Contents

**No table of contents entries found.**

1. Why is Java so popular?
2. What is platform independence?
3. What is bytecode?
4. Compare JDK vs JVM vs JRE&nbsp;
5. What are the important differences between C++ and Java?
6. What is the role for a classloader in Java?
7. What are Wrapper classes?
8. Why do we need Wrapper classes in Java?
9. What are the different ways of creating Wrapper class instances?
10. What are differences in the two ways of creating Wrapper classes?
11. What is auto boxing?
12. What are the advantages of auto boxing?
13. What is casting?
14. What is implicit casting?
15. What is explicit casting?
16. Are all String’s immutable?
17. Where are String values stored in memory?
18. Why should you be careful about String concatenation(+) operator in loops?
19. How do you solve above problem?
20. What are differences between String and StringBuffer?
21. What are differences between StringBuilder and StringBuffer?
22. Can you give examples of different utility methods in String class?
23. What is a class?
24. What is an object?
25. What is state of an object?
26. What is behavior of an object?
27. What is the super class of every class in Java?
28. Explain about toString method ?
29. What is the use of equals method in Java?
30. What are the important things to consider when implementing equals method?
31. What is the Hashcode method used for in Java?
32. Explain inheritance with examples .&nbsp;
33. What is method overloading?
34. What is method overriding?
35. Can super class reference variable can hold an object of sub class?
36. Is multiple inheritance allowed in Java?
37. What is an interface?
38. How do you define an interface?
39. How do you implement an interface?
40. Can you explain a few tricky things about interfaces?
41. Can you extend an interface?
42. Can a class extend multiple interfaces?
43. What is an abstract class?
44. When do you use an abstract class?
45. How do you define an abstract method?
46. Compare abstract class vs interface?
47. What is a constructor?
48. What is a default constructor?
49. Will this code compile?
50. How do you call a super class constructor from a constructor?
51. Will this code compile?
52. What is the use of this()?
53. Can a constructor be called directly from a method?
54. Is a super class constructor called even when there is no explicit call from a sub class constructor?
55. What is polymorphism?
56. What is the use of instanceof operator in Java?
57. What is coupling?
58. What is cohesion?
59. What is encapsulation?
60. What is an inner class?
61. What is a static inner class?
62. Can you create an inner class inside a method?
63. What is an anonymous class?
64. What is default class modifier?
65. What is private access modifier?
66. What is default or package access modifier?
67. What is protected access modifier?
68. What is public access modifier?
69. What access types of variables can be accessed from a class in same package?
70. What access types of variables can be accessed from a class in different package?
71. What access types of variables can be accessed from a sub class in same package?
72. What access types of variables can be accessed from a sub class in different package?
73. What is the use of a final modifier on a class?
74. What is the use of a final modifier on a method?
75. What is a final variable?
76. What is a final argument?
77. What happens when a variable is marked as volatile?
78. What is a static variable?
79. Why should you always use blocks around if statement?
80. Guess the output
81. Guess the output
82. Guess the output of this switch block .&nbsp;
83. Guess the output of this switch block?
84. Should default be the last case in a switch statement?
85. Can a switch statement be used around a String
86. Guess the output of this for loop
87. What is an enhanced for loop?
88. What is the output of the for loop below?
89. What is the output of the program below?
90. What is the output of the program below?
91. Why is exception handling important?
92. What design pattern is used to implement exception handling features in most languages?
93. What is the need for finally block?
94. In what scenarios is code in finally not executed?
95. Will finally be executed in the program below?
96. Is try without a catch is allowed?
97. Is try without catch and finally allowed?
98. Can you explain the hierarchy of exception handling classes?
99. What is the difference between error and exception?
100. What is the difference between checked exceptions and unchecked exceptions?
101. How do you throw an exception from a method?
102. What happens when you throw a checked exception from a method?
103. What are the options you have to eliminate compilation errors when handling checked exceptions?
104. How do you create a custom exception?
105. How do you handle multiple exception types with same exception handling block?
106. Can you explain about try with resources?
107. How does try with resources work?
108. Can you explain a few exception handling best practices?
109. What are the default values in an array?
110. How do you loop around an array using enhanced for loop?
111. How do you print the content of an array?
112. How do you compare two arrays?
113. What is an enum?
114. Can you use a switch statement around an enum?
115. What are variable arguments or varargs?
116. What are asserts used for?
117. When should asserts be used?
