**Collection: Objects**

Array: Objects, **fixed size**. Index based: 0-> size-1. Elements of same datatype.

Int[] arr= new int[4];

arr[4]= 43;

List/set/map

**Wrapper class:**

Primitive datatypes( byte, short, int, long, char, float, double, boolean)

**Wrapper class:** Byte, Short, Integer, Long, Character, Float, Double, Boolean (Immutable)

1. In collections, u cannot use primitive datatypes.
2. Methods, which u can use for basic operations
   1. String -> int Integer.paseInt(str);

Array:

Employee[] arr= new Employee[14];//initialized

Arr= new Employee[15];

**Collection interface: (java.util) [ctrl+shift+T]**

add(Elem e), remove(Obj o), iterator(), size(), isEmpty()

**Interfaces extend Collection interface: List, Set**

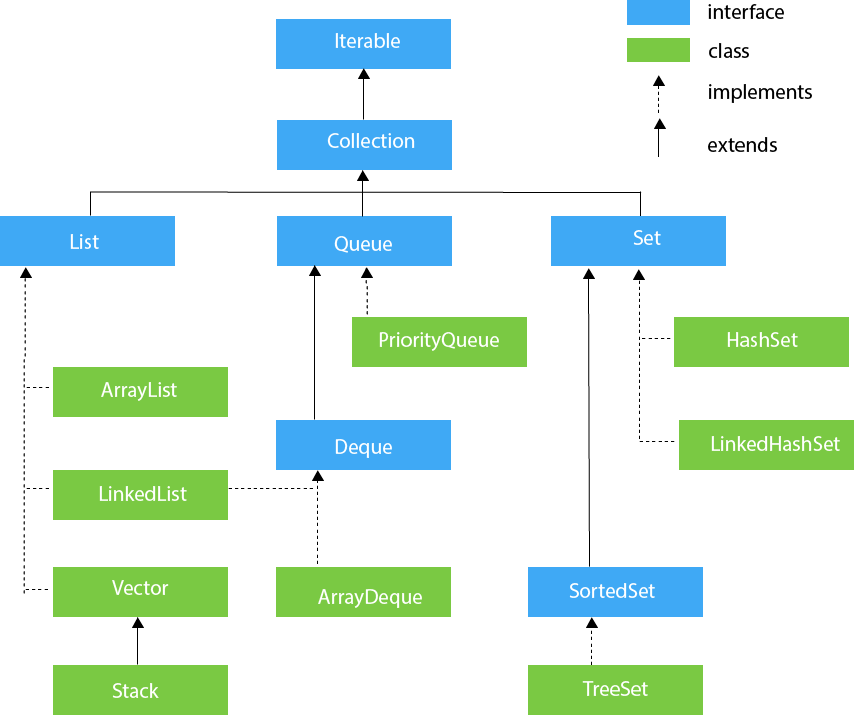
**Ctrl+T: hierarchy of the class/interface**

**List: preserves insertion order, allows duplicates**

**Set: unique elements, does not allow duplicates.**

**Map: Does not extends your Collection interface but it is also a collection api (java.util)**

**Key -value pair (search)**



**Collection: dynamic in size**

**ArrayList: It also allow duplicates, preserves insertion order, add/remove/size/iterator/isEmpty…**

**package** com.java;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.ListIterator;

**public** **class** ListDemo {

**public** **static** **void** main(String[] args) {

/\*int[] arr= new int[4];

arr[0]= 45.3f;\*/ //compile time check

//Generics: compile time check

List<Float> list= **new** ArrayList<Float>(); //add any datatype [array of size10]

list.add(45.3f);//index 0 list.remove(0); //list.size();

list.add(32.5f);//index 1

list.add(76.4f);//index 2

list.add(76.4f);//index 3

list.get(4);//ArrayIndexOutOfBoundsException

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

System.***out***.println(list.size());

System.***out***.println(list.get(1));

list.remove(Float.*valueOf*(45.3f));

//remove is an overloaded : int index, Object o

System.***out***.println(list.size());

//List is index based:

**int** no= list.size();

**for**(**int** i=0; i<no; i++) {

list.set(i, list.get(i)+1);

list.remove(i);

}

//List iterator is a fail-fast iterator

**for**(**float** i: list) {

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

list.remove(0);//If we iterate over the list using a for-each loop or iterator;

//and we also modify the list, it immediately throws a ConcurrentModificationException

}

Iterator<Float> it=list.iterator();

**int** index=0;

**while**(it.hasNext()) {

Float f= it.next();

//list.set(index++, f+1);

it.remove();//this will remove the current element whichever ur iterator is pointing to

ListIterator<Float> li=list.listIterator();

**while**(li.hasNext()) {

System.***out***.println(li.next());

**if**(list.size()==1) {

li.add(43f);

}

}

**while**(li.hasPrevious()) {

System.***out***.println(li.previous());

}

}

/\*for(float e: list) {

}\*/

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

List<Integer> list1= **new** ArrayList<Integer>();

list1.add(1);

list1.remove(Integer.*valueOf*(1));

}

}

**Internally uses array only.**

class A{

}

class B extends A{

}

A a = new B();

