1. A Java exception is an instance of _____.

a. RuntimeException b. Exception c. Error d. Throwable

2. An instance of _____ describes programming errors, such as bad casting, accessing an out-ofbounds array, and numeric errors.

a. RuntimeException b. Exception c. Error d. Throwable

3. What exception type does the following program throw?

```
public class Test {   public static void
main(String[] args) {
System.out.println(1 / 0);
  }
}
```

a. ArithmeticException b. ArrayIndexOutOfBoundsException

c. StringIndexOutOfBoundsException d. ClassCastException

4. A method must declare to throw _____.

a. unchecked exceptions b. checked exceptions c. Error d. RuntimeException

5. Which one(s) of the following statements are true?

a. You use the keyword throws to declare exceptions in the method heading.

b. A method may declare to throw multiple exceptions.

c. To throw an exception, use the key word throw.

d. If a checked exception occurs in a method, it must be either caught or declared to be thrown from the method.

6. ArrayList<String> and ArrayList<Integer> are two types. Does the JVM load two classes ArrayList<String> and ArrayList<Integer>?

a. Yes b. No

7. Which of the following is not an advantage of Java exception handling?

a. Java separates exception handling from normal processing tasks.

b. Exception handling improves performance.

c. Exception handling makes it possible for the caller's caller to handle the exception.

d. Exception handling simplifies programming because the error-reporting and error-handling code can be placed at the catch block.

8. Which one(s) of the following statements is correct?

a. Generics can help detect type errors at compile time, thus make programs more robust.

b. Generics can make programs easy to read.

c. Generics can avoid cumbersome castings.

d. Generics can make programs run faster.

9. All the concrete classes in the Java Collections Framework implement _____.

a. the Cloneable interface b. the Serializable interfaces

c. the Comparable interface d. the Comparator interface

10. For an instance of Collection, you can obtain its iterator using _____.

a. c.getIterator() b. c.iterator() c. c.iterators() d. c.iterable()

11. You can use a for-each loop to traverse all elements in a container object that implements _____.

a. Iterator b. Collection c. Iterable d. ArrayList

12. Which one(s) of the following are true?

a. You can insert an element anywhere is an arraylist.

b. You can insert an element anywhere is a linked list.

c. You can use a linked list to improve efficiency for adding/removing elements at the beginning of a list.

d. You should use an array list if your application does not require adding and removing elements at the beginning of a list.

13. Suppose ArrayList x contains three strings [Beijing, Singapore, Tokyo]. Which one(s) of the following methods will cause runtime errors?

a. x.get(2) b. x.set(3, "New York") c. x.get(3) d. x.remove(3)

14. Suppose list list1 is [1, 2, 5] and list list2 is [2, 3, 6]. After list1.addAll(list2), list1 is _____.

a. [1, 2, 5, 2, 3, 6] b. [1, 2, 5, 3, 6] c. [1, 5] d. [2]

15. Suppose a list contains {"red", "green", "red", "green"}. What is the list after the following code?

```
list.remove("red");
```

a. {"red", "green", "red", "green"} b. {"green", "red", "green"}

c. {"green", "green"} d. {"red", "green", "green"}

16. Which of the following is correct to sort the elements in a list lst?

a. lst.sort() b. Collections.sort(lst)

c. Arrays.sort(lst) d. new LinkedList<String>(new String[]{"red", "green", "blue"})

17. Which data type should you use if you want to store duplicate elements and be able to insert or delete elements anywhere efficiently.

a. ArrayList b. LinkedList c. Vector d. Set

18. java.util.Vector is a subtype of _____.

a. java.util.ArrayList b. java.util.LinkedList

c. java.util.AbstractList d. java.util.Vector

19. The _____ method in the Queue interface retrieves and removes the head of this queue, or null if this queue is empty.

a. poll() b. remove() c. peek() d. element()

20. What is the printout of the following code?

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

```
    list.add(0);
list.add(1);
list.add(2);     list.add(1,
4);     list.set(2, 30);
    System.out.println(list);
```

a. [0, 1, 2, 4, 30] b. [0, 4, 2, 30] c. [0, 1, 30, 2] d. [0, 1, 2, 30] e. [4, 1, 2, 30]