How do you judge a Java programmer with only 5 questions?

quora.com/How-do-you-judge-a-Java-programmer-with-only-5-questions

I have interviewed a lot of technology people over the years, including many Java programmers. Many of the answers already listed have good, specific questions. I would do it a little differently though. My 5 questions aren't programming language specific either. I really don't care if they answer with Java, PHP, C, Lua, or pseudo code.

First, you have to have the interview room set up. Get a computer with an IDE and hook it up to a monitor so the interviewers can see. A laptop is ideal because the interviewee has their own screen and the interviewers can view a large monitor. Have the IDE ready to go so the interviewee can sit at it and write code for you. Also have some code for a project the interviewee is likely to work on ready to go too. It's also a good idea to have a whiteboard. Now the questions...

1. Tell me about either your current project or your favorite project, whichever one you'll be able to speak to in most technical depth. (This should be about a 2 minute discussion.)

2. Please take about 10 minutes and demonstrate a quick class or method you might create for that project using the IDE on the computer in front of you. It might feel a little odd, but talk through your thought process while you are coding. We're not interested in the code being 100% correct--we won't be compiling or running it. Trust me, none of us writes draft code that is 100% correct with other people breathing down our neck. :) We're also not interested in whether you know the shortcuts for this particular IDE.

3. What are some ways you might improve that class or method? You can use the whiteboard or the IDE or whatever feels most natural. (Spend about 5 minutes here.)

4. Now let's look at some of the code we have for a project you would likely be working on. (At this point, one of the interviewers should switch to the project code in the IDE.) What are your thoughts about the code? Don't worry, we're thick skinned and code review ourselves on a regular basis. (This can take another 5-10 minutes.)

5. What questions do you have for us? (Probably about 5 minutes, leading to roughly a 30 minute interview.)

A lot of other questions, like those Miguel Paraz listed, will naturally flow from the discussion. The point is to get them talking about something they know about so they aren't trying to memorize programming trivia questions. Having them demonstrate code will tell you a lot about how they think when coding. The code doesn't have to be correct, but it should be logical. Even if they completely screw up question 2, they have a chance to redeem themselves in question 3. As User said in his answer, it isn't practical to know everything about a language all the time. About 15 years ago I was part of a project that wrote a huge piece of enterprise software in Java. Today I probably wouldn't be able to answer too many specific technical questions about some of the stuff we used back then, but I would know where to look if I had to build something with them. I'd feel perfectly comfortable mocking some code up related to the project using things I've worked with more recently too.

I actually taught Java as a Lecturer for years. I usually give my students problems & projects that are not in the standard materials. Things that they have to look up in the API/manual.

If I was an interviewer for a job, I would much rather know how good the interviewee knows about OO concepts, and what they understand about the Java API & documentation. As I would want them to maybe obtain some java packages and use them to implement solutions.

I'd ask:

1. So you learned that it is a good practice to declare instance variables in a

class as private, and only use accessor/mutator methods to manipulate them. Why do we want to do that? Can you give me an example.
    1. *A good answer would involve talking about how we might want to adjust how variables are manipulated. What if we decided to get polar coordinates instead of cartesian coordinates. We would have to change code in many places.*
2. We talk about polymorphism in OO. Do you know how & why polymorphism works? Can you mention examples in Java API where Polymorphism is used. Have you designed programs or classes with polymorphism in mind.
    1. *Collections & Sorting methods on them are an example I personally love! Comparable & CompareTo*
    2. *Talking about how the toString method is also good!*
3. Java 5 introduced Generics. How do you think Generics are helpful?
    1. *An okay, but not good answer:* They save time writing code so that I do not have to do many type casts.
    2. *Great Answer I would look for:* They improve the development process by shifting "checking" to compile time rather than run-time. It's better to find faults earlier in the development process.
4. "non-static variable XYZ cannot be referenced from a static context". Why is that a problem?
    1. *A good chat about objects being instances, and static being some sort of a class thing would suffice.*
5. How does Java passes method parameters? by value or by reference? Explain what you know about that. What if you want to create a swap method? Can how would you do that?
    1. *Java is a bit tricky*. Java always passes by value. However, references are considered values, so it passes the "value" or the "reference"

Conclusion Question if I was happy with the 5 answers: So you have worked a lot with Java. Where does Java disappoint you? Can we chat about what things you would change about the language if you could?

