Introduction to DSA

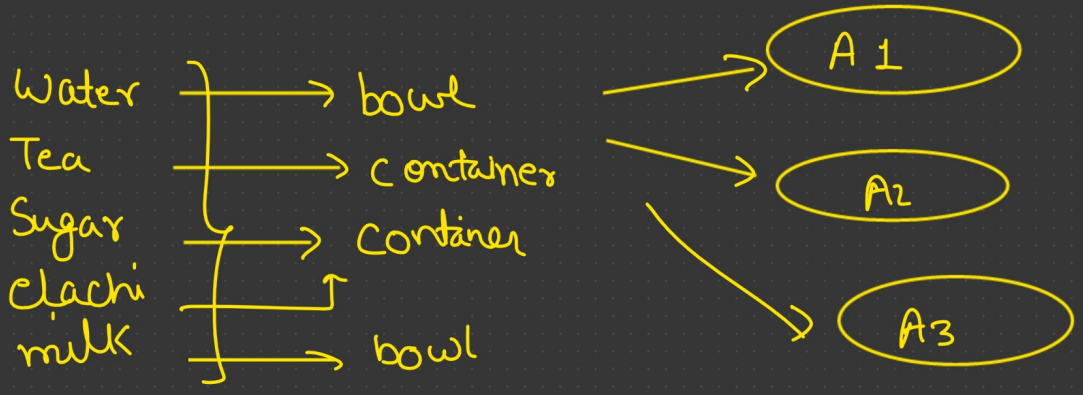
Data → information

Data Structure

Algorithm → Set of instruction to perform a specific task

Water → Bucket, Glass, Bottle

5+10



```
public Class Abc
{   public static void main (String args[])
    {
        System.out.println ("Hello");
    }
}
```

