Which of the following interfaces is at the top of collection framework hierarchy?

|  |  |
| --- | --- |
| Correct | java.util.Collection |
| **Feedback:**  All collection interfaces such as List, Set, Queue etc. inherit from the interface java.util.Collection. |
|  | java.util.List |
|  | java.util.Set |
|  | java.util.Queue |

***Score 1.00 of 1***

2. Which of these methods can be used to delete an element from an iterator object?

|  |  |
| --- | --- |
|  | boolean remove(); |
|  | void delete(); |
|  | boolean delete(); |
| Correct | void remove(); |
| **Feedback:**  The method "void remove()" removes from the underlying collection the last element that is returned by this iterator. |

***Score 1.00 of 1***

3. The class java.util.Collections provides a set of \_\_\_\_\_\_\_\_\_\_\_\_\_\_ methods to work with different types of collections.

|  |  |
| --- | --- |
|  | abstract |
|  | final |
| Correct | static |
| **Feedback:**  All the methods in the class java.util.Collections are static. |
|  | void |

***Score 1.00 of 1***

4. Which of these methods is used for getting an instance of a List interface?

|  |  |
| --- | --- |
|  | Collections.emptyCollection(); |
| Correct | Collections.emptyList(); |
| **Feedback:**  The method "Collections.emptyList()" returns an immutable and serializable empty list. |
|  | Collections.newList(); |
|  | Collections.listInstance(); |

***Score 1.00 of 1***

5. How to create an immutable list consisting of 'n' copies of the specified object?

|  |  |
| --- | --- |
| Correct | Collections.nCopies(n, obj); |
| **Feedback:**  The method "" returns an immutable list consisting of 'n' copies of the specified object. |
|  | Collections.copyOf(n, obj); |
|  | Collections.duplicate(n, obj); |
|  | Collections.clone(n, obj); |

***Score 1.00 of 1***

6. The class \_\_\_\_\_\_\_\_\_\_\_\_\_\_ was introduced in Java 1.0 to represent a collection of objects, and was retrofitted in Java 1.2 to implement the interface java.util.List.

|  |  |
| --- | --- |
|  | java.util.ArrayList |
|  | java.util.LinkedList |
| Correct | java.util.Vector |
| **Feedback:**  The collection framework was introduced in Java 1.2, and by then there was already a class called Vector with similar features. By making the class Vector extend AbstractList, it was re-written to follow the guidelines of a List. |
|  | java.util.AbstractList |

***Score 1.00 of 1***

7. Which of the following is the superclass for java.util.Hashtable?

|  |  |
| --- | --- |
| Correct | java.util.Dictionary |
| **Feedback:**  The class Hashtable was introduced in Java 1.0 and extends the class Dictionary to provide support for key-value storage. |
|  | java.util.AbstractMap |
|  | java.util.Properties |
|  | java.util.HashMap |

***Score 1.00 of 1***

8. What is the output of below code?

**int time = Integer.parseInt("4");  
 Queue queue = new LinkedList<>();  
 for (int i = time; i > 0; i--)  
 queue.offer(new Integer(i));  
 while (!queue.isEmpty()) {  
 System.out.print(queue.poll());  
 }**

|  |  |
| --- | --- |
| Correct | 4321 |
| **Feedback:**  Queue follows FIFO order. Offer method adds to the head of queue and pop method removes from the tail of the Queue |
|  | 1234 |
|  | Queue is not valid Java interface |
|  | Runtime Exception |

***Score 0.00 of 1***

9. What is the output of below code?

**List<Object> list = new ArrayList<>();  
    list.add(123);  
    list.add("John");  
    list.add(1,1.22f);  
    list.add(123);  
    list.remove(0);  
    System.out.println(list);**

|  |  |
| --- | --- |
| Wrong | [123, 1.22, John, 123] |
| Should have chosen | [1.22, John, 123] |
|  | Compilation Fails |
|  | [John, 1.22, 123] |
|  | Runtime Error |

***Score 1.00 of 1***

10. What is the output of below code?

**Stack stack = new Stack<>();  
 stack.push(new Integer(3));  
 stack.push(new Integer(2));  
 stack.pop();  
 stack.push(new Integer(5));  
 System.out.println(stack);**

|  |  |
| --- | --- |
|  | [3, 2, 5] |
| Correct | [3, 5] |
| **Feedback:**  The Stack Collection follows Last in First out order |
|  | [3, 5, 2] |
|  | [2, 5] |

***Score 1.00 of 1***

11. What is the output of below code?  
 public class Test {  
 public static void main(String argv[]) throws InterruptedException {

**TreeSet<Object> myMap = new TreeSet<>();  
        String s1 = new String("test");  
        String s2 = new String("test");  
        Sample s3 = new Sample("test");  
        Sample s4 = new Sample("test");  
        
        myMap.add(s1);  
        myMap.add(s2);  
        myMap.add(s3);  
        myMap.add(s4);  
        
