SCJP Dumps Doc 10 of 16

This document is Tenth in series of 16 documents for Java SCJP Dumps. Links to all documents are available in Facebook group JAVA Tutorials. Join the group and download the documents.

Index : Download Links for other Doc :

Collections and Generics Part2

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: 31 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?
A. 0
B. 1
C. 2
D. a
E. b
F. c
G. The result is undefined.

Answer: G

Q: 32 Given:

```
11. public static void append(List list) { list.add("0042"); }
12. public static void main(String[] args) {
13. List<Integer> intList = new ArrayList<Integer>();
14. append(intList);
15. System.out.println(intList.get(0));
16. }
```

What is the result?

A. 42
B. 0042
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 13.
E. Compilation fails because of an error in line 14.

Answer: B

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: 34

A programmer must create a generic class MinMax and the type parameter of MinMax must implement Comparable. Which implementation of MinMax will compile?

A. class MinMax<E extends Comparable<E>> {
E min = null;
E max = null;
public MinMax() {}
public void put(E value) { /* store min or max */ }
B. class MinMax<E implements Comparable<E>> {
E min = null;
E max = null;
public MinMax() {}
public void put(E value) { /* store min or max */ }
C. class MinMax<E extends Comparable<E>> {
<E> E min = null;
<E> E max = null;
public MinMax() {}
public <E> void put(E value) { /* store min or max */ }
D. class MinMax<E implements Comparable<E>> {
<E> E min = null;
<E> E max = null;
public MinMax() {}
public <E> void put(E value) { /* store min or max */ }

Answer: A

Q: 35 Given:

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

A. List myList = myArray.asList();
B. List myList = Arrays.asList(myArray);
C. List myList = new ArrayList(myArray);
D. List myList = Collections.fromArray(myArray);

Answer: B

Question: 36

Given:

```
1. public class Score implements Comparable<Score> {
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?

A. public int compareTo(Object o) {/*mode code here*/}
B. public int compareTo(Score other) {/*more code here*/}
C. public int compare(Score s1,Score s2){/*more code here*/}
D. public int compare(Object o1,Object o2){/*more code here*/}

Answer: B

Question: 37

Click the Exhibit button.

```
import java.util.*;
class KeyMaster {
public int i;
public KeyMaster(int i) { this.i = i; }
public boolean equals(Object o) { return i == ((KeyMaster)o).i; }
public int hashCode() { return i; }
}
public class MapIt {
public static void main(String[] args) {
Set<KeyMaster> set = new HashSet<KeyMaster>();
KeyMaster k1 = new KeyMaster(1);
KeyMaster k2 = new KeyMaster(2);
set.add(k1); set.add(k1);
set.add(k2); set.add(k2);
System.out.print(set.size() + “:”);
k2.i = 1;
System.out.print(set.size() + “:”);
set.remove(k1);
System.out.print(set.size() + “:”);
set.remove(k2);
System.out.print(set.size());
}
}
```

What is the result?

A. 4:4:2:2
B. 4:4:3:2
C. 2:2:1:0
D. 2:2:0:0
E. 2:1:0:0
F. 2:2:1:1
G. 4:3:2:1

Answer: F

Question: 38

Given:

```
1. import java.util.*;
2. public class Test {
3. public static void main(String[] args) {
4. List<String> strings = new ArrayList<String>();
5. // insert code here
6. }
7. }
```

Which four, inserted at line *5,* will allow compilation to succeed? (Choose four.)

A. String s = strings.get(0);
B. Iterator i1 = strings.iterator();
C. String[] array1 = strings.toArray();
D. Iterator<String> i2 = strings.iterator();
E. String[] array2 = strings.toArray(new String[1]);
F. Iterator<String> i3 = strings.iterator<String>();
Answer: ABDE

Question: 39

Given:

