Collections

1. **What is the Collection framework?**
   Collection framework represents an architecture for storing and manipulating a group of objects. Collection framework has interfaces, classes and also few algorithms.

2. **What is the root interface in Collection hierarchy?**
   Iterable interface is the root interface in collection hierarchy.

3. **What is an Iterable interface?**
   Iterable interface facilitates to iterate the elements. It iterates in forward direction only. It provides a generic way for traversal through the elements of a Collections and implements Iterator design pattern.

4. **Which design pattern is followed by Iterable interface?**
   Iterable interface follows iterator design pattern.

5. **What was the java's Collections framework equivalent in C++ called as?**
   Java's Collections framework equivalent in C++ is called as Standard Template Library.

6. **What was the java's Collections framework equivalent in smalltalk called as?**
   Java's collection framework equivalent in Smalltalk is called as Small Talk Collection hierarchy.

7. **What are the benefits of java's Collection framework?**
   Advantages of Collections framework are as follows:
   i) Reduces the effort in programming.
   ii) Increases the speed of programming and quality.
   iii) Allows inter-operability.

iv) Reduces effort to design new software.

8. **When was Collections framework introduced in java?**
   Collections framework was introduced into Java from JDK 1.2 onwards.

9. **Which were the adhoc classes that were provided in java for dealing with collection of objects before java's Collection framework was introduced?**
   Prior to Collections framework was introduced in Java, James Gosling had provided four classes in order to perform storage and retrieval of the objects namely,
   i) Vector
   ii) Stack
   iii) HashTable and
   iv) Properties

10. **What was the problem with these adhoc classes?**
    Although these legacy classes were useful, they lacked a central unifying theme. This created a lot of confusion for the programming community because the way Vector class used was different from the way Stack class or HashTable class was used. This adhocness lead to the emergence of the Collections hierarchy.

11. **What is a Dictionary?**
    Dictionary is a legacy abstract class. It contains <key-value> pair. It is much like today's Map class.

12. **What is a Hashtable?**
    HashTable is a legacy class. HashTable is the concrete implementation of Dictionary.

13. **What is a Properties class?**
    Properties class is a legacy class. It is a subclass of HashTable class. It

is used to maintain <key-value> pair such that both key and value are String.

Eg. <"Dhoni", "Mahi">

14. **When does a ClassCastException occur?**

Whenever in a program we attempt to typecast incompatible types then ClassCastException will occur.

15. **When does an UnsupportedOperationException occur?**

Whenever we attempt to modify an un modifiable Collection, then UnsupportedOperationException will occur.

16. **Name the three static variables present in Collection framework?**

1) EMPTY_SET
2) EMPTY_LIST
3) EMPTY_MAP

17.Differentiate between Array and ArrayList?

Arrays	ArrayList
Can store only primitive data.	Can store only Objects
Arrays are fixed in size. They cannot grow or shrink during program execution.	ArrayList can grow or shrink in size during program execution
Can store only homogeneous data.	Can store heterogeneous data.
Creation takes less time	Creation would take more time because auto boxing has to take place using wrapper classes
Auto boxing is not required.	Auto boxing is required.

18. **When will we use Array over ArrayList?**

1) When the size of the array is either fixed or known earlier.
2) When we have only primitive data have to be stored.
3) When we have multi-dimensional data have to be stored.

19. **What are the similarities between ArrayList and Vector?**

1) ArrayList and Vector maintains the order of insertion.
2) ArrayList and Vector are both fail-fast.
3) ArrayList and Vector allow null values, both provide index based accessing.

20. **Differentiate between ArrayList and Vector?**

ArrayList and Vector are almost similar. However, Vector class is synchronized whereas ArrayList if not synchronized.

21.Differentiate between ArrayList and LinkedList?

ArrayList	LinkedList
Internally makes use of dynamic array	Internally makes uses of linked list
Suitable for performing rear end insertion	Suitable for performing insertion at all given position
Cannot utilize dispersed memory location	Can utilize dispersed memory location

22.Differentiate between Iterator and ListIterator?

