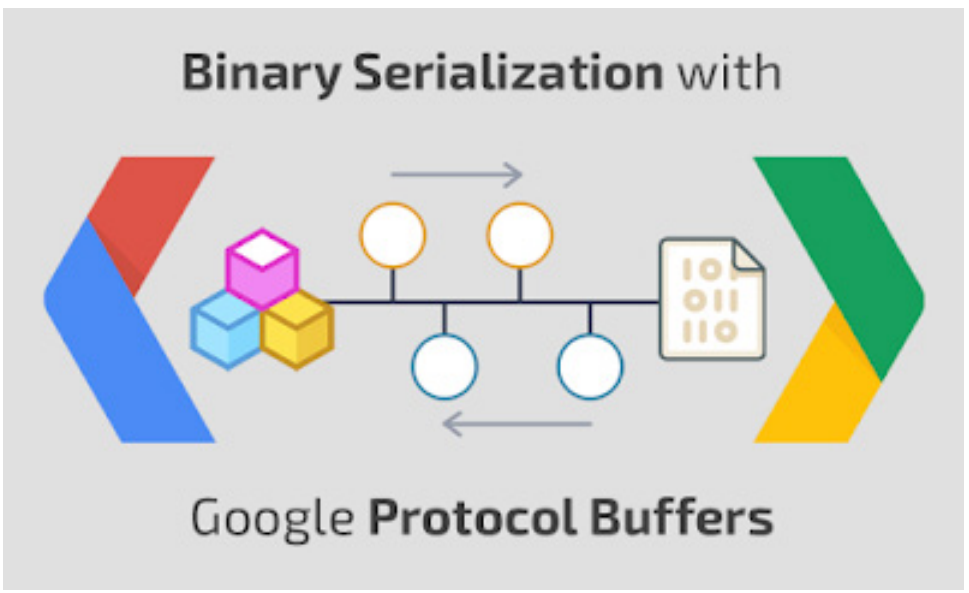
Yes, you can make the subclass NotSerializable. It's not mandatory for a subclass to be a Serializable class even if the superclass is serializable.

13. Do you know any alternative to Serialization for Java application?

Yes, there are a couple of alternatives to Serialization in Java. For example, you can develop your own encoder and decoder to convert your Java objects to a binary format like a byte array where positions are used to mark for a particular field.

This is very popular on high speed messaging and high-frequency training applications and many exchange protocols like Arrowhead of the Tokyo Stock Exchange and Omnet protocol of HongKong future exchange use this mechanism.

Another popular alternative of Serialization in Java is Google's protocol buffer which allows you to safely convert your objects to a pre-defined binary format known as protobuf. You can also use gRPC to develop server-side applications. If you want to learn more about Google Profotocl BUffer then I highly recommend you go through Complete Guide to Protocol Buffers 3 in Java by Stephane Maarek on Udemy.



That's all about the Serliaziation interview questions for Java programmers. You can use these questions to learn more about Serliazation and also fill gaps in your learning. If you haven't read then I strongly suggest you read Effective Java book, particularly all items on Serialization which contain a lot of wisdom, best practices, and advice from Joshua Bloch about effectively and safely using Serialization in Java.

DESIGN PATTERN INTERVIEW QUESTIONS

Hello guys, if you are preparing for Java interviews and looking for frequently asked design pattern interview questions then you have come to the right place.
In the past, I have shared the best courses for Java interviews, and today, I am going to share popular design pattern questions from Java interviews. You can use these questions to both practices and check your knowledge about OOP design patterns. Both OOP and GOF design pattern interview questions are an integral part of any good list of core Java interview questions. Java is a popular Object-oriented programming language and has lots of design patterns and design principles, contributed by many developers and open-source frameworks.

As a Java programmer, it's expected from you to know OOPS concepts like Abstraction, Encapsulation, and Polymorphism; what is a design pattern in Java, Some popular Java design patterns, and, most importantly, when to use those design pattern in Java application. The purpose of asking the design pattern interview

question in Java is to check whether the Java programmer is familiar with those essential design patterns or not.

Design patterns in Java interviews are as crucial as multi-threading, collection, and programming questions and one should not neglect it. Good knowledge of Software design patterns goes a long way in becoming a better Java developer which every company looks for.

These Software design patterns provide templates and tricks used to design and solve real-world software problems and tasks. If you know how to apply and use these time-tested design patterns then you can write better code, I mean the code which is more inextensible, maintainable, and flexible.

If you are a senior or experienced Java programmer, then you can expect more complex and tough design patterns in Java interviews like Momento, Chain of Responsibility, and Builder design pattern and solving real-time software design questions.

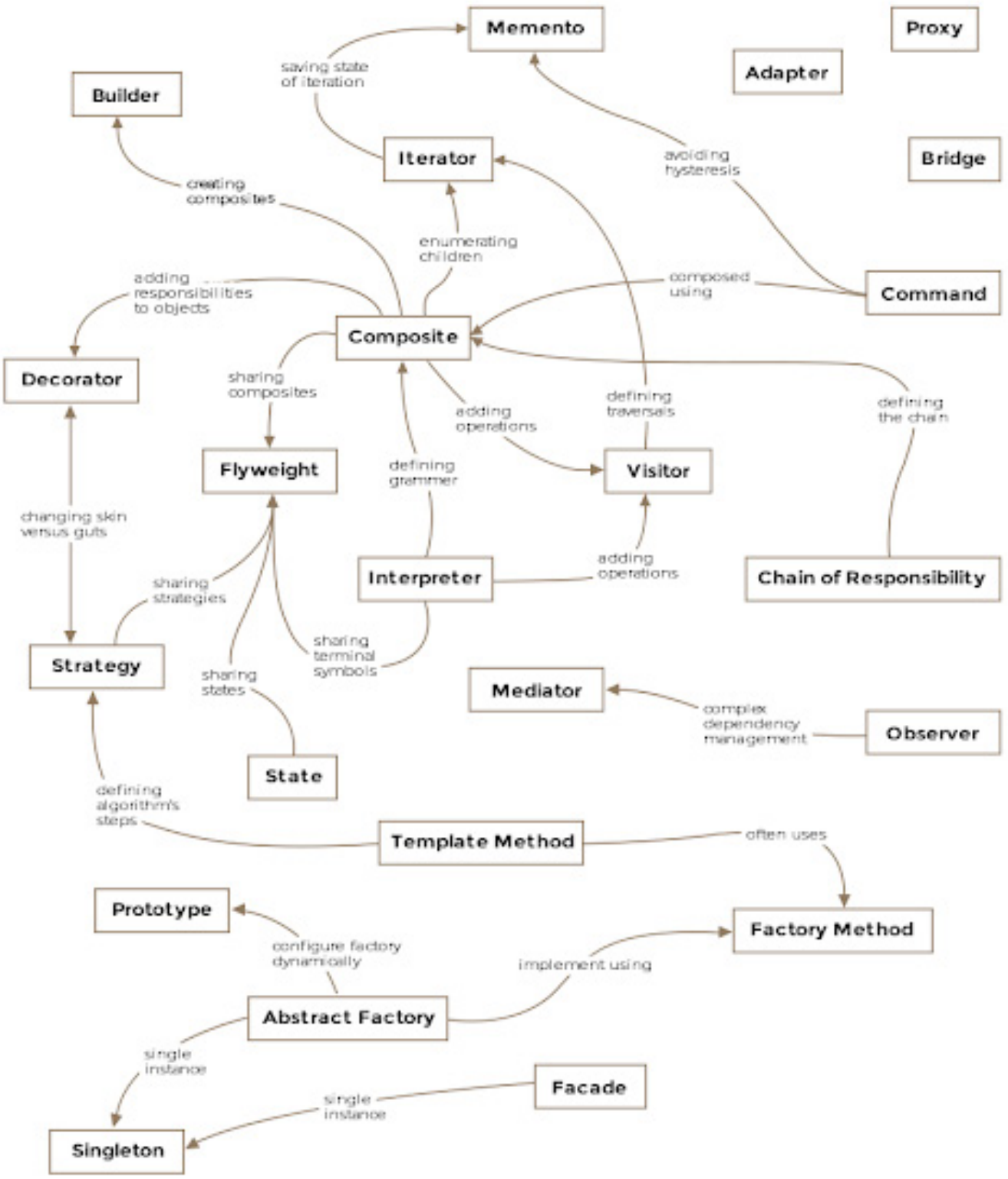
By the way, always remember, giving an example from your project creates a better impression and if you want to learn about the modern implementation of design patterns in Java, I suggest you join the Design Patterns in Java course by Dmitri Nesteruk on Udemy. In this course, you will not only learn how to recognize and apply design patterns but also how to Refactor existing designs to use design patterns.

