***Collections***

**Q: 01 Given:**

**34. HashMap props = new HashMap();**

**35. props.put("key45", "some value");**

**36. props.put("key12", "some other value");**

**37. props.put("key39", "yet another value");**

**38. Set s = props.keySet();**

**39. // insert code here**

**What, inserted at line 39, will sort the keys in the props HashMap?**

A. Arrays.sort(s);

B. s = new TreeSet(s);

C. Collections.sort(s);

D. s = new SortedSet(s);

**Answer: B**

**Q: 02 Click the Exhibit button.**



**Which statement is true about the set variable on line 12?**

A. The set variable contains all six elements from the coll collection, and the order is guaranteed to be

preserved.

B. The set variable contains only three elements from the coll collection, and the order is guaranteed to be preserved.

C. The set variable contains all six elements from the coll collection, but the order is NOT guaranteed to be preserved.

D. The set variable contains only three elements from the coll collection, but the order is NOT guaranteed to be preserved.

**Answer: D**

**Q: 03 Given:**

**23. Object [] myObjects = {**

**24. new Integer(12),**

**25. new String("foo"),**

**26. new Integer(5),**

**27. new Boolean(true)**

**28. };**

**29. Arrays.sort(myObjects);**

**30. for(int i=0; i<myObjects.length; i++) {**

**31. System.out.print(myObjects[i].toString());**

**32. System.out.print(" ");**

**33. }**

**What is the result?**

A. Compilation fails due to an error in line 23.

B. Compilation fails due to an error in line 29.

C. A ClassCastException occurs in line 29.

D. A ClassCastException occurs in line 31.

E. The value of all four objects prints in natural order.

**Answer: C**

**Q: 06 Given:**

**1. public class Person {**

**2. private String name;**

**3. public Person(String name) { this.name = name; }**

**4. public boolean equals(Person p) {**

**5. return p.name.equals(this.name);**

**6. }**

**7. }**

**Which statement is true?**

A. The equals method does NOT properly override the Object.equals method.

B. Compilation fails because the private attribute p.name cannot be accessed in line 5.

C. To work correctly with hash-based data structures, this class must also implement the hashCode method.

D. When adding Person objects to a java.util.Set collection, the equals method in line 4 will prevent duplicates.

**Answer: A**

**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: 09 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?**

A. A, B, C,

B. B, C, A,

C. Compilation fails.

D. The code runs with no output.

E. An exception is thrown at runtime.

**Answer: B**

**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: 14 Given:**

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

**2. public class WrappedString {**

**3. private String s;**

**4. public WrappedString(String s) { this.s = s; }**

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

**6. HashSet<Object> hs = new HashSet<Object>();**

**7. WrappedString ws1 = new WrappedString("aardvark");**

**8. WrappedString ws2 = new WrappedString("aardvark");**

**9. String s1 = new String("aardvark");**

**10. String s2 = new String("aardvark");**

**11. hs.add(ws1); hs.add(ws2); hs.add(s1); hs.add(s2);**

**12. System.out.println(hs.size()); } }**

**What is the result?**

A. 0

B. 1

C. 2

D. 3

E. 4

F. Compilation fails.

G. An exception is thrown at runtime.

**Answer: D**

**Q: 15 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.)**

A. public int hashCode()

B. public boolean equals(Key k)

C. public int compareTo(Object o)

D. public boolean equals(Object o)

E. public boolean compareTo(Key k)

**Answer: A, D**

**Q: 18 Given:**

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

**2.**

**3. public class LetterASort{**

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

**5. ArrayList<String> strings = new ArrayList<String>();**

**6. strings.add("aAaA");**

**7. strings.add("AaA");**

**8. strings.add("aAa");**

**9. strings.add("AAaa");**

**10. Collections.sort(strings);**

**11. for (String s : strings) { System.out.print(s + " "); }**

**12. }**

**13. }**

**What is the result?**

A. Compilation fails. B. aAaA aAa AAaa AaA

C. AAaa AaA aAa aAaA D. AaA AAaa aAaA aAa

E. aAa AaA aAaA AAaa F. An exception is thrown at runtime.

**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: 22 A programmer has an algorithm that requires a java.util.List that provides**

**an efficient implementation of add(0, object), but does NOT need to support quick random access.**

**What supports these requirements?**

A. java.util.Queue

B. java.util.ArrayList

C. java.util.LinearList

D. java.util.LinkedList

**Answer: D**

**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: 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: 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: 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: 37**

**Click the Exhibit button.**

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

**2. class KeyMaster {**

**3. public int i;**

**4. public KeyMaster(int i) { this.i = i; }**

*5.* **public boolean equals(Object o) { return i == ((KeyMaster)o).i; }**

**6. public int hashCode() { return i; }**

**7. }**

**8. public class MapIt {**

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

**10. Set<KeyMaster> set = new HashSet<KeyMaster>();**

**11. KeyMaster k1 = new KeyMaster(1);**

**12. KeyMaster k2 = new KeyMaster(2);**

**13. set.add(k1); set.add(k1);**

**14. set.add(k2); set.add(k2);**

*15.* **System.out.print(set.size() + “:”);**

**16. k2.i = 1;**

**17. System.out.print(set.size() + “:”);**

**18. set.remove(k1);**

**19. System.out.print(set.size() + “:”);**

**20. set.remove(k2);**

**21. System.out.print(set.size());**

**22. }**

**23. }**

**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: 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**

**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.**

**22. static class Thread 1 extends Thread {**

**23. public void run() {**

**24. synchronized (resource) {**

*25.* **System.out.print(”Startl “);**

**26. delay(6000);**

**27. System.out.print(”End1 “);**

**28. }**

**29. }**

**30. }**

**31.**

**32. static class Thread2 extends Thread {**

**33. public void run() {**

**34. synchronized (resource) {**

**35. System.out.print(”Start2 “***);*

**36. delay(2000);**

**37. System.out.print(”End2 “);**

**38. }**

**39. }**

**40. }**

**41. }**

**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: ABE**

**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.

**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.

**58. Given:**

**12. TreeSet map = new TreeSet();**

**13. map.add("one");**

**14. map.add("two");**

**15. map.add("three");**

**16. map.add("four");**

**17. map.add("one");**

**18. Iterator it = map.iterator();**

**19. while (it.hasNext() ) {**

**20. System.out.print( it.next() + " " );**

**21. }**

**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.