Iterator	ListIterator
Can be used to access objects present in all the seven Collections classes	Can be used to access objects present in only list based classes

Iterator can iterate upon Collections only in the forward direction	ListIterator can iterate upon Collections both in forward and reverse direction

23.Differentiate between Iterator and Enumeration?

Iterator	Enumeration
Iterator can traverse legacy and non-legacy elements	Enumeration can traverse only legacy elements
Iterator is fail-fast	Enumeration is not fail-safe
Iterator is slower than Enumeration	Enumeration is faster than Iterator

24.Differentiate between List and Set?

In List index based access is provided and also duplicates are allowed. Where as in set, index based access is not permitted and duplicates are not allowed.

25.Differentiate between HashSet and TreeSet?

HashSet	TreeSet
Internally makes use of Hashing algorithm	Internally makes use of balanced binary search tree using red-black algorithm
Objects would not get displayed in the sorted order	It would display the objects in sorted order because it would make use of in-order traversal.
Search time complexity is O(1)	Search time complexity is $O(\log_2 n)$

26.Differentiate between Set and Map?

Set	Map
Set stores only values.	Map stores <key, value> pair.
Set is a member of Collections hierarchy.	Map is not a member of Collection hierarchy.
As the complexity of an object increases, efficiency reduces.	Efficiency would not be reduced even though complexity increases because hashing would be applied on "key" and not on value.

27.Differentiate between HashSet and HashMap?

HashSet	HaspMap
It is an implementation of Set interface	It is an implementation of Map Interface
Contains only values (objects)	Contains object as <key,value> pair
Duplicates are not permitted	Duplicate keys are not permitted. However, duplicate values are permitted
add() method would be used to insert an object	put() method would be used for insertion operation
One parameter has to be passed while inserting an object. Eg. add(Object arg0);	Two parameters have to be passed for put() method. Eg. put(Object arg0, Object arg1) put(key , value)
Slow in operation	Fast in operation

28. Differentiate between HashMap and TreeMap?

HashMap	TreeMap
Internally makes use of hashing algorithm.	Internally makes use of balanced binary search tree using red-black algorithm.
Data cannot be retrieved in a sorted manner.	Data can be retrieved in a sorted manner.

29. Differentiate between HashMap and Hashtable?

HashMap	HashTable
It is not a synchronized class.	It is a synchronized class.
It is not a legacy class.	It is a legacy class.
It can contain an empty Key or Value pair.	It cannot contain an empty Key or value pair.
Suitable for multithreaded programming	Not suitable for multithreaded programming

30. Differentiate between Collection and Collections?

"Collection" is an interface whereas "Collections" is a class.

31. Differentiate between Comparable and Comparator?

Comparable	Comparator
It expects implementation for compareTo() method	It expects implementation for compare() method

The first object is referred to as "this"	The first object is not referred to as "this"
It is present in java.lang package	It is present in java.util package
Target class has to be modified	Target class need not be modified
It is useful in such scenarios where the target class is modifiable	It is useful in all the scenarios

32. **Are the List, Set and Map elements automatically synchronized in java's Collection framework?**

No. They are not automatically synchronized.

33. **How do we synchronize List, Set and Map elements?**

Using Collections.synchronizeCollection(Collection c)

34. **What is the advantage of generic Collection?**

Advantages of generic collections are:

i) It is type safe

ii) It is checked at compilation time

iii) It prevents ClassCastException

iv) It makes the code clean since we need not use casting.

35. **What is collision in Hashtable and how is it handled?**

While computing bucket number for storing an object in hash table, if hash function allocates the same bucket for two different objects, then collision occurs. Therefore, hash function maintains a load factor which is 0.75 or 75% which means, if the hash table gets filled by the objects about 75% of its capacity, then automatically the size of the hash table would increase so that collision would be avoided.

