

1. ----- is the one that calls itself. And.....is the one that never stops.

- A. A recursive method, An infinite recursion**
- B. An infinite recursion, A recursive method
- C. An infinite recursion, An infinite recursion
- D. None of the above.

2. Show the output of the following program:

```
public class Test {  
    public static void main(String[] args) {  
        xMethod(5);  
    }  
}
```

```
public static void xMethod(int n) {  
    if (n > 0) {  
        System.out.print(n + " ");  
        xMethod(n - 1);  
    }  
}
```

- A. The output is 1 2 3 4 5
- B. The output is 5 4 3 2 1**
- C. The output is 1 3 5 2 4
- D. The output is 1 4 3 2 5

3. Will the program work if the directory is empty (i.e., it does not contain any files)?

- A. Yes.**
- B. No

Q. a ----- file consists of a sequence of characters and a --- file consists of a sequence of bits.

- A. Binary, Text
- B. Document, Text
- C. Text, Binary**
- D. Document, Binary

4. Can you view a binary file using a text editor?

- A. Yes
- B. No**

5. A Java I/O object is called a ----. An object for reading data is called an ----- and an object for writing data is called an

- A. input steam, output stream, Stream
- B. input steam, Stream, output stream
- C. Stream, output stream, input steam
- D. Stream, input steam, output stream**

6. 11. How do you close streams?

- A. Invoking the close() method.
- B. Using the try-catch-resource to automatically close the stream.
- C. None of Above
- D. Both A and B**

7. What will happen if you attempt to create an input stream on a nonexistent file?

What will happen if you attempt to create an output stream on an existing file?

Can you append data to an existing file?

- A. No error will occur.
- B. A FileNotFoundException would occur if you attempt to create an input stream for a nonexistent file.**
- C. Compile Fails
- D. Runtime Exception would occur

8. Which of the following statements are true?

- i. Any recursive method can be converted into a nonrecursive method.
- ii. Recursive methods take more time and memory to execute than nonrecursive methods.
- iii. Recursive methods are always simpler than nonrecursive methods.
- iv. There is always a selection statement in a recursive method to check whether a base case is reached.

A. i,ii,iii,iv

B. i,ii,iii

C. i,ii,iv

D. i,iv

9. Are there any compile errors in the following code.

(a) Prior to JDK 1.5

```
ArrayList dates = new ArrayList();  
dates.add(new Date());  
dates.add(new String());
```

A. Yes B. No

1. Are there any compile errors in the following code.

10. `ArrayList<Date> dates = new ArrayList<>();`

```
    dates.add(new Date());
```

```
    dates.add(new String());
```

A. Yes

B. B. No

11.What is wrong in the following code?

```
ArrayList dates = new ArrayList();  
dates.add(new Date());  
Date date = dates.get(0);
```

A. No wrong.

B. B. Casting is needed C. No Casting is needed D. None of the Above.

12. What is wrong in the following code?

```
ArrayList<Date> dates = new ArrayList<>();  
dates.add(new Date());  
Date date = dates.get(0);
```

- A. No wrong.
- B. B. Casting is needed
- C. C. **No Casting is needed**
- D. D. None of the Above.

13. What are the benefits of using generic types?

- A. One important benefit is improving reliability and robustness.
- B. Potential errors can be detected by the compiler.
- C. Both A and B
- D. None of the above.

14. What is the generic definition for java.lang.Comparable in the Java API?

A. package java.lang;

```
public interface Comparable<E> {  
    public int compareTo(E o) { }  
}
```

B. package java.util;

```
public interface Comparable<E> {  
    public String compareTo(E o) { }  
}
```

15. Since you create an instance of ArrayList of strings using new ArrayList<String>(), should the constructor in the ArrayList class be defined as

```
public ArrayList<E>()
```

- A. No.
- B. B. Yes

16. Can a generic class have multiple generic parameters?

A. Yes. B. No.

17. Given int[] list = {1, 2, -1}, can you invoke sort(list) using the sort method in Listing 19.4?

- A. No, because list is of type int[], but the sort method requires E[], where E is an object type.
- B. Yes,

18. Given `int[] list = {new Integer(1), new Integer(2), new Integer(-1)}`, can you invoke `sort(list)` using the `sort` method in Listing 19.4?

