**SCJP Mock exam for Collection**

Step-by-step questions for Collections, in order to help you prepare for the Sun Certified Java Programmer for Java SE 6.

Questions 1-40 are about [sets](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-set), 41-85 about [maps](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-map), 86-110 about [queues](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-queue), 111-130 about [lists](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-list).

|  |  |
| --- | --- |
| **questions** |  |
| [1-16](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-set) | [SortedSet](http://java.sun.com/javase/6/docs/api/java/util/SortedSet.html) |
| [17-30](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-navigableset) | [NavigableSet](http://java.sun.com/javase/6/docs/api/java/util/NavigableSet.html) |
| [31-40](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-hashset) | [LinkedHashSet](http://java.sun.com/javase/6/docs/api/java/util/LinkedHashSet.html), [HashSet](http://java.sun.com/javase/6/docs/api/java/util/HashSet.html), [TreeSet](http://java.sun.com/javase/6/docs/api/java/util/TreeSet.html) |
| [41-56](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-map) | [SortedMap](http://java.sun.com/javase/6/docs/api/java/util/SortedMap.html) |
| [57-67](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-hashmap) | [LinkedHashMap](http://java.sun.com/javase/6/docs/api/java/util/LinkedHashMap.html), [HashMap](http://java.sun.com/javase/6/docs/api/java/util/HashMap.html), [TreeMap](http://java.sun.com/javase/6/docs/api/java/util/TreeMap.html) |
| [68-85](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-navigablemap) | [NavigableMap](http://java.sun.com/javase/6/docs/api/java/util/NavigableMap.html) |
| [86-110](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-queue) | [PriorityQueue](http://java.sun.com/javase/6/docs/api/java/util/PriorityQueue.html), [Queue](http://java.sun.com/javase/6/docs/api/java/util/Queue.html) |
| [111-130](http://nikojava.wordpress.com/2008/10/06/scjp-mock-exam-for-collections/#scjp-collections-list) | [List](http://java.sun.com/javase/6/docs/api/java/util/List.html), [LinkedList](http://java.sun.com/javase/6/docs/api/java/util/LinkedList.html), [ArrayList](http://java.sun.com/javase/6/docs/api/java/util/ArrayList.html) |
|  |  |