36. **What is load factor in hashing base Collection?**

While computing bucket number for storing an object in hash table, if hash function allocates the same bucket for two different objects, then collision occurs. Therefore, hash function maintains a load factor which is 0.75 or 75% which means, if the hash table gets filled by the objects about 75% of its capacity, then automatically the size of the hash table would increase so that collision would be avoided.

37. **Why is Map not a part of Collection framework?**

Map is not a part of Collections framework because map contains <key, value> pair and does not fit into the Collections framework paradigm which contains only values.

38. **What is the difference between fail-fast and fail-safe?**

Fail-fast	Fail-safe
It would result in termination of the program by generating an Exception if structural modification is attempted	Does not terminate the program by generating an Exception if structural modification is attempted
No clone is used	Clone is used
All the Collections based classes present in java.util package exhibit fail-fast behavior	All the Collections based classes present in java.util.concurrent package exhibit fail-safe behavior

39. **How do we avoid ConcurrentModificationException while iterating Collection?**

To avoid ConcurrentModificationException we can make use of the classes present in java.util.concurrent package.

40. **What are the different Collection views provided by Map interface?**

The different Collection views are:

1) Keyset
2) EntrySet
3) values.

41. **How do you decide between using HashMap and TreeMap?**

In the project, if insertion, deletion and locating elements operations are of utmost important, then HashMap must be used. However, if traversing the keys in a sorted order is important then TreeMap must be used.

42. **Which Collection classes are Thread-safe?**

The legacy classes such as Vector, Hashtable, Properties and Stack are thread safe classes because they are synchronized.
However, from jdk 1.5 onwards, Collections based classes that are present in java.util.concurrent package such as CopyOnWriteArrayList, CopyOnWriteArraySet, ConcurrentHashMap etc. are also thread safe classes.

43. **What is a blocking Queue?**

Blocking Queue is a queue in which retrieval is possible if and only if the queue is non-empty and insertion is possible only when space is available. It is used for implementing producer-consumer problem.

44. **How can we sort a list of objects?**

We can sort a list of objects by :

i) Using TreeSet class
ii) Using Collections.sort() method.

45. **While passing a Collection as argument to a function how can we make sure that a function will not be able to modify it?**

By creating read only collection using:
Collections.unmodifiableCollection(Collection c)

46. What are the common algorithms implemented in Collections class?

i) Sorting
ii) Shuffling
iii) Routine Data Manipulation
iv) Searching
v) Composition
vi) Finding Extreme Values etc.

47. What are the best practices related to java Collections framework?

i) Choose the best type of Collection depending upon objects to be stored.
ii) If the number of objects to be stored is known in advance then preferably we can use Collections classes that allow us to specify the initial capacity.
iii) We can always use generics for type safety and to avoid ClassCastException at run time.
iv) We can also use Collections utility class methods since it enhances code re-usability, greater stability and maintainability.

48. Which are the classes that have implemented the List interface?

ArrayList and LinkedList classes.

49. Which are the classes that have implemented the Set interface?

HashSet and LinkedHashSet classes.

50. Which are the classes that have implemented the Queue interface?

ArrayDeque and PriorityQueue classes.

51. **Which are the classes that have implemented the Map interface?**

HashMap, TreeMap and LinkedHashMap classes.

52. **Which methods do we have to override in order to use an object as a key in HashMap?**

We have to override hashCode() and equals() methods.

53. **What is the difference between Stack and Queue?**

Stack	Queue
Uses LIFO (Last-In-First-Out) or FILO (First-In-Last-Out) to add and access data.	Uses FIFO (First-In-First-Out) or LILO (Last-In-Last-Out) to add and access data.
Insertion and deletion would happen only at one end	Insertion and deletion would happen at separate ends. For insertion, rear end would be used. For deletion, front end would be used.
Implementation is relatively easy as Stack has no variants.	Implementation is a bit complex as Queue is having variants such as Circular Queue, Priority Queue, Double ended Queue etc.

54. **How do we reverse the List in Collection?**

We can reverse the list in collection by using Collections.reverse() method.

