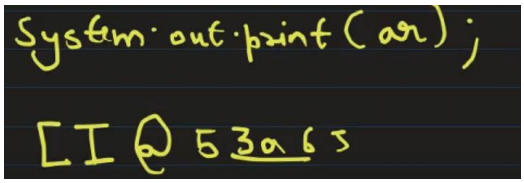
Introduction

Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.

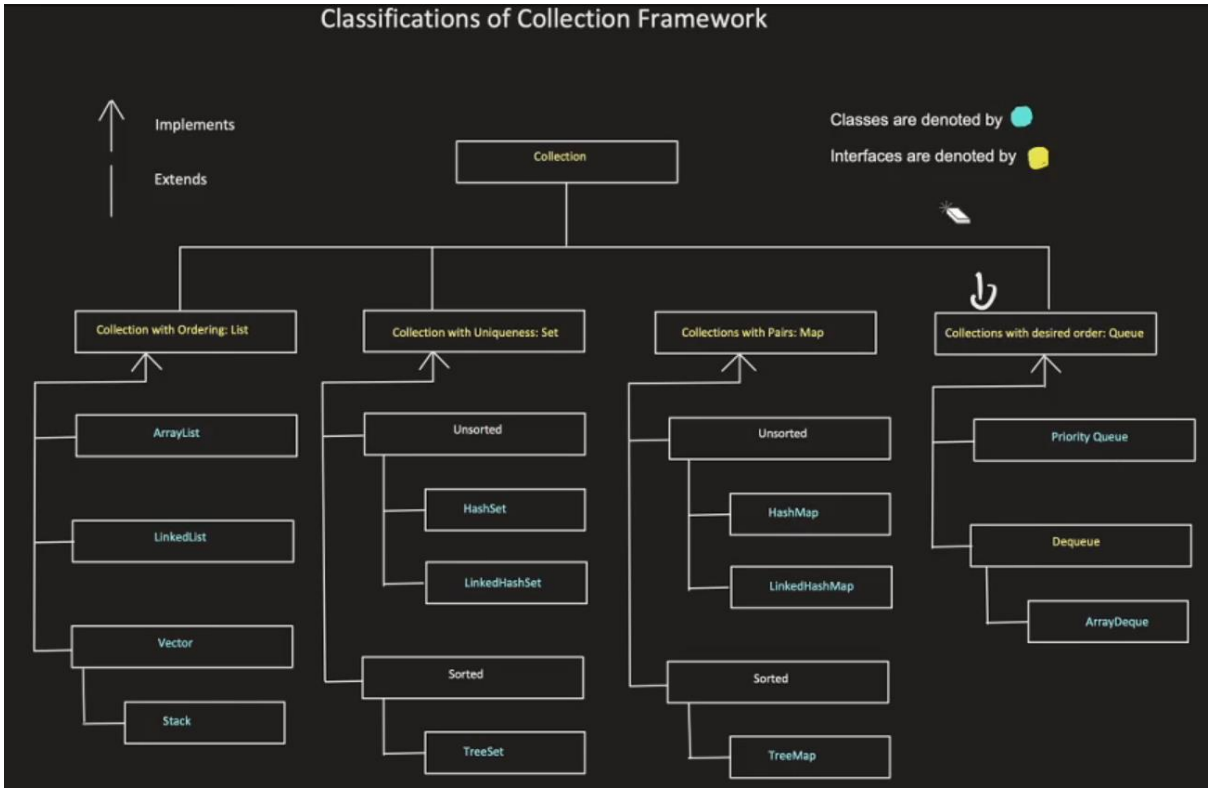
Advantages of Collection Framework

- Less Programming effort
- Immense classes and interfaces present in the Collection Framework
- Makes things convenient and quality

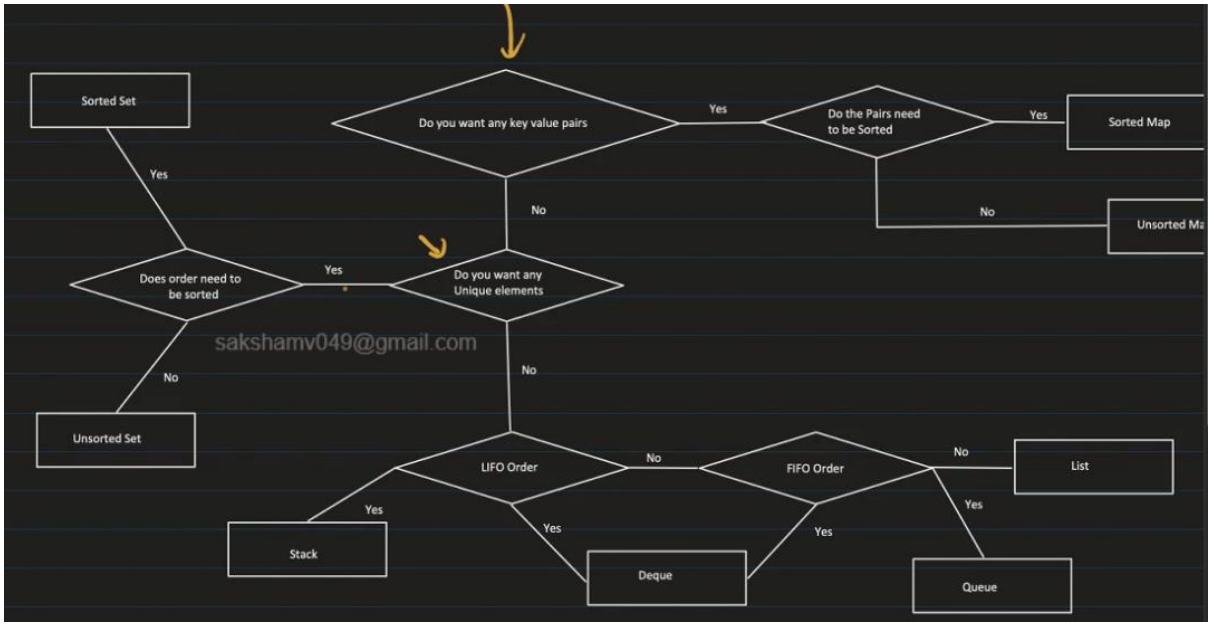
Array → Uniqueness(Add a particular thing) / Size issues(size not present at the time of starting)/ Same data type