```
classA {}
class B extends A {}
class C extends A {}
class D extends B {}
```

Which three statements are true? (Choose three.)

A. The type List<A> is assignable to List.
B. The type List<B> is assignable to List<A>.
C. The type List<Object> is assignable to List<?>.
D. The type List<D> is assignable to List<? extends B>.
E. The type List<? extends A> is assignable to List<A>.
F. The type List<Object> is assignable to any List reference.
G. The type List<? extends B> is assignable to List<? extends A>.

Answer: CDG

Question:40

Given:

11. public void addStrings(List list) {
12. list.add(”foo”);
13. list.add(”bar”);
14. }

What must you change in this method to compile without warnings?

A. add this code after line 11:
list = (List<String>) list;

B. change lines 12 and 13 to:
list.add<String>(”foo”);
list.add<String>(”bar”);

C. change the method signature on line 11 to:
public void addStrings(List<? extends String> list) {

D. change the method signature on line 11 to:
public void addStrings(List<? super String> list) {

E. No changes are necessary. This method compiles without warnings.

Answer: D

Question: 41

Given:

```
1. public class Test {
2. public <T extends Comparable> T findLarger(T x, T y) {
3. if(x.compareTo(y) > 0) {
4. return x;
5. } else {
6. return y;
7. }
8. }
9. }
```

and:

```
22. Test t = new Test();
23. // insert code here
```

Which two will compile without errors when inserted at line 23?

(Choose two.)

A. Object x = t.findLarger(123, *“456”);*
B. int x = t.findLarger(123, new Double(456));
C. int x = t.findLarger(123, new Integer(456));
D. int x = (int) t.findLarger(new Double(123), new Double(456));

Answer: A, C

Question: 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

Question: 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

Question: 44

Given:

```
11. public class Counter {
12. public static void main(String[] args) {
13. int numArgs = /* insert code here */;
14. }
15. }
```

and the command line:

```
java Counter one fred 42
```

Which code, inserted at line 13, captures the number of arguments passed into the program?

A. args.count
B. args.length
C. args.count()
D. args.length()
E. args.getLength()

Answer: B

45. Given:

```
import java.util.*;
class Test {
public static void main(String[] args) {
// insert code here
x.add("one");
x.add("two");
x.add("TWO");
System.out.println(x.poll());
}
}
```

Which, inserted at // insert code here, will compile? (Choose all that apply.)

A. List<String> x = new LinkedList<String>();
B. TreeSet<String> x = new TreeSet<String>();
C. HashSet<String> x = new HashSet<String>();
D. Queue<String> x = new PriorityQueue<String>();
E. ArrayList<String> x = new ArrayList<String>();
F. LinkedList<String> x = new LinkedList<String>();

Answer:

-> D and F are correct. The poll() method is associated with Queues. The LinkedList class implements the Queue interface.

->A is incorrect because the List interface does not implement Queue,

Question: 46

Click the Exhibit button.

Given:

```
1. public class TwoThreads {
2
3. private static Object resource = new Object();
4.
5. private static void delay(long n) {
6. try { Thread.sleep(n); }
7. catch (Exception e) { System.out.print(”Error “); }
8. }
9
10. public static void main(String[] args) {
11. System.out.print(”StartMain “);
12. new Thread1().start();
13. delay(1000);
14. Thread t2 = new Thread2();
15. t2.start();
16. delay(1000);
17. t2.interrupt
18. delay(1000);
19. System.out.print(”EndMain “);
20. }
21.
static class Thread 1 extends Thread {
public void run() {
synchronized (resource) {
System.out.print(”StartI “);
delay(6000);
System.out.print(”End1 “);
}
}
}

static class Thread2 extends Thread {
public void run() {
synchronized (resource) {
System.out.print(”Start2 “);
delay(2000);
System.out.print(”End2 “);
}
}
}
}
```

Assume that sleep(n) executes in exactly m milliseconds, and all other code executes in an insignificant amount of time. What is the output if the main() method is run?

A. Compilation fails.
B. Deadlock occurs.
C. StartMain Start1 Error EndMain End1
D. StartMain Start1 EndMain End1 Start2 End2
E. StartMain Start1 Error Start2 EndMain End2 End1
F. StartMain Start1 Start2 Error End2 EndMain End1
G. StartMain Start1 EndMain End1 Start2 Error End2

Answer: G

Question: 47

Click the Exhibit button.