55. **How do we convert an array of String into a List?**

We can convert an array of String into a list by using Collections.addAll() or Arrays.asList() method.

56. **What is the difference between peek(), poll() and remove() methods of Queue interface?**

peek() and poll() would return null if the queue is empty whereas remove() would throw NoSuchElementException.

57. **What is the difference between HashMap and ConcurrenthashMap?**

ConcurrentHashMap is synchronized and does not allow null key and null value, whereas HashMap is not synchronized and allows null key and null value.

58. **Arrange the following in the descending order of performance- ConcurrentHashMap, Hashtable, HashMap, Collections.SynchronizedMap.**

HashMap, ConcurrenthashMap, Collections.SynchronizedMap, Hashtable.

59. **How does HashMap work?**

(Refer class notes).

60. **What is identity HashMap?**

Identity HashMap implements Map interface by using reference equality instead of object equality.

61. **What is weakHashMap?**

WeakHashMap is an implementation of Map interface with weak keys.

62. **What is a Data Structure?**

A data structure is a specialized format for organizing and storing the data.

Eg. Array, Queue, LinkedList, Tree etc.

63. **Can any of the Collection class store primitive data?**

No. In all the Collections classes data would be stored in the object format.

64. **What happens when a primitive data is given to a Collection class?**

When primitive data is given to a collections class, it would be automatically converted into an object format. This process is called as auto boxing.

65. **What is meant by boxing?**

(Refer class notes).

66. **What is meant by auto-boxing?**

(Refer class notes).

67. **What is meant by unboxing?**

(Refer class notes).

68. **What is meant by auto-unboxing?**

(Refer class notes).

69. **What is the initial capacity of a Vector?**

Initial capacity of a vector is 10.

70. **By what factor does the capacity increase in case of a Vector if the existing capacity is filled?**

If the existing capacity is filled then the capacity doubles in existing

size.

71. **What are the different ways of printing a Vector?**

We can print a vector by using Enumeration.

72. **What are the different ways of printing a ArrayList?**

We can print ArrayList by using:

i) for loop

ii) enhanced for loop

iii) iterator and

iv) ListIterator.

73. **What are the different ways of printing a LinkedList?**

We can print LinkedList by using:

i) for loop

ii) enhanced for loop

iii) iterator

iv) ListIterator and

v) DescendingIterator

74. **What are the different ways of printing a ArrayDeque?**

We can print ArrayDeque by using:

i) enhanced for loop

ii) iterator and

iii) DescendingIterator

75. **What are the different ways of printing a PriorityQueue?**

We can print a PriorityQueue by using:

i) enhanced for loop and

ii) iterator.

76. **What are the different ways of printing a Treeset?**

We can print a TreeSet using:

i) enhanced for loop

ii) iterator and

iii) DescendingIterator.

77. **What are the different ways of printing a HashSet?**

We can print a HashSet using:

i) enhanced for loop and

ii) iterator.

78. **What are the different ways of printing a LinkedHashSet?**

We can print a HashSet using:

i) enhanced for loop and

ii) iterator.

79. **What are the different ways of printing a Treemap?**

We can print a TreeMap using:

1) enhanced for loop

2) iterator and

3) DescendingIterator.

80. **What are the different ways of printing a LinkedMap?**

We can print a LinkedssMap using

1) enhanced for loop and

2) iterator.

81. **What are the different ways of printing a LinkedHashMap?**

We can print a LinkedHashMap using:

1) enhanced for loop and

2) iterator

82. **What is the role of clone()?**

A clone() method is used to create a new Collections object. It is an alternative method to new operator and factory method to create the objects.

83. **How can an Array be converted to a List?**

We can convert an Array into a List by using Arrays.asList() method.

84. **How can a List be converted to an Array?**

We can convert a List into an Array by using toArray() method.

85. **What is the difference between add() and set()?**

add() method is used to insert a new object at the specified position. If any other object already exists in that position, then it would be shifted to the right so that empty location would be created to insert a new object.