Class ID → I here is because of Integer



Map doesn't extends Collection interface but is a part of Collection framework .



1. `sort(List<T> list)`: sorts the elements in a list in ascending order.
2. `reverse(List<T> list)`: reverses the order of the elements in a list.
3. `shuffle(List<T> list)`: randomly shuffles the order of the elements in a list.
4. `binarySearch(List<? extends Comparable<? super T>> list, T key)`: searches a sorted list for a specified element using the binary search algorithm.
5. `addAll(Collection<? super T> c, T... elements)`: adds one or more elements to a collection.
6. `frequency(Collection<?> c, Object o)`: returns the number of times a specified element appears in a collection.
7. `max(Collection<? extends T> coll)`: returns the maximum element in a collection, based on the natural ordering of the elements.
8. `min(Collection<? extends T> coll)`: returns the minimum element in a collection, based on the natural ordering of the elements.
9. `copy(List<? super T> dest, List<? extends T> src)`: copies the elements from one list to another.
10. `unmodifiableCollection(Collection<? extends T> c)`: returns an unmodifiable view of a collection, which prevents modification of the collection.

These are just a few examples of the many useful functions provided by the Collections class in Java.

size()	Gives the size / no. of elements in the Collection
isEmpty()	It is used to check whether the collection is empty or not. (Returns true or false)
add(element)	It is used to add element to the Collection
addAll(collection)	It is used to add elements of the argument collection into the collection
remove(element)	It is used to remove element from a Collection
removeAll(collection)	It is used remove elements of the argument collection from the collection
retainAll(collection)	It is used to remove elements from the collection not present in argument collection
contains(element)	It is used to check whether the collection contains argument element or not. (Returns true or false)
containsAll(collection)	It is used to check whether the collection contains all elements of argument collection or not
clear()	Removes all the elements from Collection

LIST

Sublist(form,to)[from,to)

Default Cap of ArrayList and CurrentCap →10

ArrayList →

Currentcap = prevcap+ prevcap/2

```
ar1.add(12);

System.out.println(ar.size());
System.out.println(ar.contains(1));

  System.out.println(ar.remove(1));
  System.out.println(ar.set(1,4));
System.out.println(ar.indexOf(2));

System.out.println(ar.lastIndexOf( o: 3));

System.out.println(ar.toString());

ar.addAll(ar1);

System.out.println(ar);

// removeAll()
//retainAll()

System.out.println(ar.get(1));
```

LIST

```
list.add(3);

System.out.println(list.contains(3));

LinkedList<Integer> list1 = (LinkedList<Integer>) list.clone();
System.out.println(list1);

System.out.println(list.size());
System.out.println(list.getFirst());
System.out.println(list.getLast());
list.addLast( e: 45);
list.addFirst( e: 41);

  System.out.println(list.removeFirstOccurrence(3));

System.out.println(list.removeLastOccurrence( o: 3));
System.out.println(list);
System.out.println(list.peek());
System.out.println(list.poll());
System.out.println(list.pollLast());
```

Vector →

Currentcap = prevcap*2

```
vector.add(2);

System.out.println(vector.size());
  System.out.println(vector.);

System.out.println(vector.contains(4));

System.out.println();

ListIterator<Integer> list= vector.listIterator();

while(list.hasNext()){
    System.out.println(list.next());
}

while(list.hasPrevious()){
    System.out.println(list.previous());
}

System.out.println(vector.);
```

Vector<String> vector = new Vector<>();

vector.addElement("Apple");

vector.addElement("Banana");

vector.addElement("Cherry");

String secondElement = vector.elementAt(1);

System.out.println(secondElement);

Set and HashSet

```
set.add(5);

  System.out.println(set.add(1));

System.out.println(set.size());
System.out.println(set.contains(4));
System.out.println(set.isEmpty());

Iterator<Integer> iterator= set.iterator();

while(iterator.hasNext()){
    System.out.println(iterator.next());
}
```

TreeSet

```
treeSet.add(10);


SortedSet set = treeSet.headSet( toElement: 2,  inclusive: true);
System.out.println(set);

SortedSet set1= treeSet.tailSet( fromElement: 3, inclusive: false);
System.out.println(set1);

System.out.println(treeSet.pollFirst());
```