3RD JULY

COLLECTION AND MAP API IN JAVA

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

   Ans: The Collection framework in Java is a set of classes and interfaces that provide an organized and efficient way to manage groups of objects. It offers various data structures such as lists, sets, queues, and maps to store, manipulate, and retrieve collections of objects.

2. What is the difference between ArrayList and LinkedList?

Ans: The main difference between ArrayList and LinkedList lies in their underlying implementation.

ArrayList is implemented as a dynamic array, where elements are stored in contiguous memory locations. It provides fast random access and is efficient for retrieving elements by index. However, inserting or deleting elements in the middle of the list can be slower.

LinkedList, on the other hand, is implemented as a doubly linked list, where each element holds a reference to the previous and next elements. It is efficient for inserting or deleting elements at any position, but random access is slower compared to ArrayList.

3. What is the difference between Iterator and Listiterator?

Ans: Iterator and ListIterator are both interfaces in Java that provide a way to iterate over the elements of a collection, but they have some differences:

Iterator: It is the base interface for all collection iterators. It allows traversing the elements in a forward direction and supports basic operations like iterating, removing elements, and querying the presence of the next element.

ListIterator: It extends the Iterator interface and provides additional functionality specifically for lists. It allows traversing the elements in both forward and backward directions. It supports operations like adding elements, replacing elements, obtaining the previous element, and more.

4. What is the difference between Iterator and Enumeration?

Ans: The key differences between Iterator and Enumeration are as follows:

Iterator: It is a more recent addition to the Java collections framework, introduced in Java 1.2. It is an interface that provides a universal way to iterate over elements in various collection classes. It supports removing elements during iteration and has more functionality compared to Enumeration.

Enumeration: It is an older interface that was present in Java before the collections framework was introduced. It is primarily used for iterating legacy classes like Hashtable and Vector. Enumeration does not support removing elements during iteration and has limited functionality compared to Iterator.

5. What is the difference between List and Set?

Ans: The main difference between List and Set lies in their characteristics:

List: It is an ordered collection that allows duplicate elements. Elements in a list have a specific index, and they can be accessed by their index position. Examples of list implementations are ArrayList, LinkedList, and Vector.

Set: It is a collection that does not allow duplicate elements. Unlike a list, a set does not have a specific order, and the order of elements may vary. Examples of set implementations are HashSet, TreeSet, and LinkedHashSet.

6. What is the difference between HashSet and TreeSet?

Ans: HashSet and TreeSet are both implementations of the Set interface, but they have some differences:

HashSet: It stores elements in a hash table using the hash code of the objects. It does not maintain any specific order of elements. HashSet offers constant-time complexity for basic operations like add, remove, and contains. However, the order in which elements are iterated is unpredictable.

TreeSet: It stores elements in a sorted tree structure (typically a red-black tree). It maintains the elements in sorted order, which allows for efficient retrieval of elements in ascending or descending order. TreeSet provides operations like add, remove, and contains with a time complexity of O(log n). It also offers additional operations for finding elements within a specified range.

7. What is the difference between Array and ArrayList?

Ans: The main differences between an Array and an ArrayList are as follows:

Array: It is a fixed-size data structure that stores elements of the same type in contiguous memory locations. Arrays have a fixed length defined at the time of declaration and cannot be dynamically resized. Accessing elements by index is efficient, but adding or removing elements requires manual shifting of elements.

ArrayList: It is a dynamic data structure that internally uses an array to store elements. Unlike arrays, ArrayLists can dynamically grow and shrink as needed. Elements can be easily added or removedfrom an ArrayList without the need for manual shifting. ArrayList provides additional methods for manipulating the collection, such as adding elements at specific positions, removing elements by object reference, and more.

8. What is a Map in Java?

Ans: A Map in Java is an interface that represents a collection of key-value pairs. It provides an efficient way to store, retrieve, and manipulate data based on unique keys. Each key in a map is associated with a corresponding value, and the keys are typically used to access and retrieve the corresponding values. Maps do not allow duplicate keys, and each key can map to at most one value.

9. What are the commonly used implementations of Map in Java?

Ans: Some commonly used implementations of the Map interface in Java are:

HashMap: It stores key-value pairs in a hash table. HashMap provides constant-time performance for basic operations like put and get, making it a commonly used map implementation. However, it does not guarantee any specific order of the keys.

TreeMap: It stores key-value pairs in a sorted tree structure (typically a red-black tree). TreeMap maintains the keys in sorted order, allowing efficient retrieval of elements in ascending or descending order. The operations on TreeMap have a time complexity of O(log n).

10. What is the difference between HashMap and TreeMap?

Ans: The main differences between HashMap and TreeMap are as follows:

HashMap: It does not maintain any specific order of the keys. The keys are stored in a hash table based on their hash code, allowing for fast retrieval of values using keys. HashMap offers constant-time performance for basic operations like put and get.

TreeMap: It maintains the keys in sorted order. The keys are stored in a sorted tree structure (typically a red-black tree), allowing efficient retrieval of elements in ascending or descending order. TreeMap provides operations with a time complexity of O(log n).

11. How do you check if a key exists in a Map in Java?

Ans: To check if a key exists in a Map in Java, you can use the containsKey(Object key) method provided by the Map interface. This method returns true if the map contains the specified key, and false otherwise. Here's an example:

```
Map<String, Integer> map = new HashMap<>();

map.put("key1", 10);

map.put("key2", 20);


if (map.containsKey("key1")) {

   System.out.println("Key 'key1' exists in the map.");

} else {

   System.out.println("Key 'key1' does not exist in the map.");

}
```

In this example, the containsKey() method is used to check if the key "key1" exists in the map. If it does, the corresponding message is printed.