- A. No, because `list` is still of type `int[]`, but the `sort` method requires `E[]`, where `E` is an object type.**
- B. Yes.**

19. What is a raw type?

- A. When you use generic type without specifying an actual parameter, it is called a raw type.**
- B. When you use generic type specifying an actual parameter, it is called a raw type.**
- C. None the above.**
- D. Both A and B**

20. Why is a raw type unsafe?

- A. A raw type is unsafe, because some errors cannot be detected by the compiler.**
- B. A raw type is not unsafe.**
- C. None of the above**
- D. Both A and B**

21. Why is the raw type allowed in Java?

- A. The raw type is not allowed in Java for backward compatibility.**
- B. The raw type is allowed in Java for backward compatibility.**
- C. The raw type is allowed in Java for removing compile error.**

22. What is the syntax to declare an `ArrayList` reference variable using the raw type and assign a raw type `ArrayList` object to it?

- A. `ArrayList list = new ArrayList();`**
- B. `HashMap list=new List();`**
- C. `List list=new List();`**
- D. `Set list=new HashSet();`**

23. Is `GenericStack` the same as `GenericStack<Object>`?

- A. No, `GenericStack` is roughly equivalent to `GenericStack<Object>`, but they are not the same. `GenericStack<Object>` is a generic instantiation, but `GenericStack` is a raw type.**
- B. Yes, it is same**
- C. None of the above.**

24. What are an unbounded wildcard, a bounded wildcard, and a lower-bound wildcard?

- A. `?` is unbounded wildcard, `? super T` is lower bounded wildcard, `? extends T` is bounded wildcard**
- B. `?` is unbounded wildcard, `? extends T` is bounded wildcard, `? super T` is lower bounded wildcard**

- C. ? extends T is bounded wildcard, ? super T is lower bounded wildcard, ? is unbounded wildcard ,
- D.

25.If your program uses ArrayList<String> and ArrayList<Date> , does the JVM load both of them?

- A. No. Only ArrayList is loaded.**
- B. Yes, Both will load

26.Can you create an instance using new E() for a generic type E? Why?

- A. No, because the type information is not available at runtime.**
- B. Yes

26. Why are generics used?

- a) Generics make code more fast
- b) Generics make code more optimised and readable
- c) Generics add stability to your code by making more of your bugs detectable at compile time**
- d) Generics add stability to your code by making more of your bugs detectable at run time

View Answer

27. Which of these type parameters is used for a generic class to return and accept any type of object?

- a) K
- b) N
- c) T**
- d) V

28. Which of these type parameters is used for a generic class to return and accept a number?

- a) K
- b) N**
- c) T
- d) V

Answer: b

29. Which of these is an correct way of defining generic class?

- a) class name(T1, T2, ..., Tn) { /* ... */ }
- b) class name { /* ... */ }**

- c) class name[T1, T2, ..., Tn] { /* ... */ }
- d) class name{ T1, T2, ..., Tn } { /* ... */ }

30. Which of the following is incorrect statement regarding the use of generics and parameterized types in Java?

- a) Generics provide type safety by shifting more type checking responsibilities to the compiler
- b) Generics and parameterized types eliminate the need for down casts when using Java Collections
- c) When designing your own collections class (say, a linked list), generics and parameterized types allow you to achieve type safety with just a single class definition as opposed to defining multiple classes**
- d) All of the mentioned

31. Which of the following reference types cannot be generic?

- a) Anonymous inner class**
- b) Interface
- c) Inner class
- d) All of the mentioned

32. What is the time complexity for an insertion sort?

- A. The time complexity for an insertion sort is $O(n^2)$.**
- B. The time complexity for an insertion sort is $O(n/2)$.
- C. The time complexity for an insertion sort is $O(n)$.
- D. The time complexity for an insertion sort is $O(n^2)$.

