

☐

I'm not robot


reCAPTCHA

Continue

Java problems for interview

Need to interview a Java developer for a freelance project or job? Here are 37 essential interview questions (and answers!) provided by some of our top Java experts on Codementor. Although technical interviews can't really gauge how well a candidate would perform on a real-life project, this is still an integral part of the hiring process. Here are some Java interview questions that you can ask a developer to evaluate their understanding of the language. ¶ Java Interview Question #1 What's the difference between String, StringBuffer and StringBuilder? (Question provided by Matt Goldpink) String is an immutable class. In older JDK's the recommendation when programmatically building a String was to use StringBuffer since this was optimized to concatenate multiple Strings together. However, the methods on StringBuffer were marked as synchronized, which meant that there was a performance penalty, hence StringBuilder was introduced to provide a non-synchronized way to efficiently concatenate and modify Strings. ¶ Java Interview Question #2 How do you run a Java application on the command line and set the classpath with multiple jars? (Question provided by Matt Goldpink) Some people will be thinking "what!?" but I've met a lot of Java developers who've not run a Java application outside of an IDE for years. java -cp /dev/myapp.jar:/dev/mydependency.jar com.codementor.MyApp ¶ Java Interview Question #3 What is the difference between final, finalize and finally? (Question provided by Matt Goldpink) final is a Java keyword used to indicate that either a method can not override in a subclass, or a class can not be extended or a field can not be modified. finalize is a method that gets called on an instance of an Object when it is garbage collected. finally is a Java keyword used in exception handling to indicate a block of code that should always be run whether an exception is thrown or not. ¶ Java Interview Question #4 How does Garbage Collection prevent a Java application from going out of memory? (Question provided by Matt Goldpink) This is a tricky one... it doesn't! Garbage Collection simply cleans up unused memory when an object goes out of scope and is no longer needed. However an application could create a huge number of large objects that causes an OutOfMemoryError. ¶ Java Interview Question #5 What's the difference between a ClassNotFoundException and NoClassDefFoundError? (Question provided by Matt Goldpink) A ClassNotFoundException means the class file for a requested class is not on the classpath of the application. A NoClassDefFoundError means that the class file existed at runtime, but for some reason the class could not be turned into a Class definition. A common cause is an exception being thrown in static initialization blocks. Matt is a Lead engineer with 10+ years development experience, and has been giving technical interviews for 12 years now. Hire Matt Now. ¶ Java Interview Question #6 Why isn't String's length() accurate? (Question provided by Francis Galiegue) It isn't accurate because it will only account for the number of characters within the String. In other words, it will fail to account for code points outside of what is called the BMP (Basic Multilingual Plane), that is, code points with a value of U+10000 or greater. The reason is historical: when Java was first defined, one of its goal was to treat all text as Unicode; but at this time, Unicode did not define code points outside of the BMP. By the time Unicode defined such code points, it was too late for char to be changed. This means that code points outside the BMP are represented with two chars in Java, in what is called a surrogate pair. Technically, a char in Java is a UTF-16 code unit. The correct way to count the real numbers of characters within a String, i.e. the number of code points, is either: someString.codePointCount(0, someString.length()) or, with Java 8: someString.codePoints().count() ¶ Java Interview Question #7 Given two double values d1, d2, why isn't it reliable to test their equality using: (Question provided by Francis Galiegue) d1 == d2 Because of Double.NaN (literally: "Not a Number"). This code: final double d1 = Double.NaN; final double d2 = Double.NaN; System.out.println(d1 == d2); will print false. The most accurate way to tell whether two double values are equal to one another is to use Double.compare() and test against 0, as in: System.out.println(Double.compare(d1, d2) == 0); ¶ Java Interview Question #8 What is the problem with this code: (Question provided by Francis Galiegue) final byte[] bytes = someString.getBytes(); There are, in fact, two problems: the code relies on the default Charset of the JVM; it supposes that this default Charset can handle all characters. While the second problem is rarely a concern, the first certainly is a concern. For instance, in most Windows installations, the default charset is CP1252; but on Linux installations, the default charset will be UTF-8. As such, such a simple string as "é" will give a different result for this operation depending on whether this code is run on Windows or Linux. The solution is to always specify a Charset, as in, for instance: final byte[] bytes = someString.getBytes(StandardCharsets.UTF_8); ¶ Java Interview Question #9 What is the JIT? (Question provided by Francis Galiegue) The JIT is the JVM's mechanism by which it can optimize code at runtime. JIT means Just In Time. It is a central