Here is my list of top 10 design pattern interview question in Java. I have also provided an answer to those Java design pattern questions as a link. No matter which level of Java interview is you going like a programmer, software engineer, a senior software engineer in Java, you can expect a few questions from Java design pattern.

1. What is the Decorator pattern in Java? Can you give an example of a Decorator pattern?

The Decorator pattern is another popular Java design pattern question, which is common because of its heavy usage in java.io package. BufferedReader and BufferedWriter are an excellent example of decorator pattern in Java.

If you want to learn more about how a Decorator pattern can help you to write better code, I also suggest you check out the Software Design Patterns: Best Practices for Software Developers course on Educative. This course provides a lot of real-world examples in Java programming language which makes it easy to understand and apply these design patterns and write better code.



Design Pattern Relationships

2. When to use the Strategy Design Pattern in Java?

Java design pattern interview question and answers for senior and experience programmerStrategy pattern is quite useful for implementing a set of related algorithms like compression algorithms, filtering

strategies, etc. The strategy design pattern allows you to create Context classes, which use Strategy implementation classes for applying business rules.

One good example of a Strategy pattern from JDK itself is a Collections.sort() method and the Comparator interface, which is a strategy interface and defines a strategy for comparing objects. Because of this pattern, we don't need to modify the sort() method (closed for modification) to compare any object; at the same time, we can implement a Comparator interface to define a new comparing strategy (open for extension).

This pattern is actually based upon the open-closed design principle and if you understand that principle then its quite easy for you to understand the Strategy pattern as well. It's actually better to know about SOLID principles as many design patterns are based upon that.

If you're looking for a complete course on Object-Oriented design principles, I recommend checking out SOLID Principles of Object-Oriented Design by Steve Smith on Pluralsight. This is a useful course for anyone looking to strengthen their overall knowledge of software architecture.

3. What is the Observer design pattern in Java? When do you use the Observer pattern in Java?

This is one of the most common Java design pattern interview questions. The observer pattern is based upon notification, there are two kinds of object Subject and Observer. Whenever there is a change in the subject's state observer will receive a notification.

4. What is the difference between Strategy and State design Pattern in Java?

This is an interesting Java design pattern interview question as both Strategy and State pattern has the same structure. If you look at the UML class diagram for both patterns, they look exactly the same, but their intent is totally different.

The state design pattern is used to define and manage the state of an object, while the Strategy pattern is used to define a set of an interchangeable algorithm, and let's client choose one of them. So Strategy pattern is a client-driven pattern while Object can manage their state itself.

5. When to use the Composite design pattern in Java? Have you used it previously in your project?

This design pattern question is asked on Java interview not just to check familiarity with the Composite pattern but also, whether a candidate has the real-life

experience or not.

The Composite pattern is also a core Java design pattern, which allows you to treat both whole and part object to treat similarly. Client code, which deals with a Composite or individual object, doesn't differentiate between them, it is possible because the Composite class also implements the same interface as their individual part.

One of the good examples of the Composite pattern from JDK is the JPanel class, which is both Component and Container. When the paint() method is called on JPanel, it internally called the paint() method of individual components and let them draw themselves. In the second part of this design pattern interview question, be truthful; if you have used, then say yes, otherwise say that you are familiar with the concept and used it on your own.

6. What is the Singleton pattern in Java?

Singleton pattern in Java is a pattern that allows only one instance of Singleton class available in the whole application. java.lang.Runtime is a good example of a Singleton pattern in Java. There are lots of follow up questions on the Singleton pattern see 10 Java singleton interview question answers for those follow-ups.

7. Can you write thread-safe Singleton in Java?

There are multiple ways to write thread-safe Singleton in Java, like by writing singleton using double-checked locking, by using static Singleton instance initialized during class loading. By the way, using Java enum to create thread-safe singleton is the most simple way.

8. When to use the Template method design pattern in Java?

The Template pattern is another popular core Java design pattern interview question. I have seen it appear many times in real life project itself. The template pattern outlines an algorithm in the form of a template method and lets the subclass implement individual steps.

The critical point to mention, while answering this question, is that the template method should be final so that the subclass can not override and change the steps of the algorithm. Still, the same time individual steps should be abstract, so that child classes can implement them.

9. What is the Factory pattern in Java? What is the advantage of using a static factory method to create an object?

The Factory pattern in Java is a creation Java design pattern and a favorite on many Java interviews. Factory pattern used to create an object by providing

static factory methods. There are many advantages of providing factory methods like caching immutable objects, easy to introduce new objects, etc. See What is Factory pattern in Java and benefits for more details.

10. What is the difference between Decorator and Proxy pattern in Java?

Another tricky Java design pattern question and trick here is that both Decorator and Proxy implements the interface of the Object they decorate or encapsulate. As I said, many Java design patterns can have similar or exactly the same structure, but they differ in their intent.

The Decorator pattern is used to implement functionality on an already created object, while a Proxy pattern is used for controlling access to an object.

One more difference between Decorator and the Proxy design pattern is that the Decorator doesn't create an object; instead, it gets the Object in its constructor, while Proxy actually creates objects. You can also read the Head First Analysis and Design to understand the difference between them.

11. When to use Setter and Constructor Injection in the Dependency Injection pattern?

Use Setter injection to provide optional dependencies of an object while use Constructor injection to provide a mandatory dependency of an object, without which it can not work. This question is related to the Dependency Injection design pattern and mostly asked in the context of the Spring framework, which is now become a standard for developing Java application.

Since Spring provides an IOC container, it also gives you a way to specify dependencies either by using setter methods or constructors. You can also take a look at my previous section on the same topic.

12. What is the difference between Factory and Abstract Factory in Java

I have already answered this question in detail with my section with the same title. The main difference is that the Abstract Factory creates a factory while the Factory pattern creates objects. So both abstract the creation logic, but one abstract is for factory and the other for items.

13. When to use the Adapter pattern in Java? Have you used it before in your project?

Use the Adapter pattern when you need to make two class works with incompatible interfaces. Adapter pattern can also be used to encapsulate third party

code so that your application only depends upon Adapter, which can adapt itself when third party code changes or you moved to a different third party library.

By the way, this Java design pattern question can also be asked by providing an actual scenario. You can further read Head First Design Pattern to learn more about the Adapter pattern and its real-world usage. The book is updated for Java 8 as well, so you will learn new Java 8 ways to implement these old design patterns.

14. Can you write code to implement a producer-consumer design pattern in Java?

The Producer-consumer design pattern is a concurrency design pattern in Java, which can be implemented using multiple ways. If you are working in Java 5, then it's better to use Concurrency util to implement producer-consumer pattern instead of plain old wait and notify in Java. Here is an excellent example of implementing the producer-consumer problem using BlockingQueue in Java.

15. What is the Builder design pattern in Java? When do you use the Builder pattern?

Builder pattern in Java is another creational design pattern in Java and often asked in Java interviews because of its specific use when you need to build an object which requires multiple properties, some optional and some mandatory.

16. What is the Open closed design principle in Java?

The Open closed design principle is one of the SOLID principles defined by Robert C. Martin, popularly known as Uncle Bob in his most popular book, Clean Code. This principle advises that code should be open for extension but closed for modification.

At first, this may look conflicting, but once you explore the power of polymorphism, you will start finding patterns that can provide stability and flexibility of this principle.