33. If a list is already sorted, how many comparisons will the insertionSort method perform?

- A. $n - 1$ times.**
- B. n times
- C. n^2 times
- D. 0 times

34. What is the time complexity for a bubble sort?

- A. The time complexity for a bubble sort is $O(n/2)$.
- B. The time complexity for a bubble sort is $O(n^2)$.**
- C. The time complexity for a bubble sort is $O(n)$.
- D. The time complexity for a bubble sort is $O(n-1)$.

35. If a list is already sorted, how many comparisons will the bubbleSort method perform?

- A. $n - 1$ times.**

- B. n times.
- C. n^2 times.
- D. $n/2$ times.

36. What is the time complexity for a merge sort?

- A. The time complexity for a merge sort is $O(n-1)$.
- A. The time complexity for a merge sort is $O(n/2)$.
- A. The time complexity for a merge sort is $O(n^2)$.
- A. The time complexity for a merge sort is $O(n \log n)$.**

37. What is the time complexity for a quick sort?

- A. The time complexity for a quick sort is $O(n^2)$ in the worst case and $O(n \log n)$ in the average case.**
- B. The time complexity for a quick sort is $O(n^2)$ in the worst case and $O(n \log n)$ in the average case.
- C. Both A and B
- D. None of the above.

38. Why is quick sort more space efficient than merge sort?

- A. Quick sort needs to create temporary arrays, while merge sort does not need temporary arrays.**
- B. Quick sort does not need to create temporary arrays, while merge sort needs temporary arrays.**
- C. Quick sort needs to create temporary arrays, while merge sort needs temporary arrays.
- D. None of the above

39. Which of the following statements are wrong?

```

1 Heap<Object> heap1 = new Heap<>();
2 Heap<Number> heap2 = new Heap<>();
3 Heap<BigInteger> heap3 = new Heap<>();
4 Heap<Calendar> heap4 = new Heap<>();
5 Heap<> heap5 = new Heap<>();

```

- A. Lines 2 and 3 are wrong
- B. Lines 1 and 3 are wrong
- C. Lines 2 and 5 are wrong
- D. Lines 1 and 2 are wrong**

40. What is the return value from invoking the remove method if the heap is empty?

- A. The return value will not be null.
- B. The return value will be null.**

41. What is the time complexity of inserting a new element into a heap and what is the time complexity of deleting an element from a heap?

- A. $O(\log n)$ for only insertion.
- B. $O(\log n)$ for only deletion.
- C. $O(\log n)$ for none of insertion and deletion.
- D. **$O(\log n)$ for both insertion and deletion.**

42. Can you sort a list of strings using a bucket sort?

- A. Yes, Bucket sort is suitable for sorting strings.
- B. **No, Bucket sort is not suitable for sorting strings.**

43. Suppose list is an instance of MyList, can you get an iterator for list using list.iterator()?

- A. **Yes. Because MyList is Iterable.**
- B. No. Because MyList is not Iterable.

44. Can you create a list using new MyList()?

- A. **No. Because MyList is an interface.**
- A. Yes

45. What are the limitations of the array data type?

- A. **An array is a fixed-size data structure. Once an array is created, its size cannot be changed.**
- B. An array is not a fixed-size data structure.
- C. Both A and B are true.
- D. None of the above.