1. SortedSet IS-A Set. (1 correct answer)
   1. true
   2. false
2. SortedSet belongs to the java.util package. (1 correct answer)
   1. true
   2. false
3. How many of the following methods of SortedSet may throw an exception at runtime? (1 correct answer)  
       ⇒ first()  
       ⇒ last()  
       ⇒ headSet()  
       ⇒ tailSet()  
       ⇒ subSet()  
       ⇒ comparator()
   1. None.
   2. Two.
   3. Three.
   4. Five.
4. What’s the signature of the method headSet()? (1 correct answer)
   1. SortedSet<E> headSet(E)
   2. SortedSet<E> headSet(E, E)
   3. SortedSet<E> headSet(E, boolean)
5. What happens when this code gets executed? (1 correct answer)
6. public static void main(String[] args) {
7. SortedSet<Integer> set = new TreeSet<Integer>();
8. set.last();
9. }
   1. A NoSuchElementException is thrown.
   2. It compiles and runs fine.
   3. Compilation fails.
10. What happens when this code gets executed? (1 correct answer)
11. public static void main(String[] args) {
12. SortedSet<Integer> set = new TreeSet<Integer>();
13. set.add(7);
14. set.add(-12);
15. SortedSet<Integer> sub = set.subSet(-100, 100);
16. System.out.format("%d set.size(), sub.size());
17. }
    1. It prints “2 2″.
    2. It prints “2 0″.
18. What happens when this code gets executed? (1 correct answer)
19. public static void main(String[] args) {
20. SortedSet<Integer> set = new TreeSet<Integer>();
21. set.add(7);
22. set.add(-12);
23. SortedSet<Integer> sub = set.subSet(-100, 100);
24. **sub.add(9)**;
25. System.out.format("%d set.size(), sub.size());
26. }
    1. An IllegalArgumentException is thrown.
    2. It prints “2 3″.
    3. It prints “2 1″.
    4. It prints “3 3″.
27. What happens when this code gets executed? (1 correct answer)
28. public static void main(String[] args) {
29. SortedSet<Integer> set = new TreeSet<Integer>();
30. set.add(7);
31. set.add(-12);
32. SortedSet<Integer> sub = set.subSet(-100, 100);
33. **sub.add(-100)**;
34. System.out.format("%d set.size(), sub.size());
35. }
    1. An IllegalArgumentException is thrown.
    2. It prints “2 3″.
    3. It prints “2 1″.
    4. It prints “3 3″.
36. What happens when this code gets executed? (1 correct answer)
37. public static void main(String[] args) {
38. SortedSet<Integer> set = new TreeSet<Integer>();
39. set.add(7);
40. set.add(-12);
41. SortedSet<Integer> sub = set.subSet(-100, 100);
42. **sub.add(100)**;
43. System.out.format("%d set.size(), sub.size());
44. }
    1. An IllegalArgumentException is thrown.
    2. It prints “2 3″.
    3. It prints “2 1″.
    4. It prints “3 3″.
45. What happens when this code gets executed? (1 correct answer)
46. public static void main(String[] args) {
47. SortedSet<Integer> set = new TreeSet<Integer>();
48. set.add(7);
49. set.add(-12);
50. SortedSet<Integer> sub = set.subSet(-100, 100);
51. **sub.add(99)**;
52. System.out.format("%d set.size(), sub.size());
53. }
    1. An IllegalArgumentException is thrown.
    2. It prints “2 3″.
    3. It prints “2 1″.
    4. It prints “3 3″.
54. What happens when this code gets executed? (1 correct answer)
55. public static void main(String[] args) {
56. SortedSet<Integer> set = new TreeSet<Integer>();
57. set.add(7);
58. set.add(-12);
59. SortedSet<Integer> sub = set.subSet(-100, 100);
60. **sub.add(7)**;
61. System.out.format("%d set.size(), sub.size());
62. }
    1. An IllegalArgumentException is thrown.
    2. It prints “2 1″.
    3. It prints “2 2″.
63. What happens when this code gets compiled and executed? (1 correct answer)
64. public static void main(String[] args) {
65. SortedSet set = new TreeSet();
66. set.add(7);
67. set.add(-12);
68. SortedSet sub = set.subSet("Hello!", 100);
69. System.out.format("%d set.size(), sub.size());
70. }
    1. An IllegalArgumentException is thrown at runtime.
    2. A ClassCastException is thrown at runtime.
    3. Compilation fails.
    4. It prints “2 0″.
    5. It prints “2 2″.
71. What happens when this code gets compiled and executed? (1 correct answer)
72. public static void main(String[] args) {
73. SortedSet<Integer> set = new TreeSet<Integer>();
74. set.add("Hello}
    1. A ClassCastException is thrown at runtime.
    2. Compilation fails.
    3. None of the above.
75. What happens when this code gets compiled and executed? (1 correct answer)
76. public static void main(String[] args) {
77. SortedSet set = new TreeSet();
78. set.add("Hello}
    1. A ClassCastException is thrown at runtime.
    2. Compilation fails.
    3. None of the above.
79. What happens when this code gets compiled and executed? (1 correct answer)
80. public static void main(String[] args) {
81. SortedSet set = new TreeSet();
82. set.add("Hello    **set.add(1);**
83. }
    1. A ClassCastException is thrown at runtime.
    2. Compilation fails.
    3. None of the above.
84. What happens when this code gets compiled and executed? (1 correct answer)
85. public static void main(String[] args) {
86. SortedSet set = new TreeSet();
87. set.add(**null**);
88. set.add(1);
89. }
    1. An NullPointerException is thrown at runtime.
    2. A ClassCastException is thrown at runtime.
    3. Compilation fails.
    4. None of the above.
90. NavigableSet extends SortedSet. (1 correct answer)
    1. true
    2. false
91. How many of the following methods of NavigableSet may throw an exception at runtime? (1 correct answer)  
        ⇒ lower()  
        ⇒ higher()  
        ⇒ floor()  
        ⇒ ceiling()  
        ⇒ pollFirst()  
        ⇒ pollLast()
    1. None.
    2. Two.
    3. Four.
    4. All.
92. What’s the signature of the method lower()? (1 correct answer)
    1. E lower(E)
    2. boolean lower(E)
    3. E lower(NavigableSet<E>)
93. In the NavigableSet interface there are 2 overloaded methods with the name **subSet**. (1 correct answer)
    1. true
    2. false
94. What is the output of this code? (1 correct answer)
95. public static void main(String[] args) {
96. NavigableSet<Integer> set = new TreeSet<Integer>();
97. set.add(-12);
98. set.add(24);
99. System.out.format("%dd %d",
100. set.**lower**(-12),
101. set.lower(0),
102. set.lower(24),
103. set.lower(100)
104. );
105. }
     1. An exception is thrown at runtime.
     2. It prints “null -12 -12 24″.
     3. It prints “-12 -12 24 24″.
106. What is the output of this code? (1 correct answer)
107. public static void main(String[] args) {
108. NavigableSet<Integer> set = new TreeSet<Integer>();
109. set.add(-12);
110. set.add(24);
111. System.out.format("%dd %d",
112. set.**floor**(-12),
113. set.floor(0),
114. set.floor(24),
115. set.floor(100)
116. );
117. }
     1. An exception is thrown at runtime.
     2. It prints “null -12 -12 24″.
     3. It prints “-12 -12 24 24″.
118. What is the output of this code? (1 correct answer)
119. public static void main(String[] args) {
120. NavigableSet<Integer> set = new TreeSet<Integer>();
121. set.add(-12);
122. set.add(24);
123. System.out.format("%dd %d",
124. set.**higher**(-12),
125. set.higher(0),
126. set.higher(24),
127. set.higher(100)
128. );
129. }
     1. An exception is thrown at runtime.
     2. It prints “24 24 null null”.
     3. It prints “-12 24 24 null”.
130. What is the output of this code? (1 correct answer)
131. public static void main(String[] args) {
132. NavigableSet<Integer> set = new TreeSet<Integer>();
133. set.add(-12);
134. set.add(24);
135. System.out.format("%dd %d",
136. set.**ceiling**(-12),
137. set.ceiling(0),
138. set.ceiling(24),
139. set.ceiling(100)
140. );
141. }
     1. An exception is thrown at runtime.
     2. It prints “24 24 null null”.
     3. It prints “-12 24 24 null”.
142. What is the output of this code? (1 correct answer)
143. public static void main(String[] args) {
144. NavigableSet<Integer> set = new TreeSet<Integer>();
145. set.add(-12);
146. set.add(24);
147. set.add(-28);
148. set.add(-0);
149. set.add(0);
150. set.add(+0);
151. set.add(11);
152. set.add(145);
153. System.out.format("%dd %d",
154. set.higher(-28),
155. set.lower(24),
156. set.floor(-0),
157. set.ceiling(100)
158. );
159. }
     1. An exception is thrown at runtime.
     2. It prints “-12 11 0 100″.
     3. It prints “-12 11 0 145″.
     4. It prints “-28 24 0 100″.
     5. It prints “-28 24 0 145″.
160. What is the output of this code? (1 correct answer)
161. public static void main(String[] args) {
162. NavigableSet<Integer> set = new TreeSet<Integer>();
163. set.pollFirst();
164. System.out.println(set.size());
165. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “0”.
166. What is the output of this code? (1 correct answer)
167. public static void main(String[] args) {
168. NavigableSet<Integer> set = new TreeSet<Integer>();
169. set.first();
170. System.out.println(set.size());
171. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “0”.
172. What is the output of this code? (1 correct answer)
173. public static void main(String[] args) {