One of the critical examples of this is the State and Strategy design pattern, where Context class is closed for modification, and new functionality is provided by writing new code by implementing a new state of strategy. See this section to know more about Open closed principle.

17. Can you give an example of SOLID design principles in Java?

There are lots of SOLID design pattern which forms acronym SOLID like:
1. Single Responsibility Principle or SRP
3. Open Closed Design Principle or OCD
3. Liskov Substitution Principle
4. Interface Segregation Principle
5. Dependency Inversion Principle.

18. What is the difference between Abstraction and Encapsulation in Java?

Even though both Abstraction and Encapsulation look similar because both hide complexity and make the external interface simpler, there is a subtle difference between them. Abstraction hides logical complexity while Encapsulation hides Physical Complexity.

Abstraction	Encapsulation
1. Abstraction solves the problem in the design level.	1. Encapsulation solves the problem in the implementation level.
2. Abstraction is used for hiding the unwanted data and giving relevant data.	2. Encapsulation means hiding the code and data into a single unit to protect the data from outside world.
3. Abstraction lets you focus on what the object does instead of how it does it	3. Encapsulation means hiding the internal details or mechanics of how an object does something.
4. **Abstraction**- Outer layout, used in terms of design. For Example:- Outer Look of a Mobile Phone, like it has a display screen and keypad buttons to dial a number.	4. **Encapsulation**- Inner layout, used in terms of implementation. For Example:- Inner Implementation detail of a Mobile Phone, how keypad button and Display Screen are connect with each other using circuits.

GARBAGE COLLECTION INTERVIEW QUESTIONS

Garbage collection interview questions are very popular in both core Java and advanced Java Interviews. Apart from Java Collection and Thread many tricky Java questions stem Garbage collections which are tough to answer. In this Java Interview section, I will share some questions from GC which is asked in various core Java interviews. These questions are based upon the concept of How Garbage collection works, Different kinds of Garbage collectors, and JVM parameters used for garbage collection monitoring and tuning. As I said GC is an important part of any Java interview so make sure you have good command in GC. One more thing which is getting very important is the ability to comprehend and understand Garbage collection Output, more and more interviewer are checking whether the candidate can understand GC output or not.

During Java interview, they may provide a snippet of GC output and ask various questions based on that like Which Garbage collector is used, whether the output is from the major collection or minor collection, How

much memory is free from GC, What is the size of new generation and old generation after GC, etc.

I have included a few Garbage collection interview questions and answers from GC output to help with that. It's recommended to prepare questions from Java collection, multithreading, and programming along with Garbage collection to do well in Java interviews at any stage.

And, if you are serious about improving your advanced JVM skill and learn things like taking and analyzing heap dumps then highly recommend you to join Java Application Performance and Memory Management course on Udemy. It's one of the advanced courses for Java programmers to learn more about Performance and Memory management including troubleshooting memory leaks in Java.

Here are some Garbage collection Interview questions from my personal collection, which I have created from my experience and with the help of various friends and colleagues who have shared GC interview questions with me.

Actually, there are a lot many questions than What I am sharing here but to keep this section small I thought to only share some questions, I can think of a second part of the GC interview question if you guys find this useful.

Question 1 - What is the structure of Java Heap? What is Perm Gen space in Heap?

Answer: In order to better perform in Garbage collection questions in any Java interview, It's important to have a basic understanding of Java Heap space.

By the way, Heap is divided into different generation e.g. new generation, old generation, and PermGen space.PermGen space is used to store class's metadata and filling of PermGen space can cause java.lang. OutOfMemory: PermGen space. It's also worth noting to remember the JVM option to configure PermGen space in Java.

Question 2 - How do you identify minor and major garbage collection in Java?

Answer: Minor collection prints "GC" if garbage collection logging is enabled using –verbose:gc or -XX:PrintGCDetails, while Major collection prints "Full GC". This Garbage collection interview question is based on an understanding of the Garbage collection output. As more and more Interviewer is asking a question to check the candidate's ability to understand GC output, this topic becomes even more important.

Question 3 - What is the difference between ParNew and DefNew Young Generation Garbage collector?

Answer: This Garbage Collection interview question is recently asked one of my friends. It requires more than average knowledge on GC to answer this question.

By the way, ParNew and DefNew are two young generation garbage collector. ParNew is a multi-threaded GC used along with concurrent Mark Sweep while DefNew is single-threaded GC used along with Serial Garbage Collector.

Question 4 - How do you find GC resulted due to calling System.gc()?

Answer: Another GC interview question which is based on GC output. Similar to the major and minor collection, there will be the word “System” included in the Garbage collection output.

Question 5 - What is the difference between Serial and Throughput Garbage collectors?

Answer: Serial Garbage collector is a stop the world GC which stops application thread from running during both minor and major collection. Serial Garbage collector can be enabled using JVM option -XX:UseSerialGC and it's designed for Java application which doesn't have pause time requirement and has client configuration.

The Serial Garbage collector also defaulted GC in JDK 1.4 before ergonomics was introduced in JDK 1.5. Serial GC is most suited for small applications with fewer threads while throughput GG is more suited for large applications. On the other hand Throughput garbage collector is a parallel collector where minor and major collection happens in parallel taking full advantage of all the system resources available like multiple processor.

Though both major and minor collection runs on stop-the-world fashion and introduced pause in the application. Throughput Garbage collector can be enable using -XX:UseParallelGC or -XX:UseOldParallelGC.

It increases the overall throughput of application my minimizing time spent in Garbage collection but still has long pauses during full GC.

This is a kind of Garbage collection interview question

that gives you an opportunity to show your knowledge in detail while answering. I always suggest answering these kinds of questions in detail.

Question 6 – When does an Object becomes eligible for Garbage collection in Java?

Answer : An object becomes eligible for garbage collection when there is no live reference for that object or it can not be reached by any live thread. Cyclic reference doesn't count as live reference and if two objects are pointing to each other and there is no live reference for any of them, than both are eligible for GC. Also Garbage collection thread is a daemon thread which will run by JVM based upon GC algorithm and when runs it collects all objects which are eligible for GC.

Question 7 - What is finalize method in Java ? When does Garbage collector calls finalize method in Java?

Answer : Finalize method in Java also called finalizer is a method defined in java.lang.Object and called by Garbage collector before collecting any object which is eligible for GC. Finalize() method provides last chance to object to do cleanup and free any remaining resource.

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

This Garbage collection interview questions is related question 5 "When object become eligible for Garbage collection". An object becomes eligible for Garbage collection if there is no live reference for it. It can not be accessible from any Thread and cyclic dependency doesn't prevent Object from being Garbage collected. Which means in this case both Object A and Object B are eligible of Garbage collection.