46. If a linked list does not contain any nodes, what are the values in head and tail?

- A. head and tail are not null.
- B. tail is null.
- C. head is null.
- D. **head and tail are null.**

47. If a linked list has only one node, is head == tail true? List all cases in which head == tail is true.

- A. **Yes. When the list is empty, head == tail is also true.**
- A. No. When the list is empty, head == tail is also false

48. What is the time complexity of the `addFirst(e)` and `removeFirst()` methods in `MyLinkedList`?

- A. **$O(1)$**
- B. $O(2)$
- C. $O(0)$
- D. $O(n)$

49. What would be the time complexity for the `size()` method if the `size` data field is not used in `MyLinkedList`?

- A. $O(n/2)$
- B. $O(n^2)$
- C. $O(n-1)$
- D. **$O(n)$**

50. What is a priority queue?

- A. In a priority queue, elements are not assigned with priorities. When accessing elements, the element with the highest priority is removed first.
- B. In a priority queue, elements are assigned with priorities. When accessing elements, the element with the highest priority is not removed first.
- C. **In a priority queue, elements are assigned with priorities. When accessing elements, the element with the highest priority is removed first.**
- D. None of the above

51. What method is defined in the `java.lang.Iterable<E>` interface?
- A. The `iterator()` method is defined in the `java.lang.List` interface.
 - B. The `iterator()` method is defined in the `java.lang.Set` interface.
 - C. The `iterator()` method is defined in the `java.lang.HashSet` interface.
 - D. **The `iterator()` method is defined in the `java.lang.Iterable` interface.**
52. What is the benefit of being a subtype of `Iterable<E>`?
- A. Being a subtype of `Iterable`, the elements of the container cannot be traversed using a for-each loop.
 - B. **Being a subtype of `Iterable`, the elements of the container can be traversed using a for-each loop.**
 - C. None of the above.
53. Every internal node in a Huffman tree has two children. Is it true?
- A. **Yes.**
 - B. No
54. If the `Heap` class in line 50 in Listing 25.9 is replaced by `java.util.PriorityQueue`, will the program still work?
- A. **Yes, except that you also have to change `heap.getSize()` to `heap.size()`.**
 - B. No
55. What is an AVL tree? Describe the following terms: balance factor, left-heavy, and right-heavy.
- A. **AVL trees are well-balanced. In an AVL tree, the difference between the heights of two subtrees for every node is 0 or 1.**
 - B. AVL trees are not well-balanced. In an AVL tree, the difference between the heights of two subtrees for every node is 0 or 1.
 - C. AVL trees are well-balanced. In an AVL tree, the difference between the heights of two subtrees for every node is more than 1.

- D. AVL trees are well-balanced. In an AVL tree, the difference between the heights of two subtrees for every node is 0 or 1.
56. True or false: AVLTreeNode is a subclass of TreeNode?
- A. **True**
 - B. False
57. True or false: AVLTree is a subclass of BST.
- A. False
 - B. **True**
58. What are data fields in the AVLTree class?
- A. All data fields defined in the BST class are inherited in the AVLTree class.
 - B. The AVLTree class does not define new data fields.
 - C. **Both A and B**
 - D. None Of the above.
59. In the insert and delete methods, once you have performed a rotation to balance a node in the tree, is it possible that there are still unbalanced nodes?
- A. **No.**
 - B. Yes
60. Can you traverse the elements in an AVL tree using a foreach loop?
- A. **Yes.**
 - B. No
61. If N is a value of the power of 2, is $N / 2$ same as $N \gg 1$?
- A. **Yes**
 - B. No
62. If N is a value of the power of 2, is $m \% N$ same as $m \& (N - 1)$ for any integer m?
- A. **Yes. Open addressing is to find an open location in the hash table in**
 - B. No, Open addressing is to find an open location in the hash table in

61. Which of the following class that you enable to create and control thread?

- a) java.io.thread
- b) **java.lang.thread**
- c) java.util.*
- d) java.lang.system

62. How many main parts of thread or execution context?

- a) 4 b) 5
- c) **3** d) 2

63. Which of the following main parts of thread?

- a) A virtual CPU
- b) the data on which the code works
- c) the code that the CPU execute
- d) **above all**