        System.out.println(myMap);  
 }  
}**

**class Sample{  
    private String str;  
    
    Sample(String str ){  
        this.str = str;  
    }  
}**

|  |  |
| --- | --- |
|  | test test |
|  | test test test test |
| Correct | java.lang.ClassCastException |
| **Feedback:**  Object in TreeSet should be Comparable. Object of class Sample does not implement Comparable and cannot be compared to a String Object resulting in ClassCastException |
|  | None of the options |

***Score 1.00 of 1***

12. What is the output of below code?

**public class Test {  
 public static void main(String[] args) {  
  HashSet<Vehicle> s = new HashSet<>();  
  s.add(new Vehicle());  
  s.add(new Vehicle());  
  for (Vehicle v : s) {  
   System.out.println(v);  
  }  
 }  
}**

**class Vehicle {  
 int vehicleNo = 123;**

**Vehicle() {  
  vehicleNo = 456;  
 }**

**@Override  
 public String toString() {  
  return "Vehicle No " + vehicleNo;  
 }  
}**

|  |  |
| --- | --- |
|  | Vehicle No 123 Vehicle No 123 |
| Correct | Vehicle No 456 Vehicle No 456 |
| **Feedback:**  HashSet removes duplicates by running the hashCode() and equals() method for an object. Since Vehicle class doesn't override these methods, every Vehicle object is considered as a unique object |
|  | Vehicle No 123 |
|  | Vehicle No 456 |

***Score 0.00 of 1***

13. LinkedList is an implementation of …….

|  |  |
| --- | --- |
| Wrong | Queue |
| **Feedback:**  Partially Correct. Linked List is an Implementation of Queue, List and Collection interfaces. |
|  | List |
|  | Collection |
| Should have chosen | All of them |

***Score 0.00 of 1***

14. Which of the following data structure can be used to implement PriorityQueue?

|  |  |
| --- | --- |
| Should have chosen | array |
|  | map |
|  | set |
| Wrong | iterator |
| **Feedback:**  Incorrect. Iterator is used to iterate a collection. |

***Score 1.00 of 1***

15. Which of the following statement is false for Deque?

|  |  |
| --- | --- |
|  | Deque stands for "Double ended Queue". |
|  | It is not Thread-safe. |
| Correct | It can store limited number of elements. |
| **Feedback:**  False. Deque has no fixed capacity limit. |
|  | LinkedList is a concrete implementation of Deque. |

***Score 0.00 of 1***

16. Which collection is most suitable for LRU Cache?

|  |  |
| --- | --- |
|  | ArrayList |
| Should have chosen | LinkedHashMap |
| Wrong | TreeSet |
| **Feedback:**  Incorrect. |
|  | Stack |

***Score 1.00 of 1***

17. Which method of the interface java.util.Iterator returns the next element in the iteration?

|  |  |
| --- | --- |
|  | E nextElement() |
|  | E getNext() |
| Correct | E next(); |
| **Feedback:**  The method "E next()" returns the next element in the iteration. |
|  | E getCurrent() |

***Score 1.00 of 1***

18. Which of these methods from java.util.Iterator returns true if the iteration has more elements?

|  |  |
| --- | --- |
|  | boolean hasNextElement(); |
|  | boolean hasMore(); |
|  | boolean hasMoreElements(); |
| Correct | boolean hasNext(); |
| **Feedback:**  The method "boolean hasNext()" returns true if the iteration has more elements. |

***Score 1.00 of 1***

19. Which of these methods must be implemented by a comparator object?

|  |  |
| --- | --- |
| Correct | int compare(T o1, T o2); |
| **Feedback:**  The method "int compare(T o1, T o2)" compares its two arguments for order. It returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second. |
|  | int compareTo(T o1, T o2); |
|  | int doCompare(T o1, T o2); |
|  | int order(T o1, T o2); |

***Score 1.00 of 1***

20. Which of the following does not allow duplicate objects in a collection?

|  |  |
| --- | --- |
|  | java.util.Collection |
|  | java.util.List |
| Correct | java.util.Set |
| **Feedback:**  A Set represents a unique collection of objects and inherits from java.util.Collection. |
|  | java.util.Queue |

***Score 1.00 of 1***

21. Which method in the List interface can be used for checking the existence of an element?

|  |  |
| --- | --- |
|  | boolean exists(Object obj); |
| Correct | boolean contains(Object obj); |
| **Feedback:**  The method "boolean contains(Object obj)" returns true if this list contains the specified element. |
|  | boolean has(Object obj); |
|  | boolean available(Object obj); |