List<A> list= new ArrayList<>();//diamond operator

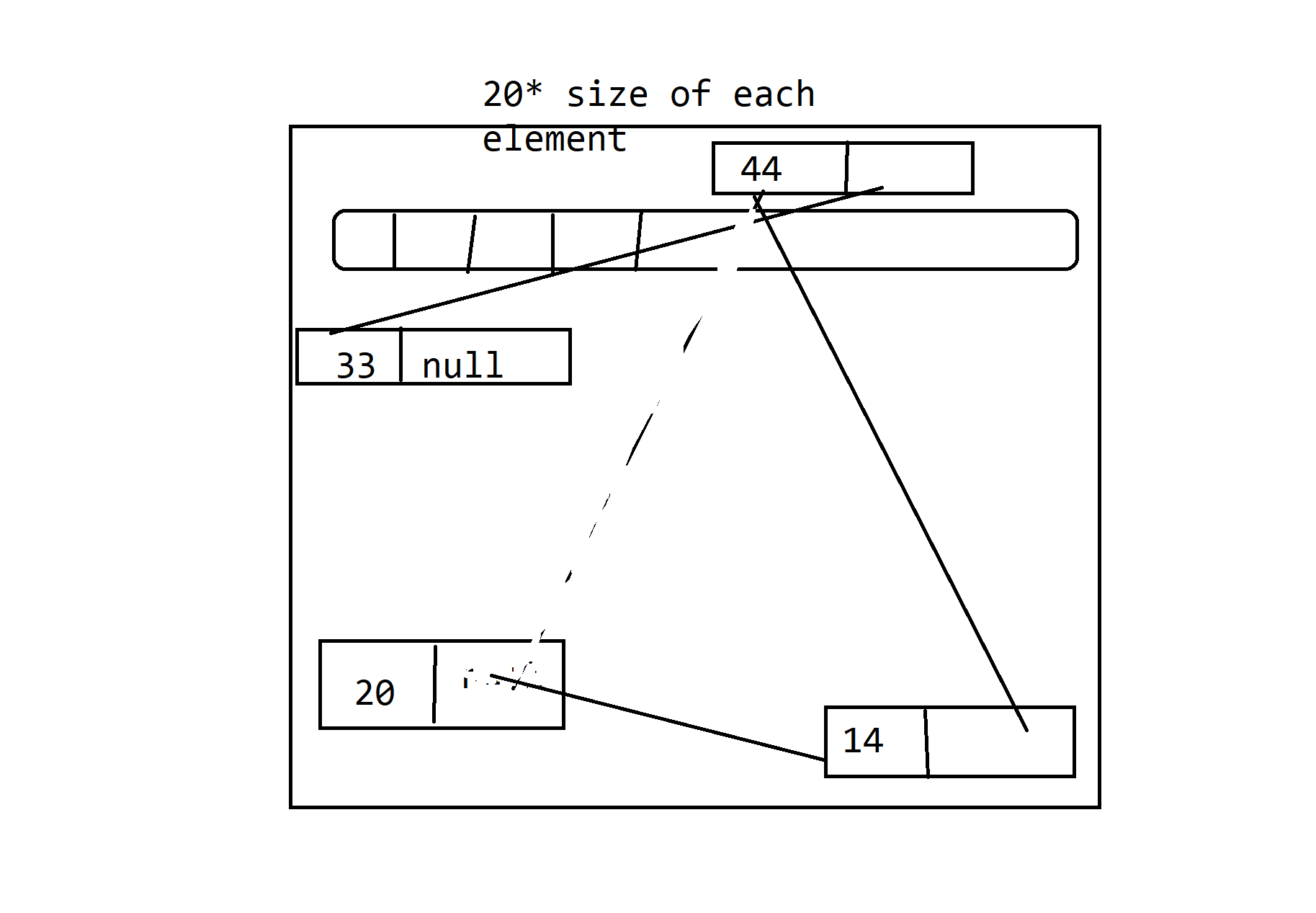
List list= new ArrayList<A>();//generics works at compile time

list.add(1); //fine

list.add("hello");//fine.

Generics is not working in this case.

**LinkedList: Adding / removing an element is faster in LinkedList as compared to ArrayList**

****

**public** **static** **void** main(String[] args) {

List<Integer> aList= **new** ArrayList<>(20);//10

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

list.add(20);

list.add(44);

list.add(11);

list.add(1, 14);

}

**Generics:**

**Collections class: utility class: static methods**

Class MyList{

List list= new ArrayList();

void add(Object o){

list.add(o);

}

void size(){

Sysout(list.size());

}

void remove(Object o){

list.remove(o);

}

}

//No type check at the compile time. I can add integers and try to remove string. Runtime error.

MyList obj= new LyList();

Obj.add(“apple”);

Obj.add(“grapes”);

Obj.remove(“apple”);

Sysout(obj.size());

List<String> list= new ArrayList()

list.add(“apple”);

list.add(“grapes”);

Obj.remove(“apple”);

Sysout(obj.size());

MyList obj1= new MyList();

Obj1.add(11);

Obj1.remove(11);

Generic class: When u create the object of this class, tell it it should be of what type.

Class MyList<T>{

List<T> list= new ArrayList<>();

void add(T obj){

list.add(obj);

}

void remove(T obj){

list.remove(obj);

}

void size(){

Sysout(list.size();

}

}

MyList<String> list= new MyList<>();

List.add(“apple”);

List.add(1);//compile time error {generics}

MyList<Integer> list1= new MyList<>();

List1.add(12);

List1.add(“jfdhf”); //error