|  |  |  |
| --- | --- | --- |
| **APCS Exposure Java** | **Exercises 18.01-05** | **Date:** |
| **Name:** | | **Period:** |

1. Is the *essence* of an algorithm different depending on what language it is written in?

2. What is an *algorithm*?

3. What is a synonym for *arrays*?

4. Look at program ***Java1801.java***. Does the *List* class have a default constructor?

5. Look again at program ***Java1801.java***.

What values are put into the array by the single parameter constructor?

6. Look yet again at program ***Java1801.java***.

In the double parameter constructor, what do the *s* and *n* signify?

7. Look at program ***Java1802.java***.

In the new triple parameter constructor, what do the 3 parameters signify?

8. Look at program ***Java1803.java***. What method is added to the *List* class in this program?

9. Refer to the previous question. What is the purpose of this method?

10. Look at program ***Java1804.java***. How was the *List* class improved in this program?

11. What is the simplest of all searches?

12. Refer to the previous question. Explain how this search works.

13. What is a synonym for *linear*?

14. Look at program ***Java1805.java***. What is inefficient about the search used in this program?

15. The *linear search* is made more efficient in program ***Java1806.java***. How was this done?

16. Look again at program ***Java1806.java***.

What is still wrong or *impractical* about the search in this program?

17. Look at program ***Java1807.java***.

Instead of returning a **boolean** this search method returns an **int**. What does this **int** signify?

18. Refer to the previous question.

What happens if the item you are looking for is not in the list?

19. Why is sorting important?

20. What is considered one of the most inefficient sorts?

21. Refer to the previous question. Why then are we looking at this sort?

22. Where does the *Bubble Sort* get its name?

Questions 23 - 26 all refer to the Bubble Sort.

23. What elements are compared?

24. Refer to the previous question. What happens if these 2 elements are in the wrong order?

25. After 1 comparison pass is completed, what has been accomplished?

26. If there are *n* elements in the array, how many comparison passes will be necessary?

27. Look at program ***Java1808.java***. This program does not contain a complete Bubble Sort.

What is accomplished by this *partial sort*?

28. Look at program ***Java1809.java***. How would you make this program sort in *decending* order instead of *ascending* order. *(Hint: It involves changing just 1 character!)*

29. Look again at program ***Java1809.java***. Aside from the fact that is the *Bubble Sort*, what is particularly inefficient about the *bubbleSort* method used in this program?

30. Why would a *swap* method be declared **private**?

31. Look closely at the *bubbleSort* methods in both programs ***Java1809.java*** and ***Java1810.java***.

Aside from the *swap* method, one other line was changed which makes the sort more efficient.

What is that change?

32. Look at program ***Java1811.java***. This program used a **do…while** loop for the outer loop in the sort instead of a **for** loop. Why is that?

33. Look again at program ***Java1811.java***.

How does the *Smart Bubble Sort* know when the list is *sorted*?