Java Array List Interview Questions

Array List

Java Array List Interview Questions, Algorithms and Array List Programs.

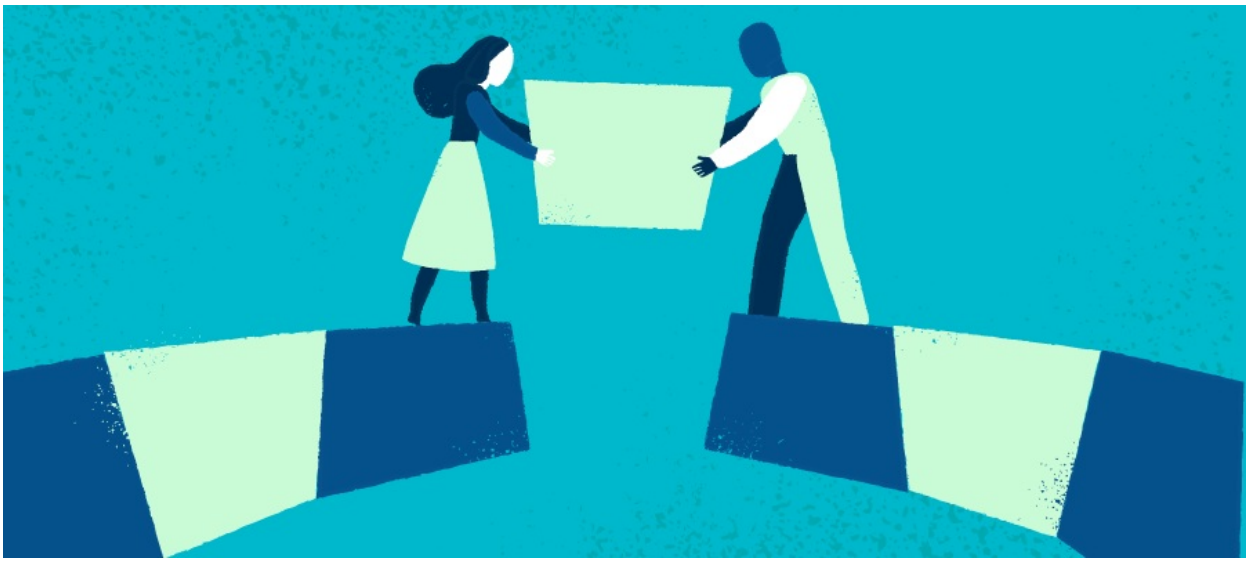

Table of Contents:

Top four Java ArrayList Interview Questions

Java Array List Interview Questions?

How to describe Array Lists in interview?

- Java Array List is the resizable array implementation of list interface .
- Java ArrayList is not synchronized.
- Size of the ArrayList grows and shrinks as you add or remove the elements.
- ArrayLists in Java supports generics.
- Big-O Complexity Access Θ(1) SearchΘ(n) InsertionΘ(n) DeletionΘ(n)

Question: Write a program to implement your own Array List in Java.

This implementation should contain following methods add(), get(),remove(),size() and ArrayList should increase its size when it reaches its size threshold.

Note: You can download the working project at the end of the page(implemented in Java and eclipse)

Answer: Now we will write code to implement our arrayList in a class called CustomArrayList and then test it via another class called AppRunner.

```
package codespaghetti.com;
import java.util.Arrays;

public class CustomArrayList {
 private Object[] arrayPlaceHolder;
 private int actSize = 0;

 public CustomArrayList() {
  arrayPlaceHolder = new Object[10];
 }

 public Object get(int index) {
  if (index < actSize) {
   return arrayPlaceHolder[index];
  } else {
   throw new ArrayIndexOutOfBoundsException();
  }
 }

 public void add(Object obj) {
  if (arrayPlaceHolder.length - actSize <= 5) {
   increaseListSize();
  }
  arrayPlaceHolder[actSize++] = obj;
 }

 public Object remove(int index) {
  if (index < actSize) {
   Object obj = arrayPlaceHolder[index];
   arrayPlaceHolder[index] = null;
   int tmp = index;
   while (tmp < actSize) {
    arrayPlaceHolder[tmp] = arrayPlaceHolder[tmp + 1];
    arrayPlaceHolder[tmp + 1] = null;
    tmp++;
   }
   actSize--;
   return obj;
  } else {
   throw new ArrayIndexOutOfBoundsException();
  }

 }

 public int size() {
  return actSize;
 }

 private void increaseListSize() {
  arrayPlaceHolder = Arrays.copyOf(arrayPlaceHolder, arrayPlaceHolder.length * 2);
  System.out.println("nNew length: " + arrayPlaceHolder.length);
 }
}
```