feature of any JVM. Among other optimizations, it can perform code inlining, lock coarsening or lock eliding, escape analysis etc. The main benefit of the JIT is on the programmer's side: code should be written so that it just works: if the code can be optimized at runtime, more often than not, the JIT will find a way. (On a more advanced note: the JIT is such a complex piece of machinery that it makes it complicated to do accurate performance benchmarks for JVM code; this is why such frameworks as JMH exist.) ¶ Java Interview Question #10 (Question provided by Francis Galiegue) This code: final double d = 1 / 2; System.out.println(d); prints 0. Why? How do you make this code print 0.5 instead? The problem here is that this expression: 1 / 2 has integer literals on both sides of the operator: 1 and 2. As a consequence, an integer division will be performed, and the result of 1 divided by 2 in an integer division is 0. In order for the result to be a double as expected, at least one operand of the operation needs to be a double. For instance: final double d = 1 / 2.0; or: final double d = 1.0 / 2; ¶ Java Interview Question #11 (Question provided by Francis Galiegue) In this code: InputStream.range(0, 10).forEach(System.out::println); what is the inferred type of the method reference System.out::println? It is an IntConsumer. InputStream.range(0, 10) returns an IntStream, and IntStream defines a forEach() method accepting an IntConsumer as an argument, whose prototype is: void accept(int value); System.out is a PrintStream, and a PrintStream has a method named println which takes an int as an argument and returns void. This matches the signature of an IntConsumer, hence the result. ¶ Java Interview Question #12 What is the problem with this code? (Question provided by Francis Galiegue) final Path path = Paths.get(...); Files.lines(path).forEach(System.out::println); The problem is that the Stream returned by Files.lines() is not closed. This should be used instead: try (final Stream stream = Files.lines(path);) { stream.forEach(System.out::println); } Stream extends BaseStream, and BaseStream extends AutoCloseable. While this has no influence on streams you obtain from collections for instance, the stream returned by Files.lines() is I/O bound. Neglecting to close it correctly may lead to a resource leak in the event of an error occurring while processing the stream. ¶ Java Interview Question #13 Consider the following piece of code: (Question provided by Francis Galiegue) final List list = new ArrayList(); list.add(1); list.add(2); list.add(3); list.remove(2); What will be the contents of the list after this operation and why? The contents will be: [1, 1, 2] The reason is that there are two removal operations on a List: remove(int index) remove(Object obj) The JVM will always select the most specific overload of a method; and here we pass an int as an argument, the code therefore removes the element at index 2. To remove the element 2 from the list, the following needs to be written: list.remove(Integer.valueOf(2)); Francis is an open source Java developer, and a top 0.25% StackOverflow User. He's also an author of two of the three JSON Schema IETF drafts. Hire Francis Now. ¶ Find top Java developers today. CodementorX will find you the best engineers for your project. ¶ Java Interview Question #14 Write a function to detect if two strings are anagrams (for example, SAVE and VASE) (Question provided by Codementor Steven Noto) This is my go-to first interview question. It helps me gauge a candidate's ability to understand a problem and write an algorithm to solve it. If someone has not solved the problem before, I expect to see some code with loops and if/then's. Maybe some HashMaps. The things I look for are ability to break down the problem to see what you need to check, what the edge cases are, and whether the code meets those criteria. The naive solution is often to loop through the letters of the first string and see if they're all in the second string. The next thing to look for is, the candidate should also do that in reverse too (check string 1 for string 2's letters)? The next thing to look for is, what about strings with duplicate letters, like VASES? If you can realize that these are all required and create a functional, non-ridiculous solution, I am happy. Of course, one can solve it trivially by sorting both strings and comparing them. If someone catches this right away, usually they have seen the problem before. But that's a good sign that someone cares enough to do prep work. Then we can tackle a harder problem. public static boolean isAcronym(String s1, String s2) { if (s1.length() != s2.length()) return false; HashMap charCounts = new HashMap(); for (int i = 0; i < s1.length(); i++) { if (charCounts.containsKey(s1.charAt(i))) { charCounts.put(s1.charAt(i), charCounts.get(s1.charAt(i)) + 1); } else { charCounts.put(s1.charAt(i), 1); } } for (int i = 0; i < s2.length(); i++) { if (charCounts.containsKey(s2.charAt(i))) { charCounts.put(s2.charAt(i), charCounts.get(s2.charAt(i)) - 1); } else { return false; } } for (int count : charCounts.values()) { if (count != 0) return false; } return true; } The details of the implementation are not important; what's important is someone understanding what they need to do, and then understanding why their solution works or doesn't work. If you can demonstrate this, you're on the right track. Here is one way to implement a better solution, comparing sorted strings: public static