Collections TEST

After finishing please update on github

1. Given:

```
public static void main(String[] args) {
// INSERT DECLARATION HERE
for (int i = 0; i <= 10; i++) {
List<Integer> row = new ArrayList<Integer>();
for (int j = 0; j <= 10; j++)
row.add(i * j);
table.add(row);
}
for (List<Integer> row : table)
System.out.println(row);
}
```

Which statements could be inserted at // INSERT DECLARATION HERE to allow this code to compile and run? (Choose all that apply).

A. List<List<Integer>> table = new List<List<Integer>>();
B. List<List<Integer>> table = new ArrayList<List<Integer>>();
C. List<List<Integer>> table = new ArrayList<ArrayList<Integer>>();
D. List<List, Integer> table = new List<List, Integer>();
E. List<List, Integer> table = new ArrayList<List, Integer>();
F. List<List, Integer> table = new ArrayList<ArrayList, Integer>();
G. None of the above

2. Which statements are true about comparing two instances of the same class, given that the equals() and hashCode() methods have been properly overridden? (Choose all that apply.)

A. If the equals() method returns true, the hashCode() comparison == might return false
B. If the equals() method returns false, the hashCode() comparison == might return true
C. If the hashCode() comparison == returns true, the equals() method must return true
D. If the hashCode() comparison == returns true, the equals() method might return true
E. If the hashCode() comparison != returns true, the equals() method might return true

3. Given:

```
public static void before() {
Set set = new TreeSet();
set.add("2");
set.add(3);
set.add("1");
Iterator it = set.iterator();
while (it.hasNext())
```

System.out.print(it.next() + " ");
}
Which statements are true?

A. The before() method will print 1 2
B. The before() method will print 1 2 3
C. The before() method will print three numbers, but the order cannot be determined
D. The before() method will not compile
E. The before() method will throw an exception at runtime
4. 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.)
5. Given:
import java.util.*;
class MapEQ {
public static void main(String[] args) {
Map<ToDos, String> m = new HashMap<ToDos, String>();
ToDos t1 = new ToDos("Monday");
ToDos t2 = new ToDos("Monday");
ToDos t3 = new ToDos("Tuesday");
m.put(t1, "doLaundry");
m.put(t2, "payBills");
m.put(t3, "cleanAttic");
System.out.println(m.size());
} }
class ToDos{
String day;
ToDos(String d) { day = d; }
public boolean equals(Object o) {
return ((ToDos)o).day == this.day;
}
// public int hashCode() { return 9; }
}

Which is correct? (Choose all that apply.)

A. As the code stands it will not compile
B. As the code stands the output will be 2
C. As the code stands the output will be 3
D. If the hashCode() method is uncommented the output will be 2
E. If the hashCode() method is uncommented the output will be 3
F. If the hashCode() method is uncommented the code will not compile
6. 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
7. 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
8. Given a method declared as

```
public static <E extends Number> List<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

9. Given:

```
3. import java.util.*;
4. public class Mixup {
5. public static void main(String[] args) {
6. Object o = new Object();
7. // insert code here
8. s.add("o");
9. s.add(o);
10. }
11. }
```

And these three fragments:
I. Set s = new HashSet();
II. TreeSet s = new TreeSet();
III. LinkedHashSet s = new LinkedHashSet();
When fragments I, II, or III are inserted, independently, at line 7, which are true? (Choose all that apply.)

A. Fragment I compiles
B. Fragment II compiles
C. Fragment III compiles
D. Fragment I executes without exception
E. Fragment II executes without exception
F. Fragment III executes without exception

10. Given the proper import statement(s), and

```
13. PriorityQueue<String> pq = new PriorityQueue<String>();
14. pq.add("2");
15. pq.add("4");

16. System.out.print(pq.peek() + " ");
17. pq.offer("1");
18. pq.add("3");
19. pq.remove("1");
20. System.out.print(pq.poll() + " ");
21. if(pq.remove("2")) System.out.print(pq.poll() + " ");
```

22. System.out.println(pq.poll() + " " + pq.peek());
What is the result?

A. 2 2 3 3
B. 2 2 3 4
C. 4 3 3 4
D. 2 2 3 3 3
E. 4 3 3 3 3
F. 2 2 3 3 4
G. Compilation fails
H. An exception is thrown at runtime

11. Given the proper import statement(s), and:

```
13. TreeSet<String> s = new TreeSet<String>();
14. TreeSet<String> subs = new TreeSet<String>();
15. s.add("a"); s.add("b"); s.add("c"); s.add("d"); s.add("e");
16.
17. subs = (TreeSet)s.subSet("b", true, "d", true);
18. s.add("g");
19. s.pollFirst();
20. s.pollFirst();
21. s.add("c2");
22. System.out.println(s.size() +" "+ subs.size());
```

Which are true? (Choose all that apply.)

A. The size of s is 4
B. The size of s is 5
C. The size of s is 7
D. The size of subs is 1
E. The size of subs is 2
F. The size of subs is 3
G. The size of subs is 4
H. An exception is thrown at runtime

14. Given:

```
public static void main(String[] args) {
// INSERT DECLARATION HERE
for (int i = 0; i <= 10; i++) {
List<Integer> row = new ArrayList<Integer>();
for (int j = 0; j <= 10; j++)
row.add(i * j);
table.add(row);
}
for (List<Integer> row : table)
System.out.println(row);
}
```

Which statements could be inserted at // INSERT DECLARATION HERE to allow this code to compile and run? (Choose all that apply).

15. Which statements are true about comparing two instances of the same class, given that the equals() and hashCode() methods have been properly overridden? (Choose all that apply.)

A. If the equals() method returns true, the hashCode() comparison == might return false
B. If the equals() method returns false, the hashCode() comparison == might return true
C. If the hashCode() comparison == returns true, the equals() method must return true
D. If the hashCode() comparison == returns true, the equals() method might return true
E. If the hashCode() comparison != returns true, the equals() method might return true

16. Given:

```
public static void before() {
Set set = new TreeSet();
set.add("2");
set.add(3);
set.add("1");
Iterator it = set.iterator();
while (it.hasNext())
System.out.print(it.next() + " ");
}
```

Which statements are true?

A. The before() method will print 1 2
B. The before() method will print 1 2 3
C. The before() method will print three numbers, but the order cannot be determined
D. The before() method will not compile
E. The before() method will throw an exception at runtime

A. List<List<Integer>> table = new List<List<Integer>>();
B. List<List<Integer>> table = new ArrayList<List<Integer>>();
C. List<List<Integer>> table = new ArrayList<ArrayList<Integer>>();
D. List<List, Integer> table = new List<List, Integer>();
E. List<List, Integer> table = new ArrayList<List, Integer>();
F. List<List, Integer> table = new ArrayList<ArrayList, Integer>();
G. None of the above

17. Given:

```
import java.util.*;
class MapEQ {
public static void main(String[] args) {
Map<ToDos, String> m = new HashMap<ToDos, String>();
ToDos t1 = new ToDos("Monday");
ToDos t2 = new ToDos("Monday");
ToDos t3 = new ToDos("Tuesday");
m.put(t1, "doLaundry");
m.put(t2, "payBills");
m.put(t3, "cleanAttic");
System.out.println(m.size());
} }
```

```
class ToDos{
String day;
ToDos(String d) { day = d; }
public boolean equals(Object o) {
return ((ToDos)o).day == this.day;
}
// public int hashCode() { return 9; }
}
```

Which is correct? (Choose all that apply.)

A. As the code stands it will not compile
B. As the code stands the output will be 2
C. As the code stands the output will be 3
D. If the hashCode() method is uncommented the output will be 2
E. If the hashCode() method is uncommented the output will be 3
F. If the hashCode() method is uncommented the code will not compile

18. 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

19. Given the proper import statement(s), and

```
13. PriorityQueue<String> pq = new PriorityQueue<String>();
14. pq.add("2");
15. pq.add("4");
```

```
16. System.out.print(pq.peek() + " ");
17. pq.offer("1");
18. pq.add("3");
19. pq.remove("1");
20. System.out.print(pq.poll() + " ");
21. if(pq.remove("2")) System.out.print(pq.poll() + " ");
22. System.out.println(pq.poll() + " " + pq.peek());
```

What is the result?

A. 2 2 3 3
B. 2 2 3 4
C. 4 3 3 4
D. 2 2 3 3 3
E. 4 3 3 3 3
F. 2 2 3 3 4
G. Compilation fails
H. An exception is thrown at runtime

20. Given:

```
3. import java.util.*;
4. class Turtle {
5. int size;
6. public Turtle(int s) { size = s; }
7. public boolean equals(Object o) { return (this.size == ((Turtle)o).size); }
8. // insert code here
9. }
10. public class TurtleTest {
11. public static void main(String[] args) {
12. LinkedHashSet<Turtle> t = new LinkedHashSet<Turtle>();
13. t.add(new Turtle(1)); t.add(new Turtle(2)); t.add(new Turtle(1));
14. System.out.println(t.size());
15. }
16. }
```

And these two fragments:
I. public int hashCode() { return size/5; }
II. // no hashCode method declared
If fragment I or II is inserted, independently, at line 8, which are true? (Choose all that apply.)

A. If fragment I is inserted, the output is 2
B. If fragment I is inserted, the output is 3
C. If fragment II is inserted, the output is 2
D. If fragment II is inserted, the output is 3
E. If fragment I is inserted, compilation fails
F. If fragment II is inserted, compilation fails

21. Given:

```
3. import java.util.*;
4. class Turtle {
5. int size;
6. public Turtle(int s) { size = s; }
7. public boolean equals(Object o) { return (this.size == ((Turtle)o).size); }
8. // insert code here
9. }
10. public class TurtleTest {
11. public static void main(String[] args) {
12. LinkedHashSet<Turtle> t = new LinkedHashSet<Turtle>();
13. t.add(new Turtle(1)); t.add(new Turtle(2)); t.add(new Turtle(1));
14. System.out.println(t.size());
15. }
16. }
```

And these two fragments:
I. public int hashCode() { return size/5; }
II. // no hashCode method declared
If fragment I or II is inserted, independently, at line 8, which are true? (Choose all that apply.)

A. If fragment I is inserted, the output is 2
B. If fragment I is inserted, the output is 3
C. If fragment II is inserted, the output is 2
D. If fragment II is inserted, the output is 3
E. If fragment I is inserted, compilation fails
F. If fragment II is inserted, compilation fails
22. Given the proper import statement(s), and:

```
13. TreeSet<String> s = new TreeSet<String>();
14. TreeSet<String> subs = new TreeSet<String>();
15. s.add("a"); s.add("b"); s.add("c"); s.add("d"); s.add("e");
16.
17. subs = (TreeSet)s.subSet("b", true, "d", true);
18. s.add("g");
19. s.pollFirst();
20. s.pollFirst();
21. s.add("c2");
22. System.out.println(s.size() +" "+ subs.size());
```

Which are true? (Choose all that apply.)

A. The size of s is 4
B. The size of s is 5
C. The size of s is 7
D. The size of subs is 1
E. The size of subs is 2
F. The size of subs is 3
G. The size of subs is 4

H. An exception is thrown at runtime

23. Given:

```
3. import java.util.*;
4. public class Magellan {
5. public static void main(String[] args) {
6. TreeMap<String, String> myMap = new TreeMap<String, String>();
7. myMap.put("a", "apple"); myMap.put("d", "date");
8. myMap.put("f", "fig"); myMap.put("p", "pear");
9. System.out.println("1st after mango: " + // sop 1
10. myMap.higherKey("f"));
11. System.out.println("1st after mango: " + // sop 2
12. myMap.ceilingKey("f"));
13. System.out.println("1st after mango: " + // sop 3
14. myMap.floorKey("f"));
15. SortedMap<String, String> sub = new TreeMap<String, String>();
16. sub = myMap.tailMap("f");
17. System.out.println("1st after mango: " + // sop 4
18. sub.firstKey());
19. }
20. }
```

Which of the System.out.println statements will produce the output 1st after mango: p?
(Choose all that apply.)

A. sop 1
B. sop 2
C. sop 3
D. sop 4
E. None; compilation fails
F. None; an exception is thrown at runtime

23. Given:

```
3. import java.util.*;
4. class Dog { int size; Dog(int s) { size = s; } }
5. public class FirstGrade {
6. public static void main(String[] args) {
7. TreeSet<Integer> i = new TreeSet<Integer>();
8. TreeSet<Dog> d = new TreeSet<Dog>();
9.
10. d.add(new Dog(1)); d.add(new Dog(2)); d.add(new Dog(1));
11. i.add(1); i.add(2); i.add(1);
12. System.out.println(d.size() + " " + i.size());
13. }
14. }
```

What is the result?

A. 1 2

B. 2 2
C. 2 3
D. 3 2
E. 3 3
F. Compilation fails
G. An exception is thrown at runtime

24. Given:

```
3. import java.util.*;
4. public class GeoCache {
5. public static void main(String[] args) {
6. String[] s = {"map", "pen", "marble", "key"};
7. Othello o = new Othello();
8. Arrays.sort(s,o);

9. for(String s2: s) System.out.print(s2 + " ");
10. System.out.println(Arrays.binarySearch(s, "map"));
11. }
12. static class Othello implements Comparator<String> {
13. public int compare(String a, String b) { return b.compareTo(a); }
14. }
15. }
```

Which are true? (Choose all that apply).

A. Compilation fails
B. The output will contain a 1
C. The output will contain a 2
D. The output will contain a –1
E. An exception is thrown at runtime
F. The output will contain "key map marble pen"
G. The output will contain "pen marble map key"

25. Given:

```
public static void main(String[] args) {
// INSERT DECLARATION HERE
for (int i = 0; i <= 10; i++) {
List<Integer> row = new ArrayList<Integer>();
for (int j = 0; j <= 10; j++)
row.add(i * j);
table.add(row);
}
for (List<Integer> row : table)
System.out.println(row);
}
```

Which statements could be inserted at // INSERT DECLARATION HERE to allow this code to compile and run? (Choose all that apply).

A. List<List<Integer>> table = new List<List<Integer>>();
B. List<List<Integer>> table = new ArrayList<List<Integer>>();
C. List<List<Integer>> table = new ArrayList<ArrayList<Integer>>();
D. List<List, Integer> table = new List<List, Integer>();
E. List<List, Integer> table = new ArrayList<List, Integer>();
F. List<List, Integer> table = new ArrayList<ArrayList, Integer>();
G. None of the above
26. Given a method declared as
public static <E extends Number> List<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
27. 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
28. 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

29. 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());

30. Given:

```
public static void before() {
Set set = new TreeSet();
set.add("2");
set.add(3);
set.add("1");
Iterator it = set.iterator();
while (it.hasNext())
```

System.out.print(it.next() + " ");
}
Which statements are true?

A. The before() method will print 1 2
B. The before() method will print 1 2 3
C. The before() method will print three numbers, but the order cannot be determined
D. The before() method will not compile
E. The before() method will throw an exception at runtime
31. Which statements are true about comparing two instances of the same class, given that the equals() and hashCode() methods have been properly overridden? (Choose all that apply.)

A. If the equals() method returns true, the hashCode() comparison == might return false
B. If the equals() method returns false, the hashCode() comparison == might return true
C. If the hashCode() comparison == returns true, the equals() method must return true
D. If the hashCode() comparison == returns true, the equals() method might return true
E. If the hashCode() comparison != returns true, the equals() method might return true
32. Given:
```
public static void main(String[] args) {
// INSERT DECLARATION HERE
for (int i = 0; i <= 10; i++) {
List<Integer> row = new ArrayList<Integer>();
for (int j = 0; j <= 10; j++)
row.add(i * j);
table.add(row);
}
for (List<Integer> row : table)
System.out.println(row);
}
```
Which statements could be inserted at // INSERT DECLARATION HERE to allow this code to compile and run? (Choose all that apply).

A. List<List<Integer>> table = new List<List<Integer>>();
B. List<List<Integer>> table = new ArrayList<List<Integer>>();
C. List<List<Integer>> table = new ArrayList<ArrayList<Integer>>();
D. List<List, Integer> table = new List<List, Integer>();
E. List<List, Integer> table = new ArrayList<List, Integer>();
F. List<List, Integer> table = new ArrayList<ArrayList, Integer>();
G. None of the above
33. Given:
```
3. import java.util.*;
4. public class GeoCache {
5. public static void main(String[] args) {
6. String[] s = {"map", "pen", "marble", "key"};
7. Othello o = new Othello();
8. Arrays.sort(s,o);
for(String s2: s) System.out.print(s2 + " ");
System.out.println(Arrays.binarySearch(s, "map"));
}
static class Othello implements Comparator<String> {
public int compare(String a, String b) { return b.compareTo(a); }
}
}
```

Which are true? (Choose all that apply).

A. Compilation fails
B. The output will contain a 1
C. The output will contain a 2
D. The output will contain a –1
E. An exception is thrown at runtime
F. The output will contain "key map marble pen"
G. The output will contain "pen marble map key"