118. What is garbage collection?
119. Can you explain garbage collection with an example?
120. When is garbage collection run?
121. What are best practices on garbage collection?
122. What are initialization blocks?
123. What is a static initializer?
124. What is an instance initializer block?
125. What is tokenizing?
126. Can you give an example of tokenizing?
127. What is serialization?
128. How do you serialize an object using serializable interface?
129. How do you de-serialize in Java?
130. What do you do if only parts of the object have to be serialized?
131. How do you serialize a hierarchy of objects?
132. Are the constructors in an object invoked when it is de-serialized?
133. Are the values of static variables stored when an object is serialized?
134. Why do we need collections in Java?
135. What are the important interfaces in the collection hierarchy?
136. What are the important methods that are declared in the collection interface?
137. Can you explain briefly about the List interface?
138. Explain about ArrayList with an example?
139. Can an ArrayList have duplicate elements?
140. How do you iterate around an ArrayList using iterator?
141. How do you sort an ArrayList?
142. How do you sort elements in an ArrayList using comparable interface?
143. How do you sort elements in an ArrayList using comparator interface?
144. What is vector class? How is it different from an ArrayList?
145. What is linkedList? What interfaces does it implement? How is it different from an ArrayList?
146. Can you briefly explain about the Set interface?
147. What are the important interfaces related to the Set interface?
148. What is the difference between Set and sortedSet interfaces?
149. Can you give examples of classes that implement the Set interface?
150. What is a HashSet?
151. What is a linkedHashSet? How is different from a HashSet?
152. What is a TreeSet? How is different from a HashSet?
153. Can you give examples of implementations of navigableSet?
154. Explain briefly about Queue interface?
155. What are the important interfaces related to the Queue interface?
156. Explain about the Deque interface?
157. Explain the BlockingQueue interface?
158. What is a priorityQueue?
159. Can you give example implementations of the BlockingQueue interface?
160. Can you briefly explain about the Map interface?
161. What is difference between Map and sortedMap?
162. What is a HashMap?
163. What are the different methods in a Hash Map?
164. What is a TreeMap? How is different from a HashMap?
165. Can you give an example of implementation of navigableMap interface?
166. What are the static methods present in the collections class?
167. What is the difference between synchronized and concurrent collections in Java?
168. Explain about the new concurrent collections in Java?
169. Explain about copyonwrite concurrent collections approach?
170. What is compareandswap approach?
171. What is a lock? How is it different from using synchronized approach?
172. What is initial capacity of a Java collection?
173. What is load factor?
174. When does a Java collection throw UnsupportedOperationException?
175. What is difference between fail-safe and fail-fast iterators?
176. What are atomic operations in Java?
177. What is BlockingQueue in Java?
178. What are Generics?
179. Why do we need Generics? Can you give an example of how Generics make a program more flexible?
180. How do you declare a generic class?
181. What are the restrictions in using generic type that is declared in a class declaration?
182. How can we restrict Generics to a subclass of particular class?
183. How can we restrict Generics to a super class of particular class?
184. Can you give an example of a generic method?
185. What is the need for threads in Java?
186. How do you create a thread?
187. How do you create a thread by extending thread class?
188. How do you create a thread by implementing runnable interface?
189. How do you run a thread in Java?
190. What are the different states of a thread?
191. What is priority of a thread? How do you change the priority of a thread?
192. What is executorservice?
193. Can you give an example for executorservice?
194. &nbsp;
195. How do you check whether an executionservice task executed successfully?
196. What is callable? How do you execute a callable from executionservice?
197. What is synchronization of threads?
198. Can you give an example of a synchronized block?
199. Can a static method be synchronized?
200. What is the use of join method in threads?
201. Describe a few other important methods in threads?
202. What is a deadlock?
203. What are the important methods in Java for inter-thread communication?
204. What is the use of wait method?
205. What is the use of notify method?
206. What is the use of notifyall method?
207. Can you write a synchronized program with wait and notify methods?
208. What is functional programming?
209. Can you give an example of functional programming?
210. What is a stream?
211. Explain about streams with an example?
212. what are intermediate operations in streams?
213. What are terminal operations in streams?
214. What are method references?
215. What are lambda expressions?
216. Can you give an example of lambda expression?
217. Can you explain the relationship between lambda expression and functional interfaces?
218. What is a predicate?
219. What is the functional interface – function?
220. What is a consumer?
