***Collections***

**Q: 08 Given:**

**1. import java.util.\*;**

**2. public class Example {**

**3. public static void main(String[] args) {**

**4. // insert code here**

**5. set.add(new Integer(2));**

**6. set.add(new Integer(1));**

**7. System.out.println(set);**

**8. }**

**9. }**

**Which code, inserted at line 4, guarantees that this program will output [1, 2]?**

A. Set set = new TreeSet();

B. Set set = new HashSet();

C. Set set = new SortedSet();

D. List set = new SortedList();

E. Set set = new LinkedHashSet();

**Answer: A**

**Q: 11 Given:**

**11. public static Iterator reverse(List list) {**

**12. Collections.reverse(list);**

**13. return list.iterator();**

**14. }**

**15. public static void main(String[] args) {**

**16. List list = new ArrayList();**

**17. list.add("1"); list.add("2"); list.add("3");**

**18. for (Object obj: reverse(list))**

**19. System.out.print(obj + ", ");**

**20. }**

**What is the result?**

A. 3, 2, 1, B. 1, 2, 3,

C. Compilation fails. D. The code runs with no output.

E. An exception is thrown at runtime.

**Answer: C**

**Q: 13 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?**

A. apple:apple

B. carrot:apple

C. apple:banana

D. banana:apple

E. carrot:carrot

F. carrot:banana

**Answer: C**

**Q: 20 Which two statements are true about the hashCode method? (Choose two.)**

A. The hashCode method for a given class can be used to test for object equality and object inequality for that class.

B. The hashCode method is used by the java.util.SortedSet collection class to order the elements within that set.

C. The hashCode method for a given class can be used to test for object inequality, but NOT object equality, for that class.

D. The only important characteristic of the values returned by a hashCode method is that the distribution of values must follow a Gaussian distribution.

E. The hashCode method is used by the java.util.HashSet collection class to group the elements within that set into hash buckets for swift retrieval.

**Answer: C, E**

**Q: 27 Given:**

**1. public class Drink implements Comparable {**

**2. public String name;**

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

**4. return 0;**

**5. }**

**6. }**

**and:**

**20. Drink one = new Drink();**

**21. Drink two = new Drink();**

**22. one.name= "Coffee";**

**23. two.name= "Tea";**

**23. TreeSet set = new TreeSet();**

**24. set.add(one);**

**25. set.add(two);**

**A programmer iterates over the TreeSet and prints the name of each Drink object.**

**What is the result?**

A. Tea

B. Coffee

C. Coffee Tea

D. Compilation fails.

E. The code runs with no output.

F. An exception is thrown at runtime.

**Answer: B**

**Q: 30 Given:**

**10. abstract public class Employee {**

**11. protected abstract double getSalesAmount();**

**12. public double getCommision() {**

**13. return getSalesAmount() \* 0.15;**

**14. }**

**15. }**

**16. class Sales extends Employee {**

**17. // insert method here**

**18. }**

**Which two methods, inserted independently at line 17, correctly complete the Sales class? (Choose two.)**

A. double getSalesAmount() { return 1230.45; }

B. public double getSalesAmount() { return 1230.45; }

C. private double getSalesAmount() { return 1230.45; }

D. protected double getSalesAmount() { return 1230.45; }

**Answer: B, D**

**Q: 33 Given:**

**11. public class Person {**

**12. private name;**

**13. public Person(String name) {**

**14. this.name = name;**

**15. }**

**16. public int hashCode() {**

**17. return 420;**

**18. }**

**19. }**

**Which statement is true?**

A. The time to find the value from HashMap with a Person key depends on the size of the map.

B. Deleting a Person key from a HashMap will delete all map entries for all keys of type Person.

C. Inserting a second Person object into a HashSet will cause the first Person object to be removed as a duplicate.

D. The time to determine whether a Person object is contained in a HashSet is constant and does NOT depend on the size of the map.

**Answer: A**

**Q: 42**

**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?**

A. Collections.reverseSort(list, new MyComparator());

B. Collections.sort(list, new MyComparator());

list.reverse();

C. Collections.sort(list, new InverseComparator(

new MyComparator()));

D. Collections.sort(list, Collections.reverseOrder(

new MyComparator()));

**Answer: D**

**Q: 43**

**Given:**

**ArrayList a = new ArrayList();**

**containing the values {“1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”}**

**Which code will return 2?**

A. Collections. sort(a, a.reverse());

int result = Collections.binarySearch(a, “6”);

B. Comparator c = Collections.reverseOrder();

Collections.sort(a, c);

int result = Collections.binarySearch(a, “6”);

C. Comparator c = Collections.reverseOrder();

Collections.sort(a, c);

int result = Collections.binarySearch(a, “6”,c);

D. Comparator c = Collections.reverseOrder(a);

Collections.sort(a, c);

int result = Collections.binarySearch(a, “6”,c);

E. Comparator c = new InverseComparator(new Comparator());

Collections.sort(a);

int result = Collections.binarySearch(a, “6”,c);

**Answer: C**

**Q: 54. Which collection class(es) allows you to grow or shrink its size and provides indexed access to its elements, but whose methods are not synchronized? (Choose all that apply.)**

A. java.util.HashSet

B. java.util.LinkedHashSet

C. java.util.List

D. java.util.ArrayList

E. java.util.Vector

F. java.util.PriorityQueue

**Answer:**

-> D is correct. All of the collection classes allow you to grow or shrink the size of your collection. ArrayList provides an index to its elements. The newer collection classes tend not to have synchronized methods. Vector is an older implementation of ArrayList functionality and has synchronized methods; it is slower than ArrayList.

-> A, B, C, E, and F are incorrect based on the logic described above; Notes: C, List is an interface, and F, PriorityQueue does not offer access by index.