174. NavigableSet<Integer> set = new TreeSet<Integer>();
175. set.add(1);
176. set.add(2);
177. set.add(4);
178. NavigableSet<Integer> sub = set.headSet(4);
179. System.out.println(sub.last());
180. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “4”.
     4. It prints “2”.
181. What is the output of this code? (1 correct answer)
182. public static void main(String[] args) {
183. NavigableSet<Integer> set = new TreeSet<Integer>();
184. set.add(1);
185. set.add(2);
186. set.add(4);
187. NavigableSet<Integer> sub = set.headSet(4, **true**);
188. System.out.println(sub.last());
189. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “4”.
     4. It prints “2”.
190. What is the output of this code? (1 correct answer)
191. public static void main(String[] args) {
192. NavigableSet<Integer> set = new TreeSet<Integer>();
193. set.add(1);
194. set.add(2);
195. set.add(4);
196. for (Iterator iterator = set.descendingSet().iterator();
197. iterator.hasNext();) {
198. System.out.format("%diiterator.next());
199. }
200. }
     1. It prints “1 2 4 “.
     2. It prints “4 2 1 “.
     3. Compilation fails.
201. What is the output of this code? (1 correct answer)
202. import java.util.\*;
203. public class TestSet {
204. static void add(Set<? super String> set) {
205. set.add(null);
206. System.out.println(set.size());
207. }
208. public static void main(String[] args) {
209. Set<String> set = new **HashSet**<String>();
210. add(set);
211. }
212. }
     1. It prints “0”.
     2. It prints “1”.
     3. An exception is thrown at runtime.
213. What is the output of this code? (1 correct answer)
214. import java.util.\*;
215. public class TestSet {
216. static void add(Set<? super String> set) {
217. set.add(null);
218. **set.add(null);**
219. System.out.println(set.size());
220. }
221. public static void main(String[] args) {
222. Set<String> set = new HashSet<String>();
223. add(set);
224. }
225. }
     1. It prints “0”.
     2. It prints “1”.
     3. It prints “2”.
     4. An exception is thrown at runtime.
226. What is the output of this code? (1 correct answer)
227. import java.util.\*;
228. public class TestSet {
229. static void add(Set<? super String> set) {
230. set.add("Hi       set.add("Hi       System.out.println(set.size());
231. }
232. public static void main(String[] args) {
233. Set<String> set = new HashSet<String>();
234. add(set);
235. }
236. }
     1. It prints “0”.
     2. It prints “1”.
     3. It prints “2”.
     4. An exception is thrown at runtime.
237. What is the output of this code? (1 correct answer)
238. import java.util.\*;
239. public class TestSet {
240. static void add(Set set) {
241. set.add("Hi       set.add(1);
242. System.out.println(set.size());
243. }
244. public static void main(String[] args) {
245. Set<String> set = new HashSet<String>();
246. add(set);
247. }
248. }
     1. It prints “0”.
     2. It prints “1”.
     3. It prints “2”.
     4. An exception is thrown at runtime.
249. What is the output of this code? (1 correct answer)
250. import java.util.\*;
251. public class TestSet {
252. static void add(Set<? super String> set) {
253. set.add(null);
254. System.out.println(set.size());
255. }
256. public static void main(String[] args) {
257. Set<String> set = new **TreeSet**<String>();
258. add(set);
259. }
260. }
     1. It prints “0”.
     2. It prints “1”.
     3. An exception is thrown at runtime.
261. What is the output of this code? (1 correct answer)
262. import java.util.\*;
263. public class TestSet {
264. static void add(Set<? super String> set) {
265. set.add(null);
266. **set.add(null);**
267. System.out.println(set.size());
268. }
269. public static void main(String[] args) {
270. Set<String> set = new TreeSet<String>();
271. add(set);
272. }
273. }
     1. It prints “0”.
     2. It prints “1”.
     3. It prints “2”.
     4. An exception is thrown at runtime.
274. What is the output of this code? (1 correct answer)
275. import java.util.\*;
276. public class TestSet {
277. static void add(Set<? super String> set) {
278. set.add("Hi       set.add("Hi       System.out.println(set.size());
279. }
280. public static void main(String[] args) {
281. Set<String> set = new TreeSet<String>();
282. add(set);
283. }
284. }
     1. It prints “0”.
     2. It prints “1”.
     3. It prints “2”.
     4. An exception is thrown at runtime.
285. What is the output of this code? (1 correct answer)
286. import java.util.\*;
287. public class TestSet {
288. static void add(Set set) {
289. set.add("Hi       set.add(1);
290. System.out.println(set.size());
291. }
292. public static void main(String[] args) {
293. Set<String> set = new TreeSet<String>();
294. add(set);
295. }
296. }
     1. It prints “0”.
     2. It prints “1”.
     3. It prints “2”.
     4. An exception is thrown at runtime.
297. What is the output of this code? (1 correct answer)
298. import java.util.\*;
299. public class TestSet {
300. static void print(Set<? extends String> set) {
301. for (String element : set) {
302. System.out.format("%slement);
303. }
304. }
305. public static void main(String[] args) {
306. Set<String> set = new LinkedHashSet<String>();
307. set.add("A       set.add("J       set.add("A       set.add("X       print(set);
308. }
309. }
     1. It prints “A J A X “.
     2. It prints “A J X “.
     3. The output cannot be determined.
310. What is the output of this code? (1 correct answer)
311. import java.util.\*;
312. public class TestSet {
313. static void print(Set<? extends String> set) {
314. for (String element : set) {
315. System.out.format("%slement);
316. }
317. }
318. public static void main(String[] args) {
319. Set<String> set = new **HashSet**<String>();
320. set.add("A");
321. set.add("J");
322. set.add("A");
323. set.add("X");
324. print(set);
325. }
326. }
     1. It prints “A J A X “.
     2. It prints “A J X “.
     3. The output cannot be determined.
327. SortedMap IS-A Collection. (1 correct answer)
     1. true
     2. false
328. SortedMap IS-A Map. (1 correct answer)
     1. true
     2. false
329. All the following methods are defined in SortedMap. (1 correct answer)  
         ⇒ firstKey()  
         ⇒ lastKey()  
         ⇒ headMap()  
         ⇒ tailMap()  
         ⇒ subMap()  
         ⇒ comparator()
     1. true
     2. false
330. How many of the following methods of SortedMap may throw an exception at runtime? (1 correct answer)  
         ⇒ firstKey()  
         ⇒ lastKey()  
         ⇒ headMap()  
         ⇒ tailMap()  
         ⇒ subMap()  
         ⇒ comparator()
     1. None.
     2. Two.
     3. Three.
     4. Five.
331. Consider SortedMap<K,V>. What’s the signature of the method firstKey()? (1 correct answer)
     1. K firstKey()
     2. V firstKey()
     3. K firstKey(K)
     4. V firstKey(K)
332. Consider SortedMap<K,V>. What’s the signature of the method tailMap()? (1 correct answer)
     1. SortedMap<K,V> tailMap(K)
     2. SortedMap<K,V> tailMap(K, V)
     3. SortedMap<K,V> tailMap(K, boolean)
333. What happens when this code gets executed? (1 correct answer)
334. public static void main(String[] args) {
335. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
336. map.last();
337. }
     1. A NoSuchElementException is thrown.
     2. It compiles and runs fine.
     3. Compilation fails.
338. What happens when this code gets executed? (1 correct answer)
339. public static void main(String[] args) {
340. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
341. map.**lastKey**();
342. }
     1. A NoSuchElementException is thrown.
     2. It compiles and runs fine.
     3. Compilation fails.
343. What happens when this code gets executed? (1 correct answer)
344. public static void main(String[] args) {
345. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
346. map.lastEntry();
347. }
     1. A NoSuchElementException is thrown.
     2. It compiles and runs fine.
     3. Compilation fails.
348. What happens when this code gets executed? (1 correct answer)
349. public static void main(String[] args) {
350. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
351. map.put("B;
352. System.out.println(map.put("B);
353. }
     1. It prints “B”.
     2. It prints “null”.
     3. It prints “1”.
     4. It prints “2”.
354. What happens when this code gets executed? (1 correct answer)
355. public static void main(String[] args) {
356. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
357. map.put("B;
358. map.put("B;
359. map.put("a;
360. System.out.format("%s
361. map.lastKey(),
362. map.size());
363. }
     1. It prints “a 2″.
     2. It prints “a 3″.
     3. It prints “B 2″.
     4. It prints “B 3″.
364. What happens when this code gets executed? (1 correct answer)
365. public static void main(String[] args) {
366. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
367. map.put("K;
368. map.put("B;
369. map.put("F;