221. Can you give examples of functional interfaces with multiple arguments?
222. What are Static imports
223. org/overriding-in-java/)
224. Rules of overloading? ()
225. what is dilect ?
226. Difference between result set and rowset?
227. Will connection established even after resultset ? (Yes throughout result set connection will be established and it is non sealizable)
228. Can we write test case for private methods? (Yes, Using reflections we can write)
229. Generics in List collection?
230. Enumerations vs Iterations ?What are Collection related features in Java 8?
231. What is Java Collections Framework? List out some benefits of Collections framework?
232. What is the benefit of Generics in Collections Framework?
233. What are the basic interfaces of Java Collections Framework?
234. Why Collection doesn’t extend Cloneable and Serializable interfaces?
235. Why Map interface doesn’t extend Collection interface?
236. What is an Iterator?
237. What is the difference between Enumeration and Iterator interface?
238. Why there is not a method like Iterator.add() to add elements to the collection?
239. Why Iterator don’t have a method to get next element directly without moving the cursor?
240. What is different between Iterator and ListIterator?
241. What are different ways to iterate over a list?
242. What do you understand by iterator fail-fast property?
243. What is difference between fail-fast and fail-safe?
244. How to avoid ConcurrentModificationException while iterating a collection?
245. Why there are no concrete implementations of Iterator interface?
246. What is UnsupportedOperationException?
247. How HashMap works in Java?
248. What is the importance of hashCode() and equals() methods?
249. Can we use any class as Map key?
250. What are different Collection views provided by Map interface?
251. What is difference between HashMap and Hashtable?
252. How to decide between HashMap and TreeMap?
253. What are similarities and difference between ArrayList and Vector?
254. What is difference between Array and ArrayList? When will you use Array over ArrayList?
255. What is difference between ArrayList and LinkedList?
256. Which collection classes provide random access of its elements?
257. What is EnumSet?
258. Which collection classes are thread-safe?
259. What are concurrent Collection Classes?
260. What is BlockingQueue?
261. What is Queue and Stack, list their differences?
262. What is Collections Class?
263. What is Comparable and Comparator interface?
264. What is difference between Comparable and Comparator interface?
265. How can we sort a list of Objects?
266. While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?
267. How can we create a synchronized collection from given collection?
268. What are common algorithms implemented in Collections Framework?
269. What is Big-O notation? Give some examples?
270. What are best practices related to Java Collections Framework?
271. What is Java Priority Queue?
272. Why can’t we write code as List<Number> numbers = new ArrayList<Integer>();?
273. Why can’t we create generic array? or write code as List<Integer>[] array = new ArrayList<Integer>[10];
274. ConcurrentHashMap vs Hashtable vs Synchronized Map
275. What is the output of below program?
     1. package com.journaldev.strings;
     2. public class StringTest {
        1. public static void main(String[] args) {
           1. String s1 = new String("pankaj");
           2. String s2 = new String("PANKAJ");
           3. System.out.println(s1 = s2);
        2. }
     3. }
     4. It’s a simple yet tricky program, it will print “PANKAJ” because we are assigning s2 String to s1. Don’t get confused with == comparison operator.
     5. What is the output of below program?
     6. package com.journaldev.strings;
     7. public class Test {
        1. public void foo(String s) {
        2. System.out.println("String");
        3. }
        4. public void foo(StringBuffer sb){
        5. System.out.println("StringBuffer");
        6. }
        7. public static void main(String[] args) {
           1. new Test().foo(null);
        8. }
     8. }
     9. The above program will not compile with error as “The method foo(String) is ambiguous for the type Test”. For complete clarification read Understanding the method X is ambiguous for the type Y error.
276. What will be the output of below program?
     1. String s1 = "abc";
     2. String s2 = new String("abc");
     3. s2.intern();
     4. System.out.println(s1 ==s2);
     5. It’s a tricky question and output will be false. We know that intern() method will return the String object reference from the string pool, but since we didn’t assigned it back to s2, there is no change in s2 and hence both s1 and s2 are having different reference. If we change the code in line 3 to s2 = s2.intern(); then output will be true.
277. What is use of @qualifier can we use qualifier without @autowired
278. How json serlization and deserlization works in java (@JsonComponent , @Jsonsearlization ,@jsondeserlization)
279. Write a program to remove duplicates in LinkedList
280. Changes done in Hashmap in Java 1.8
281. {1,2,3,4,5} if key=2 we need to transfer first two elements to last {3,4,5,1,2}
282. Usage of Http2 VS Http1 (In http1 we use TCP connection we migrated to TLP connection which will improve latency)
283. Difference between post and PUT with idopnent example <https://restfulapi.net/rest-put-vs-post/>

