ArrayList vs. Array: A comparison chart

Basis of Differentiation	Array	ArrayList
Size of data structure	An array contains a data structure of fixed length. The size of the Array cannot be changed once the object has been defined. It is static.	The size of an ArrayList is dynamic. The elements/ items in the data structure can be modified to change the size of the object as and when required.
Resizing property	As an Array length is static across the program, its size will remain unchanged.	The size of an ArrayList is capable of changing dynamically based on the capacity and load it has to function with.
Insertion and storage of elements	The assignment operator is put to use for the storage of elements.	The *add()* attribute is utilized for the insertion of elements in an ArrayList.
Nature of datatypes	An Array can store primitive data types as well as other objects that are of the different or same data type.	ArrayLists can only store object types. They are not able to contain primitives.
Generics	Generics are not compatible with Arrays.	ArrayLists allow the use of Generics.
Multi-dimensional nature	Arrays are multi-dimensional.	ArrayLists are single-dimensional.
Storage in contiguous memory locations	An Array is a native programming component wherein the elements are contained in adjacent memory locations.	An ArrayList belongs to a class belonging to Java's collections framework. Here, the objects are incapable of being contained in contiguous locations.
Determination of length	The *Length* variable is responsible for determining an Array's length.	The length of an ArrayList is set by the *Size ()* method.
Memory space consumed	Arrays take more memory for the storage of specified objects or elements.	ArrayLists take less memory space for storing objects or elements.
Iteration	Iterating across an Array takes lesser time than what it does in the case of ArrayLists.	Iterating across an ArrayList takes more time, and the performance is slower.

Array in JAVA

Arrays are strongly-typed data collections that consist of values of the same type. They have a fixed length, which can't be altered during runtime. The Array elements can be accessed by using their indices that starts at zero.

While the value of the reference elements are kept at `null`, the default value of the numeric array elements is usually set to zero.

Example of Array

```
Int[ ] intArray =new int [ ] {2};
intArray [0] = 1;
intArray [2] = 2;
```

ArrayList

An Array list is different from an Array as it is not like a strongly-typed collection. It is a resizable array which is present in the `java.util package`. It is capable of storing data types that are either similar or different. Its overall size and quality can decrease or increase dynamically to capture values of all sizes, and that too from any data type.

An ArrayList feature is one of the most flexible data structures in the C# Collections. It presents a simple and easy-to-implement list of values.

When you are using the ArrayList, you will find that it implements the IList interface that is compatible with Arrays. You can use it to modify, add, insert, delete, or view the data types entered by you.

In Java, you can access an element of the ArrayList using the *get()* method.

```
students.get(3);
```

Example of an Array List

```
ArrayList Arrlst = new ArrayList ( );
Arrlst.Add (“Uma”);
Arrlst.Add (“1”);
Arrlst.Add (“null”);
```

Array vs. ArrayList: Head-to-head comparison

The **difference between Array and ArrayList** is described in the following section using 8 comparative points - size, performance, primitives, iterator, type safety, length, adding elements, and multi-dimensions.

1. Size

Arrays are static in their length and size. It is not possible to change their length once the developer has created the array object. They contain similar datatype items/elements that have been defined sequentially.

On the other hand, an Array List is dynamic. An ArrayList object will showcase an instance of variable capacity that appropriately depicts the size of the ArrayList. The capacity of the ArrayList is extensible and keeps growing automatically when more variables are added to it.

2. Performance

The **performance of the ArrayList and Array** is depended upon the operation performed on them. For instance, in the *resize()* operation, automatic resizing of ArrayList decreases the performance of the operation. This happens because it uses a temporary array for copying elements to the new array from the old one.

When you use an Array list, it is backed internally by an Array in the process of calling the native implemented method as given below:

```
System.arrayCopy(src,srcPos,dest,destPos,length)
```

add() **or** ***get()*** **operation:** The procedure of retrieving an element from or adding any element from the ArrayList or Array object projects has the same performance levels. In the case of an ArrayList object, the operations would be carried out in constant time.

Overall, an ArrayList is slower than an Array.

3. Primitives

Another difference between ArrayList and array in Java is that an ArrayList cannot hold primitive data types such as int, float, double, etc. It contains objects only.

On the other hand, Arrays are designed to contain both objects and primitive data types together.

4. Iteration of values

You can iterate through the values in an ArrayList using an iterator. The iterators that are returned by an ArrayList class's list iterator are fail-fast.

On the other hand, the commands `‘for each loop’` or `‘for loop’` are used by Java developers to iterate through an Array.

5. Type-safety

Java developers find it easy to ensure the programming language’s type-safety feature with the help of Generics. Here, it is important to understand that an Array comprises of tightly-typed similar or homogeneous data structure.

It may contain primitives belonging to specific data types only. You cannot store different types of data here. In case you try it, the `ArrayStoreException` error will be thrown.

This is not true in the case of ArrayLists.

For instance:

```
String temp[] =  new String[3];
```

This will generate a string array of size 3

```
temp[0] = new Integer(12);
```

This command will throw `ArrayStoreException`. This is because the code is attempting to include an Integer object in *String[ ].*

6. Length:

The *size( )* allows users to define the length of the ArrayList. On the other hand, Array objects use the *length()* variable to fetch the length of the specified array.

For instance,

This command uses the array object length variable

```
Integer arrayobject [ ] = new Integer[6];
arraylength= arrayobject.length;
```

This command uses the ArrayList object size method

```
ArrayList  arraylistobject = new ArrayList();
arraylistobject.add(10);
arraylistobject.size( );
```

7. Addition of elements:

Elements can be inserted into the ArrayList object by using the *add( )* method. In the case of an Array, elements can only be inserted with the help of an assignment operator.

For instance,

```
Integer addarrayobject[ ] = new Integer[8];
addarrayobject[0]= new Integer(3);
```

This command adds a new object to the specified array object. This is one of the most important differences between Array and ArrayList.

8. Multi-dimensional:

An Array can be multi-dimensional. But an ArrayList always has to be single dimensional.

Here is an instance of a multidimensional array:

```
Integer addarrayobject[ ][ ] = new Integer[5][6];
addarrayobject [0][0]= new Integer(8)
```

Similarities Between ArrayList and Array in Java

Add and get method

The performance levels of both are the same in this case. Both operations run in constant time.

Duplicate elements

Both Array and ArrayList contain duplicate elements.

Null Values

Both objects can contain null values. They use an index for making references to their elements.

Unordered

Array and ArrayList fail to guarantee ordered elements.