We can test our CustomArrayList class by running the AppRunner Class

```
package codespaghetti.com;
public class AppRunner {

 public static void main(String[]args){
 CustomArrayList customArrayList = new CustomArrayList();
  customArrayList.add(new Integer(1));
  customArrayList.add(new Integer(2));
  customArrayList.add(new Integer(3));
  customArrayList.add(new Integer(4));
  customArrayList.add(new Integer(5));
     for(int i=0;i<customArrayList.size();i++){
         System.out.print(customArrayList.get(i)+" ");
     }
     customArrayList.add(new Integer(9));
     System.out.println("Element at Index 3:"+customArrayList.get(3));
     System.out.println("Size: "+customArrayList.size());
     System.out.println("Removing element at index 1: "+customArrayList.remove(1));
     for(int i=0;i<customArrayList.size();i++){
         System.out.print(customArrayList.get(i)+" ");
     }
 }
}


//OutPut
1 2 3 4 5
New length: 20
Element at Index 3:4
Size: 6
Removing element at index 1: 2
1 3 4 5 9
```

Performance

Big' O' values for our CustomArrayList functions are:

Get	Add	remove
O(1)	O(1)	O(1)

Download the Source code

Download the fully functional CustomArrayList project

Question: How to synchronize Array List in Java?

By Default ArrayList is not synchronized, but it is possible to create a synchronized arrayList by following two ways

1. Using **Collections.synchronizedList()** method
2. Using thread-safe variant of ArrayList which is :**CopyOnWriteArrayList**

In real interview there is a 100% chance the next question they will ask is

Question: What is CopyOnWriteArrayList ?

CopyOnWriteArrayList: is a concurrent Collection class and it implements List interface like ArrayList, Vector and LinkedList.

But it is a thread-safe collection and it achieves its thread-safety in a slightly different way than Vector.

As name suggest CopyOnWriteArrayList creates copy of underlying ArrayList with every mutation operation e.g. add or set.

Normally CopyOnWriteArrayList is very expensive because it involves costly Array copy with every write operation.

But its very efficient if you have a List where Iteration outnumber mutation e.g. you mostly need to iterate the ArrayList and don't modify it too often.

Iterator of CopyOnWriteArrayList is fail-safe and doesn't throw ConcurrentModificationException even if underlying CopyOnWriteArrayList is modified.

Once Iteration begins because Iterator is operating on separate copy of ArrayList. Consequently all the updates made on CopyOnWriteArrayList is not available to Iterator.

Question: How to get sublist from ArrayList in Java?

Answer: We can get a list of elements by using subList() method.This would be very helpful in case of sorted list.

Question: How do you increase the current capacity of an ArrayList?

The capacity of an ArrayList is automatically increased when we try to add more elements than the current capacity. However to manually increase the current capacity, **ensureCapacity()** method is used.

Question: What is difference between length() of array and size() of ArrayList methods?

If we call length() method on array, it will return how many elements. We can store in this array, this is known as capacity, but if we call size() function of ArrayList class then

It will return total number of elements currently stored in ArrayList, which is always less than or equal to capacity.

Question: What is CopyOnWriteArrayList in Java?

Its a concurrent collection class which is introduced as an alternative of synchronized List in Java. This class take advantage of advanced thread-safety technique instead of locking. Its very efficient if ArrayList is mostly used for reading purpose. Because it allows multiple threads to read data without locking, which was not possible with synchronized ArrayList. See answer to learn more about CopyOnWriteArrayList class.

```
public class MainClass
{
public static void main(String[] args)
{
ArrayList list = new ArrayList();

//Now 'list' can hold 10 elements (Default Initial Capacity)

list.ensureCapacity(20);

//Now 'list' can hold 20 elements.
}
```

Question: While passing an ArrayList to a method or returning an ArrayList from a method, when is it considered to be a security violation? How to fix this problem?