If the interviewee nailed it, I'd hire him!

11.7k views · View 46 Upvoters
Not limited to **Java**.

1. Pick 5 programming languages other than Java, explain when you'd choose them instead of Java and why.
2. Implement a URI parser.
3. Desgin your own programming language/environment/toolset, and explain why.
4. Suggestions/solutions for current business.
5. Why you'd like to be a programmer.

Actually to judge a man is not only from what's he doing and what's done by him, you should consider his experience, skillset, methodology and philosophy etc. to see if he is good and if he can be better.

20.2k views · View 43 Upvoters

I don't think you can make a very good judgement about a programmer in only 5 questions, but you can get some idea.

In interviews I used to ask a lot of detailed Java-specific questions like "what are the 8 primitive data types" and "what's the difference between method overloading and overriding" but have found they're not really that useful. Everyone could answer the second one and almost nobody could answer the first (most people forget short). Nowadays I'd focus more on finding out what kind of programmer they are, but without losing sight of the aim of making sure they know some Java. Here are 5 examples:

1. Have you read "The Pragmatic Programmer"?
2. What's your view on checked vs unchecked exceptions?
3. What's the problem with static singletons?
4. Can a Java object instance exist without any of its constructors having been called? If yes, in what situations?
5. Give a Java example of an Adapter, a Proxy, a Facade and a Visitor.

My reasons for asking, in case it's not clear:

1. If they've read it (or similar books), it tells me they care about programming as a craft, not just as a living
2. Something every Java programmer will have encountered, and so should

have formed some sort of view on - bonus points for being aware that there has been debate on this for some time in the Java community and what the consensus is - extra bonus points for their opinion agreeing with mine
3. Tells me something about how much testing they've done - bonus points if they've used a mocking library - more bonus points if they explain the problem in terms of general principles like encapsulation and contracts
4. An esoteric corner of Java, but anyone who's learnt Java really thoroughly should know of one. Bonus points for understanding reflection as well, extra bonus for having encountered multiple contexts where something like this happens.
5. Tests whether they're aware of GoF patterns which are an important communication tool between Java programmers. Bonus points for accurate Java examples, especially of Visitor which is necessarily more complex than the others.

These questions test not just some Java knowledge, but how much Java code the programmer has really worked with, and whether they understand some important principles, and whether they are going to be able to communicate well with other developers.

Don’t hire a Java programmer. They’re six-a-penny. Hire a software developer who understands that writing code is the easy part, and can tell you about stuff that matters, like requirements capture, design patterns, test planning, impact analysis, good logging practice and project documentation. If they’re a walking API reference, they probably don’t know what the internet’s for.

Update: It’s been (fairly) pointed out that I haven’t actually answered the given question, so I’ll address that here.

1. What are some common pit-falls when working with dates and/or times, and what can/should be done to mitigate them?
2. What is inversion of control, and why is it a good thing?
3. What design principles can be used to facilitate coding for accessibility?
4. In a service-oriented architecture, what considerations should be borne in mind when deciding between client-side and server-side validation?

5. Stakeholders are debating whether some existing behaviour should be changed, or left as-is. One of them asks you “Can’t you just make it configurable?” What is your answer?

It may be argued that these questions aren’t exactly tailored for Java programmers, but I’d consider them more important to *any* programmer than anything language-specific, and as such, they are questions I’d ask to gauge the quality of a Java programmer.

Update 2: Notes on the above:

1. In my experience, dates and times account for a disproportionate share of coding defects. Recognising common errors with time-zones (and their impact on dates), daylight saving time, locale-based formatting, etc. is vital. But more so are recognising the importance of suitable representations when persisting or (de)serialising data; *consistent* use of mature, non-lossy APIs throughout business logic; avoiding *any* string manipulation until last-minute rendering, and so on. These are design considerations; getting them right from the start makes coding a breeze. What’s more, these principles apply to pretty much every other complex data type; if a candidate can get dates/times right, the rest should follow.
2. Largely speaks for itself, and tests their “big picture” aptitude, but in particular, this a great opportunity for them to demonstrate an appreciation for test-driven development.
3. Understanding the uses of, say, encapsulation and abstraction is a good thing. But the real point of this question is to see whether a candidate is used to considering accessibility from the get-go. It should never be an afterthought.
4. Another “big picture” question, and good to see how security-conscious they are.
5. The point here isn’t to see how they’d handle stakeholders. It is to see whether they themselves recognise the ongoing costs (complexity, testing, maintenance, etc.) associated with adding configurable behaviour properly. If they don’t, there’s a good chance they will cut corners to deliver.