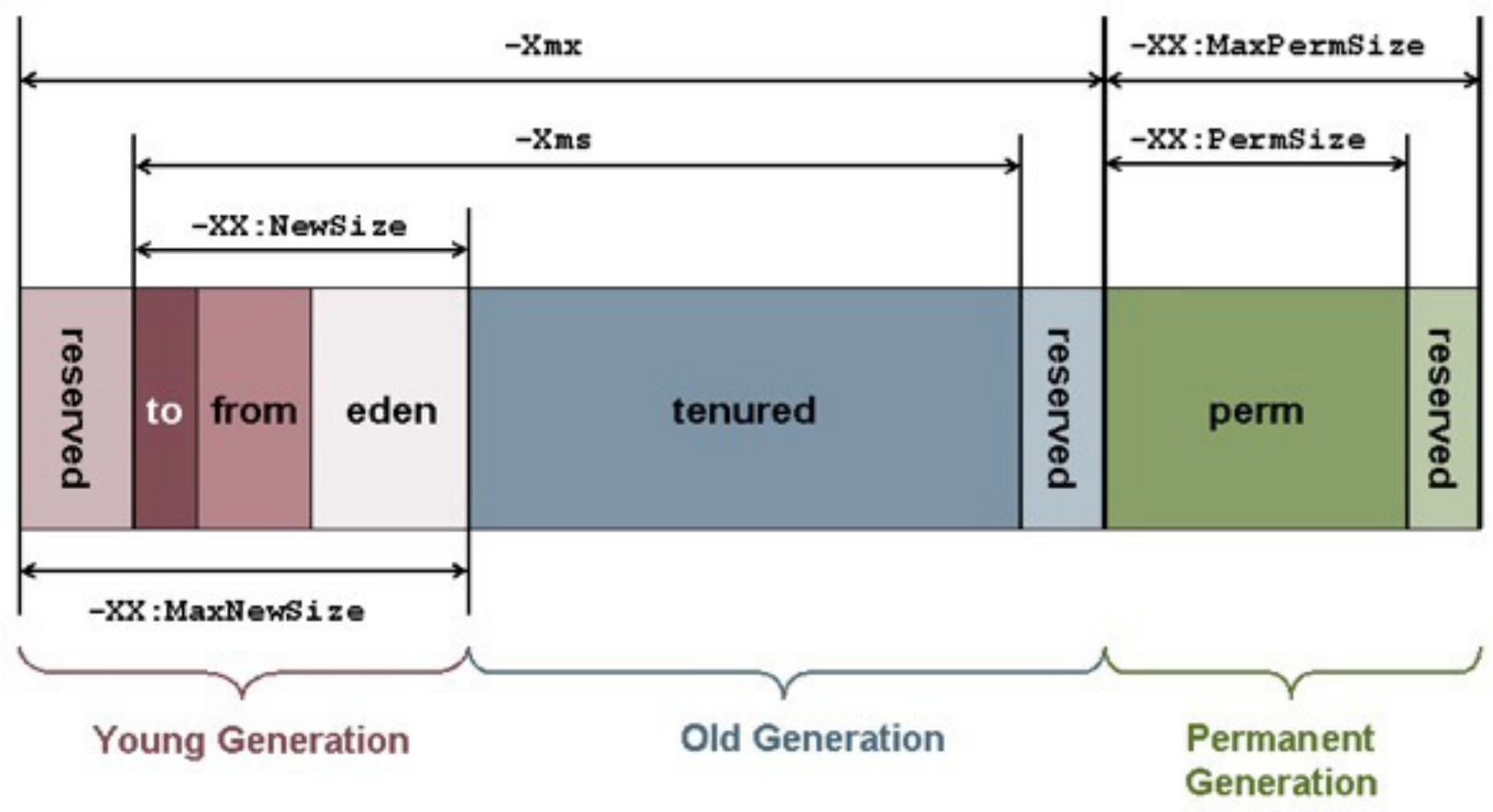
Question 9 -Can we force the Garbage collector to run at any time?

Answer : No, you can not force Garbage collection in Java. Though you can request it by calling Sytem.gc() or its cousin Runtime.getRuntime().gc(). It's not guaranteed that GC will run immediately as result of calling these method.

Question 10 - Does Garbage collection occur in permanent generation space in JVM?

Answer : This is a tricky Garbage collection interview question as many programmers are not sure whether PermGen space is part of Java heap space or not and since it maintains class Meta data and String pool, whether its eligible for garbage collection or not. By the way Garbage Collection does occur in PermGen

space and if PermGen space is full or cross a threshold, it can trigger Full GC. If you look at output of GC you will find that PermGen space is also garbage collected. This is why correct sizing of PermGen space is important to avoid frequent full GC. You can control size of PermGen space by JVM options -XX:PermGenSize and -XX:MaxPermGenSize.



Question 11 : How to you monitor garbage collection activities?

Answer : One of my favorite interview questions on Garbage collection, just to check whether candidate has ever monitored GC activities or not. You can monitor garbage collection activities either offline or real-time. You can use tools like JConsole and VisualVM VM with its Visual GC plug-in to monitor real time garbage collection activities and memory status of JVM or you can

redirect Garbage collection output to a log file for offline analysis by using -XlogGC=<PATH> JVM parameter. Anyway you should always enable GC options like -XX:PrintGCDetails -X:verboseGC and -XX:PrintGCTimeStamps as it doesn't impact application performance much but provide useful states for performance monitoring.

Question 12: Look at below Garbage collection output and answer following question :

```
[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]
```

1. Is this output of Major Collection or Minor Collection?
2. Which young Generation Garbage collector is used?
3. What is size of Young Generation, Old Generation and total Heap Size?
4. How much memory is freed from Garbage collection?
5. How much time is taken for Garbage collection?
6. What is current Occupancy of Young Generation?

This Garbage collection Interview questions is completely based on GC output. Following are answers of above GC questions which will not only help you to answer these question but also help you to understand and interpret GC output.

Answer 1: It's Minor collection because of "GC" word,

In case of Major collection, you would see “Full GC”.

Answer 2: This output is of multi-threaded Young Generation Garbage collector “ParNew”, which is used along with CMS concurrent Garbage collector.

Answer 3: [1512K] which is written in bracket is total size of Young Generation, which include Eden and two survivor space. 1512K on left of arrow is occupancy of Yong Generation before GC and 64K is occupancy after GC. On the next line value if bracket is total heap size which is (600345K). If we subtract size of young generation to total heap size we can calculate size of Old Generation. This line also shows occupancy of heap before and after Garbage collection.

Answer 4: As answered in previous garbage collection interview question, second line shows heap occupancy before and after Garbage collection. If we subtract value of right side 13569K, to value on left side 15604K, we can get total memory freed by GC.

Answer 5: 0.0636056 secs on second line denotes total time it took to collect dead objects during Garbage collection. It also include time taken to GC young generation which is shown in first line (0635032 secs).

Answer 6: 64K

Here is few more interesting Garbage collection Interview question for your practice.
Question - What is the difference between

-XX:ParallelGC and -XX:ParallelOldGC?

Question - When do you ConcurrentMarkSweep Garbage collector and Throughput GC?

Question - What is difference between ConcurrentMarkSweep and G1 garbage collector?

Question - Have you done any garbage collection tuning? What was your approach?

GENERICS INTERVIEWS QUESTIONS

Generic interview questions in Java interviews are getting more and more common with Java 5 around there for a considerable time and many applications either moving to Java 5 and almost all new Java development happening on Tiger(code name of Java 5). Importance of Generics and Java 5 features like Enum, Autoboxing, varargs and Collection utilities like CountDownLatch, CyclicBarrier and BlockingQueue are getting more and more popular on Java interviews. Generic interview question can get real tricky if you are not familiar with bounded and unbounded generic wildcards, How generics works internally, type erasure and familiarity with writing parametrized generics classes and methods in Java.

Best way to prepare for Generics interview is to try simple program best on various features of generics. Anyway In this Java interview section we will see some popular Java Generics interview questions and there answer.

By the way there are lot of material available in Javarevisited to better preparing for Java and J2EE interviews, you can prepare multi-threading and Collections using 15 thread interview question and Top 10 Java collection interview question along with several other questions answers sections on Spring, Struts, JSP and Servlet.

If you are GUI developers and working in Java Swing technology than you can also check interview questions on Java Swing mostly asked in Investment banks

Without wasting anymore of your time, here is a list of frequently asked Generices concept interview questions from Java interviews

1. What is Generics in Java ? What are advantages of using Generics?

This is one of the first interview questions asked on generics in any Java interview, mostly at beginners and intermediate level. Those who are coming from prior to Java 5 background knows that how inconvenient it was to store object in Collection and then cast it back to correct Type before using it. Generics prevents from those. it provides compile time type-safety and ensures that you only insert correct Type in collection and avoids ClassCastException in runtime.

2. How Generics works in Java ? What is type erasure ?

This is one of better interview question in Generics. Generics is implemented using Type erasure, compiler erases all type related information during compile time and no type related information is available during runtime. for example List<String> is represented by only List at runtime.

This was done to ensure binary compatibility with the libraries which were developed prior to Java 5. you don't have access to Type argument at runtime and Generic type is translated to Raw type by compiler during runtime. you can get lot of follow up question based on this Generic interview question based upon your response e.g. Why Generics is implemented using Type erasure or presenting some invalid generic code which results in compiler error.

