**1. What is Collection in Java?**

Answer: A collection is a group of objects. In Java, these objects are known as elements of the collection.

For example, In childhood, we had a kiddy bank. In the kiddy bank, we had collected a lot of coins. This kiddy bank is called collection and the coins are nothing but objects.

Technically, a collection is a container object that stores a group of other objects as a single unit or single entity.

**2. Does a collection object store copies of other objects?**

Answer: No, a collection object works with reference types. It never stores copies of other objects. It stores references of other objects.

**3. Can we store a primitive data type into a collection?**

Answer: No, collections store only objects, not primitive data types values.

**4. Which types of objects can be stored in a collection or container object?**

Answer: We can store both homogeneous and heterogeneous objects in the collection or container object.

**5. What are duplicate objects in Java?**

Answer: The multiple objects of a class that contains the same data are called duplicate objects in java.

**6. What are unique objects in java?**

Answer: The multiple objects of a class that contains different data are called unique objects in java.

**7. Why do we need Collections in Java?**

Answer: Collections in Java can be used for storing multiple homogeneous and heterogeneous, duplicate, and unique elements without any size limitation.

**8. What is Framework in Java?**

Answer: A set of several classes and interfaces which provide a ready-made architecture is called framework in java.

**9. What is Collections Framework in Java?**

Answer: Collections framework in Java is a sophisticated hierarchy of numerous predefined interfaces and implementation classes that can be used to handle a group of objects as a single entity.

**10. In which java package is the collections framework placed?**

Answer: The collections framework is placed in java.util package.

**11. What is the main difference between Collection and Collections Framework in Java?**

Answer: Collection in java is an object that has multiple elements of the same type in a single entity. These multiple elements are accessed through one Collection object.

Whereas, Collections Framework is a library of interfaces and implementation classes that provides the common architecture for creating, accessing, and updating the different types of collections.

**12. What are the different components of collections framework?**

Answer: The different components of collections framework are interfaces, implementation classes, and algorithms.

**13. What is the difference between Arrays and Collections in Java?**  
Or, Explain the difference between Collections and Arrays.

Answer: The main difference between arrays and collections are as follows:

a) Arrays are fixed in size/length but collections are growable in nature. We can increase or decrease size based on our requirements.

b) Arrays are not recommended to use with respect to memory whereas collections are recommended to use with respect to memory.

c) Arrays are recommended to use with respect to performance but collections are not recommended to use with respect to performance.

d) Arrays can store only homogeneous data elements (similar type of data) but collections can hold both homogeneous and heterogeneous elements.

e) Arrays do not support any method whereas collections support numerous types of methods.

g) Arrays can hold both primitives and object types whereas, collections can hold only objects but not primitive.

**14. What are the main benefits/advantages of Collections Framework in Java?**

a) **Reusability:** Collections Framework in Java provides common classes and utility methods that can be used with various types of collections. It enables the reusability of the code.

b) **Quality:** Collections Framework improves the quality of code since the code is already tested and used by thousands of programmers.

c) **Speed:** Collections Framework improves the performance of the applications, reduces the development time and the burden of designers, programmers, and users.

d) **Maintenance:** Since Java Collections framework code is open source and API documents are widely available, therefore, the code is easier to maintain written with the help of the collections framework.

e) **Growable:** The size of the collection container is growable in nature.

**15. What are the limitations of Collections Framework in Java?**

Answer: There are two limitations of collections framework in Java. They are as follows:

* Care must be taken to use the appropriate cast operation.
* Compile type checking is not possible.

**16. What is the root interface of collection hierarchy in Java?**

Answer: The root interface of collection hierarchy in Java is collection interface. Since collection interface extends Iterable interface, therefore, some people consider Iterable interface as the root interface.

**17. In which Java package Iterable interface is present?**

Answer: Iterable interface is present in java.lang package. Whereas, Collection interface is present in java.util package.

**19. Which method of iterable interface is used to iterate over elements of the collection?**

Answer: Iterable interface provides only one method called iterator() method.

**20. Which interfaces extend the collection interface in java?**

Answer: List, Queue, and Set are three interfaces that extend the Collection interface. The map interface is not inherited by collection interface.

The list of basic interfaces available in Java collections framework are as follows:

* Collection interface
* List interface
* Set interface
* Queue interface
* Map interface

**23. What is the difference between collections and streams in java?**

a) Collections hold an array of values whereas, streams do not hold any values.

**24. What are the two ways to traverse elements of a collection in java?**

Answer: We can traverse elements of a collection either by using for-each or Iterator.

**25. Suppose that we are traversing elements of a collection using an Iterator. Can we remove elements from a collection using Iterator? If yes, how to?**

Answer: Iterator provides remove() method that removes the last element that is returned by the next method.

For example, the following code snippet removes “John” from the List using Iterator.