44.7k views · View 180 Upvoters

These are the questions I shall ask to understand how strongly the candidate understands the internals of java:

1. What is the difference between **final object and an immutable object**?
2. What is the difference between static and dynamic(runtime) polymorphism? Feel free to explain it indepth with the help of simple examples.
3. A discussion on stack and heap, storage of instance variables, garbage collection. Follow Up Questions:
    1. Can i leave the task of freeing up resources to finalize method? **Why or why not?**
    2. **Statics and Inheritance:** Can i override a static method? How do object references behave when you have static methods with same signatures in base and sub class?
4. Implement the following from scratch without the use of available packages (I shall get a hint of his prowess with algorithms, data structures and hands on coding):
    1. A hashmap with put and get methods.
    2. Quicksort.
    3. Stringbuilder class with append method.
    4. (Emphasis will be on using arrays and primitives wherever possible).
5. Lets talk about **Generics** for the next 20 minutes.....

I shall grill the candidate with follow up and if/else questions. Testing algorithms, data structures and OO design shall of course come in other rounds.

P.S: These questions are meant for candidates with 0–3 years of experience. Answering these questions with confidence not only demonstrates the command and passion the candidate has on the language but will also assure me that he would learn anything on the job with similar zeal.

3.7k views · View 14 Upvoters

A great question to answer with questions indeed... Judging a Java programmer with only 5 questions is certainly a difficult task as Java has a wide scope as a programming language. Still, if I am supposed to do that, I

will preferably choose the following 5 questions to judge a programmer's clarity on Java concepts.

Top 5 Java Questions to Judge a Java Programmer:

1. What will happen to the finally block if you put return statement or System.exit () on try or catch block?
2. If a method throws NullPointerException in the superclass, can we override it with a method which throws RuntimeException?
3. How do you ensure that N threads access N resources without deadlock?
4. What is the difference between factory and abstract factory pattern?
5. What is the difference between creating String as new() and literal?

So, now it's time for you to find out answers to the questions that you were not able to figure out... And if, no such question is there, then certainly you are great as a Java programmer...

Love Java!!! Live Java!!!

Some questions that will lead to interesting discussions and will help tell you more about the programmer. (And would anyone like to answer!)

- What programming languages do you use besides Java? How do they influence your Java programming? When do you choose to use Java?
- What are some complex systems have you built in Java? How does the Java code follow from the requirements? Could you explain them in terms of simpler systems, or in terms of frameworks and libraries they use?
- What is the most interesting standard Java API class, interface, or package? How does it work? How have you used it?
- What is Object-Oriented Programming? How does Java fulfill the requirements of OOP? Do you strictly follow the principles of Java OOP programming, or do you adjust it to your needs?
- How does the Java Virtual Machine work? How does the JVM work with bytecode, with memory allocation, and garbage collection?

We can ask questions related to some interesting Java solutions.

1. **Why do you like Java Programming language?** Typical answer from a candidate - more about OO and JVM (platform independence) .
2. **Explain more about Java Platforms you have worked so far.** Typical answer from a candidate - more about Java SE, Java EE and Java ME platforms. You can ask more questions related to the Java platform they worked on.
3. **Write Java Program solution for abstract class and interface?** You can ask more questions from the program solution about abstract class, abstract methods, interfaces, overloading, overriding and polymorphism.
4. **Which Java SE API you have used most frequently?** Based on the answer, you can ask more questions from the API chosen.
5. **What is the latest Java SE version you used in your projects?** You can ask questions related to Java SE version changes and new platform additions as well.

