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. }
What three changes allow the class to be used with generics and avoid an unchecked warning?(Choose three.)
A. Remove line 14.
B. Replace line 14 with “int i = iter.next();”.
C. Replace line 13 with “for(int i:intList) {”.
D. Replace line 13 with “for(Iterator iter:intList){”.
E. Replace the method declaration with “sum(List<int>intList)”.
F. Replace the method declaration with sum “sum(List<Integer>intList)”.
Para eliminar los warnings que arroja una colección de datos, debo especificar el tipo de datos que dicha colección va a almacenar (lista de la línea 11). Este tipo de dato no puede ser una primitiva, luego “Replace the method declaration with “sum(List<int>intList)” es incorrecto y “Replace the method declaration with sum “sum(List<Integer>intList)” es correcto. Por otra parte, el bucle for() puede recibir como parámetro el tipo de dato sobre el que esto iterando. Tal como acabamos de ver, este caso ese tipo de dato es un Integer, luego “Replace line 13 with 'for(int i:intList) {'” es correcto y “Replace line 13 with 'for(Iterator iter:intList){'” es incorrecto. A, C, F
Given:
11. //insert code here
12. private N min, max;
13. public N getMin(){return min; }
14. public N getMax(){return max; }
15. public void add(N added){
16. if(min == null || added.doubleValue() < min.doubleValue())
17. min = added;
18. if(max == null || added.doubleValue() > max.doubleValue())
19. max = added;
20. }
21. }
Which two, inserted at line 11, will allow the code to compile?(Choose two.)
A. public class MinMax<?>{
B. public class MinMax<? extends Number>{
C. public class MinMax<N extends Object>{
D. public class MinMax<N extends Number>{
E. public class MinMax<? Extends Object> {
F. public class MinMax<N extends Integer>{
MinMax será una clase que almacene objetos de tipo N, los cuales podrán heredar de Number y de Object. No tiene sentido especificar que hereden de Object, ya que todo hereda de object. Los tipos genéricos van siempre entre los símbolos <>. D, F
A programmer has an algorithm that requires a java.util.List that provides an efficient implementation of(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.LienarList
D. java.util.LinkedList
LinkedList allows for constant-time insertions or removals, but only sequential access of elements. In other words, you can walk the list forwards or backwards, but grabbing an element in the middle takes time proportional to the size of the list.
ArrayLists, on the other hand, allow random access, so you can grab any element in constant time. But adding or

removing from anywhere but the end requires shifting all the latter elements over, either to make an opening or fill the gap. Also, if you add more elements than the capacity of the underlying array, a new array (twice the size) is allocated, and the old array is copied to the new one, so adding to an ArrayList is O(n) in the worst case but constant on average.

D

Given:
12. import java.util.*;
13. public class Explorer1{
14. public static void main(String [] args){
15. TreeSet<Integer>s = new TreeSet<Integer>();
16. TreeSet<Integer>subs = new TreeSet<Integer>();
17. for(int i = 606; i <613; i++)
18. if(i%2 == 0)s.add(i)
19. subs = (TreeSet)s.subSet(608, true, 611, true)
20. s.add(609);
21. System.out.println(s + "" + subs);
22. }
23. }
What is the result?
A. Compilation fails.
B. An exception is thrown at runtime.
C. [608, 609, 610, 612][608, 610]
D. [608, 609, 610, 612][608, 609, 610]
E. [606, 608, 609, 610, 612][608, 610]
F. [606, 608, 609, 610, 612][608, 609, 610]

F

Given:
12. import java.util.*;
13. public class Explorer2{
14. public static void main(String args[]){
15. TreeSet<Integer> s = new TreeSet<Integer>();
16. TreeSet<Integer> subs = new TreeSet<Integer>();
17. for(int i=606; i<613; i++)
18. if(i%2 == 0)s.add(i)
19. subs = (TreeSet)s.subSet(608, true, 611, true);
20. s.add(629);
21. System.out.println(s + "" + subs);
22. }
23. }
What is the result?
A. Compilation fails.
B. An exception is thrown at runtime.
C. [608, 610, 612, 629][608, 610]
D. [608, 610, 612, 629][608, 610, 629]
E. [606, 608, 610, 612, 629][608, 610]
F. [606, 608, 610, 612, 629][608, 610, 629]

subSet(E fromElement, E toElement)
Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive. E
Given:
12. import java.util.*;
13. public class Explorer3{
14. public static void main(String [] args){
15. TreeSet<Integer> s = new TreeSet<Integer>();
16. TreeSet<Integer> subs = new TreeSet<Integer>();
17. for(int i = 606; i <613; i++)

18. if(i%2 == 0)s.add(i);
19. subs = (TreeSet)s.subSet(608, true, 611, true);
20. subs.add(629);
21. System.out.println(s + “” + subs);
22. }
23. }
What is the result?
A. Compilation fails.
B. An exception is thrown at runtime.
C. [608, 610, 612, 629][608, 610]
D. [608, 610, 612, 629][608, 610, 629]
E. [606, 608, 610, 612, 629][608, 610]
F. [606, 608, 610, 612, 629][608, 610, 629]

F

Given:
1. public class Score implements Comparable<Score>
2. private int wins, loses;
3. public Score(int w, int i){wins = w; losses i;}
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){/*more 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*/}

Para comparar un objeto de la clase Score con otro, el segundo también debe ser de la clase Score.B

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 depent on the size of the map.

