ArrayList in Java - Detailed Notes

1. Introduction to ArrayList

- Definition: ArrayList is a resizable array implementation of the List interface in Java, part of the java.util package.

- Characteristics: Unlike arrays, ArrayList can dynamically resize itself when elements are added or removed, adapting its capacity.

- Usage: Commonly used when we need a dynamic data structure that allows random access to elements by index.

2. Creating an ArrayList

- Declaration: ArrayList<Type> list = new ArrayList<>();

- With Initial Capacity: ArrayList<Type> list = new ArrayList<>(int initialCapacity);

```
ArrayList<String> names = new ArrayList<>();

ArrayList<Integer> numbers = new ArrayList<>(20); // capacity of 20
```

3. Adding Elements

- add(element): Appends the specified element to the end of the list.

- add(index, element): Inserts the element at the specified index, shifting elements if necessary.

```
names.add("Alice");

names.add(1, "Bob");
```

4. Accessing Elements

- get(index): Returns the element at the specified index.

```
String name = names.get(0);
```

5. Modifying Elements

- set(index, element): Replaces the element at the specified index with a new element.

```
names.set(1, "Charlie");
```

6. Removing Elements

- remove(index): Removes the element at the specified index.

- remove(Object): Removes the first occurrence of the specified element, if it exists.

```
names.remove(0);

names.remove("Charlie");
```

7. Size and Capacity

- size(): Returns the number of elements in the ArrayList.

- Dynamic Sizing: ArrayList expands automatically if elements exceed the initial capacity.

8. Iterating Over ArrayList

- Using a for loop

- Enhanced for-each loop

- Using Iterator

```
for (int i = 0; i < names.size(); i++) { System.out.println(names.get(i)); }

for (String name : names) { System.out.println(name); }

Iterator<String> it = names.iterator(); while (it.hasNext()) { System.out.println(it.next()); }
```

9. Common Methods in ArrayList

- contains(Object): Checks if the ArrayList contains a specific element.

- indexOf(Object): Returns the index of the first occurrence of the specified element or -1 if not present.

- isEmpty(): Returns true if the ArrayList has no elements.

- clear(): Removes all elements from the ArrayList.

10. Sorting ArrayList

- Using Collections.sort(): Sorts the ArrayList in natural order or using a custom comparator.

```
Collections.sort(names);
```

11. Conversion to Array

- Using toArray(): Converts ArrayList to an array.

```
String[] namesArray = names.toArray(new String[0]);
```

12. Synchronization

- Thread-Safety: ArrayList is not synchronized. Use Collections.synchronizedList(new ArrayList<>()) for thread safety.

```
List<String> syncList = Collections.synchronizedList(new ArrayList<>());
```

13. Advantages and Limitations

- Advantages: Dynamic resizing, random access by index, and flexible structure.
- Limitations: Inefficient for inserting/deleting elements in the middle (requires shifting), not thread-safe without external synchronization.

Additional Notes on ArrayList in Java

Key Points of ArrayList in Java:

- ArrayList is a resizable array, also known as a growable array.
- Duplicates are allowed in ArrayList.
- Insertion order is preserved.
- Heterogeneous objects are allowed.
- Null insertion is possible.

Complexity of Java ArrayList:

Operation	Time Complexity	Space Complexity
Inserting Element	O(1)	O(N)
Removing Element	O(N)	O(1)
Traversing Elements	O(N)	O(N)
Replacing Elements	O(1)	O(1)

Advantages of Java ArrayList:

- Dynamic size: ArrayList can grow and shrink in size as needed.
- Easy to use and popular among Java developers.
- Fast access to elements due to underlying array structure.
- Ordered collection: maintains the order of elements.
- Supports null values.

Disadvantages of Java ArrayList:

- Slower than arrays for certain operations, such as inserting elements in the middle.
- Higher memory usage compared to arrays due to resizing needs.