**Assignment No:-62**

Name:-Suryawanshi Sangramsingh Sambhaji

Batch: - Delta - DCA (Java) 2024 Date:-12/8/2024

**1. What is an ArrayList in Java?**

An ArrayList in Java is a part of the Java Collections Framework and implements the List

interface. It is a resizable array, which means that elements can be added and removed

dynamically. Unlike arrays, which have a fixed size once they are created, ArrayList can

grow or shrink as needed, providing greater flexibility in handling collections of data. It

is widely used due to its ease of use and efficiency for many common operations.

**2. How is an ArrayList different from an array?**

An ArrayList differs from a traditional array in several ways. Firstly, an array has a fixed

size that must be specified at the time of its creation, and it cannot be changed later. In

contrast, an ArrayList can dynamically resize itself to accommodate more elements as

needed. Secondly, arrays can store both primitive types and objects, whereas ArrayList can

only store objects. Lastly, ArrayList provides a rich set of methods for manipulating the

elements, such as adding, removing, and searching for elements, which are not available

with arrays.

**3. Explain the dynamic nature of ArrayList.**

The dynamic nature of an ArrayList refers to its ability to automatically resize itself as

elements are added or removed. Internally, ArrayList maintains an array that can grow or

shrink. When the current array becomes full, a new array is created with a larger capacity,

and the elements are copied over to the new array. This resizing mechanism allows the

ArrayList to provide flexibility in terms of storage and management of elements, making it

a convenient choice for collections that need to change size frequently.

**4. What is the default capacity of an ArrayList in Java?**

When an ArrayList is created without specifying an initial capacity, it starts with a default

capacity of 10. This means that the internal array initially has room for 10 elements. If the

number of elements exceeds this capacity, the ArrayList will automatically resize itself to

accommodate the additional elements.

**5. How does ArrayList handle resizing when it reaches its capacity?**

When an ArrayList reaches its current capacity, it needs to increase its capacity to

accommodate more elements. This is done by creating a new, larger array and copying

the elements from the old array to the new one. The new capacity is usually calculated as

1.5 times the old capacity (although the exact growth factor can vary across different

implementations). This approach helps to balance the performance trade-offs between

memory usage and the frequency of resizing operations.

**6. What are the key features of the ArrayList class?**

The ArrayList class in Java offers several key features that make it a powerful and flexible

data structure. These features include dynamic resizing, random access to elements using

indices, and a rich set of methods for manipulating elements such as adding, removing,

searching, and sorting. Additionally, ArrayList maintains the order of elements as they

were added and allows for duplicate elements.

**7. How do you create an empty ArrayList in Java?**

An empty ArrayList in Java can be created using its default constructor. For example:

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

This creates an empty ArrayList with the default initial capacity.

**8. What is the role of the capacity in an ArrayList?**

The capacity of an ArrayList refers to the size of the internal array that stores the elements.

It determines the maximum number of elements the ArrayList can hold before it needs to

resize. Managing capacity efficiently can help optimize the performance of the ArrayList

by reducing the frequency of resizing operations.

**9. Can an ArrayList contain primitive data types in Java?**

An ArrayList cannot directly contain primitive data types such as int, char, or double.

Instead, it can store objects of the wrapper classes corresponding to these primitive types,

such as Integer, Character, and Double. This allows the ArrayList to store elements in a

uniform manner, as all elements are objects.

**10. What is the initial capacity of an ArrayList?**

If no initial capacity is specified, an ArrayList starts with a default initial capacity of

10.This means the internal array can initially hold up to 10 elements before needing to

resize.

**11. How do you add elements to an ArrayList?**

Elements can be added to an ArrayList using the add method. For example:

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

list.add("Hello");

list.add("World");

This adds the strings "Hello" and "World" to the ArrayList.

**12. Explain the difference between add() and addAll() methods in ArrayList.**

The add method adds a single element to the ArrayList, while the addAll method adds all

elements from a specified collection to the ArrayList. For example:

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

list.add("Hello");

List<String> moreWords = Arrays.asList("World", "!");

list.addAll(moreWords);

This results in the ArrayList containing "Hello", "World", and "!" after both operations.

**13. How can you remove an element from an ArrayList?**

An element can be removed from an ArrayList using the remove method, which can take

either the index of the element to be removed or the element itself. For example:

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

list.add("Hello");

list.add("World");

list.remove("Hello");

list.remove(0);

**14. What happens when you call the clear() method on an ArrayList?**

Calling the clear method on an ArrayList removes all elements from the list, effectively

making it empty. The size of the ArrayList becomes zero, but the underlying array remains

the same size. For example:

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

list.add("Hello");

list.add("World");

list.clear();

**15. Discuss the difference between ArrayList and LinkedList.**

ArrayList and LinkedList are both implementations of the List interface but differ in their

internal structure and performance characteristics. ArrayList uses a dynamic array to store

elements, which allows for fast random access and efficient iteration. However, adding or

removing elements in the middle can be slow due to the need to shift elements. In

contrast, LinkedList uses a doubly-linked list, which allows for fast insertion and removal

of elements at any position, but slower random access because elements must be

traversed sequentially.

**16. Explain the significance of the ensureCapacity() method in ArrayList.**

The ensureCapacity method in ArrayList is used to increase the capacity of the internal array

to ensure that it can hold at least the specified number of elements without resizing. This

can help improve performance by reducing the number of resizing operations when a

large number of elements are added. For example:

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

list.ensureCapacity(100);

**17. What is the purpose of the trimToSize() method in ArrayList?**

The trimToSize method reduces the capacity of the ArrayList to its current size. This can

help save memory if the ArrayList has a large capacity but only a few elements. For

example:

ArrayList<String> list = new ArrayList<>(100);

list.add("Hello");

list.trimToSize();

**18. How can you check if an ArrayList contains a specific element?**

The contains method is used to check if an ArrayList contains a specific element. It returns

true if the element is found, and false otherwise. For example:

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

list.add("Hello");

boolean containsHello = list.contains("Hello");

**19. Discuss the difference between ArrayList and Vector.**

ArrayList and Vector both implement the List interface, but they have different

synchronization properties and growth policies. ArrayList is not synchronized, meaning it

is not thread-safe, whereas Vector is synchronized, making it thread-safe. However, this

synchronization can result in slower performance for single-threaded applications.

Additionally, Vector increases its capacity by doubling it when it needs to grow, while

ArrayList increases its capacity by 50%.

**20. What is the role of the set() method in ArrayList?**

The set method in ArrayList is used to replace the element at a specified position with a

new element. This method returns the element that was previously at the specified

position. For example:

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

list.add("Hello");

list.add("World");

list.set(1, "Java");

**21. How do you find the size of an ArrayList?**

The size method is used to find the number of elements in an ArrayList. It returns an

integer representing the current number of elements. For example:

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

list.add("Hello");

int size = list.size();