***Score 1.00 of 1***

22. How do you insert an element into a List at index n?

|  |  |
| --- | --- |
|  | list.insert(n, obj); |
| Correct | list.add(n, obj); |
| **Feedback:**  The method add(n, obj) of the List interface inserts the specified element at the specified position in this list. |
|  | list.set(n, obj); |
|  | list.append(n, obj); |

***Score 0.00 of 1***

23. If the insertion of an element is done always at index 0, which of the following is preferred?

|  |  |
| --- | --- |
|  | ArrayList |
|  | Vector |
| Should have chosen | LinkedList |
| Wrong | Stack |

***Score 0.00 of 1***

24. Which is the preferred implementation of java.util.List, if insertion and deletion of elements in random positions are frequently done?

|  |  |
| --- | --- |
|  | ArrayList |
| Wrong | Vector |
| Should have chosen | LinkedList |
|  | Stack |

***Score 1.00 of 1***

25. Which of the following classes stores the elements of a set in the natural ordering of an element?

|  |  |
| --- | --- |
|  | java.util.HashSet |
| Correct | java.util.TreeSet |
| **Feedback:**  The class TreeSet uses the element's Comparable property to order the elements in the Set. java.util.SortedSet. This is an interface that the TreeSet implements. |
|  | java.util.SortedSet |
|  | java.util.LinkedHashSet |

***Score 1.00 of 1***

26. The class java.util.HashSet depends on the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ method for ensuring the uniqueness of the set.

|  |  |
| --- | --- |
|  | compareTo() |
|  | rehash() |
| Correct | hashCode() |
| **Feedback:**  Using the hashCode() method, the HashSet (interanally HashMap) derives at the index of the bucket to store the element. |
|  | toString() |

***Score 1.00 of 1***

27. The elements of a set can be accessed sequentially by \_\_\_\_\_\_\_\_\_\_ .

|  |  |
| --- | --- |
|  | using the index of the element |
| Correct | using an Iterator |
| **Feedback:**  The set interface provides a method called iterator() to return an instance of Iterator, using which its elements can be accessed in a sequential way. |
|  | using Enumeration |
|  | calling the values() method |

***Score 1.00 of 1***

28. Which of the following classes is the default choice for a Set's implementation?

|  |  |
| --- | --- |
| Correct | java.util.HashSet |
| **Feedback:**  HashSet is the simplest implementation amongst all. It does not have any additional stipulations like the others. |
|  | java.util.TreeSet |
|  | java.util.SortedSet |
|  | java.util.LinkedHashSet |

***Score 1.00 of 1***

29. Which of the methods is used for storing an object in a Map?

|  |  |
| --- | --- |
| Correct | V put(K key, V value); |
| **Feedback:**  The method "put" associates the specified value with the specified key in this map. |
|  | V add(K key, V value); |
|  | V store(K key, V value); |
|  | V set(K key, V value); |

***Score 1.00 of 1***

30. Which of these methods is used for retrieving an object from the Map using its key?

|  |  |
| --- | --- |
| Correct | V get(Object key); |
| **Feedback:**  The method "get" returns the value to which the specified key is mapped. Otherwise it returns null if this map contains no mapping for the key. |
|  | V retrieve(Object key); |
|  | V getObject(Object key); |
|  | V objectAt(Object key); |

***Score 1.00 of 1***

31. Which of these methods can be called on a Map object to get a collection of all its values?

|  |  |
| --- | --- |
|  | Collection members(); |
| Correct | Collection values(); |
| **Feedback:**  The method "Collection values()" returns a Collection view of the values contained in this map. |
|  | Collection getValues(); |
|  | Collection valueSet(); |

***Score 1.00 of 1***

32. Which of these methods is used for getting the collection of its keys?

|  |  |
| --- | --- |
|  | Set getKeys(); |
| Correct | Set keySet(); |
|  | Collection keysCollection(); |
|  | List keys(); |

***Score 0.00 of 1***

33. The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ interface imposes a natural ordering on the objects of each class that implements it.

|  |  |
| --- | --- |
| Should have chosen | java.lang.Comparable |
| Wrong | java.util.Comparable |
|  | java.lang.Comparator |
|  | java.util.Comparator |

***Score 1.00 of 1***

34. Which of these methods is implemented by a comparable object?

|  |  |
| --- | --- |
|  | public boolean equals(T o); |
|  | public int compare(T o); |
| Correct | public int compareTo(T o); |
| **Feedback:**  The method "public int compareTo(T o)" compares this object with the specified object for order. It returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. |
|  | public int compare(T o1, To2); |

***Score 0.00 of 1***

35. Which method of NavigableMap returns the least key greater than or equal to the specified key ?

|  |  |
| --- | --- |
|  | higherKey() |
| Wrong | lowerKey() |
| **Feedback:**  Incorrect. It returns the greatest key less than the given key. |
| Should have chosen | ceilingKey() |
|  | floorKey() |