    System.out.println(map.tailMap("C").size

* 1. It prints “0”.
  2. It prints “1”.
  3. It prints “2”.
  4. It prints “3”.

1. What happens when this code gets executed? (1 correct answer)
2. public static void main(String[] args) {
3. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
4. map.put("K;
5. map.put("B;
6. map.put("F;

    System.out.println(map.tailMap("C").put("D

* 1. An IllegalArgumentException is thrown.
  2. It prints “null”.
  3. It prints “D”.
  4. It prints “F”.

1. What happens when this code gets executed? (1 correct answer)
2. public static void main(String[] args) {
3. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
4. map.put("K;
5. map.put("B;
6. map.put("F;
7. System.out.println(map.tailMap("**c**").size());
8. }
   1. An IllegalArgumentException is thrown.
   2. It prints “0”.
   3. It prints “1”.
   4. It prints “2”.
   5. It prints “3”.
9. What happens when this code gets executed? (1 correct answer)
10. public static void main(String[] args) {
11. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
12. map.put("K;
13. map.put("B;
14. map.put("F;
15. System.out.println(map.tailMap("**c**").put("**c**", 1));
16. }
    1. An IllegalArgumentException is thrown.
    2. It prints “null”.
    3. It prints “c”.
    4. It prints “F”.
17. What happens when this code gets executed? (1 correct answer)
18. public static void main(String[] args) {
19. SortedMap<String, Integer> map = new TreeMap<String, Integer>();
20. map.put("K;
21. map.put("B;
22. map.put("F;

    System.out.println(map.tailMap("c").put("A

* 1. An IllegalArgumentException is thrown.
  2. It prints “null”.
  3. It prints “c”.
  4. It prints “F”.