When one passes the array to a method, if array is assigned to the member variable directly without making a copy of it. It could lead to a scenario that the original array when changed by the caller will also end up changing the array passed to the method.

Question: When would you use ArrayList and when LinkedList?

ArrayList works best for cases where you're doing random access on the list and a LinkedList works better if you're doing a lot of editing in the middle of the list.

for a complete overview check the answer to performance and ArrayList VS Linked list questions below on the page.

Question: What is time complexity of different ArrayList operations in terms of big o notation?

Operation	Performance	Description
Get	O(1)	**FAST:**Since ArrayList uses array as underline data structure and array is index based data structure searching in array using array.get(index) will take O(1)

Add	O(n)	**SLOW:**Add operations is slow consider the case if underlying Array is full, we need to copy contents to new array which makes inserting an element into ArrayList of O(n) in worst case, while ArrayList also needs to update its index if you insert something anywhere except at the end of array
Contains	O(n)	**SLOW:** Same reason as above, we need to iterate all the values in order to find the one we are looking for . But it will be 0(1) if the value we are looking for is the first value in the arrayList
Remove	O(n)	**SLOW:** Again we need to iterate the whole arrayList to find the value we want to remove.

Question: Are arrayLists thread safe ?If not then how can we make it thread safe?

ArrayList are non- synchronized and therefore are not suited for multithreading environments.

But There are two ways to synchronize an ArrayList explicitly:

1. Using Collections.synchronizedList() method
2. Using thread-safe variant of ArrayList: CopyOnWriteArrayList

Question: Which access modifiers can be used to declare ArrayList

ArrayLists can be declared as PRIVATE, PUBLIC, PROTECTED, and without any modifiers. Following table gives an overview about the accessibility of arrayList for different access modifiers.

```
            | Class | Package | Subclass | Subclass | World
            |       |         |(same pkg)|(diff pkg)|
————————————+———————+—————————+——————————+——————————+———————
public      |   +   |    +    |    +     |     +    |   +
————————————+———————+—————————+——————————+——————————+———————
protected   |   +   |    +    |    +     |     +    |   o
————————————+———————+—————————+——————————+——————————+———————
no modifier |   +   |    +    |    +     |     o    |   o
————————————+———————+—————————+——————————+——————————+———————
private     |   +   |    o    |    o     |     o    |   o

+ : accessible
o : not accessible
```

Java Array List Algorithm Questions?

Question: Implement Quick sort with arrayList in Java.

In this tutorial we will implement "**Quick Sort**" algorithm using ArrayList. Quick sort is **inplace** sorting algorithm and worst case for quick sort to run is O (n^2).

Quick sort implementation in Java

```
package codespaghetti.com;

import java.util.ArrayList;
import java.util.Random;

public class QuickSort {
private static ArrayList inputArray = new ArrayList();

    public QuickSort(ArrayList inputArray){
        QuickSort.inputArray = inputArray;
    }

    public void startQuickStart(int start,int end){
        int q;
        if(start<end){
            q = partition(start, end);
            startQuickStart(start, q);
            startQuickStart(q+1, end);
        }
    }

    public ArrayList getSortedArray(){
        return QuickSort.inputArray;
    }

    int partition(int start,int end){
        System.out.println("n---------Iteration Starts----------");
        System.out.println("nSorting Window from index number:"+start+" to "+end);
```

```
        int init = start;
        int length = end;

        Random r = new Random();
        int pivotIndex = nextIntInRange(start,end,r);
        int pivot = inputArray.get(pivotIndex);

        System.out.println("Pivot Element "+pivot+" at index:"+pivotIndex);

        while(true){
            while(inputArray.get(length)>pivot && length>start){
                length--;
            }

            while(inputArray.get(init) max) {
              throw new IllegalArgumentException("Cannot draw random int from
invalid range [" + min + ", " + max + "].");
           }
           int diff = max - min;
           if (diff >= 0 && diff != Integer.MAX_VALUE) {
              return (min + rng.nextInt(diff + 1));
           }
           int i;
           do {
              i = rng.nextInt();
           } while (i < min || i > max);
           return i;
        }
}
```