Arguments

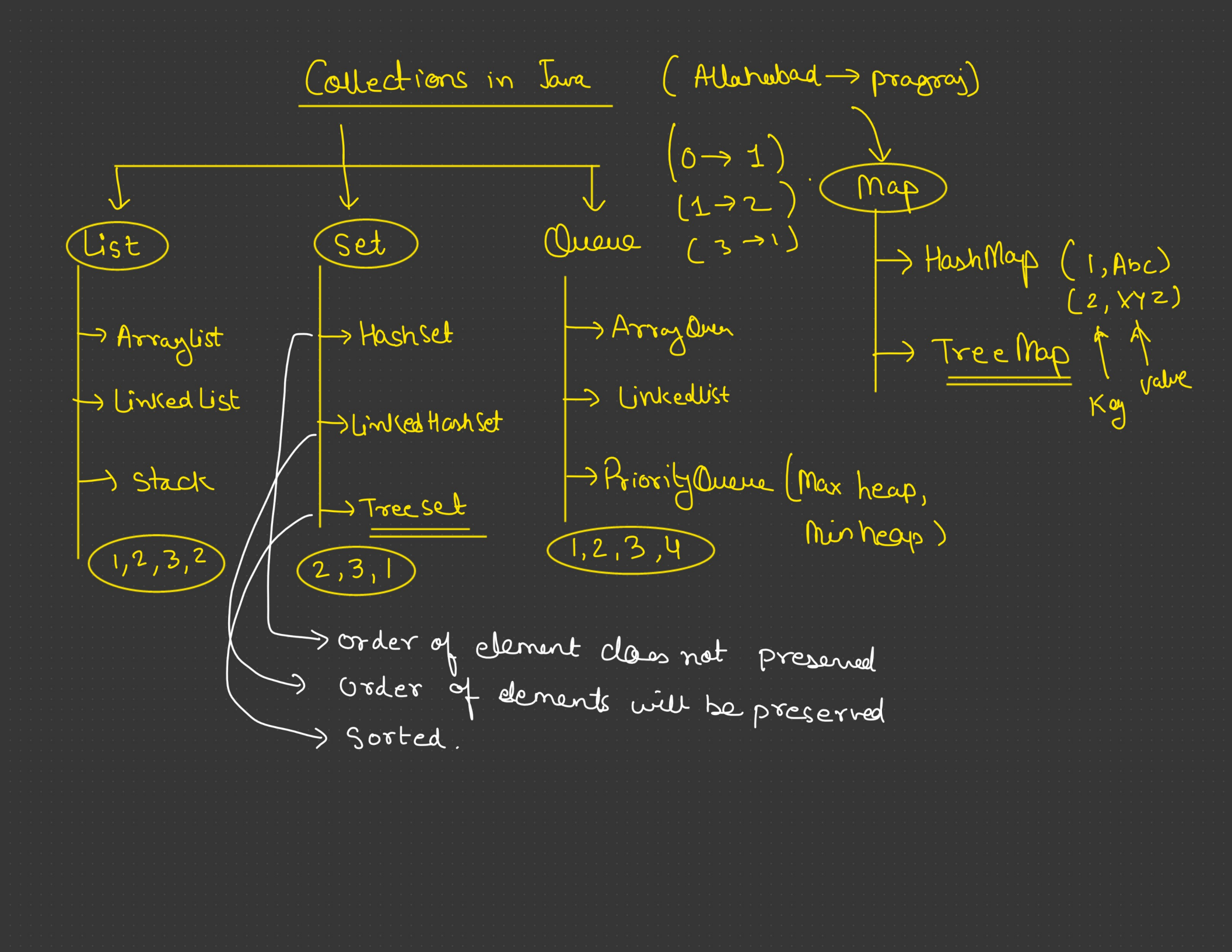
Abc.java

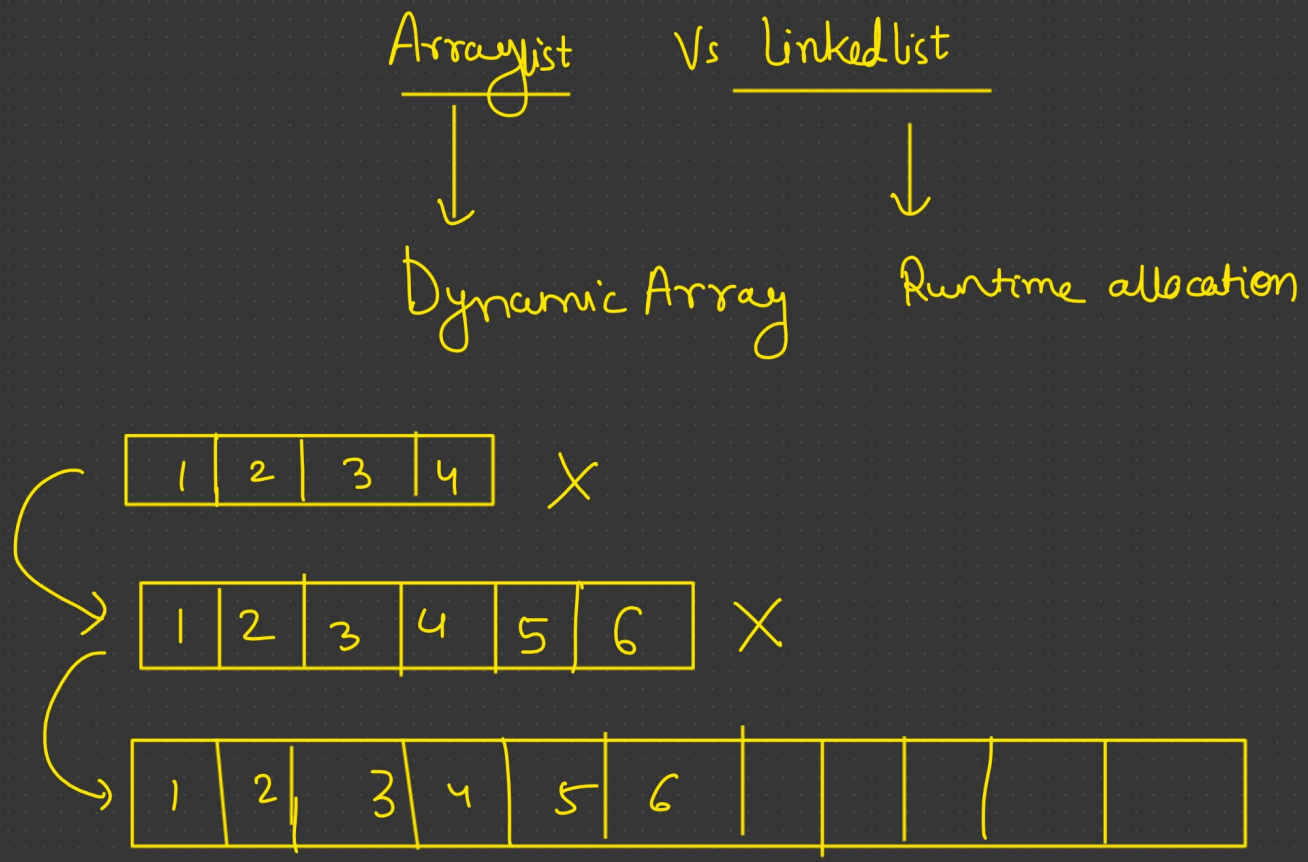
```
public class Abc
{


}