List<String> list = new ArrayList<String>();

list.add("Ivaan");

list.add("John");

list.add("Merry");

Iterator<String> itr = list.iterator();

while(itr.hasNext())

{

String element = itr.next();

if(element.equals("John"))

{

element.remove();

}

}

**26. Is it possible to convert a collection into an array?**

Answer: Collection interface provides toArray() method to convert a collection to an array. A sample of code is given below:

Object[ ] obj = list.toArray();

**27. Suppose we have a collection of String type and want to convey it into an Array of String type. How can it be achieved?**

Answer: list.toArray() will always return an array of Object type. To Array of type String, the following code snippet can be used:

String[ ] myArray = myStringList.toArray(new String[0]);

**28. Which of Collection classes implement Set interface in java?**

Answer: There are several collection classes in java that implement Set interface. They are:

* AbstractSet
* ConcurrentSkipListSet
* CopyOnWriteArraySet
* EnumSet
* HashSet
* LinkedHashSet
* TreeSet

**29. What are Collection classes that implement List interface in java?**

Answer: Collection classes that implement List interface, are:

* AbstractList
* AbstractSequentialList
* ArrayList
* AttributeList
* CopyOnWriteArrayList
* LinkedList
* RoleList
* RoleUnresolvedList
* Stack
* Vector

**30. What are Java collection classes that implement Queue interface?**

Answer: There are the following collection classes that implement Queue interface in java, are as:

* LinkedList
* PriorityQueue
* ArrayQueue
* PriorityBlockingQueue
* LinkedBlockingQueue
* **5. What are the important features of ArrayList in Java?**
* Answer: There are several significant features of ArrayList in Java that are as follows:
* a) ArrayList in Java uses an index-based structure.
* b) The size of ArrayList can increase or decrease at runtime. Once ArrayList is created, we can add any number of elements.
* c) An ArrayList allows adding elements into the middle of collection.
* d) It allows to delete elements.
* e) Duplicate elements are allowed in the array list.
* f) Any number of null elements can be added to ArrayList.
* g) ArrayList maintains the insertion order in Java. That is insertion order is preserved.
* h) ArrayList is not synchronized. That means multiple threads can use the same ArrayList objects simultaneously.
* i) Since ArrayList implements random access interface, we can get, set, insert, and remove elements of the array list from any arbitrary position.
* j) The performance of ArrayList is slow because if any element is removed from ArrayList, a lot of shifting takes place.
* **9. Does ArrayList allow to insert duplicate elements?**
* Answer: Yes, ArrayList allows to insert duplicate elements.
* **10. Is it possible to add null into ArrayList?**
* Answer: Yes, we can add any number of nulls into the array list.
* **11. Is it possible to join two or more ArrayLists in Java?**
* Answer: Yes, we can join two or more ArrayLists in java. List interface provides a method addAll() to join two or more lists in java.
* If we have one list list1 and another list2, we can join them with the help of addAll() like this: list1.addAll(list2);

**21. Why adding or inserting elements to ArrayList can be slow?**

Answer: When the size of ArrayList is unknown then adding elements to ArrayList is slow. When the size of ArrayList grows, a lot of shifting takes place in the memory while adding elements. Due to which the performance of ArrayList becomes slow.

**22. What is the difference between the length of an array and size of ArrayList?**

Answer: The length of an array can be determined by using property length. But ArrayList does not support the length property. It provides size() method that can be used to find the number of elements in the list.

**23. Suppose we want to add an element in the middle of list. Which list implementation will provide you better performance? ArrayList or LinkedList?**

Answer: For the above scenario, LinkedList is a better choice because in the case of LinkedList, when we add an element at the specified position, internally, a node is created and only two links are changed.

But in the case of ArrayList, a lot of shifting is done in the memory when we add an element in the middle of the list or anywhere, except at the end.

So, LinkedList gives faster performance when we add an element in the middle of list.

**24. Both ArrayList and LinkedList provide get() method to retrieve an element at the specified position from the list. Which one is faster, ArrayList or LinkedList?**

Answer: ArrayList’s get() method is faster than LinkedList’s get() because LinkedList does not implement Random Access Interface.

Due to which it will traverse from the beginning or ending over the list until it reaches the index specified.

**25. What are the advantages of ArrayList over Arrays?**

Answer: The advantages of ArrayList over Arrays are as follows:

* ArrayList can grow or shrink dynamically.
* ArrayList provides a more powerful insertion and search mechanism as compared to arrays

**26.Is it possible to convert an array to ArrayList in java?**

* Answer: Yes, it is possible to convert an array to ArrayList using asList() method of Arrays class. The asList() method is a static method provided by Arrays class that accepts the List objects.
* **1. What is LinkedList in Java?**
* Answer: LinkedList in Java is a linear data structure that uses a doubly linked list internally to store a group of elements or data.
* A doubly linked list consists of a group of nodes that together represents a sequence in the list. It stores the group of elements in the sequence of nodes.
* **2. What is the initial capacity of Java LinkedList?**
* Answer: The initial capacity of linked list is zero because the size of linked list automatically grows with the addition of elements. When the element is removed from the list, its size automatically shrinks.
* **3. In which scenario, LinkedList is better to use than ArrayList in Java?**
* Answer: Java LinkedList is better to use than ArrayList when the frequent operations are addition or deletion of elements in the middle of the list.
* No shifting of elements takes place after removal. Only the reference of the next and previous nodes will change.
* But in the case of ArrayList, if we remove an element from the middle of ArrayList, a lot of shifting of elements will take place after removal that will reduce the performance of ArrayList.
* **6. What are the key features of Java LinkedList?**
* Answer: The key features of Java LinkedList are as follows:
* a) The underlying data structure of LinkedList is a doubly LinkedList data structure.
* b) LinkedList allows storing duplicate elements.
* c) Null elements can be added to the linked list.
* d) Heterogeneous elements are allowed in the linked list.
* e) LinkedList is not synchronized. Therefore, it is not thread-safe.
* f) Since LinkedList is not synchronized. Hence, its operation is faster.
* g) Insertion and removal of elements in the LinkedList are fast.
* h) Retrieval (getting) of elements is very slow in LinkedList