3. What is Bounded and Unbounded wildcards in Generics ?

This is another very popular Java interview questions on Generics. Bounded Wildcards are those which impose bound on Type. there are two kinds of Bounded wildcards <? extends T> which impose an upper bound by ensuring that type must be sub class of T and <? super T> where its imposing lower bound by ensuring Type must be super class of T. This Generic Type must be instantiated with Type within bound otherwise it will result in compilation error.

On the other hand <?> represent and unbounded type because <?> can be replace with any Type.

4. What is difference between List<? extends T> and List <? super T> ?

This is related to previous generics interview questions, some time instead of asking what is bounded and unbounded wildcards interviewer present this question to gauge your understanding of generics. Both of List declaration is example of bounded wildcards, List<? extends T> will accept any List with Type extending T while List<? super T> will accept any List with type super class of T. for Example List<? extends Number> can accept List<Integer> or List<Float>. see more on above link.

5. How to write a generic method which accepts generic argument and return Generic Type?

Writing generic method is not difficult, instead of using raw type you need to use Generic Type like T, E or K,V which are well known placeholders for Type, Element and Key, Value. Look on Java Collection framework for examples of generics methods. In simplest form a generic method would look like:

```
public V put(K key, V value) {
        return cache.put(key, value);
}
```

6. How to write parametrized class in Java using Generics ?

This is an extension of previous Java generics interview question. Instead of asking to write Generic method Interviewer may ask to write a type safe class using generics. again key is instead of using raw types you need to used generic types and always use standard place holder used in JDK.

7. Write a program to implement LRU cache using Generics ?

This is an exercise for anyone who like Coding in Java. One hint is that LinkedHashMap can be used implement fixed size LRU cache where one needs to remove eldest entry when Cache is full. LinkedHashMap provides a method called removeEldestEntry() which is called by put() and putAll() and can be used to instruct to remove eldest entry. you are free to come up with your own implementation as long as you have a written a working version along with JUnit test.

8. Can you pass List<String> to a method which accepts List<Object>

This generic interview question in Java may look confusing to any one who is not very familiar with Generics as in fist glance it looks like String is object so List<String> can be used where List<Object> is required but this is not true. It will result in

compilation error. It does make sense if you go one step further because List<Object> can store any any thing including String, Integer etc but List<String> can only store Strings.

```
List<Object> objectList;
List<String> stringList;

objectList = stringList;  //compilation error
incompatible types
```

9. Can we use Generics with Array?

This was probably most simple generics interview question in Java, if you know the fact that Array doesn't support Generics and that's why Joshua Bloch suggested in Effective Java to prefer List over Array because List can provide compile time type-safety over Array.

10. How can you suppress unchecked warning in Java ?

javac compiler for Java 5 generates unchecked warnings if you use combine raw types and generics types e.g.

```
List<String> rawList = new ArrayList()
Note: Hello.java uses unchecked or unsafe
operations.;
```

which can be suppressed by using @SuppressWarnings ("unchecked") annotation.

I got few more interview questions on Generics in Java to share with you guys, These questions focus on What is difference between Generics type and Raw type and Can we use Object in place of bounded wildcards etc:

Difference between List<Object> and raw type List in Java?

Main difference between raw type and parametrized type List<Object> is that, compiler will not check type-safety of raw type at compile time but it will do that for parametrized type and by using Object as Type it inform compiler that it can hold any Type of Object e.g. String or Integer.

This Java Generics interview question is based on correct understanding of raw type in Generics. Any way second difference between them is that you can pass any parametrized type to raw type List but you can not pass List<String> to any method which accept List<Object> it will result in compilation error.

Difference between List<?> and List<Object> in Java?

This generics interview question may look related to previous interview questions but completely different. List<?> is List of unknown type while List<Object> is essentially List of any Type. You can assign List<String>, List<Integer> to List<?> but you can not assign List<String> to List<Object>.

```
List<?> listOfAnyType;
List<Object> listOfObject = new ArrayList<Object>();
List<String> listOfString = new ArrayList<String>();
List<Integer> listOfInteger = new
ArrayList<Integer>();

listOfAnyType = listOfString; //legal
listOfAnyType = listOfInteger; //legal
listOfObjectType = (List<Object>) listOfString;
//compiler error - in-convertible types
```

to know more about wildcards see Generics Wildcards Examples in Java

Difference between List<String> and raw type List.

This Generics interview question is similar to difference between raw type and parametrized type. Parametrized type are type-safe and type-safety will be guaranteed by compiler but List raw type is not type safe. You can not store any other Object on List of String but you can not store any Object in raw List. There is no casting required in case of Parametrized type with Generics but explicit casting will be needed for raw type.

```
List listOfRawTypes = new ArrayList();
listOfRawTypes.add("abc");
listOfRawTypes.add(123); //compiler will allow this -
exception at runtime
String item = (String) listOfRawTypes.get(0); //
explicit cast is required
item = (String) listOfRawTypes.get(1); //
ClassCastException because Integer can not be cast in
String

List<String> listOfString = new ArrayList();
listOfString.add("abcd");
listOfString.add(1234); //compiler error, better than
runtime Exception
item = listOfString.get(0); //no explicit casting is
required - compiler auto cast
```

These were some of the frequently asked generics interview questions and answers in Java. None of these generic interview questions are tough or hard, Indeed they are based on fundamental knowledge of generics. Any Java programmer who has decent knowledge of Generics must be familiar with these generics questions in Java.

JDBC INTERVIEW QUESTIONS

JDBC Questions are an integral part of any Java interview, I have not seen any Java Interview which is completed without asking the single JDBC Interview question, there is always at least one or two questions from JDBC API. Some of the popular questions like Why you should use PreparedStatement in Java, Difference between PreparedStatement and CallableStatement in Java and few questions is related to improving the performance of the JDBC layer by applying some JDBC performance tips. In this section, I have summarized a few frequently asked questions in JDBC, they range from easy to difficult and beginner to advanced.

Questions like distributed transaction management and 2 phase commit is tough to answer until you have real experience but mostly asked in various J2EE interviews. This is not an extensive list of JDBC question answers but practicing or revising this question before going to any Java interview certainly helps.

Btw, if you are new to JDBC then I also recommend you join a comprehensive online course to learn JDBC from scratch. If you need a recommendation then you can take a look at the Complete JDBC Programming course on Udemy. This course is a comprehensive

guide on how to use JDBC in Java to connect to different databases. You will learn the right ways of doing things with respect to Java and the database.

Here is my list of frequently asked JDBC question in Java, I have tried to provide the answer to most of the question. If you have any interesting JDBC question which you have faced and not in this list then please share with us.

Question 1: What is JDBC?

Answer: One of the first JDBC interview questions in most of the interviews. JDBC is java database connectivity as name implies it's a java API for communicating to relational database, API has java classes and interfaces using that developer can easily interact with database. For this we need database specific JDBC drivers. Some time this also result in followup questions like Difference between type 2 and type 4 JDBC drivers. See the link for answer.

Question 2: What are the main steps in java to make JDBC connectivity?

Answer : Another beginner level JDBC Interview question, mostly asked on telephonic interviews. Here are main steps to connect to database.

Load the Driver: First step is to load the database specific driver which communicates with database.

Make Connection: Next step is get connection from the database using connection object, which is used to send SQL statement also and get result back from the database.

Get Statement Object: From connection object we can get statement object which is used to query the database

Execute the Query: Using statement object we execute the SQL or database query and get result set from the query.

Close the connection: After getting resultset and all required operation performed the last step should be closing the database connection.

Question 3: What is the mean of “dirty read“ in database?

Answer : This kind of JDBC interview question is asked on 2 to 4 years experience Java programmer, they are expected to familiar with database transaction and isolation level etc. As the name it self convey the meaning of dirty read “read the value which may or may not be correct”. in database when one transaction is executing and changing some field value same time some another transaction comes and read the change field value before first transaction commit or rollback the value ,which cause invalid value for that field, this scenario is known as dirty read.

Question 4: What is the 2 phase commit?

