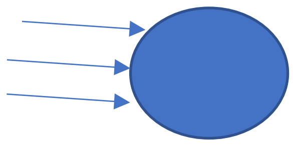
Array List
Growable
Duplicate allowed
Default capacity is 10
Insertion order preserved
Heterogenous object are allowed
Null insertion is possible
Random access Interface
For frequent retrieval array list is best choice
If thousands of element in array list and we need to add an element in middle then we (java or jvm) need to shift all element after it, means for insertion / deletion then AL is worst choice

New capacity = current capacity x (3/2) +1 = 16 (10 become 16)
Program:
```
package CollectiionProg;

import java.util.ArrayList;
import java.util.Collections;

public class ArrayListProg {

public static void main(String[] args) {

ArrayList al1 = new ArrayList();      //default capacity = 10
ArrayList al2 = new ArrayList(1000);  //customize capacity = 1000
ArrayList al3 = new ArrayList(al2);   //equivalent collection from existing

System.out.println("Is AL1 is empty = " + al1.isEmpty());
System.out.println("Size = " + al1.size());

al1.add(10);
al1.add("Harry");
al1.add("Ron");
al1.add(12222.222);
al1.add('A');
al1.add(null);

System.out.println("elements in AL1 = " + al1);
System.out.println("Is AL1 is empty = " + al1.isEmpty());
System.out.println("Size = " + al1.size());

for(int i=0;i<=10;i++)
{
al1.add(i);
}
System.out.println("elements in AL1 = " + al1);

al1.add(3, "Albus");
System.out.println("elements in AL1 = " + al1);

//                              al1.addAll(5, al1);
//                              System.out.println("elements in AL1 = " + al1);

System.out.println("Is AL1 contain A = " + al1.contains('A'));
System.out.println("Value at index 9 = " + al1.get(9));    //2
System.out.println("Index of A = " + al1.indexOf('A'));
System.out.println("First Index of 10 = " + al1.indexOf(10));
System.out.println("Last Index of 10 = " + al1.lastIndexOf(10));

al1.remove(6);
System.out.println("elements in AL1 = " + al1);
al1.set(1, "Harry Pooter");
System.out.println("elements in AL1 = " + al1);

for(int i=0;i<al1.size();i++)
{
System.out.println(al1.get(i));
}
System.out.println("-------------");
for(int i=al1.size()-1;i>=0;i--)
{
System.out.println(al1.get(i));
}


System.out.println("---------------------------------------------");
System.out.println("elements in AL1 = " + al1);
Collections.reverse(al1);
System.out.println("elements in AL1 = " + al1);
//                              Collections.sort(al1);

al2.add(20);
al2.add(2);
al2.add(1);
al2.add(22);
al2.add(100);
al2.add(5);
System.out.println("elements in AL2 = " + al2);
Collections.sort(al2);
System.out.println("elements in AL2 = " + al2);
//10 -> 16 ->  22 -> 30 -> 42
}

}
```

Vector
Vector is legacy class in collection (introduce first)
Growable
Default capacity is 10
Data structure doubly
Duplicate are allowed
Insertion order is preserved
Null insertion is possible
Heterogenous object allowed
Best for retrieval choice
Random access Interface



New capacity = 2 x current capacity
Program:
```
package CollectiionProg;

import java.util.Vector;

public class VectorProg {

public static void main(String[] args) {

Vector v = new Vector();
Vector v1 = new Vector(100);

System.out.println(v.capacity());

for(int i=0;i<10;i++)
v.add(i);

System.out.println(v);
v.add(100);
System.out.println(v);
System.out.println(v.capacity());

for(int i=0;i<100;i++)
v1.add(i);
System.out.println(v1);
System.out.println(v1.capacity());  //100
v1.add(1000);
System.out.println(v1.capacity());  //200

//Puja   Rohidas

//Sameer = Sam
}

}
```