MyExamCloud Resources for Java Professionals:

Java Certifications MyExamCloud Exam Collections

Java Interview Questions MyExamCloud Exam Collections

MyExamCloud fun Magic Quiz :

Developed by Java Code in just 2 lines of algorithm: MyExamCloud Magic Quiz

1.1k views · View 9 Upvoters

Anonymous

Answered Sep 4, 2015

I've interviewed lot of Java programmers in last 4 year. I usually focus on these 4 areas:
1. core java basics 2. problem solving 3. design 4. communication skill

1. I usually start with basic core java questions to keep the ball rolling. I don't mind if interviewer is not able to answer some tricky questions which otherwise can be found out in seconds by goggling. Some of the questions i avoid asking or not deriving any conclusion if candidate do not know the answer are:

- difference between inner class and static inner class
- overriding the methods which throws exceptions
- how many areas are in java heap?
- why Object class is not abstract
- what is volatile?
- exception hierarchy
- what is string pool

I usually ask these questions to make sure that candidate is kind of aware about basic and to make him comfortable. If he can't answer correctly, I do not conclude that he is not a good java devloper. However i'd expect candidate to answer some of the conceptual question e.g.

- Detailed knowledge on synchronization. What would happen when thread enters the synchronized method? How the lock is acquired?
- thread pools
- overall idea on collection frame work
- working of garbage collector

2. For this part, I usually start with very basic problem solving (FizzBuzz type) question which do not need any advance knowledge of programming .
I really expect the candidate to come up with the solution. examples are:

- How would you find the integer occurring maximum no of times from array
- find out sum of all the elements in an array which is occurring more than 5 times in an array
- Given 2 integer array find out the subset.

All of these are very basic and easy problem solving questions. No specific knowledge of any advance concept or technology is required. My objective is

to eliminate those who can't even think on basic problems.

3. Design - For this part, my goal is how comfortable candidate is to come up with design given the scenario. Examples are

- How would you design the distributed cache? What things you need to consider?
- How would you implement call back?
- How would you implement your own hashmap/concurrent hashmap?
- How would you design your own Arraylist whose iterator iterates in reverse order (or skip the alternative element)?
- design connection pool

4. Communication - The only thing I check how comfortable candidate is on explaining the technical concepts/projects etc.. and how confident he is.
2.1k views · View 4 Upvoters

I would be very wary of depending on a few questions to give you insight into a person. Some people simply do better answering face-to-face questions than others. In most cases, it has no bearing on a person's ability to do the job at hand.

Whenever I see someone asking a question like this, I want to ask them "Have you ever followed up on how well your questions/interview process did in predicting a good employee"? If you do not use metrics to drive your interview process, you might as well ask them their astrological sign.
"Fabulous, I am also a Pisces! OMG!!!!!"
33 views · View 2 Upvoters

1. How long do you make your lines of code?

1. What is an acceptable value for cyclomatic complexity, for you?

1. What is an acceptable value for NPath complexity, for you?

1. How deep do you nest your *if*s?

1. How do you feel about hardcoding?

2. What comes first GUI or API?

3. Do you run your unit tests at each commit?

4. Do you run a static code analysis tool at each commit?

5. Have you thought about automating this with a local Jenkins(e.g.) so that you automate some of these tasks? Have you heard about Jenkins? What about Sonar?

8.4k views · View 11 Upvoters

Answered Feb 27, 2016

Implement a method that stores an ordered set of Objects, and supports three methods: insert(Object o), get(int index), and count(). For bonus points, use generic types. (I don't allow them to use Vector, or ArrayList; I want them to write this backed with an array or a hand-rolled linked list.)

What's the big-O runtime of your implementation of those three methods?

What tradeoffs did you make in choosing to build it that way?

If they get those three easily, they can write simple code, and have an idea of algorithmic analysis and design. At that point, we should have coffee and judge if they're a culture fit; "what have you learned recently" (are they still learning?), and "do you chuckle when you hear a painfully long class name, like the WidgetFactoryBuilderWidget?"

2.6k views · View 13 Upvoters

I assume we are talking about a junior programmer because the question looks to be focused on pure java only. So I would ask:

1.a question about a certain OOP aspect and how this is possible in java
2. a question on Java API
3. a question about a java feature in general (static, exceptions, inner classes, annotations, generics, ...)
4. a question focused on algorithms and how a given one is implemented using java (java api for instance collection is encouraged)
5. a problem solving question in general

If we are not talking about a junior profile then we need to consider the context:
- server side web programming

- real time applications
- database layer related
- networking in general
- languages on top of JVM
- interests in various types of frameworks (IoC, MVC, template, persistence, message-based, micro-service,...)