Answer : This is one of the most popular JDBC Interview question and asked at advanced level, mostly to senior Java developers on J2EE interviews. Two phase commit is used in distributed environment where multiple process take part in distributed transaction process. In simple word we can understand like if any transaction is executing and it will effect multiple database then two phase commit will be used to make all database synchronized with each other.

In two phase commit, commit or rollback is done by two phases:

1. Commit request phase: in this phase main process or coordinator process take vote of all other process that they are complete their process successfully and ready to commit if all the votes are “yes” then they go ahead for next phase. And if “No “then rollback is performed.

2. Commit phase: according to vote if all the votes are yes then commit is done.

Similarly when any transaction changes multiple database after execution of transaction it will issue pre commit command on each database and all database send acknowledgement and according to acknowledgement if all are positive transaction will issue the commit command otherwise rollback is done.

Question 5: What are different types of Statement?

Answer : This is another classical JDBC interview question. Variants are Difference between Statement, PreparedStatemetn and CallableStatement in Java. Statement object is used to send SQL query to database and get result from database, and we get statement object from connection object.

There are three types of statement:

1. Statement: it's a commonly used for getting data from database useful when we are using static SQL statement at runtime. it will not accept any parameter.

```
Statement   stmt = conn.createStatement( );
      ResultSet rs = stmt.executeQuery();
```

2. PreparedStatement: when we are using same SQL statement multiple time its is useful and it will accept parameter at runtime.

```
          String SQL = "Update stock SET limit = ?
WHERE stockType = ?";
      PreparedStatement  pstmt = conn.
prepareStatement(SQL);
      ResultSet rs = pstmt.executeQuery();
```

3. Callable Statement: when we want to access stored procedures then callable statement are useful and they also accept runtime parameter. It is called like this

```
    CallableStatement cs = con.prepareCall("{call
SHOW_SUPPLIERS}");
      ResultSet rs = cs.executeQuery();
```

Question 6: How cursor works in scrollable result set?

Answer : Another tough JDBC Interview question, not many Java programmer knows about using Cursor in Java.

In JDBC 2.0 API new feature is added to move cursor in resultset backward forward and also in a particular row .

There are three constant define in result set by which we can move cursor.

- **TYPE_FORWARD_ONLY:** creates a nonscrollable result set, that is, one in which the cursor moves only forward
- **TYPE_SCROLL_INSENSITIVE :** a scrollable result set does not reflects changes that are made to it while it is open
- **TYPE_SCROLL_SENSITIVE:** a scrollable result set reflects changes that are made to it while it is open

Question 7: What is connection pooling?

Answer : This is also one of the most popular question asked during JDBC Interviews. Connection pooling is the mechanism by which we reuse the recourse like connection objects which are needed to make connection with database .In this mechanism client are not required every time make new connection and then interact with database instead of that connection objects are stored in connection pool and client will get it from there. so it's a best way to share a server resources among the client and enhance the application performance. If you use Spring framework, then you can also refer How to setup JDBC Connection Pool using Spring in Java

Question 8: What do you mean by cold backup, hot backup?

Answer: This question is not directly related to JDBC but some time asked during JDBC interviews. The cold back is the backup technique in which backup of files are taken before the database restarted. In hot backup of files and table is taken at the same time when database is running. A warm is a recovery technique where all the tables are locked and users cannot access at the time of backing up data.

Question 9: What are the locking system in JDBC

Answer : One more tough JDBC question to understand and prepare. There are 2 types of locking in JDBC by which we can handle multiple user issue using the record. if two user are reading the same record then there is no issue but what if users are updating the record , in this case changes done by first user is gone by second user if he also update the same record .so we need some type of locking so no lost update.

Optimistic Locking: optimistic locking lock the record only when update take place. Optimistic locking does not use exclusive locks when reading

Pessimistic locking: in this record are locked as it selects the row to update

Question 10: Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?

Answer: No, we can open only one Statement object when using the JDBC-ODBC Bridge.

That's all on this list of 10 JDBC Interview questions with answers. As I said JDBC API and their concepts are integral parts of any Java interview and there is always at least one question from JDBC. Since most application uses a database in the backend, JDBC becomes critical for any Java developer.

STREAM AND FUNCTIONAL PROGRAMMING INTERVIEW QUESTIONS

The JDK 8 release has changed the way we write Java. With new functional programming idioms and a powerful Stream API, most of the new Java code is written in a functional style. This also means that Stream and Functional programming related questions are increasing on Java interviews. If you are not familiar with Java 8 changes, then it's difficult to crack a Java interview nowadays. Though it's not stated anywhere, most of the companies, particularly Investment banks like Barclays, Citi, and Goldman Sachs, now expect Java developers to know at least Java 8, which is also good, right? Java 11 is already out, and we are looking forward to Java 12 in a couple of months, it makes sense to know at least Java 8 changes.

All the questions are meant for Java developers and related to whatever functional programming concepts and tools available to Java developers in JDK 8.

To be honest, even though these are frequently asked Java 8 Interview questions, they are not at all very

difficult and just focus on essential Java 8 concepts, rather than it's complex implementation.

They are also not functional heavy, but, if you are interested in true functional programming something like the Scala and Haskel has implemented, I suggest you check the Functional Programming for Beginners course with Scala. I bought it for just $11 a couple of days ago, and it's really awesome.

Without any further ado, here is a list of some of the common yet popular functional programming and Stream interview questions for Java programmers. If you have any other Java functional programming question or anything related to JDK 8 Stream API, feel free to share with us, and I'll try to find an answer for you.

1. What is the difference between Collection and Stream?

The main difference between a Collection and Stream is that Collection contains their elements, but Stream doesn't. Stream work on a view where elements are actually stored by Collection or array, but unlike other views, any change made on Stream doesn't reflect on the original Collection.

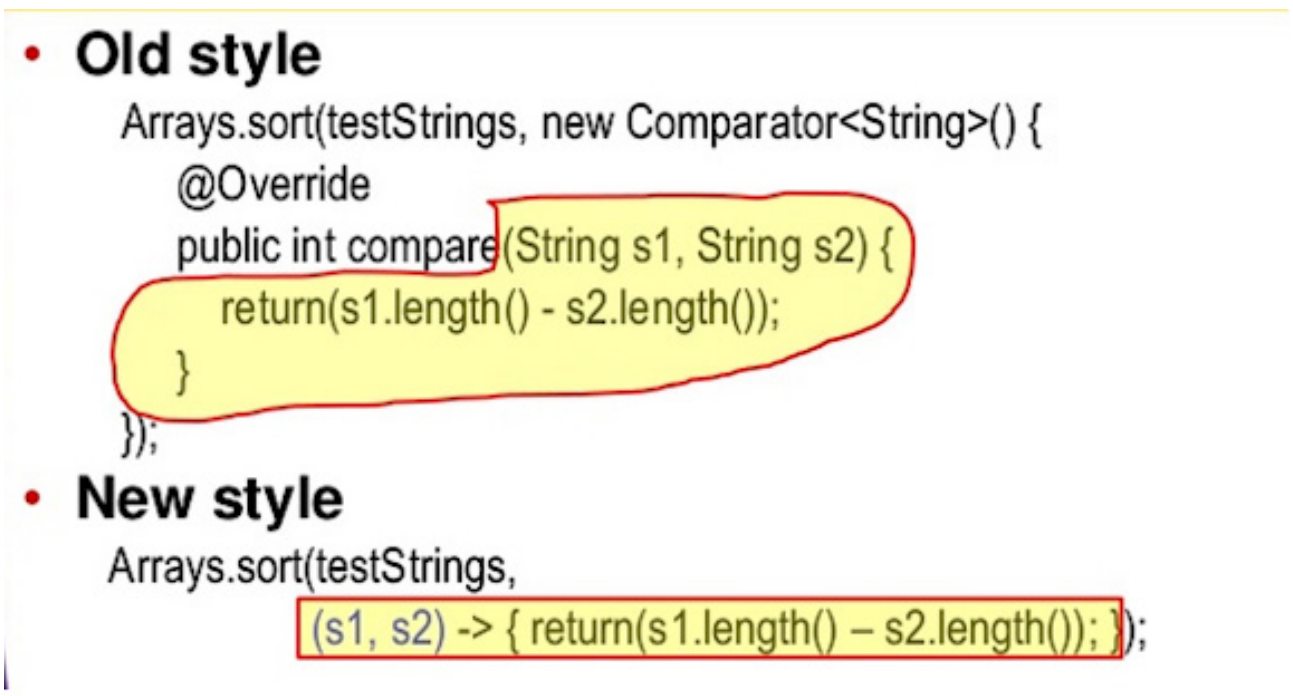
2. What does the map() function do? why you use it?