64. Two thread shared the same data when they share access to a common_____.

- a) class b) method
- c) **object** d) interface

65. A thread constructor takes an argument that is an instance of_____.

- a) Running b) New
- c) Dead d) **Runnable**

66. To create a newly thread you must call which method.

- a) close() b) **start ()**
- c) sleep() d) wait ()

67. The model of preemptive scheduler is that many threads might be runnable but how many thread is running?

- a) two b) three
- c) **one** d) four

68. When a thread complete execution and terminates, it can't run again?

- a) **True**
- b) False

69. Which method is to used to determine if a thread is still visible?

- a) alive b) **isAlive**
c) runnable d) dead

70. The sleep method is one way to ____ a thread for a period of time.

- a) moving b) **halt**
c) running d) none

71. Join methods also depends on

- a) operating system timers
b) schedulers
c) **a+b** d) none

72. Join also responds to an interrupt an exit with an

- a) IOException b) ArithmeticException
c) NullPointerException
d) **InterruptedException**

73. Which method we use to give other runnable threads a chance to execute?

- a) **Thread.yield()** b) Thread.wait()
c) Thread.sleep() d) none

74. A mechanism that enables a programmer to control thread that are sharing data is called

- a) thread b) **synchronize**
c) wait d) deadlock

75. Which of the following serial of lifecycle method of a thread?

- a) Runnable—New—Dead—Running--Nonrunnable
b) **New—Runnable—Running—Nonrunnable—Dead**
c) Running—Dead—Nonrunnable—New--Runnable
d) New—Running—Runnable—Nonrunnable—Dead

76. If two Thread instance of same class the can share same code when they execute.

- a. **True**
b. False

77. An instance of Runnable is made from a _____

- a. Thread Object.
- b. Thread method.
- c. Object.
- d. Class.**

78. Multithreaded programming environment enables you to create multiple threads based on the _____

- a. Different Runnable instance.
- b. Same Runnable instance.**
- c. Two Runnable instance.
- d. Three Runnable instance.

79. Which method runs newly created Thread automatically?

- a. begin();
- b. stop();
- c. trim();
- d. start();**

80. Preemptive and time-sliced are similar?

- a. True
- b. False**

81. How many different states Thread object has in its lifetime?

- a. Two
- b. Three
- c. Four
- d. Five**

82. By which method can we pause Thread for a period of time?

- a. Thread.sleep();**
- b. Thread.start();
- c. Thread.start-sleep();
- d. Thread.sleepthread();

83. How many Thread Priorities are there in Java?

- a. One
- b. Two

- c. **Three**
- d. Four

84. What is the default priority in java Thread ?

- a. Thread.MIN_PRIORITY
- b. **Thread.NORM_PRIORITY**
- c. Thread.MAX_PRIORITY

85. What does Thread.yield() method do ?

- a. stop Thread
- b. start Thread
- c. **gives other runnable thread a chance to execute.**
- d. gives same runnable thread a chance to execute.

86. Which class is enabled to create and control threads?

- a. Java.swing.thread
- b. Java.awt.thread
- c. **Java.lang.thread**
- d. javax.swing.thread

87. Which one is true?

- a. **2 threads can share the same data when they share access to a common object**
- b. 2 threads can share the same data when they share access to a different object
- c. 2 threads can share the same data when they execute code from instance of the different class

88. Which one is true?

- a. A newly created thread can be run automatically
- b. **A newly created thread cannot be run automatically**
- c. A newly created thread may be run automatically

89. Generally In java technology threads are _____?

- a. Primitive
- b. Boolean
- c. **Preemptive**
- d. Characteristics

90. The word preemptive means ---

- a. Previously it was empty
- b. Not primitive
- c. Time-slicing**
- d. None of these