1. What happens when this code gets executed? (1 correct answer)
2. public static void main(String[] args) {
3. Map map = new TreeMap();
4. map.put("K;
5. map.put("L");
6. }
   1. An exception is thrown at runtime.
   2. Compilation fails.
   3. None of the above.
7. What happens when this code gets executed? (1 correct answer)
8. public static void main(String[] args) {
9. Map map = new TreeMap();
10. map.put("K;
11. map.put(**2**, "L");
12. }
    1. An exception is thrown at runtime.
    2. Compilation fails.
    3. None of the above.
13. What happens when this code gets executed? (1 correct answer)
14. public static void main(String[] args) {
15. Map map = new **LinkedHashMap**();
16. map.put("K;
17. map.put(2, "L");
18. }
    1. An exception is thrown at runtime.
    2. Compilation fails.
    3. None of the above.
19. What happens when this code gets executed? (1 correct answer)
20. public static void main(String[] args) {
21. Map map = new LinkedHashMap();
22. map.put("K;
23. map.put(2, "L");
24. for (Object element : map.keySet()) {
25. System.out.print(element);
26. }
27. }
    1. An exception is thrown at runtime.
    2. Compilation fails.
    3. None of the above.
28. What happens when this code gets executed? (1 correct answer)
29. public static void main(String[] args) {
30. new LinkedHashMap<String, Integer>().put(null, null);
31. new HashMap<Object, Number>().put(null, null);
32. new TreeMap<String, Queue<String>>().put(null, null);
33. }
    1. An exception is thrown at runtime.
    2. It compiles and runs fine.
    3. Compilation fails.
34. What happens when this code gets executed? (1 correct answer)
35. public static void main(String[] args) {
36. new Hashtable().put(1, null);
37. }
    1. An exception is thrown at runtime.
    2. It compiles and runs fine.
    3. Compilation fails.
38. What happens when this code gets executed? (1 correct answer)
39. public static void main(String[] args) {
40. new Hashtable().put(null, 1);
41. }
    1. An exception is thrown at runtime.
    2. It compiles and runs fine.
    3. Compilation fails.
42. What happens when this code gets compiled and executed? (1 correct answer)
43. public static void main(String[] args) {
44. Map<String, Integer> map = new **LinkedHashMap**<String, Integer>();
45. map.put("J", 1);
46. map.put("A", 2);
47. map.put("V", 3);
48. map.put("A", 4);
49. for (Object element : map.keySet()) {
50. System.out.format("%s ", element);
51. }
52. }
    1. It prints “J A V A”
    2. It prints “J A V”
    3. It prints “A J V”
    4. The output order cannot be guaranteed.
53. What happens when this code gets compiled and executed? (1 correct answer)
54. public static void main(String[] args) {
55. Map<String, Integer> map = new **TreeMap**<String, Integer>();
56. map.put("J", 1);
57. map.put("A", 2);
58. map.put("V", 3);
59. map.put("A", 4);
60. for (Object element : map.keySet()) {
61. System.out.format("%s ", element);
62. }
63. }
    1. It prints “J A V A”
    2. It prints “J A V”
    3. It prints “A J V”
    4. The output order cannot be guaranteed.
64. What happens when this code gets compiled and executed? (1 correct answer)
65. public static void main(String[] args) {
66. Map<String, Integer> map = new HashMap<String, Integer>();
67. map.put("T;
68. map.put("M;
69. **map.keySet().add("A");**
70. System.out.println(map.size());
71. }
    1. It prints “2”
    2. It prints “3”
    3. An exception is thrown at runtime.
72. What happens when this code gets compiled and executed? (1 correct answer)
73. public static void main(String[] args) {
74. Map<String, Integer> map = new HashMap<String, Integer>();
75. map.put("T;
76. map.put("M;
77. **map.keySet().remove("T");**
78. System.out.println(map.size());
79. }
    1. It prints “1”
    2. It prints “2”
    3. An exception is thrown at runtime.
80. All the following methods belong to NavigableMap. (1 correct answer)  
        ⇒ lowerEntry()  
        ⇒ higherEntry()  
        ⇒ floorEntry()  
        ⇒ ceilingEntry()
    1. true
    2. false
81. All the following methods belong to NavigableMap. (1 correct answer)  
        ⇒ pollFirstEntry()  
        ⇒ pollLastEntry()  
        ⇒ firstEntry()  
        ⇒ lastEntry()
    1. true
    2. false
82. What is the output of this code? (1 correct answer)
83. public static void main(String[] args) {
84. NavigableMap<Integer, String> map =
85. new TreeMap<Integer, String>();
86. map.put(5, "D");
87. map.put(-1, "A");
88. map.put(7, "O");
89. System.out.format("%dd %d",
90. map.**lowerKey**(-100),
91. map.lowerKey(5),
92. map.lowerKey(6),
93. map.lowerKey(100)
94. );
95. }
    1. It prints “null -1 5 7″.
    2. It prints “null 5 5 7″.
96. What is the output of this code? (1 correct answer)
97. public static void main(String[] args) {
98. NavigableMap<Integer, String> map =
99. new TreeMap<Integer, String>();
100. map.put(5, "D");
101. map.put(-1, "A");
102. map.put(7, "O");
103. System.out.format("%dd %d",
104. map.**floorKey**(-100),
105. map.floorKey(5),
106. map.floorKey(6),
107. map.floorKey(100)
108. );
109. }
     1. It prints “null -1 5 7″.
     2. It prints “null 5 5 7″.
110. What is the output of this code? (1 correct answer)
111. public static void main(String[] args) {
112. NavigableMap<Integer, String> map =
113. new TreeMap<Integer, String>();
114. map.put(5, "D");
115. map.put(-1, "A");
116. map.put(7, "O");
117. System.out.format("%dd %d",
118. map.**higherKey**(-100),
119. map.higherKey(5),
120. map.higherKey(6),
121. map.higherKey(100)
122. );
123. }
     1. It prints “-1 7 7 null”.
     2. It prints “-1 5 7 null”.
124. What is the output of this code? (1 correct answer)
125. public static void main(String[] args) {
126. NavigableMap<Integer, String> map =
127. new TreeMap<Integer, String>();
128. map.put(5, "D");
129. map.put(-1, "A");
130. map.put(7, "O");
131. System.out.format("%dd %d",
132. map.**ceilingKey**(-100),
133. map.ceilingKey(5),
134. map.ceilingKey(6),
135. map.ceilingKey(100)
136. );
137. }
     1. It prints “-1 7 7 null”.
     2. It prints “-1 5 7 null”.
138. What is the output of this code? (1 correct answer)
139. public static void main(String[] args) {
140. NavigableMap<Integer, String> map =
141. new TreeMap<Integer, String>();
142. map.put(5, "D");
143. map.put(-1, "A");
144. map.put(7, "O");
145. System.out.format("%ss %s",
146. map.**lowerEntry**(-100),
147. map.lowerEntry(5),
148. map.lowerEntry(6),
149. map.lowerEntry(100)
150. );
151. }
     1. It prints “null -1=A 5=D 7=O”.
     2. It prints “null 5=D 5=D 7=O”.
152. What is the output of this code? (1 correct answer)
153. public static void main(String[] args) {
154. NavigableMap<Integer, String> map =
155. new TreeMap<Integer, String>();
156. map.put(5, "D");
157. map.put(-1, "A");
158. map.put(7, "O");
159. System.out.format("%ss %s",
160. map.**floorEntry**(-100),
161. map.floorEntry(5),
162. map.floorEntry(6),
163. map.floorEntry(100)
164. );
165. }
     1. It prints “null -1=A 5=D 7=O”.
     2. It prints “null 5=D 5=D 7=O”.
166. What is the output of this code? (1 correct answer)
167. public static void main(String[] args) {
168. NavigableMap<Integer, String> map =
169. new TreeMap<Integer, String>();
170. map.put(5, "D");
171. map.put(-1, "A");
172. map.put(7, "O");
173. System.out.format("%ss %s",
174. map.**higherEntry**(-100),
175. map.higherEntry(5),
176. map.higherEntry(6),
177. map.higherEntry(100)
178. );
179. }
     1. It prints “-1=A 7=O 7=O null”.
     2. It prints “-1=A 5=D 7=O null”.
180. What is the output of this code? (1 correct answer)
181. public static void main(String[] args) {
182. NavigableMap<Integer, String> map =
183. new TreeMap<Integer, String>();
184. map.put(5, "D");
185. map.put(-1, "A");
186. map.put(7, "O");
187. System.out.format("%ss %s",
188. map.**ceilingEntry**(-100),
189. map.ceilingEntry(5),
190. map.ceilingEntry(6),
191. map.ceilingEntry(100)
192. );
193. }
     1. It prints “-1=A 7=O 7=O null”.
     2. It prints “-1=A 5=D 7=O null”.
194. What is the output of this code? (1 correct answer)