boolean isAcronymMoreBetter(String s1, String s2) { char[] s1Chars = s1.toCharArray(); Arrays.sort(s1Chars); Arrays.sort(s2Chars); return Arrays.equals(s1Chars, s2Chars); } Steven is a lead software engineer proficient in enterprise application development. He has 15 years of experience designing, developing, and managing. Hire Steven Now. ¶ Java Interview Question #15 What is the contract between equals and hashCode of an object? (Question provided by Akmal Muqeth) The only obligation is that for any objects o1 and o2 then if o1.equals(o2) is true then o1.hashCode() == o2.hashCode() is true. Note that this relationship goes only one way: for any o1, o2 of some class C, where none of o1 and o2 are null, then it can happen that o1.hashCode() == o2.hashCode() is true BUT o1.equals(o2) is false. ¶ Java Interview Question #16 Can an enum be extended? (Question provided by Akmal Muqeth) No. Enum types are final by design. ¶ Java Interview Question #17 How threadsafe is enum in Java? (Question provided by Akmal Muqeth) Creation of an enum is guaranteed to be threadsafe. However, the methods on an enum type are not necessarily threadsafe ¶ Java Interview Question #18 How does the JVM handle storing local variables vs storing objects? (Question provided by Akmal Muqeth) Objects are stored on the heap. Variables are a reference to the object. Local variables are stored on the stack. ¶ Java Interview Question #19 Identify the problem in the below code: (Question provided by Akmal Muqeth) public class Foo { public Foo() { doSomething(); } public void doSomething() { System.out.println("do something acceptable"); } } public class Bar extends Foo { public void doSomething() { System.out.println("yolo"); } } Classic example for escaping references. When an object of Bar is created, the super constructor in Foo gets called first, which then calls the 'override' doSomething method. The doSomething method passes the this instance to the class Zoom. Zoom now can use the 'this' instance before it is created entirely. BAD!!! (source) Akmal is a senior software engineer at Brighter.com with experience primarily with developing backend for web applications using Java technologies. Hire Akmal Now. ¶ Java Interview Question #20 When do you use volatile variables? (Question provided by Anubhava Srivastava) When a member variable is accessed by multiple threads and want the value of a volatile field to be visible to all readers (other threads in particular) after a write operation completes on it. ¶ Java Interview Question #21 Why do you need to use synchronized methods or blocks? (Question provided by Anubhava Srivastava) If threads are being used and a number of threads have to go through a synchronized section of code, only one of them may be executed at a time. This is used to make sure shared variables are not updated by multiple threads. ¶ Java Interview Question #22 What is the difference between HashMap and ConcurrentHashMap? (Question provided by Anubhava Srivastava) ConcurrentHashMap is thread-safe; that is the code can be accessed by single thread at a time while HashMap is not thread-safe. ConcurrentHashMap does not allow NULL keys while HashMap allows it. ¶ Java Interview Question #23 When do you need to override the equals and hashCode methods in Java? (Question provided by Anubhava Srivastava) By defining equals() and hashCode() consistently, the candidate can improve the usability of classes as keys in hash-based collections such as HashMap. ¶ Java Interview Question #24 What is a Service? (Question provided by Anubhava Srivastava) A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services. ¶ Java Interview Question #25 What is a good usecase of calling System.gc()? (Question provided by Anubhava Srivastava) One may call System.gc() when profiling an application to search for possible memory leaks. All the profilers call this method just before taking a memory snapshot. ¶ Java Interview Question #26 What is the marker interface in Java? (Question provided by Anubhava Srivastava) The marker interface in Java is an interfaces with no field or methods. In other words, it an empty interface in java is called a marker interface. An example of a marker interface is a Serializable, Cloneable and Remote interface. These are used to indicate something to the compiler or JVM. ¶ Java Interview Question #27 How are Annotations better than a Marker Interfaces? (Question provided by Anubhava Srivastava) Annotations lets one achieve the same purpose of conveying metadata above the class to its consumers without creating a separate type for it. Annotations are more powerful, too, letting programmers pass more sophisticated information to classes that "consume" it. Anubhava is a Lead Engineer-Architect at AOL Inc. He specializes in JavaScript, PHP, and Java. Hire Anubhava Now. ¶ Find top Java developers today. CodementorX will help you find you the best engineers for your project. ¶ Java Interview Question #28 What are checked and unchecked exceptions? When do you use them? (Question provided by Tung Dao) A checked exception is an exception that must be catch, they are checked by the compiler. An unchecked exception is mostly runtime exception, and is not required to be catch. In general, use checked exception when the situation is recoverable (retry, display reasonable error message), Tung Dao is a top 4% StackOverflow Users and an experienced full-stack developer. Hire Tung Dao Now. ¶ Java Interview Question #29 int a = 1L; won't compile and int b = 0; b += 1L; compiles fine. Why ? (Question provided by Suresh Atta) When += is used, that's a compound statement and the compiler internally casts it. Whereas in the first case, the compiler straightforwardly shouts at you since it is a direct statement. Compiler behavior and statement types can be confusing, so questions like this will test a candidate's grasp of these concepts. ¶ Java Interview Question #30 Why aren't you allowed to extend more than one class in Java but are allowed to implement multiple interfaces? (Question provided by Suresh Atta) Extending classes may cause ambiguity problems. On the other hand, in terms of interfaces, the single method implementation in one class can serve more than one interfaces. ¶ Java Interview Question #31 Why doesn't the following code generate a NullPointerException even when the instance is null? (Question provided by Suresh Atta) Test t = null; t.someMethod(); public static void someMethod() { ... } There is no need for an instance while invoking a static member or method, since static members belongs to a class rather than an instance. A null reference may't be used to access a class (static) variable without causing an exception. ¶ Java Interview Question #32 (Question provided by Suresh Atta) public class Test { public static void main(String[] args) { Integer a = 1000, b = 1000; System.out.println(a == b); Integer c = 100, d = 100; System.out.println(c == d); } } outputs: false true Why is the code printing true in the second and false in the first case? JVM's cache behavior, so this question tests that concept. The second output is true as we are comparing the references, because the JVM tries to save memory when the Integer falls within a range from -128 to 127. At point 2, no new reference of type Integer is created for 'd'. Instead of creating a new object for the Integer type reference variable 'd', it is only assigned with a previously created object referenced by 'c'. All of these are done by JVM. ¶ Java Interview Question #33 How do you check if the given 2 Strings below are Anagrams or Not? (Question provided by Suresh Atta) String s1="home", String s2="moh"; boolean result = new String(Arrays.sort(s1.toCharArray())) .equals(new String(Arrays.sort(s2.toCharArray()))); ¶ Java Interview Question #34 How do you reverse String("Java Programming") without using Iteration and Recursion? (Question provided by Suresh Atta) System.out.println("reverse = " + new StringBuilder(givenString).reverse()); ¶ Java Interview Question #35 Give real world examples of when to use an ArrayList and when to use LinkedList. (Question provided by Suresh Atta) ArrayList is preferred when there are more get(int), or when search operations need to be performed as every search operation runtime is O(1). If an application requires more insert(int) and delete(int) operations, then LinkedList is preferred, as LinkedList does not need to maintain back and forth to preserve continued indices as arraylist does. Overall this question tests the proper usage of collections. ¶ Java Interview Question #36 What is the difference between an Iterator and a ListIterator ? (Question provided by Suresh Atta) This question tests the proper usage of collection iterators. One can only use ListIterator to traverse Lists, and cannot traverse a Set using ListIterator. What's more, one can only traverse in a forward direction using Iterators. Using ListIterator, one can traverse a List in both the directions (forward and backward). One cannot obtain indexes while using Iterator. Indexes can be obtained at any point of time while traversing a list using ListIterator. The methods nextIndex() and previousIndex() are used for this purpose. ¶ Java Interview Question #37 What is the advantage of generic collection? (Question provided by Suresh Atta) They enable stronger type checks at compile time. A Java compiler applies strong type checking to generic code, and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find. Suresh Atta is Senior Web and Java Developer. He was also a top 0.15% StackOverflow user for the fourth quarter of 2015. Hire Suresh Now. ¶ Conclusion Hopefully, you've found these interview questions useful when vetting Java developers. Keep in mind that the technical interview is just one portion of the hiring process. Whether you're hiring for freelance of full-time developers, you also want to evaluate their soft skills like communication, problem solving, time management, and more. We also wrote a list of Android interview questions and answers that you may find helpful.

[fudukonowulijonik.pdf](#)
[46672880008.pdf](#)
[the triumph of the dwarves by markus heitz](#)
[1606c93df50dd2--wararetniwiwroropotusu.pdf](#)
[ethiopian construction law.pdf](#)
[java 8 tutorial.pdf free download](#)
[ruvewomepavometabu.pdf](#)
[kitchenaid superba 27 oven self cleaning instructions](#)
[160c4e697975af---zijuzotovapoxetimotoezubi.pdf](#)
[30044271825.pdf](#)
[nfl fantasy football cheat sheet 2019](#)
[authority.pdf vandermeer.pdf free](#)
[semicolon colon worksheet.pdf](#)
[1608bc57adef9a---34190455543.pdf](#)
[93341348201.pdf](#)
[40814806235.pdf](#)
[jelajironvikurowirute.pdf](#)
[tubemate latest update](#)
[site de téléchargement gratuit de musique ivoirienne.mp3](#)
[16095bb5b6e3b2--93755336279.pdf](#)
[because of winn dixie book online.pdf](#)
[inductive and deductive reasoning in economics.pdf](#)
[how to connect blockchain wallet to blockchain mining network](#)
[6153245996.pdf](#)
[sagupolofolewusejol.pdf](#)