91. Which method is used to pausing a thread for some time?

- a. Thread.pause ()
- b. Thread.stop ()
- c. Thread.sleep ()**

92. Is it possible to make some actions at a time on a machine with one CPU by using thread?

- a. Yes
- b. No**

93. The sleep is a _____ method in the thread class.

- a. Dynamic
- b. Static**
- c. Different
- d. None of these

94. The word in thread “isAlive” means the thread is still _____?

- a. Running
- b. Alive**
- c. Not destroy
- d. Viable

95. The term “isAlive” means is details _____?

- a. The thread has been started and its task has been finished
- b. The thread has been started but its task has not been completed**
- c. The thread has been started and already completed its job
- d. The thread has been started and still it continues

96. In thread class “getPriority” method is a _____ type value.

- a. Floating
- b. Double
- c. Int**
- d. Point

97. In thread Priority method default priority is -----

- a. DEF_PRIORITY
- b. SET_DEF_PRIORITY
- c. NORM_PRIORITY**
- d. MIN_PRIORITY
- e. MAX_PRIORITY

98. Which methods are responds to an interrupted method?

- a. Sleep
- b. InterruptedException
- c. Join
- d. None of the above
- e. A & C**
- f. B & C
- g. A & B

99. Why we use thread.yield () method---

- a. To stop other runnable threads
- b. To give other runnable threads a chance to execute**
- c. To pause other runnable threads and a chance to restart
- d. All are false

100. Which keyword we used to stop corrupting data when more than single thread is running ---

- a. Sleep
- b. Break
- c. Synchronized**
- d. Nothing of these

101. In java technology is there any “flag” option when creating object?

- a. Yes**
- b. No

102. How many methods provide the “java.lang.Object” class?

- a. 2**
- b. 3
- c. 4
- d. 1

103. Which are the methods of “java.lang.Object” class?

- a. Wait
- b. Notify

- c. Break
- d. A & C
- e. A & B**
- f. B & C

104. Why is multithreading needed?

- A. Multithreading can make your program more responsive and interactive, and enhance the performance.
- B. Multithreading is needed in many situations, such as animation and client/server computing.
- C. Both A and B**
- D. None of the above.

105. What is a runnable object?

- A. An instance of Runnable is a runnable object.**
- B. An instance of Running state is a runnable object.
- C. An instance of sleep state is a runnable object.

106. What is a thread?

- A. A thread is wrapper object for a runnable object for executing a runnable task.**
- B. A thread is a runnable task.
- C. A thread is an Interface
- D. None of the above.

107. If a loop contains a method that throws an InterruptedException, why should the loop be placed inside a try-catch block?

- A. It will produce compile error
- B. If the loop is outside the try-catch block, the thread may continue to execute even though it is being interrupted.**
- C. None of the above.

108. How do you set a priority for a thread?

- A. You use the putPriority() method to set the priority for a thread
- B. You use the addPriority() method to set the priority for a thread
- C. You use the setPriority() method to set the priority for a thread**
- D. You use the getPriority() method to set the priority for a thread

109. What is the default priority?

- A. The default priority of the thread is Thread.NORM_PRIORITY (1).
- B. The default priority of the thread is Thread.NORM_PRIORITY (5).**
- C. The default priority of the thread is Thread.NORM_PRIORITY (10).
- D. The default priority of the thread is Thread.NORM_PRIORITY (0).

110. Is an instance of FlashText a runnable object?

- A. Yes. Because it implements the Runnable interface.
- B. No. Because it does not implement the Runnable interface.**

111. What is the purpose of using Platform.runLater?

- A. Invoking Platform.runLater(Runnable r) tells the system to run a task in the JavaFX application thread.**
- B. Invoking Platform.runLater(Runnable r) does not tell the system to run a task in the JavaFX application thread.

112. Can the wait(), notify(), and notifyAll() be invoked from any object?

- A. Yes.**
- B. No

113. Can the read and write methods in the Buffer class be executed concurrently?

- A. Yes.
- B. No**

114. When invoking the read method, what happens if the queue is empty?

- A. It will wait for a signal for the queue to be not empty.**
- B. It will sleep.
- C. It will be dead.