```

Data + Operation

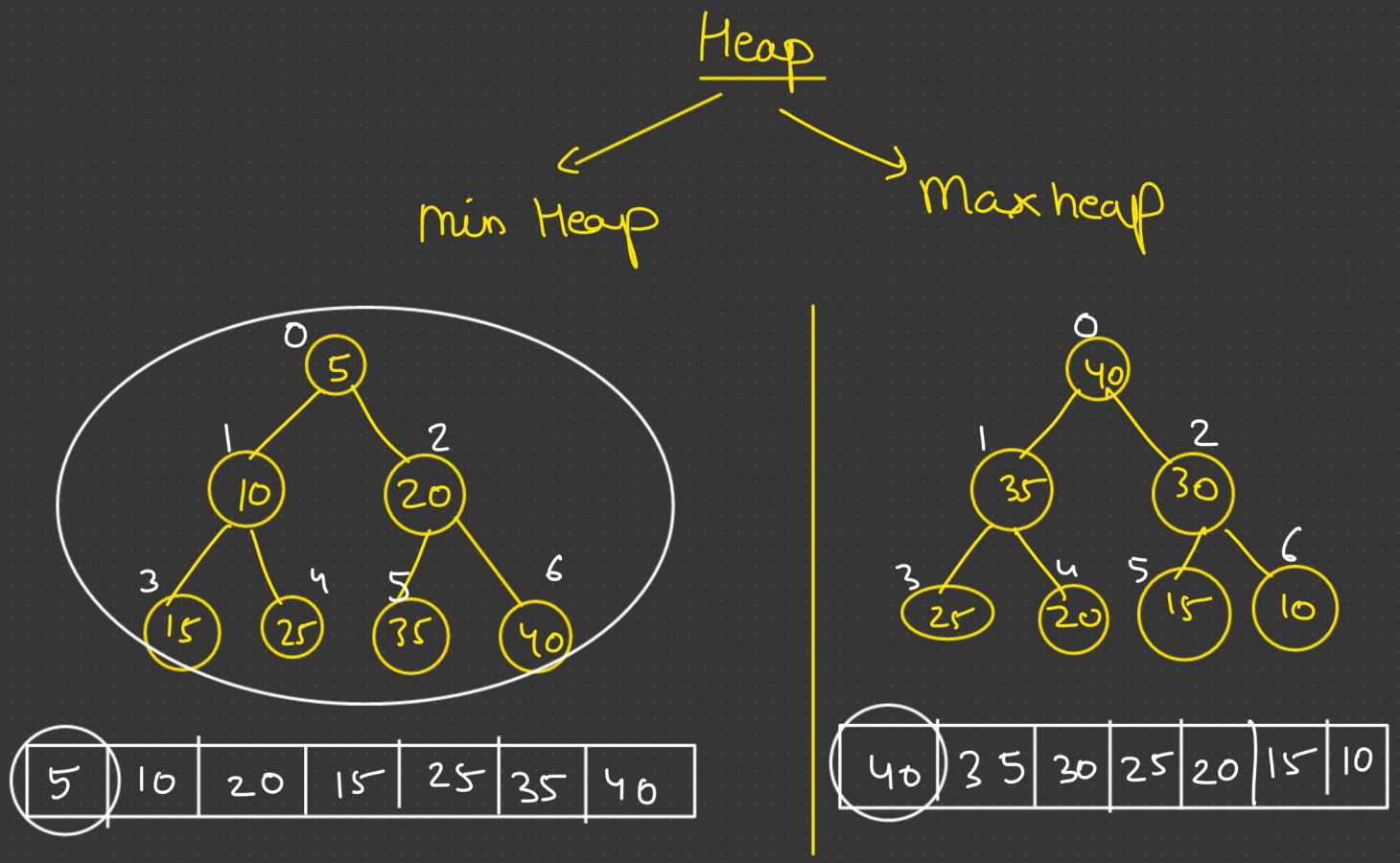
Class → [variables + add( ), sub( )]

object.add()
object.sub()