195. public static void main(String[] args) {
196. NavigableMap<Integer, String> map =
197. new TreeMap<Integer, String>();
198. map.put(34, "J");
199. map.put(-1, "a");
200. map.put(70, "v");
201. map.put(5, "a");
202. map.put(-1, "2");
203. System.out.format("%ss %s",
204. map.ceilingEntry(-100),
205. map.floorEntry(5),
206. map.higherEntry(6),
207. map.lowerKey(5)
208. );
209. }
     1. It prints “-1=2 5=a 34=J -1″.
     2. It prints “-1=a 5=a 34=J -1″.
     3. It prints “-1=2 5=a 34=J 5″.
     4. It prints “-1=a 5=a 34=J 5″.
210. What is the output of this code? (1 correct answer)
211. public static void main(String[] args) {
212. NavigableMap<Integer, String> map =
213. new TreeMap<Integer, String>();
214. map.lastEntry();
215. System.out.println(map.size());
216. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “0”.
217. What is the output of this code? (1 correct answer)
218. public static void main(String[] args) {
219. NavigableMap<Integer, String> map =
220. new TreeMap<Integer, String>();
221. map.pollFirstEntry();
222. System.out.println(map.size());
223. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “0”.
224. What is the output of this code? (1 correct answer)
225. public static void main(String[] args) {
226. NavigableMap<Integer, String> map =
227. new TreeMap<Integer, String>();
228. map.firstKey();
229. System.out.println(map.size());
230. }
     1. An exception is thrown at runtime.
     2. Compilation fails.
     3. It prints “0”.
231. What is the output of this code? (1 correct answer)
232. public static void main(String[] args) {
233. NavigableMap<Integer, String> map =
234. new TreeMap<Integer, String>();
235. map.put(1, "A");
236. map.put(2, "B");
237. map.put(3, "C");
238. NavigableMap<Integer, String> sub =
239. map.subMap(0, 3);
240. System.out.println(sub.lastKey());
241. }
     1. It prints “2”.
     2. It prints “3”.
     3. Compilation fails.
     4. An exception is thrown at runtime.
242. What is the output of this code? (1 correct answer)
243. public static void main(String[] args) {
244. NavigableMap<Integer, String> map =
245. new TreeMap<Integer, String>();
246. map.put(1, "A");
247. map.put(2, "B");
248. map.put(3, "C");
249. NavigableMap<Integer, String> sub =
250. map.subMap(0, **false**, 3, **false**);
251. System.out.println(sub.lastKey());
252. }
     1. It prints “2”.
     2. It prints “3”.
     3. Compilation fails.
     4. An exception is thrown at runtime.
253. What is the output of this code? (1 correct answer)
254. public static void main(String[] args) {
255. NavigableMap<Integer, String> map =
256. new TreeMap<Integer, String>();
257. map.put(1, "A");
258. map.put(2, "B");
259. map.put(3, "C");
260. NavigableMap<Integer, String> sub =
261. map.subMap(0, false, 3, false);
262. **map**.put(4, "D");
263. System.out.format("%d map.size(), sub.size());
264. }
     1. It prints “2 2″.
     2. It prints “3 2″.
     3. It prints “3 3″.
     4. It prints “4 2″.
     5. It prints “4 3″.
     6. Compilation fails.
     7. An exception is thrown at runtime.
265. What is the output of this code? (1 correct answer)
266. public static void main(String[] args) {
267. NavigableMap<Integer, String> map =
268. new TreeMap<Integer, String>();
269. map.put(1, "A");
270. map.put(2, "B");
271. map.put(3, "C");
272. NavigableMap<Integer, String> sub =
273. map.subMap(0, false, 3, false);
274. **sub**.put(4, "D");
275. System.out.format("%d map.size(), sub.size());
276. }
     1. It prints “2 2″.
     2. It prints “3 2″.
     3. It prints “3 3″.
     4. It prints “4 2″.
     5. It prints “4 3″.
     6. Compilation fails.
     7. An exception is thrown at runtime.
277. Queue is an interface. (1 correct answer)
     1. true
     2. false
278. PriorityQueue IS-A Queue. (1 correct answer)
     1. true
     2. false
279. PriorityQueue IS-A AbstractQueue. (1 correct answer)
     1. true
     2. false
280. All the following are methods of Queue. (1 correct answer)  
         ⇒ offer()  
         ⇒ poll()  
         ⇒ peek()  
         ⇒ add()  
         ⇒ remove()  
         ⇒ element()
     1. true
     2. false
281. How many of the following methods of Queue may throw an exception at runtime? (1 correct answer)  
         ⇒ offer()  
         ⇒ poll()  
         ⇒ peek()
     1. One.
     2. Two.
     3. Three.
282. How many of the following methods of Queue may throw an exception at runtime? (1 correct answer)  
         ⇒ add()  
         ⇒ remove()  
         ⇒ element()
     1. One.
     2. Two.
     3. Three.
283. Consider Queue<E>. What’s the signature of the method offer()? (1 correct answer)
     1. E offer(E)
     2. void offer(E)
     3. boolean offer(E)
284. Consider Queue<E>. What’s the signature of the method peek()? (1 correct answer)
     1. E peek()
     2. E peek(E)
     3. boolean peek(E)
285. In the Queue interface there are 2 overloaded methods with the name **remove**. (1 correct answer)
     1. true
     2. false
286. In the Queue interface both add() and offer() add an element to the queue. (1 correct answer)
     1. true
     2. false
287. What is the output of this code? (1 correct answer)
288. public static void main(String[] args) {
289. Queue<String> queue = new PriorityQueue<String>();
290. System.out.println(queue.**add**(null));
291. }
     1. It prints “true”.
     2. It prints “false”.
     3. An exception is thrown at runtime.
292. What is the output of this code? (1 correct answer)
293. public static void main(String[] args) {
294. Queue<String> queue = new PriorityQueue<String>();
295. System.out.println(queue.**offer**(null));
296. }
     1. It prints “true”.
     2. It prints “false”.
     3. An exception is thrown at runtime.
297. What is the output of this code? (1 correct answer)
298. public static void main(String[] args) {
299. Queue<String> queue = new PriorityQueue<String>();
300. System.out.println(queue.**peek**());
301. }
     1. It prints “null”.
     2. An exception is thrown at runtime.
302. What is the output of this code? (1 correct answer)
303. public static void main(String[] args) {
304. Queue<String> queue = new PriorityQueue<String>();
305. System.out.println(queue.**poll**());
306. }
     1. It prints “null”.
     2. An exception is thrown at runtime.
307. What is the output of this code? (1 correct answer)
308. public static void main(String[] args) {
309. Queue<String> queue = new PriorityQueue<String>();
310. System.out.println(queue.**element**());
311. }
     1. It prints “null”.
     2. An exception is thrown at runtime.
312. What is the output of this code? (1 correct answer)
313. public static void main(String[] args) {
314. Queue<String> queue = new PriorityQueue<String>();
315. System.out.println(queue.**remove**());
316. }
     1. It prints “null”.
     2. An exception is thrown at runtime.
317. What is the output of this code? (1 correct answer)
318. public static void main(String[] args) {
319. Queue<String> queue = new PriorityQueue<String>();
320. System.out.format("%bd",
321. queue.add("A       queue.add("A       queue.size());
322. }
     1. It prints “true true 2″.
     2. It prints “true false 1″.
     3. An exception is thrown at runtime.
323. What is the output of this code? (1 correct answer)
324. public static void main(String[] args) {
325. Queue<String> queue = new PriorityQueue<String>();
326. queue.add("E   queue.add("J   queue.add("B   queue.add("3   System.out.println(queue.element());
327. }
     1. It prints “B”.
     2. It prints “J”.
     3. It prints “3”.
328. What is the output of this code? (1 correct answer)
329. public static void main(String[] args) {
330. Queue<String> queue = new PriorityQueue<String>(
331. **Collections.reverseOrder()**);
332. queue.add("E   queue.add("J   queue.add("B   queue.add("3   System.out.println(queue.element());
333. }
     1. It prints “B”.
     2. It prints “J”.
     3. It prints “3”.
     4. Compilation fails.
334. What is the output of this code? (1 correct answer)
335. public static void main(String[] args) {
336. Queue<String> queue = new PriorityQueue<String>(
