DURGA ONLINE EXAMS

DURGAJOBS.COM
Continuous Job Updates For Every hour

Test Your Knowledge

HOME

251) **Given:**

```
1. public class Score implements Comparable {
2. private int wins, losses;
3. public Score(int w, int 1) { wins = w; losses = 1; }
4. public int getWins() { return wins; }
5. public int getLosses() { return losses; }
6. public String toString() {
7. return "<" + wins + "," + losses + ">";
8. }
9. // insert code here
10. }
```

Which method will complete this class?

1) **public int compare(Object o1,Object o2){/*more code here*/}**
2) **public int compare(Score s1,Score s2){/*more code here*/}**
3) **public int compareTo(Score other) {/*more code here*/}**
4) **public int compareTo(Object o) {/*mode code here*/}**

Your Selected options :: none
Correct Options :: 3

Click Here for Explanation

252) **Given a pre-generics implementation of a method:**

```
11. public static int sum(List list) {
12. int sum = 0;
13. for ( Iterator iter = list.iterator(); iter.hasNext(); ) {
14. int i = ((Integer)iter.next()).intValue();
15. sum += i;
16. }
17. return sum;
18. }
```

Which three changes must be made to the method sum to use generics? (Choose three.)

1) **remove line 14**
2) **replace line 14 with "int i = iter.next();"**
3) **replace line 13 with "for (int i : intList) {"**
4) **replace line 13 with "for (Iterator iter : intList) {"**
5) **replace the method declaration with "sum(List<int> intList)"**
6) **replace the method declaration with "sum(List<Integer> intList)"**

Your Selected options :: none
Correct Options :: 1, 3, 6

Click Here for Explanation

253) **Given:**

```
3. import java.util.*;
4. public class Quest {
5. public static void main(String[] args) {
6. String[] colors = {"blue", "red", "green", "yellow", "orange"};
7. Arrays.sort(colors);
8. int s2 = Arrays.binarySearch(colors, "orange");
9. int s3 = Arrays.binarySearch(colors, "violet");
10. System.out.println(s2 + " " + s3);
11. }
12. }
```

What is the result?

1) **2 -1**
2) **2 -4**
3) **2 -5**

4) **3 -1**

5) **3 -4**

6) **3 -5**

7) **Compilation fails.**

8) **An exception is thrown at runtime.**

Your Selected options :: none

Correct Options :: 3

Click Here for Explanation

254) **Given:**

```
11. public class Key {
12. private long id1;
13. private long id2;
14.
15. // class Key methods
16. }
```

A programmer is developing a class Key, that will be used as a key in a standard java.util.HashMap.Which two methods should be overridden to assure that Key works correctly as a key? (Choose two.)

1) **public int hashCode()**

2) **public boolean equals(Key k)**

3) **public int compareTo(Object o)**

4) **public boolean equals(Object o)**

5) **public boolean compareTo(Key k)**

Your Selected options :: none

Correct Options :: 1, 4

Click Here for Explanation

255) **Given:**

```
1. import java.util.*;
2. public class PQ {
3. public static void main(String[] args) {
4. PriorityQueue<String> pq = new PriorityQueue<String>();
5. pq.add("carrot");
6. pq.add("apple");
7. pq.add("banana");
8. System.out.println(pq.poll() + ":" + pq.peek());
9. }
10. }
```

What is the result?

1) **apple:apple**

2) **carrot:apple**

3) **apple:banana**

4) **banana:apple**

5) **carrot:carrot**

6) **carrot:banana**

Your Selected options :: none

Correct Options :: 3

Click Here for Explanation

256) **Given:**

```
13. public static void search(List<String> list) {
14. list.clear();
15. list.add("b");
16. list.add("a");
17. list.add("c");
18. System.out.println(Collections.binarySearch(list, "a"));
19. }
```

What is the result of calling search with a valid List implementation?

1) **0**

2) **1**

3) **2**

4) **a**

5) **b**

6) **c**

7) **The result is undefined.**

Your Selected options :: none

Correct Options :: 7

Click Here for Explanation

257) **Given:**

```
1. import java.util.*;
2. public class Old {
3. public static Object get0(List list) {
4. return list.get(0);
5. }
6. }
```

Which three will compile successfully? (Choose three.)

1) **Object o = Old.get0(new LinkedList());**

2) **Object o = Old.get0(new LinkedList<?>());**

3) **String s = Old.get0(new LinkedList<String>());**

4) **Object o = Old.get0(new LinkedList<Object>());**

5) **String s = (String)Old.get0(new LinkedList<String>());**

Your Selected options :: none

Correct Options :: 1, 4, 5

Click Here for Explanation

258) **Given:**

```
11. public static Collection get() {
12. Collection sorted = new LinkedList();
13. sorted.add("B"); sorted.add("C"); sorted.add("A");
14. return sorted;
15. }
16. public static void main(String[] args) {
17. for (Object obj: get()) {
18. System.out.print(obj + ", ");
19. }
20. }
```

What is the result?

1) **A, B, C,**

2) **B, C, A,**

3) **Compilation fails.**

4) **The code runs with no output.**

5) **An exception is thrown at runtime.**

Your Selected options :: none

Correct Options :: 2

Click Here for Explanation

259) **Given:**

***int[] myArray = new int[] {1, 2, 3, 4, 5};* What allows you to create a list from this array?**

1) **List myList = myArray.asList();**

2) **List myList = Arrays.asList(myArray);**

3) **List myList = new ArrayList(myArray);**

4) **List myList = Collections.fromArray(myArray);**

Your Selected options :: none

Correct Options :: 2

Click Here for Explanation

260) **Given:**
11. List list = // more code here
12. Collections.sort(list, new MyComparator());
Which code will sort this list in the opposite order of the sort in line 12?

1) **Collections.reverseSort(list, new MyComparator());**
2) **Collections.sort(list, new MyComparator());**
list.reverse();
3) **Collections.sort(list, new InverseComparator(**
new MyComparator()));
4) **Collections.sort(list, Collections.reverseOrder(**
new MyComparator()));

Your Selected options :: none

Correct Options :: 4

Click Here for Explanation

« Prev | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | **26** | 27 | 28 | 29 | 30 |
Next »

Total No.of Questions	::	292
Total No.of Answered Questions	::	0
Total No.of Unanswered Questions	::	292
Marks	::	0/292(0%)

feedback :: **feedback@durgajobs.com**