So especially for non junior (java) programmers I would say context is king.
5.9k views · View 7 Upvoters

1. Where do we use static and final keywords in java?
2. How to run a java program? Explain with intermediate steps and outputs.
3. How is Object-Oriented Programming implemented in java? Explain polymorphism and inheritance in java.
4. What is the mechanism of implementing threads in java?
5. Describe any built-in data structure or a GUI package in java.

These questions will be helpful to test knowledge of a java developer, since it unveils both basic and API level of understanding of Java.

3.9k views · View 5 Upvoters
Originally Answered: How do you judge with only 5 questions a student who just finished a CS bachelor, has no experience and wants to work as a Java programmer?

You don't. You ask to see some of their code on github or some other online repository. Make sure that they didn't just upload it once because it's most likely not their code then. Check to see their trail of updates and how often they push and ask to have them walk through the code with you and explain what they were doing and explain why. THAT is how you judge a java developer.

Anything else will be too specific as Java is a VAST library going on into the horizon. Some Java developers never mess with rewriting 'List', some don't use Trees because they don't need to search their datasets that often.

It is easiest to judge them based on their recent work.
3.5k views · View 8 Upvoters

You cannot judge any programmer with just 5 questions.
I have seen it happening at the place I work and it just doesn't work. If you hare a regular (not a specialist) what's most important is to get a good feel of the person you want to hire, not if he or she can answer 5 questions.
Talk to the guy, discuss the project's he has done before. Get a feel if this person will fit in your team and get a good understanding if his past work will match the work you will give him in the first year at his new spot.

In most/all situations you can by contract and low always get rid of somebody if there is no good match, it's ok if that happens. As always, communication is key..
2k views · View 5 Upvoters
1. Do you know C ?
No. I'm java guy.

2. Do you know C++?
No. As I said I'm java guy.

3. How about C# ?
No. I am a java professional.

4. Should earth have one more moon?
What?

5. Celestial body that rotates around earth is called moon. Isn't earth sun's moon?
what the fuck are you talking about.

Me: I was asked to ask you five questions. So after second question, I was incrementing the question count. Don't hold your breath waiting for the result of the interview.

I'm a freaking C guy and I'm so damn biased towards java.

65 views

Anonymous

Answered May 2, 2015

I usually do it with three questions, and it works for any programming language.

Explain, and give examples of

1) Abstraction
2) Inheritance
3) Interfaces

The clarity, and level of understanding to these three basic principles will be very telling. No need for trick question, language specific stuff.

If you get someone who is fluent on these concepts, you can move onto asynchronous programming (callbacks) or other such topics.
536 views

After seeing so many Java developers I can tell you one thing, a good java developer is hard to find and you can't judge someone if he is good java developer or not in just 5 questions.
But, you can judge a good Java developer with a simple test project
For example if the job posting is for java web developer: setup a web project with servlet container (Tomcat or Jetty) and jdbc driver and one entity with 20+ fields.
The task for the java programmer is to create a web form and save the data into the database without using getters/setters and other third party jars.
A good java developer will produce a simple and elegant solution.

1.3k views

Nathan Doromal, HFT dev to Google SRE SWE and back
Answered Apr 22, 2015 · Author has 1.8k answers and 2.5m answer views

5 questions is not that many questions so I would prefer open-ended questions that would generate a lot of information. I would also focus on questions that target the entire stack.

1. What's the most complex project you have ever worked?

2. Can you tell me what happens behind the scenes when I call "Foo obj =

new Foo()"?

3. If you had to implement generics in the Java compiler how would you do it?

4. Your executable running on a server is not responding and appears to be hung. How would you debug it?

5. If you could add one new piece of functionality to the Java language what would it be?

2.2k views · View 3 Upvoters

Peter Lawrey, 26 years working in IT.
Answered Apr 27, 2016 · Author has 707 answers and 857.8k answer views

I wouldn't ask them questions they can just look up online once they have the job.

I would start with

- What interests you?
- Tell me about a challenging project; what did you learn/what would you do differently.
- If you could change three things in Java or the JDK what would it be?
- When an application doesn't do what you expect, how to you approach this problem.
- When the business doesn't know what they need, how do you handle this?

2.6k views · View 9 Upvoters

A good Java interview entails

600+ Java & JEE Interview questions are answered with lots of diagrams.

