ArrayList<Type> obj= new ArrayList<Type>();

or you can omit Type on right side, ie,

ArrayList<Type> obj= new ArrayList<>(); <> is called Diamond Syntax.

C++ note :

-Vectors in c++ are similar to ArrayList in Java, moreover C++ vector template overloads [] so it must

use explicit call while java doesn't overload [].

-Vectors are call by value, i.e., a=b means a new vector is created of same length and element value while

in java they are referenced to one another, so any changes in one reflects in another.

ArrayList<int> obj= new ArrayList<>(100) - initial size

obj.add(100); adding

obj.size(); return size

obj.ensureCapacity(int capacity); fixed size given

obj.trimToSize() -> storage capacity to current size of array

rarely used and shouldn't be used, use only when sure about it.

obj.set(i, 10); sets ith value with 10. But to use this there has to be a value before. So add() before set()

obj.add(i,e); sometimes adding element in middle of the array.

obj.get(i); returns value at i

obj.remove(i); removes value at i and left shifts remaining element.

Trick for flexible growth and convenient element access->

ArrayList<X> list = new ArrayList<>(); //First make a Flexible list

while(...)

{ x=...

list.add(x);

}

X[] a= new X[list.size()]; //Then convert it into an array for easy element access.

list.toArray(a);

**@author** Adarsh