```
10. public class Transfers {
11. public static void main(String[] args) throws Exception {
12. Record r1 = new Record();
13. Record r2 = new Record();
14. doTransfer(r1, r2, 5);
15. doTransfer(r2, r1, 2);
16. doTransfer(r1, r2, 1);
17. // print the result
18. System.out.println(”rl = “ + r1.get() +“, r2=” + r2.get());
19. }
20. private static void doTransfer(
21. final Record a, final Record b, final int amount) {
22. Thread t = new Thread() {
23. public void run() {
24. new Clerk().transfer(a, b, amount);
25. }
26. };
27. t.start();
28. }
29. }
30. class Clerk {
31. public synchronized void transfer(Record a, Record b, int amount){
32. synchronized (a) {
33. synchronized (b) {
34. a.add(-amount);
35. b.add(amount);
36. }
37. }
38. }
39. }
40. class Record {
41.int num=10;
42. public int get() { return num; }
43. public void add(int n) { num = num + n; }
44. }
```

If Transfers.main() is run, which three are true? (Choose three.)

A. The output may be “r1 = 6, r2 = 14”.
B. The output may be “r1 = *5,* r2 = *15”.*
C. The output may be “r1 = 8, r2 = 12”.
D. The code may run (and complete) with no output.
E. The code may deadlock (without completing) with no output.
F. M IllegalStateException or InterruptedException may be thrown at runtime.

Answer: A, B, E

48. Given:

```
public class Messager implements Runnable {
public static void main(String[] args) {
new Thread(new Messager("Wallace")).start();
new Thread(new Messager("Gromit")).start();
}
private String name;
public Messager(String name) { this.name = name; }
public void run() {
message(1); message(2);
}
private synchronized void message(int n) {
System.out.print(name + "-" + n + " ");
}
}
```

Which of the following is a possible result? (Choose all that apply.)

A. Wallace-1 Wallace-2 Gromit-1
B. Wallace-1 Gromit-2 Wallace-2 Gromit-1
C. Wallace-1 Gromit-1 Gromit-2 Wallace-2
D. Gromit-1 Gromit-2
E. Gromit-2 Wallace-1 Gromit-1 Wallace-2
F. The code does not compile.
G. An error occurs at run time.

Answer:

-> **C** is correct. Both threads will print two messages each. Wallace-1 must be before Wallace-2, and Gromit-1 must be before Gromit-2. Other than that, the Wallace and Gromit messages can be intermingled in any order.

-> **A, B, D, E, F,** and **G** are incorrect based on the above.

51. Given:

```
12. public class AccountManager {
13. private Map accountTotals = new HashMap();
14. private int retirementFund;
15.
16. public int getBalance(String accountName) {
17. Integer total = (Integer) accountTotals.get(accountName);
18. if (total == null)
19. total = Integer.valueOf(0);
20. return total.intValue();
21. }
23. public void setBalance(String accountName, int amount) {
24. accountTotals.put(accountName, Integer.valueOf(amount));
25. } }
```

This class is to be updated to make use of appropriate generic types, with no changes in behavior (for better or worse). Which of these steps could be performed? (Choose three.)

A. Replace line 13 with
private Map<String, int> accountTotals = new HashMap<String, int>();

B. Replace line 13 with
private Map<String, Integer> accountTotals = new HashMap<String, Integer>();

C. Replace line 13 with
private Map<String<Integer>> accountTotals = new HashMap<String<Integer>>();

D. Replace lines 17–20 with
int total = accountTotals.get(accountName);
if (total == null) total = 0;
return total;

E. Replace lines 17–20 with
Integer total = accountTotals.get(accountName);
if (total == null) total = 0;
return total;

F. Replace lines 17–20 with
return accountTotals.get(accountName);

G. Replace line 24 with
accountTotals.put(accountName, amount);

H. Replace line 24 with
accountTotals.put(accountName, amount.intValue());

Answer:

-> B , E, and G are correct.

-> A is wrong because you can't use a primitive type as a type parameter. C is wrong because a Map takes two type parameters separated by a comma. D is wrong because an int can't autobox to a null, and F is wrong because a null can't unbox to 0. H is wrong because you can't autobox a primitive just by trying to invoke a method with it.

52. Given a properly prepared String array containing five elements, which range of results could a proper invocation of Arrays.binarySearch() produce?

A. 0 through 4
B. 0 through 5
C. -1 through 4
D. -1 through 5
E. -5 through 4
F. -5 through 5
G. -6 through 4
H. -6 through 5

Answer:

-> G is correct. If a match is found, binarySearch()will return the index of the element that was matched. If no match is found, binarySearch() will return a negative number that, if inverted and then decremented, gives you the insertion point (array index) at which the value searched on should be inserted into the array to maintain a proper sort.

->A, B, C, D, E, F, and H are incorrect based on the above.

53. Given:

```
interface Hungry<E> { void munch(E x); }
interface Carnivore<E extends Animal> extends Hungry<E> {}
interface Herbivore<E extends Plant> extends Hungry<E> {}
abstract class Plant {}
class Grass extends Plant {}
abstract class Animal {}
class Sheep extends Animal implements Herbivore<Sheep> {
public void munch(Sheep x) {}
}
class Wolf extends Animal implements Carnivore<Sheep> {
public void munch(Sheep x) {}
}
```

Which of the following changes (taken separately) would allow this code to compile?
(Choose all that apply.)

A. Change the Carnivore interface to
interface Carnivore<E extends Plant> extends Hungry<E> {}

B. Change the Herbivore interface to
interface Herbivore<E extends Animal> extends Hungry<E> {}

C. Change the Sheep class to
```
class Sheep extends Animal implements Herbivore<Plant> {
public void munch(Grass x) {}
}
```

D. Change the Sheep class to
```
class Sheep extends Plant implements Carnivore<Wolf> {
public void munch(Wolf x) {}
}
```

E. Change the Wolf class to
```
class Wolf extends Animal implements Herbivore<Grass> {
public void munch(Grass x) {}
}
```

F. No changes are necessary.

Answer:

-> B is correct. The problem with the original code is that Sheep tries to implement Herbivore<Sheep> and Herbivore declares that its type parameter E can be any type that extends Plant. Since a Sheep is not a Plant, Herbivore<Sheep> makes no sense— the type Sheep is outside the allowed range of Herbivore's parameter E. Only solutions that either alter the definition of a Sheep or alter the definition of Herbivore will be able to fix this. So A, E, and F are eliminated. B works, changing the definition of an Herbivore to allow it to eat Sheep solves the problem. C doesn't work because an Herbivore<Plant> must have a munch(Plant) method, not munch(Grass). And D doesn't work, because in D we made Sheep extend Plant, now the Wolf class breaks because its munch(Sheep) method no longer fulfills the contract of Carnivore.

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.

55. Given:

```
import java.util.*;
public class Group extends HashSet<Person> {
public static void main(String[] args) {
Group g = new Group();
g.add(new Person("Hans"));
g.add(new Person("Lotte"));
g.add(new Person("Jane"));
g.add(new Person("Hans"));
g.add(new Person("Jane"));
System.out.println("Total: " + g.size());
}
public boolean add(Object o) {
System.out.println("Adding: " + o);
return super.add(o);
}
}
class Person {
private final String name;
public Person(String name) { this.name = name; }
public String toString() { return name; }
}
```

Which of the following occur at least once when the code is compiled and run?
(Choose all that apply.)

A. Adding Hans
B. Adding Lotte
C. Adding Jane
D. Total: 3
E. Total: 5
F. The code does not compile.
G. An exception is thrown at runtime.

Answer:

-> F is correct. The problem here is in Group's add() method—it should have been add(Person), since the class extends HashSet<Person>. So this doesn't compile. Pop Quiz: What would happen if you fixed this code, changing add(Object) to add(Person)? Try running the code to see if the results match what you thought.

-> A, B, C, D, E, and G are incorrect based on the above.

56. Given:

```
import java.util.*;
class AlgaeDiesel {
public static void main(String[] args) {
String[] sa = {"foo", "bar", "baz" };
// insert method invocations here
}
}
```

What java.util.Arrays and/or java.util.Collections methods could you use to convert sa to a List and then search the List to find the index of the element whose value is "foo"? (Choose from one to three methods.)

A. sort()
B. asList()
C. toList()
D. search()
E. sortList()
F. contains()
G. binarySearch()

Answer:

-> A, B, and G are required. The as List() method converts an array to a List. You can find the index of an element in a List with the binarySearch() method, but before you do that you must sort the list using sort().

-> F is incorrect because contains() returns a boolean, not an index. C, D, and E are incorrect, because these methods are not defined in the List interface.

57. Given that