1.3k views · View 1 Upvoter

Situational vs. Q&A are always more telling. A lot of people can memorize interview questions, while obvious to some, it can skew opinions. A few questions that are centered around strategic approach to code and implementation will show a candidates skill set more cleanly.

Some examples:

How do you test your code?

What is the difference between an integration and unit test?

Use object-oriented design to model a simple zoo consisting of a lion, giraffe, and elephant (believe it or not 90%+ failure rate; when they do it shows areas you can follow and discuss).

What happens when you mark an variable final?
(Answer: yes, it becomes immutable but ask why. If you are not familiar with the memory organization of the JVM; final variables are managed in PermGen vs. the heap. Another way to ask this is: why are method local variables or parameters accessed in anonymous inner classes required to be marked final? (note: Java 8 no longer has a PermGen; hello Metaspace! Where Has the Java PermGen Gone?)

How does a HashMap work? Is it O(1)? How are the keys calculated? What are some other collections you use NOT including List!

I hope this helps!
11.9k views · View 9 Upvoters

You don't.

There is only one kind of company that could *conceivably* judge any kind of programmer by only five questions.

A *bad* company. A company that is going to fail in the intermediate or long term.

Hiring carefully and correctly is literally the single most important thing that any company can do to assure its long-term success. Companies that hire using cute and superficial interview-screen shortcuts will not be in business very long.

The solution -- the ONLY solution I know of that actually *works* -- is to take the time and trouble to do multi-level phone screens and multiple in-person interviews, with people at many levels of the organization.

The majority of companies hire badly, using "clever" techniques like the one you describe here.

Don't be one of them.
4.5k views · View 10 Upvoters
I work as Java Dev for over 5 years and interview Java Dev as our company candidate, both junior level and/or intermediate. Some questions I asked:

1. Why you choose java to build a system, why you don’t choose other language (PHP, Ruby, or Scala) ?
2. Please explain to me how JVM work? how object stored in memory? how GC worked?
3. What is difference between Interface and Abstract Class? explain to me the case when you should use Interface and when you should Abstract Class?
4. Tell to me about java pattern you have been implement and/or learned and why you used that pattern?
5. What is difference between parsing by value and parsing by reference?

And outside the java language I also ask about what the best project you have done and/or involved? tell me why? by this question we’ll know whether he/she like a challenge or avoid a challenge

542 views · View 1 Upvoter
You don’t need 5 questions, you need 1 assignment.

If you want to test someone’s Java expertise:

1. Write a test that verifies volatile semantics in Java, in 30 min or less.

If you want to test someone’s programming expertise in general, thereby extending to Java:

1. Write a lock-free implementation of a thread-safe linked list, in 30 min or less.

Both tests verify hands-on ability, which is first and foremost, and then both test verify abstract understanding, which is next in priority.

7.1k views · View 4 Upvoters

Quintin Evans, Languages are tools...therefore I look to choose the best one. I favor none.
Answered Nov 6, 2015

I judge a developer on design experience, ability to learn, drive to grow, communication, problem solving ability, team work. Hence I'd ask these questions

1. What is the last technical book you've read and when? How did it impact your craft?
2. What is the biggest technical challenge you've faced in the past year and how did you solve it?
3. Which of your software projects are you most proud of? Why? What design tradeoffs did you make and why?
4. If you disagreed with someone on your team about a coding standard or a code review comment, what would you do to try and resolve it?
5. Could you please write FizzBuzz? :-p

(Like my answers? Check out my blog at http://thenerddojo.com. Follow me as I learn the ancient art of Software.)
4k views · View 5 Upvoters

1. Is Java pass by value or pass by reference
2. What is the role of equals() and hashCode() methods and how are they used
3. What is immutability and how can I create an Immutable class
4. What the different types of Exceptions in Java and what is the difference b/w them and their use
5. Will updates done to a String from a different thread reflect in my thread immediately or after some time or is there no guarantee

These are questions from my understanding to judge the basics of the developer !!!

Hope you like it !!!!!!!!