337. **4**, Collections.reverseOrder());
338. queue.add("E   queue.add("J   queue.add("B   queue.add("3   System.out.println(queue.element());
339. }
     1. It prints “B”.
     2. It prints “J”.
     3. It prints “3”.
     4. Compilation fails.
340. What is the output of this code? (1 correct answer)
341. public static void main(String[] args) {
342. Queue<String> queue = new PriorityQueue<String>();
343. queue.add("J   queue.add("A   queue.add("V   queue.add("A   **Arrays.sort(queue.toArray());**
344. for (String element : queue) {
345. System.out.format("%slement);
346. }
347. }
     1. It prints “J A V A”.
     2. It prints “A A J V”.
     3. The output order is not guaranteed.
348. What is the output of this code? (1 correct answer)
349. public static void main(String[] args) {
350. Queue<String> queue = new PriorityQueue<String>();
351. queue.add("J   queue.add("A   queue.add("V   queue.add("A   String[] **array** = queue.toArray(new String[0]);
352. Arrays.sort(**array**);
353. for (String element : **array**) {
354. System.out.format("%slement);
355. }
356. }
     1. It prints “J A V A”.
     2. It prints “A A J V”.
     3. The output order is not guaranteed.
357. What happens when this code is executed? (1 correct answer)
358. public static void main(String[] args) {
359. Queue queue = new PriorityQueue();
360. queue.add(null);
361. }
     1. An exception is thrown at runtime.
     2. It runs just fine.
362. What is the output of this code? (1 correct answer)
363. public static void main(String[] args) {
364. Queue queue = new PriorityQueue();
365. queue.add(1);
366. queue.add("J   System.out.print(queue.element());
367. }
     1. It prints “J”.
     2. It prints “1”.
     3. An exception is thrown at runtime.
368. What is the output of this code? (1 correct answer)
369. public static void main(String[] args) {
370. Queue<String> queue = new PriorityQueue<String>();
371. queue.add("Z   queue.add("T   queue.add("F   queue.add("A   queue.add("A   queue.add("C   queue.offer("G   queue.poll();
372. queue.peek();
373. queue.remove();
374. queue.element();
375. System.out.format("%s
376. queue.poll(),
377. queue.size()
378. );
379. }
     1. It prints “A 6″.
     2. It prints “A 5″.
     3. It prints “C 5″.
     4. It prints “C 4″.
     5. It prints “F 4″.
     6. It prints “F 3″.
380. List is an interface. (1 correct answer)
     1. true
     2. false
381. LinkedList IS-A List. Also, LinkedList IS-A Queue since Java 1.5. (1 correct answer)
     1. true
     2. false
382. In the List interface there are two overloaded methods with the name **remove**.
     1. true
     2. false
383. In the List interface there are two overloaded methods with the name **addAll**.
     1. true
     2. false
384. Consider List<E>. What is the signature of the method**lastIndexOf**?
     1. E lastIndexOf(int)
     2. int lastIndexOf(E)
     3. int lastIndexOf(Object)
385. Consider List<E>. What is the signature of the method **set**?
     1. E set(int, E)
     2. int set(int, E)
     3. boolean set(int, E)
386. What is the output of this code? (1 correct answer)
387. public static void main(String[] args) {
388. List<Integer> list = new ArrayList<Integer>();
389. List<Integer> sub = list.subList(0, 1);
390. System.out.println(sub.size());
391. }
     1. It prints “0”.
     2. It prints “1”.
     3. Compilation fails.
     4. An exception is thrown at runtime.
392. What is the output of this code? (1 correct answer)
393. public static void main(String[] args) {
394. List<Integer> list = new ArrayList<Integer>();
395. list.add(1);
396. list.add(2);
397. List<Integer> sub = list.subList(0, 0);
398. System.out.println(sub.size());
399. }
     1. It prints “0”.
     2. It prints “1”.
     3. Compilation fails.
     4. An exception is thrown at runtime.
400. What is the output of this code? (1 correct answer)
401. public static void main(String[] args) {
402. List list = new ArrayList();
403. list.add(1);
404. list.add(2);
405. for (int element : list) {
406. System.out.format("%dlement);
407. }
408. }
     1. It prints “1 2 “.
     2. Compilation fails.
409. What is the output of this code? (1 correct answer)
410. public static void main(String[] args) {
411. List**<Integer>** list = new ArrayList**<Integer>**();
412. list.add(1);
413. list.add(2);
414. for (int element : list) {
415. System.out.format("%dlement);
416. }
417. }
     1. It prints “1 2 “.
     2. Compilation fails.
418. What is the output of this code? (1 correct answer)
419. public static void main(String[] args) {
420. List<Integer> list = new ArrayList<Integer>();
421. list.add(1);
422. list.add(2);
423. for (**Object** element : list) {
424. System.out.format("%dlement);
425. }
426. }
     1. It prints “1 2 “.
     2. Compilation fails.
427. What is the output of this code? (1 correct answer)
428. public static void main(String[] args) {
429. List list = new ArrayList();
430. list.add(1);
431. list.add(2);
432. for (Object element : list) {
433. System.out.format("%dlement);
434. }
435. }
     1. It prints “1 2 “.
     2. Compilation fails.
436. What is the output of this code? (1 correct answer)
437. public static void main(String[] args) {
438. List<String> list = new LinkedList<String>();
439. list.add("X   list.add("M   list.add("L   for (String element : list) {
440. System.out.format("%slement);
441. }
442. }
     1. It prints “X M L “.
     2. Compilation fails.
443. What is the output of this code? (1 correct answer)
444. public static void main(String[] args) {
445. List<String> list = new LinkedList<String>();
446. list.add("X   list.add("M   list.add("L   System.out.println(list.peek());
447. }
     1. It prints “X”.
     2. It prints “L”.
     3. Compilation fails.
448. What is the output of this code? (1 correct answer)
449. public static void main(String[] args) {
450. **LinkedList**<String> list = new LinkedList<String>();
451. list.add("X");
452. list.add("M");
453. list.add("L");
454. System.out.println(list.peek());
455. }
     1. It prints “X”.
     2. It prints “L”.
     3. Compilation fails.
456. What is the output of this code? (1 correct answer)
457. public static void main(String[] args) {
458. List<Number> list = new ArrayList<Number>();
459. list.add(7);
460. list.add(8);
461. list.add(9);
462. list.remove(7);
463. System.out.println(list.size());
464. }
     1. It prints “2”.
     2. It prints “3”.
     3. An exception is thrown at runtime.
465. What is the output of this code? (1 correct answer)
466. public static void main(String[] args) {
467. List<Number> list = new ArrayList<Number>();
468. list.add(7);
469. list.add(8);
470. list.add(9);
471. list.remove(**Integer.valueOf(7)**);
472. System.out.println(list.size());
473. }
     1. It prints “2”.
     2. It prints “3”.
     3. An exception is thrown at runtime.
474. What is the output of this code? (1 correct answer)
475. public static void main(String[] args) {
476. List<Number> list = new ArrayList<Number>();
477. list.add(7);
478. list.add(8);
479. list.add(7);
480. Number index = list.get(7);
481. System.out.println(index);
482. }
     1. It prints “0”.
     2. It prints “2”.
     3. An exception is thrown at runtime.
483. What is the output of this code? (1 correct answer)
484. public static void main(String[] args) {
485. List<Number> list = new ArrayList<Number>();
486. list.add(7);
487. list.add(8);
488. list.add(7);
489. Number index = list.get(Integer.valueOf(7));
490. System.out.println(index);
491. }
     1. It prints “0”.
     2. It prints “2”.
     3. An exception is thrown at runtime.
492. What is the output of this code? (1 correct answer)
493. public static void main(String[] args) {
494. List<Number> list = new ArrayList<Number>();
495. System.out.format("%bb %d",
496. list.add(7),
497. list.add(null),
498. list.add(7),
499. list.size());
500. }
     1. It prints “true false false 1″.
     2. It prints “true true false 2″.
     3. It prints “true false true 2″.
     4. It prints “true true true 3″.
     5. An exception is thrown at runtime.

