Generics
Bounded and unbounded
Comparable comparator
Iterator

```
interface Collection<T>{
void add(T element);
boolean remove();
}
```

```
Comparable<T>{
int compareTo(Employee obj);
}
```

```
Comparator<T>{
int compare(T o1, T o2);
}
```

```
class ArrayList<T> impl Coll{
T [] arr;

@Override
void add(T element){

}
```

```
class LinkedList<T>{
T [] arr;

void add(T element){
//
}
}
```

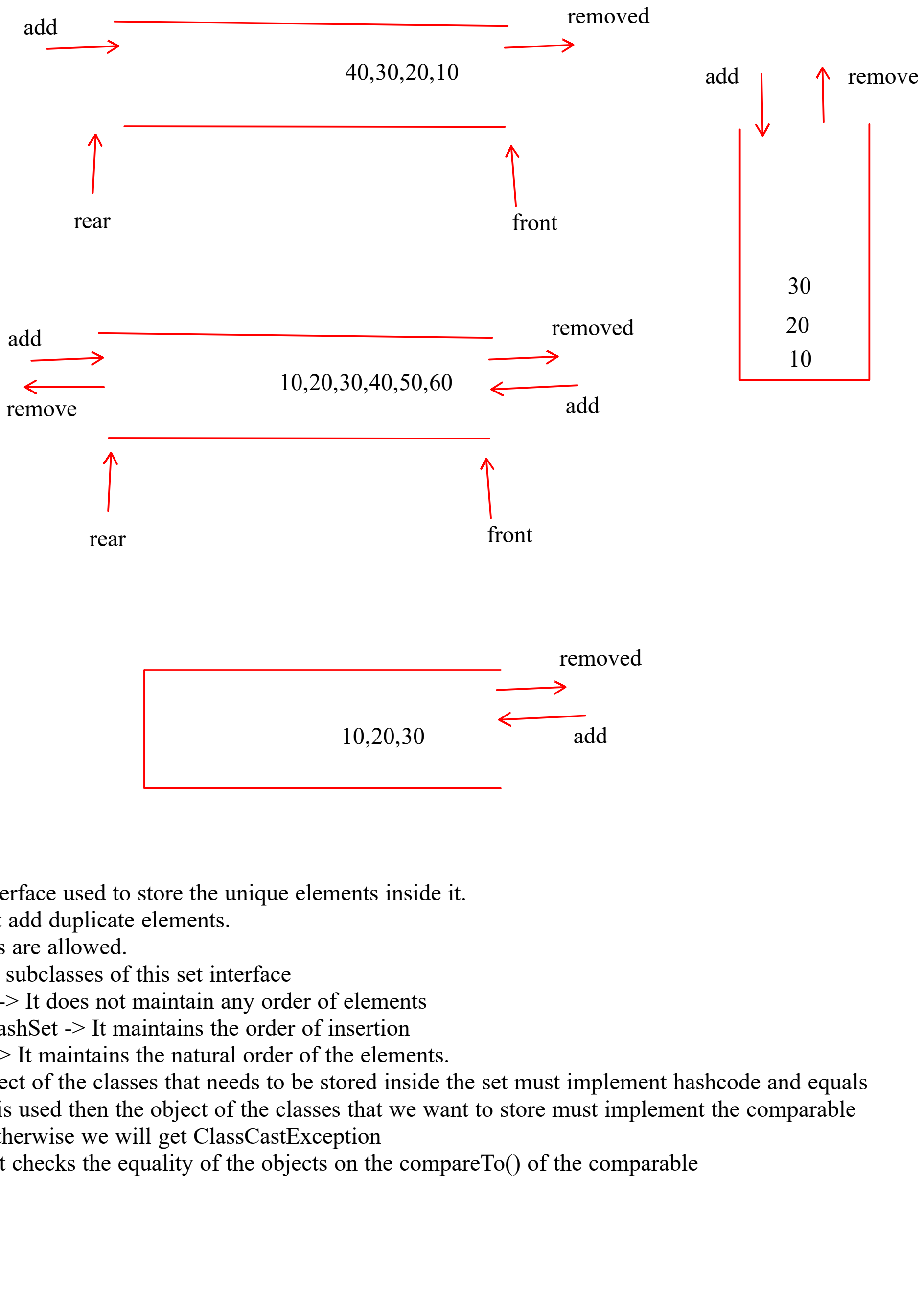
```
class Employee{

}

class Product implements Comparable<Product>,Comparator<Employee>{

int compareTo(Product p){

}

int compare(Employee o1, Employee o2){

}
}
```

```
LinkedList<Integer> l1= new LinkedList<Integer>;
LinkedList<String> l1= new LinkedList<String>;

LinkedList<Number> l1= new LinkedList<Integer>;// NOT OK
LinkedList<? extends Number> l1= new LinkedList<Integer>;// OK
```

```
Iterator<E> iterator()
{

}
```

```
List<Student> l1 = new ArrayList<Student>();

// case 1
Student s = new Student();
s.accept();
l1.add(s);

//case 2
Interator<Student> itr = l1.iterator();
while()


Student s = new Student(3);
int index = indexof(s);
sysout(l1.get(index));
```

```
class EmpNameComp implements Comparator<Employee>{

}
```



Set
- It is an interface used to store the unique elements inside it.
- We cannot add duplicate elements.
- null values are allowed.
- we have 3 subclasses of this set interface

1. HashSet -> It does not maintain any order of elements
2. LinkedHashSet -> It maintains the order of insertion
3. TreeSet -> It maintains the natural order of the elements.

- all the object of the classes that needs to be stored inside the set must implement hashcode and equals
- if Treeset is used then the object of the classes that we want to store must implement the comparable interface. otherwise we will get ClassCastException
- The treeset checks the equality of the objects on the compareTo() of the comparable