POST is NOT idempotent.

GET, PUT, DELETE, HEAD, OPTIONS and TRACE are idempotent.

1. Usage of LinkedList and ArrayList
2. Http Response code

100 Continue Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request.

101 Switching Protocols The server switches protocol.

200 OK The request is OK.

201 Created The request is complete, and a new resource is created .

202 Accepted The request is accepted for processing, but the processing is not complete.

203 Non-authoritative Information The information in the entity header is from a local or third-party copy, not from the original server.

204 No Content A status code and a header are given in the response, but there is no entity-body in the reply.

205 Reset Content The browser should clear the form used for this transaction for additional input.

206 Partial Content The server is returning partial data of the size requested. Used in response to a request specifying a Range header. The server must specify the range included in the response with the Content-Range header.

300 Multiple Choices A link list. The user can select a link and go to that location. Maximum five addresses .

301 Moved Permanently The requested page has moved to a new url .

302 Found The requested page has moved temporarily to a new url .

303 See Other The requested page can be found under a different url .

304 Not Modified This is the response code to an If-Modified-Since or If-None-Match header, where the URL has not been modified since the specified date.

305 Use Proxy The requested URL must be accessed through the proxy mentioned in the Location header.

306 Unused This code was used in a previous version. It is no longer used, but the code is reserved.

307 Temporary Redirect The requested page has moved temporarily to a new url.

400 Bad Request The server did not understand the request.

401 Unauthorized The requested page needs a username and a password.

402 Payment Required You cannot use this code yet.

403 Forbidden Access is forbidden to the requested page.

404 Not Found The server can not find the requested page.

405 Method Not Allowed The method specified in the request is not allowed.

406 Not Acceptable The server can only generate a response that is not accepted by the client.

407 Proxy Authentication Required You must authenticate with a proxy server before this request can be served.

408 Request Timeout The request took longer than the server was prepared to wait.

409 Conflict The request could not be completed because of a conflict.

410 Gone The requested page is no longer available .

411 Length Required The "Content-Length" is not defined. The server will not accept the request without it .

412 Precondition Failed The pre condition given in the request evaluated to false by the server.

413 Request Entity Too Large The server will not accept the request, because the request entity is too large.

414 Request-url Too Long The server will not accept the request, because the url is too long. Occurs when you convert a "post" request to a "get" request with a long query information .