A

Given:
5. import java.util.*;
6. public class SortOf{
7. public static void main(String [] args){
8. ArrayList<Integer> a = new ArrayList<Integer>();
9. a.add(1); a.add(5); a.add(3);

11. Collection.sort(a);
12. a.add(2);
13. Collections.reverse(a);
14. System.out.println(a);
15. }
16. }
What is the result?
A. [1, 2, 3, 5]
B. [2, 1, 3, 5]
C. [2, 5, 3, 1]
D. [5, 3, 2, 1]
E. [1, 3, 5, 2]
F. Compilation fails.
G. An exception is thrown at runtime.

C

Given:
3. import java.util.*;
4. public class Mapit{
5. public static void main(String [] args){
6. Set <Integer> set = new HashSet<Integer>();
7. Integer i1 = 45;
8. Integer i2 = 46;
9. set.add(i1);
10. set.add(i1);
11. set.add(i2); System.out.print(set.size() + “”);
12. set.remove(i1); System.out.print(set.size() + “”);
13. i2 = 47;
14. set.remove(i2); System.out.prin(set.size() + “”);
15. }
16. }
What is the result?
A. 210
B. 211
C. 321
D. 322
E. Compilation fails.
F. An exception is thrown at runtime.

En un Set no puede haber valores repetidos. B

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[i].toString()){
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.

En un arregle todos los elementos deben tener el mismo tipo. C

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.

D

Given a class whose instances, when found in a collection of objects, are sorted by using the compareTo() method, which two statements are true?(Choose two.)
A. The class implements java.lang.Comparable
B. The class implements java.util.Comparator
C. The interface used to implement sorting allows this class to define only one sort sequence.
D. The interface used to implement sorting allows this class to define many different sort sequences.

A,C

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();

A

Given:
1. class TestException extends Exception{}
2. class A{
3. public String sayHello(Sring name) throws TestException{
4. if(name == null)throw new TestException();

```
5. return “Hello” + name;
6. }
7. }
8. public class TestA{
9. public static void main(String [] args){
10. new A().sayHello(“Aiko”);
11. }
12. }
```

Which statement is true?

A. Compilation succeeds.
B. Class A does not compile.
C. The method declared on line 9 cannot be modified to throw TestException.
D. TestA compiles if line 10 is enclosed in a try/catch block that catches TestException.

D

Given:

```
34. HashMap props = new HashMap();
35. props.put(“key45”, “some value”):
36. props.put(“ley12”, “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 int the props HashMap?

A. Arrays.sort(s)
B. s = new TreeSet(s);
C. Collections.sort(s);
D. s = new SortedSet();

Collections.sort(list) debe recibir como parámetro una lista. Un Set es una colección de datos que no admite valores duplicados. En un Tree los datos estarán almacenados de acuerdo a lo establecido por el método compareTo. Un TreeSet tiene ambas características. B

Given:

```
3. import java.util.*;
4. public class Hancock{
5. // insert code here
6. list.add(“foo”);
7. }
8. }
```

Which two code fragments, inserted independently at line 5, will compile without warnings?(Choose two.)

A. public void addStrings(List list);
B. public void addStrings(List<String>list)
C. public void addString(List<? Super String > list){
D. public void addStrings(List<? Extends String > list){

B, C

Given:

```
11. public void getNumbers(){
12. ArrayList numbers = new ArrayList();
13. for( int i=0; i<10; i++){
14. int value = i*((int)Math.random());
15. Integer intObj = new Integer(value);
16. numbers.add(intObj);
17. }
18. System.out.println(numbers);
19. }
```

Which line of code marks the earliest point that an object referenced by intObj becomes a candidate for garbage collection?

A. Line 16
B. Line 17
C. Line 18
D. Line 19
E. The object is NOT a candidate for garbage collection.

D.

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.

La línea 16 está mal. No es posible iterar sobre una instancia de la clase IteratorC

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. }
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.

B

Given that the elements of a PriorityQueue are ordered according to natural ordering, and:
2. import java.util.*;
3. public class GetInLine{
4. public static void main(String [] args){
5. PriorityQueue <String> pq = new PriorityQueue<String>();
6. pq.add("banana");
7. pq.add("pear");
8. pq.add("apple");
9. System.out.println(pq.poll() + "" + pq.peek());
10. }
11. }
What is the result?
A. apple pear

B. banana pear
C. apple apple
D. apple banana
E. banana banana

PriorityQueue almacena elementos ordenándolos de acuerdo al método compareTo. Por tanto, los objetos quedarán almacenado en el siguiente orden alfabético: apple, banana, pear. El método peek devuelve el primer elemento de la lista y el método poll devuelve y elimina el primer elemento de la lista..D

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 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*/}
```

A

Given:
```
3. import java.util.*;
4. public class G1{
5. public void takeList(List<? extends String > list ){
6. // insert code here
7. }
8. }
```
Which three code fragments, inserted independently at line 6, will compile? (Choose three.)
A. list.add(“foo”);
B. Object o = list;
C. String s = lig.get(0);
D. list = new ArrayList<String>();
E. list = new ArrayList<Object>();

B, C, D

Given:
```
1. public 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”;
24. TreeSet set = new TreeSet();
25. set.add(one);
26. 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.
Un Set no permite elementos duplicados y al sobreescribir el método compareTo siempre estamos devolviendo 0, por lo que la aplicación supone que todos los objetos de la clase Drink son iguales. B