The map() function perform map functional operation in Java. This means it can transform one type of object to others by applying a function.

For example, if you have a List of String and you want to convert that to a List of Integer, you can use map() to do so.

Just supply a function to convert String to Integer e.g., parseInt() to map() and it will apply that to all elements of List and give you a List of Integer. In other words, the map can convert one object to another.

If you want to learn more about these new methods introduced in Java S.E. 8, I suggest you take a look at The Complete Java Masterclass course on Udemy. One of the most comprehensive courses which cover everything Java developers need to know.



3. What does the filter() method do? when you use it?

The filter method is used to filter elements that satisfy a certain condition that is specified using a Predicate function.

A predicate function is nothing but a function that takes an Object and returns a boolean. For example, if you have a List of Integer and you want a list of even integers.

In this case, you can use the filter to achieve that. You supply a function to check if a number is even or odd, just like this function and filter will apply this to stream elements and filter the elements which satisfy the condition and which doesn't.

4. What does the flatmap() function do? why you need it?

The flatmap function is an extension of the map function. Apart from transforming one object into another, it can also flatten it.

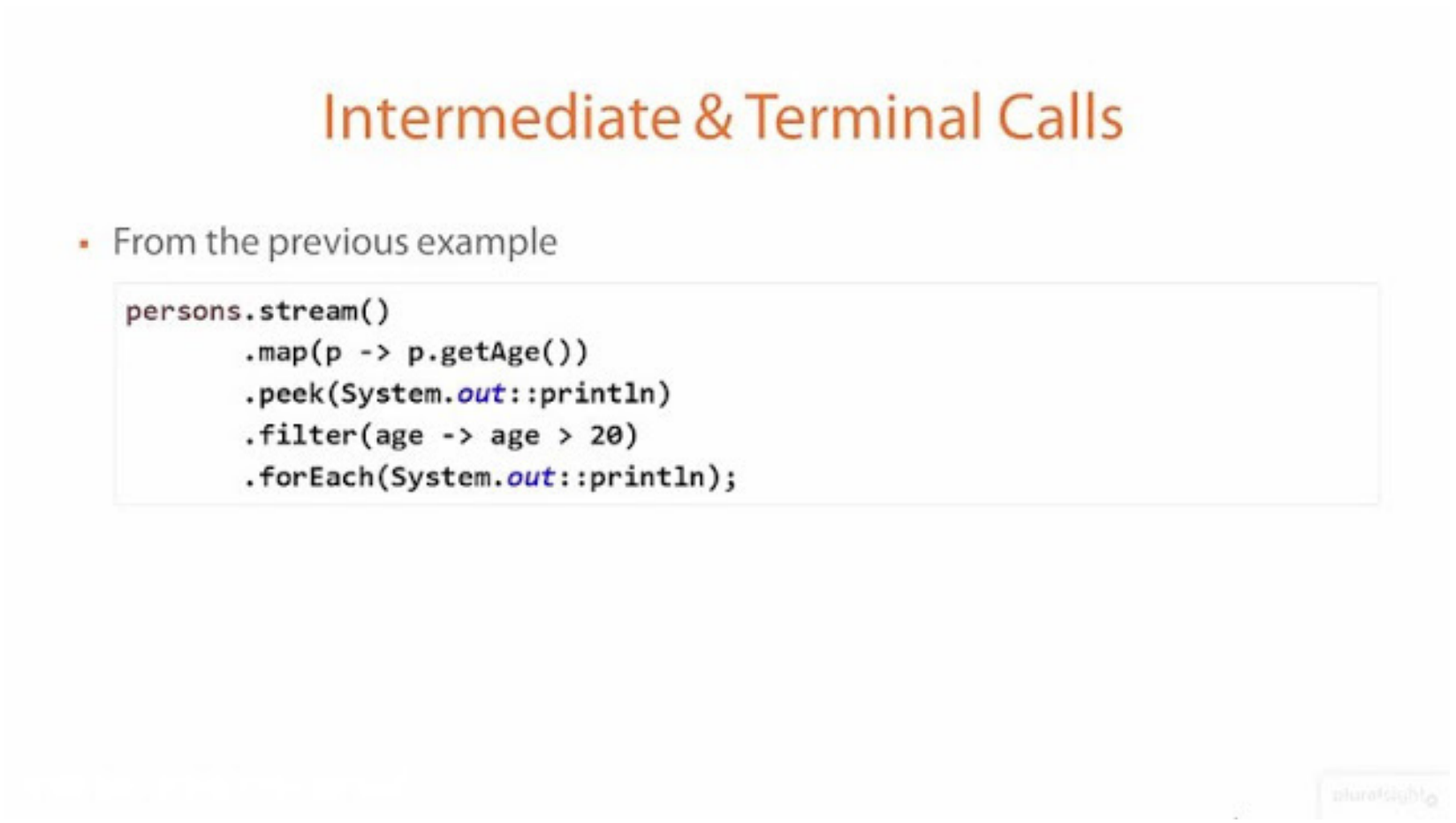
For example, if you have a list of the list but you want to combine all elements of lists into just one list. In this case, you can use flatMap() for flattening. At the same time, you can also transform an object like you do use map() function.

5. What is difference between flatMap() and map() functions?

Even though both map() and flatMap() can be used to transform one object to another by applying a method on each element.

The main difference is that flatMap() can also flatten the Stream. For example, if you have a list of the list, then you can convert it to a big list by using flatMap() function.

Btw, if you find trouble understanding these functional programming methods like map, flatMap, filter, etc., I suggest you go through From Collections to Streams in Java 8 course on Pluralsight. It's an advanced course but very good.



By the way, you would need a Pluralsight membership to access this course, which costs around $29 monthly or $299 annually (14% discount). I have one, and I also suggest all developers have that plan because Pluralsight is like NetFlix for Software developers.

It has more than 5000+ good quality courses on all the latest topics. Since we programmers have to learn new things every day, an investment of $299 USD is not bad.

Btw, it also offers a 10-day free trial without any obligation, which allows you to watch 200 hours of content. You can watch this course for free by signing for that trial.