415 Unsupported Media Type The server will not accept the request, because the mediatype is not supported .

416 Requested Range Not Satisfiable The requested byte range is not available and is out of bounds.

417 Expectation Failed The expectation given in an Expect request-header field could not be met by this server.

500 Internal Server Error The request was not completed. The server met an unexpected condition.

501 Not Implemented The request was not completed. The server did not support the functionality required.

502 Bad Gateway The request was not completed. The server received an invalid response from the upstream server.

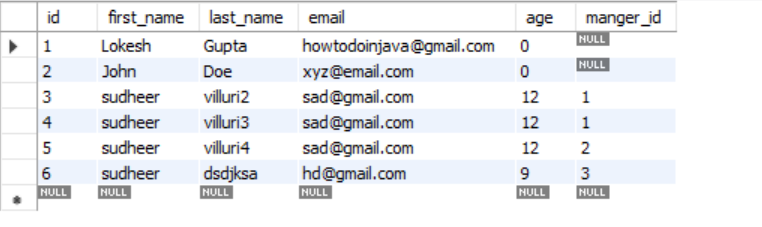
503 Service Unavailable The request was not completed. The server is temporarily overloading or down.

504 Gateway Timeout The gateway has timed out.

505 HTTP Version Not Supported The server does not support the "http protocol" version.

1. Write a program from Queue using two stacks
2. AAABBCCDDAA -> A3B2C2D2A2
3. [4,5,67,12,32] -> k=2 -> rotate elements [67l]
4. N+1 Problem in ORM and Rest how you will overcome that

Avoid using One to many relationship by created a query for selecting required element rather than fetching all dependents which will improve the performance of 15 min to 1.5 sec

1. Types of Overloading (method overloading and constructor overloading )
2. Internal implementation of Collections.sort() will it use comporator or comparable
3. What is Diamond interface and how it handled in java8
4. 

Print the managers name who have more than one reporties

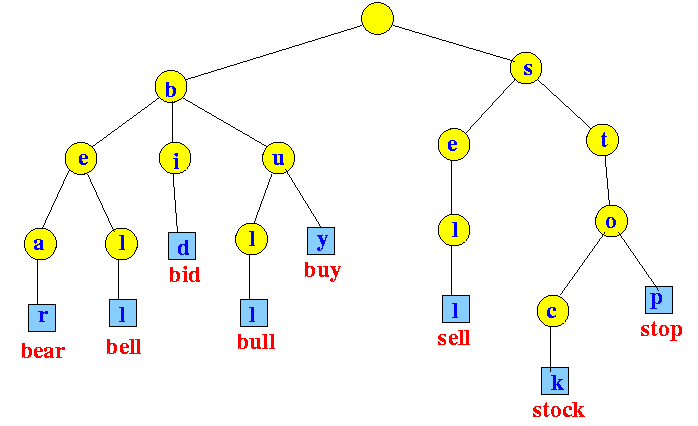
select Emp.\* from test.tbl\_employees Emp join (SELECT id,count(manger\_id) as count2,manger\_id from test.tbl\_employees group by manger\_id) mang

ON Emp.id = mang.manger\_id AND mang.count2 >1

1. Find 3rd Highest salary

select ename,sal from test.emp em1 where 1 = (select count(DISTINCT sal) from test.emp em2 WHERE em2.sal > em1.sal)

1. What are design pattern you used with examples
2. IOC
3. PROXY Design pattern (EXP AOP)
4. Singleton Design pattern (IOC Container)
5. MVC Desing patteren
6. Templete method (Spring framework provides a number of templates to kick start work and complete that piece of work as the best programming practice such as opening and closing connection for JDBC or JMS, etc. E.g., JdbcTemplate, JmsTemplate, and JpaTemplate.)
7. Why Hibernate session is thread safe
8. When to use ConcurrentHashmap vs Synchronius Hashmap
9. Usage of Hascode in Object class and Hashcode v/s Hashvalue
10. How Producer consumer concept works
11. Internal implementation of Hashmap vs TreeMap
12. trie data structure (A **Trie** is a special **data structure** used to store strings that can be visualized like a graph. It consists of nodes and edges. Each node consists of at max 26 children and edges connect each parent node to its children.)



1. types of classloaders ?
2. Ouput of Sysout(“Heloo” + A == B) false
3. Output of Sysout(“Hello” + (A==B)) true
4. Output of Sysout(A = B) updated A value
5. Internal implementation of manytomany mapping in hibernate