3.1k views · View 3 Upvoters

Good programmers can learn and think, that's he most powerful thing they (we) have. I don't know exactly what I would ask, but I would make up 5 questions to know:

1. how would you solve X problem? objective: can he think for himself?
2. You realize the servers are down suddenly and customers cannot connect. What would you do? Objective: can he work like in a team and ask for help?
3. We are starting a 5 yr project, what technology would you use? Objective: is he willing to learn something new if needed is he going to use Java for whatever he needs
4. You're an a**hole. Objective: that's not a question, but it would be interesting to see how he reacts.
5. I don't know. I ran out of questions in #4, but you get the idea. Don't ask things related to Java but to know how he is as a person. If he's a good/smart/focus person, then he's a good programmer.

2.2k views · View 1 Upvoter

Marcas Neal, 25+ years of learning software engineering the hardway -- rent-a-CTO

Answered Feb 28, 2016 · Author has 2.7k answers and 3.6m answer views

1. What are the pros and cons of inner classes?
2. What are the pros and cons of reflection?
3. What are good ways to avoid inner classes and reflection when they are not needed?
4. Why is Java a good/bad language for software engineering?
5. Could you write an OS in Java from a theoretical standpoint--why or why not?
6. Why is Java the most misunderstood/hated language in wide use? (*bonus question*)

1.2k views · View 6 Upvoters · Answer requested by Miguel Paraz

I generally just ask one question when judging programmers of any sort:

Show me some of your code, and tell me about it.

When I mention this to people, sometimes I get objections like "what about people whose code is all closed source or private?" I don't care. What would you do with an artist who applied for a job and said "I have a great portfolio, I just can't show it to you"?

I want to see an example of work product and I want you to be able to talk about it intelligently. If you don't have a portfolio, get busy. I don't care about your resume; I'm not hiring you to write resumes.
4k views · View 9 Upvoters

Will Berard, Forever a learner. Engineer, armchair scientist, unwitting philosopher.
Answered May 1, 2015 · Author has 385 answers and 671.1k answer views

Java programmers are a dime a dozen. If you want to judge a Software Engineer that is comfortable with Java, you should ask her:

- What other languages does she have any experience, no matter how small, of.
- Has she dealt with allocating and freeing memory manually in C/C++, and pointers (an pointers to pointers) and can she compare that to the way GC works in Java
- Has she got any functional programming experience at all (Lisp, Scala, Haskell), and how did it influence her understanding of recursion. Ask about lambda-calculus.
- Can she describe the kind of typing Java uses and how it differs from the other languages she knows, and why that matters
- Can she describe the OOP features that are unique to Java and how they differ from say, Python or C++ - again, ask for analysis, commentary, pros and cons.

If you want to hire code monkeys, Pick any 5 questions, but Code monkeys will have learnt by rote the answer to the typical 20 used and deemed "difficult".

If on the other hand, you want to hire engineers, then be wary of Java expert who know nothing beyond Java - they look like a great investment, hiring wise, but are much more likely to leave you with a poorly architectured

codebase, because their lack of frame of comparison, and in particular their lack of experience of anything that tries someones ability to *work at different level of abstraction.*

For more, see The Perils of JavaSchools from Joel Spolsky
4.2k views · View 4 Upvoters
For me

1) Why not C#? (Just to know if he/she knows the worth of Java)

2) The built-in memory to store integer, string, char etc ...

```
Integer a = (Integer)20;
```

```
Integer b = (Integer)20;
```

```
System.out.println(a == b); // true
```

```
Integer a1 = (Integer)200;
```

```
Integer b1 = (Integer)200;
```

```
System.out.println(a1 == b1); // false
```

3) extends vs implements in interface and classes .

4) hashmap vs hashtable vs hashset, when and why ?

5) Multi-threading + heap/stack questions.

7.7k views · View 16 Upvoters

I am a Java lead and frequently need to make hiring decisions. The only way is to put the candidate in front of an IDE (have both Intellij and Eclipse available) and have them work through a problem in front of you. That's one long question. As you watch, prepare to discuss the routes they choose and drill into those. For best results, repeat with another problem or two. It's a very successful technique and can take as much or as little time as you like, although I like to spend about 90 minutes or more.