Main class to run the code

```
import java.util.ArrayList;

import daa.QuickSort;


public class MainPractise implements Cloneable {

    public static void main(String[] args) {

        ArrayList unsortedArray = new ArrayList();

        unsortedArray.add(2);
        unsortedArray.add(8);
        unsortedArray.add(6);
        unsortedArray.add(3);
        unsortedArray.add(12);
        unsortedArray.add(4);
        unsortedArray.add(7);

        QuickSort qsu = new QuickSort(unsortedArray);
        System.out.println("---------Initial Unsorted Array---------");
        for(int i:qsu.getSortedArray()){
            System.out.print(i+" ");
        }

        qsu.startQuickStart(0, unsortedArray.size()-1);


        System.out.println("nn---------Processed sorted Array---------");
        for(int i:qsu.getSortedArray()){
            System.out.print(i+" ");
        }
    }
}
```

Output

```
———Initial Unsorted Array———
2 8 6 3 12 4 7
———Iteration Starts———-

Sorting Window from index number:0 to 6
Pivot Element 12 at index:4

After Swapping
2 8 6 3 7 4 12
———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:0 to 5
Pivot Element 8 at index:1

After Swapping
```

```
2 4 6 3 7 8
———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:0 to 4
Pivot Element 3 at index:3

After Swapping
2 3 6 4 7
———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:0 to 1
Pivot Element 3 at index:1

———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:0 to 1
Pivot Element 3 at index:1

———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:0 to 1
Pivot Element 2 at index:0

———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:2 to 4
Pivot Element 7 at index:4

———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:2 to 4
Pivot Element 4 at index:3

After Swapping
4 6 7
———Iteration Ends———

———Iteration Starts———-

Sorting Window from index number:3 to 4
Pivot Element 6 at index:3

———Iteration Ends———
———Processed sorted Array———
2 3 4 6 7 8 12
```

Question: How to sort ArrayList in Java?

Answer:ArrayLists can be sorted by using Collections.sort() method, all you need to make sure is that elements implements either Comparable or Comparator interface.

Former is used to sort on natural order while later is used while sorting in custom order.

If object's stored in ArrayList doesn't implements Comparable thanthey can not be sorted using Collections.sort().

Question: How to remove repeated elements from ArrayList?

Here is the approaches we can use to remove duplicate values from ArrayList

1. First copy all the elements of ArrayList to LinkedHashSet. Why we choose LinkedHashSet? Because it removes duplicates and maintains the insertion order.
2. Copying all the elements of LinkedHashSet (non-duplicate elements) to the ArrayList.

Here is the full code for the solution

```
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
/** * Java Program to remove repeated elements from ArrayList in Java. */
public class ArrayListDuplicateDemo{
public static void main(String args[]){
// creating ArrayList with duplicate elements
List primes = new ArrayList();
primes.add(2);
primes.add(3);
primes.add(5);
primes.add(7);
duplicate primes.add(7);
primes.add(11);

// let's print arraylist with duplicate
System.out.println("list of prime numbers : " + primes);

// Now let's remove duplicate element without affecting order LinkedHashSet will
guaranteed the order and since //it's set  it will not allow us to insert
duplicates. repeated elements will automatically filtered.

Set primesWithoutDuplicates = new LinkedHashSet(primes);

// now let's clear the ArrayList so that we can copy all elements from LinkedHashSet
primes.clear();
// copying elements but without any duplicates
primes.addAll(primesWithoutDuplicates);

System.out.println("list of primes without duplicates : " + primes); } }
```

OutPut:

```
Output list of prime numbers : [2, 3, 5, 7, 7, 11] list of primes without
duplicates : [2, 3, 5, 7, 11]
```

About The Author:

Keys To Interview Success:

ArrayLists are always mentioned in relation to arrays and linkedLists.

This is because all of these three data structures are related to each other and it is vital to know the differences among them.

In real world the selection of wrong data structure can cause a lot of problems.

Interviewers are always interested in to know, If a candidate is familiar with details of internal workings, performance and structural differences of data structures.

It is therefore necessary for you to grasp all the concepts about structural differences , performance and sorting algorithms to increase your chances of success in interview.

References

- https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html
- https://www.amazon.com/Cracking-Coding-Interview-Programming-Questions/dp/098478280X
- http://stackoverflow.com/questions/18441846/how-to-sort-an-arraylist-in-java