In case of set() method, if any other object exists in the specified position, then shifting towards right would not takes place. Rather, new object would be replaced with old object by deleting it.

86. **What is the role of retainsAll()?**

The **retainAll() method** retains all matching elements in the current **ArrayList** that match all elements from the derived Collection list argument. In other words, retainAll() method is equivalent to intersection operation.

87. **Which are the Collection class in which duplicates are permitted?**

Duplicates are permitted in ArrayList, LinkedList, ArayDeque and PriorityQueue classes.

88. **Which are the Collection class in which duplicates are not permitted?**

Duplicates are not permitted in TreeSet, HashSet and LinkedHashSet classes.

89. **Which are the Collection class in which null are permitted?**
Null is permitted in ArrayList, LinkedList, HashSet and LinkedHashSet classes.

90. **Which are the Collection class in which null is not permitted?**
Null is not permitted in ArrayDeque, PriorityQueue and TreeSet.

91. **What are the two ways in which a LinkedList can be accessed in the reverse order?**
i) Using ListIterator
ii) Using DescendingIterator

(Eg. Refer class notes)

92. **What is the difference between add() and offer()?**
add() method would throw **IllegalStateException** if the space is not available to insert an object into a Collections.
offer() method would not throw any Exception if the space is not available to insert an object into a Collections. Rather, it would return false.

93. **Which Data Structure is used internally by the Vector class?**
Vector uses Dynamic array data structure.

94. **Which Data Structure is used internally by the ArayList class?**
ArrayList uses Dynamic array data structure.

95. **Which Data Structure is used internally by the LinkedList class?**
Doubly linked list data structure.

96. **Which Data Structure is used internally by the ArrayDeque class?**

Double ended queue data structure.

97. **Which Data Structure is used internally by the PriorityQueue class?**

PriorityQueue uses min-heap data structure.

98. **Which Data Structure is used internally by the TreeSet class?**

TreeSet uses balanced binary search tree data structure using red-black algorithm.

99. **Which Data Structure is used internally by the HashSet class?**

HashSet uses Hashing algorithm.

100. **Which Data Structure is used internally by the LinkedHashSet class?**

LinkedHashSet uses Hashing algorithm.

101. Which Data Structure is used internally by the TreeMap class?

min-heap data structure.

102. **Which Data Structure is used internally by the HashMap class?**

Hashing algorithm.

103. **Which Data Structure is used internally by the LinkedHashMap class?**

Hashing algorithm.

104. Under what circumstances should we use the Vector class?

(Refer class notes).

105. **Under what circumstances should we use the ArrayList class?**

(Refer class notes).

106. **Under what circumstances should we use the LinkedList class?**

(Refer class notes).

107. **Under what circumstances should we use the ArrayDeque class?**

(Refer class notes).

108. **Under what circumstances should we use the PriorityDeque class?**

(Refer class notes).

109. **Under what circumstances should we use the TreeSet class?**

(Refer class notes).

110. **Under what circumstances should we use the HashSet class?**

(Refer class notes).

111. **Under what circumstances should we use the LinkedHashSet class?**

(Refer class notes).

112. **Under what circumstances should we use the TreeMap class?**

(Refer class notes).

113. **Under what circumstances should we use the HashMap class?**

(Refer class notes).

114. **Under what circumstances should we use the LinkedHashMap class?**

(Refer class notes).

115. **What is the role of ceiling () and floor ()?**

(Refer class notes).

116. **What is the role of higher () and lower ()?**
(Refer class notes).

117. **What is the role of headset () and tailSet ()?**
(Refer class notes).

118. **Give the Collection interface hierarchy?**
(Refer class notes).

119. **Give the Map interface hierarchy?**
(Refer class notes).

120. **How can we perform reverse sorting on a Vector?**
(Refer class notes).

121. **How do you sort the data present in non-indexing data structure?**
(Refer class notes).

122. How do you sort a Map?
(Refer class notes).