Difference between ArrayList<E> and array

We have learnt [ArrayList<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-5list-interface.html) as a part of Java [Collections](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-1.html).

There are few common disadvantages of arrays over ArrayList<E> such as

Arrays are of fixed length. This length cannot be changed once array is created.

As a result of fixed length we cannot add more elements in array then the capacity.

Also the memory is allocated in array once it is created. Even if array is empty the memory is allocated.

Let us now deep into the difference of ArrayList<E> and arrays.

The major difference between the two is that arrays are fixed length data structure. But, ArrayList is variable length, which means ArrayList can grow or shrink its size dynamically. ArrayList is backed by an array. ArrayList can roughly increment its size by 1.5 times. Incrementing size requires creating new array and copying contents of old array to new array which can be time consuming operation.

Another difference is that while creating array you need to specify its size or insert elements into it to make sure size is specified. But for ArrayList that is optional. Initial size is taken care by ArrayList constructor. Initial size is 10 of ArrayList.

In terms of length, Array will always give you one integer number. We use length attribute to get length of array.

**int**[] a = **new** **int**[3];

**int** length = a.length;

In ArrayList we use size method to get the total size of ArrayList.

List<Integer> list=**new** ArrayList<Integer>();

**int** size=list.size();

Arrays allow to insert primitive type of data like int, float, double, byte, short, long. ArrayList does not allow this. Through it might seem that we can insert int type into ArrayList’s add(E) method but it is autoboxed to wrapper type i.e Integer.

|  |  |
| --- | --- |
| Primitive type | Wrapper type |
| Byte | Byte |
| Double | Double |
| Float | Float |
| Int | Integer |
| Long | Long |
| Short | short |

Generics cannot be used to store the data. Array knows which type it can store.

**int**[] a = **new** **int**[3];

String[] str = { "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat", "Sun" };

In ArrayList we can use generics for type safety. Below list definitions are legal.

List<? **extends** Number> Ilist=**new** ArrayList<Integer>();

List<? **extends** Number> Llist=**new** ArrayList<Long>();

List<? **extends** Number> Dlist=**new** ArrayList<Double>();

List<? **extends** Number> Blist=**new** ArrayList<Byte>();

List<? **extends** Number> Slist=**new** ArrayList<Short>();

List<? **extends** Number> Flist=**new** ArrayList<Float>();

Insertion in array can be done by assignment operator “=”.

**int**[] a = **new** **int**[3];

a[0]=100;

a[1]=200;

a[2]=300;

But for ArrayList we use add(E) method to insert values in ArrayList.

List<Integer> list=**new** ArrayList<Integer>();

list.add(100);

list.add(200);

list.add(300);

There are 2 ways to traverse over arrays.

**int**[] a = {1,2,3};

**for** (**int** i = 0; i < a.length; i++) {

System.***out***.println(a[i]);

}

**for**(**int** temp:a){

System.***out***.println(temp);

}

There are several different ways to [traverse over ArrayList](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-6iterating-over.html). It can be done using random access to elements.Iterator<E>, enhanced for loop, ListIterator<E>(forward and backward direction), forEach in Java 8. Click [here](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-6iterating-over.html) to learn ways to traverse over ArrayList.

That’s all on difference between ArrayList and arrays.