```
String implements java.lang.CharSequence, and:
import java.util.*;
public class LongWordFinder {
public static void main(String[] args) {
String[] array = { "123", "12345678", "1", "12", "1234567890"};
List<String> list = Arrays.asList(array);
Collection<String> resultList = getLongWords(list);
}
// INSERT DECLARATION HERE
{
Collection<E> longWords = new ArrayList<E>();
for (E word : coll)
if (word.length() > 6) longWords.add(word);
return longWords;
} }
```

Which declarations could be inserted at // INSERT DECLARATION HERE so that the program will compile and run? (Choose all that apply.)

A. public static <E extends CharSequence> Collection<? extends CharSequence> getLongWords(Collection<E> coll)

B. public static <E extends CharSequence> List<E> getLongWords(Collection<E> coll)

C. public static Collection<E extends CharSequence> getLongWords(Collection<E> coll)

D. public static List<CharSequence> getLongWords(Collection<CharSequence> coll)

E. public static List<? extends CharSequence> getLongWords(Collection<? extends CharSequence> coll)

F. static public <E extends CharSequence> Collection<E> getLongWords(Collection<E> coll)

G. static public <E super CharSequence> Collection<E> getLongWords(Collection<E> coll)

Answer:

-> F is correct.

-> A is close, but it's wrong because the return value is too vague. The last line of the method expects the return value to be Collection<String>, not Collection<? extends CharSequence>. B is wrong because longWords has been declared as a Collection<E>, and that can't be implicitly converted to a List<E> to match the declared return value. (Even though we know that longWords is really an ArrayList<E>, the compiler only know what it's been declared as.) C, D, and E are wrong because they do not declare a type variable E (there's no <> before the return value) so the getLongWords() method body will not compile. G is wrong because E super CharSequence makes no sense—super could be used in conjunction with a wildcard but not a type variable like E.

58. Given:

```
TreeSet map = new TreeSet();
map.add("one");
map.add("two");
map.add("three");
map.add("four");
map.add("one");
Iterator it = map.iterator();
while (it.hasNext() ) {
System.out.print( it.next() + " " );
}
```

What is the result?

A. Compilation fails.
B. one two three four
C. four three two one
D. four one three two
E. one two three four one
F. one four three two one
G. An exception is thrown at runtime.
H. The print order is not guaranteed.

Answer:

-> D is correct. TreeSet assures no duplicate entries; also, when it is accessed it will return elements in natural order, which for Strings means alphabetical.

-> A, B, C, E, F, G, and H are incorrect based on the logic described above. Note, even though as of Java 5 you don't have to use an Iterator, you still can.

59. Given a method declared as:

public static <E extends Number> List<? super E> process(List<E> nums)

A programmer wants to use this method like this:
// INSERT DECLARATIONS HERE
output = process(input);
Which pairs of declarations could be placed at // INSERT DECLARATIONS HERE to allow the code to compile? (Choose all that apply.)

A. ArrayList<Integer> input = null;
ArrayList<Integer> output = null;

B. ArrayList<Integer> input = null;
List<Integer> output = null;

C. ArrayList<Integer> input = null;
List<Number> output = null;

D. List<Number> input = null;
ArrayList<Integer> output = null;

E. List<Number> input = null;
List<Number> output = null;

F. List<Integer> input = null;
List<Integer> output = null;

G. None of the above.

Answer:

->B, E, and F are correct.

-> The return type of process is definitely declared as a List, not an ArrayList, so A and D are wrong. C is wrong because the return type evaluates to List<Integer>, and that can't be assigned to a variable of type List<Number>. Of course all these would probably cause a NullPointerException since the variables are still null—but the question only asked us to get the code to compile.

Index : Download Links for other Doc :

JAVA OCPJPDumps Part 14 : JAVA I/O & Serialization

JAVA OCPJPDumps Part 15 : Inner Classes

JAVA OCPJPDumps Part 16 : Miscellaneous Questions

JAVA OCPJPMisc Dumps : Misc Dump Part 1

JAVA OCPJPMisc Dumps : Misc Dump Part2

JAVA OCPJPMisc Dumps : Misc Dump Part3

JAVA OCPJPMisc Dumps : Misc Dump Part4

JAVA OCPJPMisc Dumps : Misc Dump Part5

JAVA OCPJPMisc Dumps : Misc Dump Part6

JAVA OCPJPMisc Dumps : Misc Dump Part7