4.6k views · View 4 Upvoters


1. Inheritance is a relation between Classes whereas Composition is a relation between Objects. Explain ?

2. Why is it recommended to code to an interface ?

3. What is String immutability. What is the use and benefit of keeping a string immutable ?

4. Don't you think MVC leads to extra classes and hence deterioration in performance ?

5. How would you go about finding the performance problems if you know that its DB Communication that's hurting the most ?
35 views · View 5 Upvoters
Well according to me there is no limit of difficulty of the question which you might ask and it depends on situation that upto what quality do you want to judge a person. But I would ask following questions related to the Core part of Java

- How does a Java code run on a machine ? What is JVM ?
- What is the meaning of keyword this, super in OOP ?
- What are abstract classes & Interfaces ?Why do we need them? Can we make their objects ? What is the difference between these two ?
- How do we implement threads in Java ? Explain the different methods.
- What do you know about Object Class ? Tell some of it's methods. What steps do we need to follow make our own equals() method ? (i.e. the equals & hashCode methods' contract)

787 views

Judging a Java programmer with only 5 questions is certainly a difficult task as Java has a wide scope as a programming language. Still, if I am supposed to do that, I will preferably choose the following 5 questions to judge a programmer’s clarity on Java concepts.

1. What will happen to the finally block if you put return statement or System.exit () on try or catch block?
2. If a method throws NullPointerException in the superclass, can we

override it with a method which throws RuntimeException?
3. How do you ensure that N threads access N resources without deadlock?
4. What is the difference between factory and abstract factory pattern?
5. What is the difference between creating String as new() and literal?

229 views

1. Something that requires HashMaps or other Hashes for optimal performance.
2. A Graph problem
3. A system/Architecture diagram
4. Something that requires careful avoidance of off-by-one errors.
5. Something Fun!

2.7k views · View 3 Upvoters

You can't judge a (Java or any other language) programmer with only 5 questions, but you can create a first impression about its seniority. Anyway it's better asking an interviewee some math and logic problems to get an idea on its approach on problem solving.

You don’t hire for skills, you hire for attitude. You can always teach skills.

> Great companies don’t hire skilled people and motivate them, they hire already motivated people and inspire them. People are either motivated or they are not. Unless you give motivated people something to believe in, something bigger than their job to work toward, they will motivate themselves to find a new job and you’ll be stuck with whoever’s left. (Start with WHY - Simon Sinek)

3.1k views · View 10 Upvoters

1. "Do you prefer Scala or Clojure?"
2. If he hasn't used either of the above, then: "Name a language (possibly that you'd prefer over Java."

If he can't imagine a language (it doesn't have to be Haskell; Python or Ruby would be fine) being better than Java, that's a no-hire.

Otherwise, use the remaining 3 questions to see how much he knows about the preferred language.

6k views · View 33 Upvoters
Originally Answered: How do you judge with only 5 questions a student who just finished a CS bachelor, has no experience and wants to work as a Java programmer?

With a BSCS and no experience, it's a waste of time to ask pointed Java (or any other language) questions. Expecting someone at this level to be able to write product-quality code is silly. That doesn't mean the candidate is a dummy, it means education is a foundation, and they'll learn from experience, just like every other competent professional programmer.

I'd be more inclined to ask them about their hobbies and what technologies they've heard about that they'd like to work with.

About the most pointed question I ask of programming candidates at this level is whether they learned any manner of machine code, and what they thought of it.
1.9k views

Mina Fro

12 views

Judging a programmer by asking questions doesn't prove that he has Java knowledge. Take a look at few of the projects that he has worked, those will tell lot of things like,

How much knowledge he has ?
How effective is his code ?
How much knowledge he has ?
Plus a lot more things.
230 views · View 1 Upvoter
I think the question should also include the experience range as well. That would give a better idea of the topics to be covered and one can judge the candidate better.

Rest, almost everything has been suggested.

515 views · View 1 Upvoter
Are you into containerization? How have your experiences working with other people's code effected how your structure your own code?

you may also refer-Interview Quetions of JAVA - SkyBird Technology

37 views

My answer is general, not only for java programmers.
I usually ask to tell me something about the last project he/she worked for.
Then I ask details about what he/she did in the last project.
Then I ask what was the hardest programming problem he/she found in the last project.
Then I ask what was the solution he/she found for that problem.

2.9k views · View 2 Upvoters
if you are interviewing other people and you need to post this question. you should ask other people to do the interview.

1 view

Java is vast. But here are some barebones ideas:
1. About Marker interfaces and their necessity
2. Inner classes, their need, examples
3. Given a sample program, predict the output and explain the prediction
4. Unit tests for a simple program idea

1.3k views