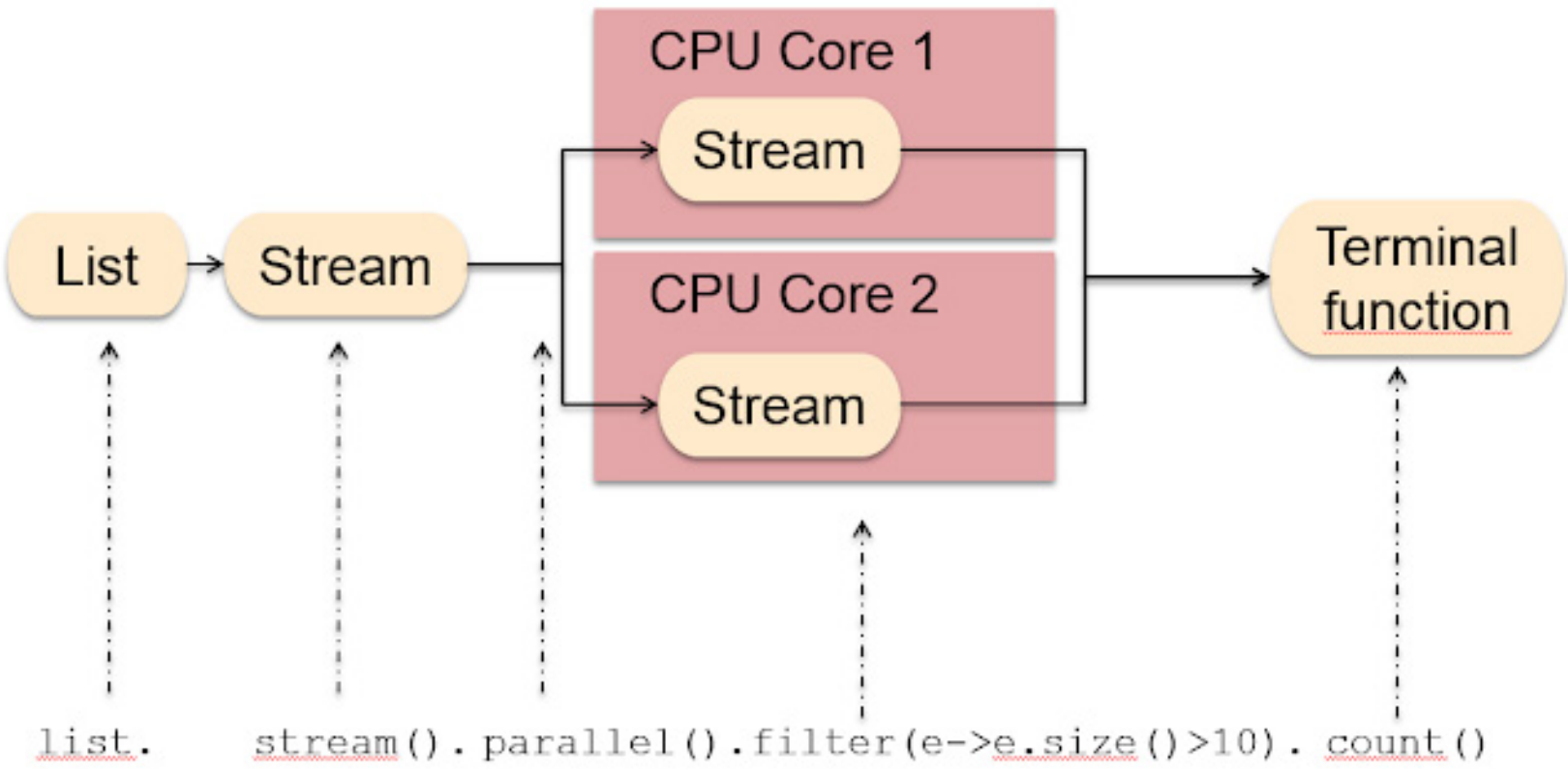
6. What is the difference between intermediate and terminal operations on Stream?

The intermediate Stream operation returns another Stream, which means you can further call other methods of Stream class to compose a pipeline.

For example after calling map() or flatMap() you can still call filter() method on Stream.

On the other hand, the terminal operation produces a result other than Streams like a value or a Collection.

Once a terminal method like forEach() or collect() is called, you cannot call any other method of Stream or reuse the Stream.



7. What does the peek() method do? When should you use it?

The peek() method of Stream class allows you to see through a Stream pipeline. You can peek through each step and print meaningful messages on the console. It's generally used for debugging issues related to lambda expression and Stream processing.

8. What do you mean by saying Stream is lazy?

When we say Stream is lazy, we mean that most of the methods defined on Java .util.stream.Stream class is lazy i.e. they will not work by just including them on Stream pipeline.

They only work when you call a terminal method on the Stream and finish as soon as they find the data they are looking for rather than scanning through the whole set of data.

9. What is a functional interface in Java 8?

As the name suggests, a functional interface is an interface that represents a function. Technically, an interface with just one abstract method is called a functional interface.

You can also use @FunctionalInterface to annotated a functional interface. In that case, the compiler will verify if the interface actually contains just one abstract method or not. It's like the @Override annotation, which prevents you from accidental errors.

Another useful thing to know is that If a method accepts a functional interface, then you can pass a lambda expression to it.

Some examples of the functional interface are Runnable, Callable, Comparator, and Comparable from old API and Supplier, Consumer, and Predicate, etc. from new function API. You can learn more about those interfaces in Java What's New in Java 8 on Pluralsight.

10. What is the difference between a normal and functional interface in Java?

The normal interface in Java can contain any number of abstract methods, while the functional interface can only contain just one abstract method.

You might be thinking why they are called functional

interfaces? Once you know the answer, it might be a little easier for you to remember the concept.

Well, they are called functional interfaces because they wrap a function as an interface. The function is represented by the single abstract method on the interface.

11. What is the difference between the findFirst() and findAny() method?

The findFirst() method will return the first element meeting the criterion i.e. Predicate, while the findAny() method will return any element meeting the criterion, very useful while working with a parallel stream. You can further see this article for a working example of the findFirst() method in Java 8.

12. What is a Predicate interface?

A Predicate is a functional interface that represents a function, which takes an Object and returns a boolean. It is used in several Stream methods like filter(), which uses Predicate to filter unwanted elements.

here is how a Predicate function looks like:

```
public boolean test(T object){
   return boolean;
}
```

You can see it just has one test() method which takes an object and returns a boolean. The method is used to

test a condition if it passes; it returns true otherwise false.

13. What are Supplier and Consumer Functional interface?

The Supplier is a functional interface that returns an object. It's similar to the factory method or new(), which returns an object.

The Supplier has get() functional method, which doesn't take any argument and return an object of type T. This means you can use it anytime you need an object.

Since it is a functional interface, you can also use it as the assignment target for a lambda expression or method reference.

A Consumer is also a functional interface in JDK 8, which represents an operation that accepts a single input argument and returns no result.

Unlike other functional interfaces, Consumer is expected to operate via side-effects. The functional method of Consumer is accept(T t), and because it's a functional interface, you can use it as the assignment target for a lambda expression or method interface in Java 8.

14. Can you convert an array to Stream? How?

Yes, you can convert an array to Stream in Java. The Stream class provides a factory method to create a Stream from an array, like Stream .of(T ...) which accepts a variable argument, that means you can also pass an array to it as shown in the following example:

```
String[] languages = {“Java”, “Python”,
“JavaScript”};
Stream numbers = Stream.of(languages);
numbers.forEach(System.out::println);

Output:
Java
Python
JavaScript
```

15. What is the parallel Stream? How can you get a parallel stream from a List?

A parallel stream can parallel execute stream processing tasks. For example, if you have a parallel stream of 1 million orders and you are looking for orders worth more than 1 million, then you can use a filter to do that.

Unlike sequential Stream, the parallel Stream can launch multiple threads to search for those orders on the different part of Stream and then combine the result.

In short, the parallel Stream can paralyze execution,

but, as Cay S. Horstman mentioned in Core Java S.E. 9 for the Impatient, there is significant overhead or parallelism, which only pays off if you are doing bulk data operation.

That's all about some of the common Java 8 Stream and Functional programming concepts based interview questions. Since Java 8 is now the modern way of writing Java code, more and more companies are looking for Java developers with good Java 8 skills.

Appendix

My favorite tip to Crack Java Interview

Hello guys, I want to share one tip with you which have helped me a lot on my Java interviews. I have given 10s of Java interviews and also taken many Java interviews and I have often seen used this tactic to make lasting impression on Interviews.

The trick here is drive the interview to your strength. Java is vast and in a limited time like 1-hour interview, its not possible for anyone to cover everything. Interviewer often have a list of questions to screen candidates during telephonic round but in face-to-face round they mostly like to check the depth of candidate's knowledge and that's where you can use this tip.

For example, if interviewer asked you about HashMap, you can not only answer his question about HashMap but also mention related details like hashing, how equals() and hashcode() are used, and how objects are stored in the HashMap (linked list vs binary tree), provided you know these topics well. When you mention these keywords, the interviewer are more likely to ask question on those topics and then you can showcase your knowledge and drive the interview.

The trick here is to high light the topic you know well and don't mention anything addition which you don't know well. This tip is really simple and once you start giving interview, you will realize that you are doing that, but if you don't do it now, I suggest setup mock interviews with your friends or online and see if you are using tip for your benefit.

All the best for your Java interview.

Index