© 2008 Nikos Pougounias. This is a free contribution to the [Java](http://java.sun.com/)community. Please distribute it for free. [http://nikojava.wordpress.com](http://nikojava.wordpress.com/)

**Answers**

1. a
2. a
3. d
4. a
5. a
6. a
7. d
8. d
9. a
10. d
11. c
12. b
13. b
14. c
15. a
16. a
17. a
18. c
19. a
20. a
21. b
22. c
23. b
24. c
25. b
26. c
27. a
28. b
29. c
30. b
31. b
32. b
33. b
34. c
35. b
36. d
37. b
38. d
39. b
40. c
41. b
42. a
43. a
44. d
45. a
46. a
47. c
48. a
49. c
50. c
51. a
52. c
53. b
54. b
55. b
56. a
57. c
58. a
59. c
60. c
61. b
62. a
63. a
64. b
65. c
66. c
67. a
68. a
69. a
70. a
71. b
72. a
73. b
74. a
75. b
76. a
77. b
78. a
79. c
80. c
81. a
82. c
83. a
84. d
85. g
86. a
87. a
88. a
89. a
90. a
91. c
92. c
93. a
94. a
95. a
96. c
97. c
98. a
99. a
100. b
101. b
102. a
103. c
104. d
105. b
106. c
107. b
108. a
109. c
110. d
111. a
112. a
113. a
114. a
115. c
116. a
117. d
118. a
119. b
120. a
121. a
122. a
123. a
124. c
125. a
126. c
127. a
128. c
129. c
130. d