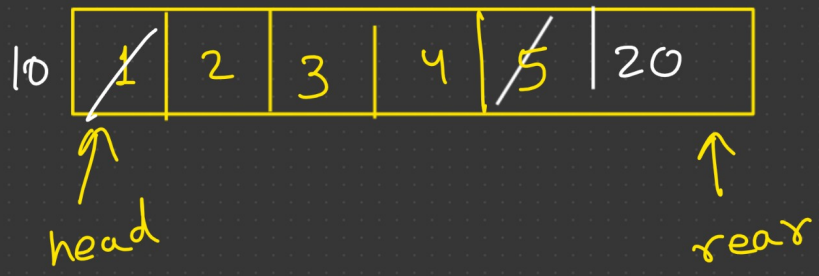

Collections in Java
(Allahabad → prayagraj)
(0 → 1)
(1 → 2)
(3 → 1)
List
Set
Queue
Map
ArrayList
LinkedList
Stack
1,2,3,2
HashSet
LinkedHashSet
TreeSet
2,3,1
ArrayQueue
LinkedList
PriorityQueue (Max heap, Min heap)
1,2,3,4
HashMap (1,Abc) (2,XYZ)
TreeMap
Key
value
order of element does not preserved
Order of elements will be preserved
Sorted.


ArrayList
Vs
LinkedList
Dynamic Array
Runtime allocation
1 2 3 4
X
1 2 3 4 5 6
X
1 2 3 4 5 6

toString();


Heap
Min Heap
Max heap
0
5
1
10
2
20
3
15
4
25
5
35
6
40
5 | 10 | 20 | 15 | 25 | 35 | 40
0
40
1
35
2
30
3
25
4
20
5
15
6
10
40 | 35 | 30 | 25 | 20 | 15 | 10

Double ended Queue:-



insertion & deletion allowed from both the end.

✓1) Collections

✓2) Hashcode funtion, equals function → Hashset & HashMap working

✓3) toString function

4) Comparator & Comparable.

✓5) Array class & collection class.

String s_1, s_2;

s_1.equals(s_2)

e_1.equals(e_2)

Object

equals function

Integer

Strings

(1) == (1)

Student $S_1 == S_2$ ✗

Employee $E_1 == E_2$ ✗

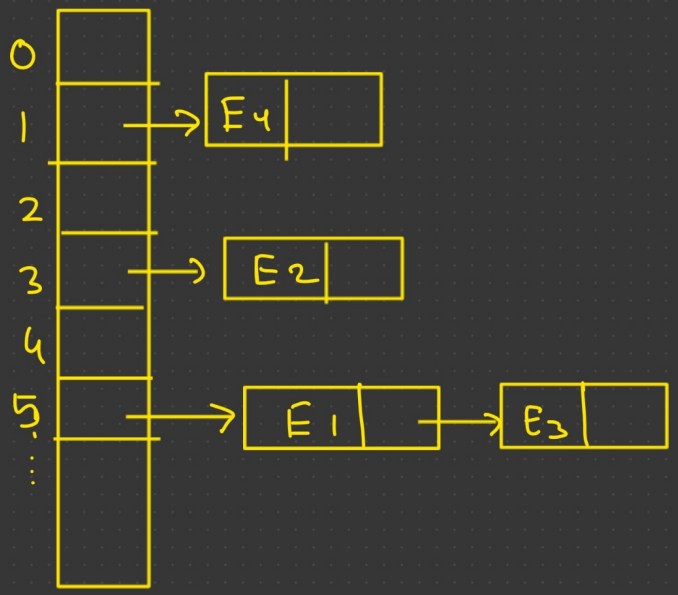
new Employee();

(1) abc → E_1 (1000) ≠ (1) abc → E_2 (2000)

1) equals() & hashcode()

Hashset:-

1) Unique
2) No fix order



hashcode(E1) ⟶ 5
hashCode(E2) ⟶ 3
hashCode(E3) ⟶ 5
-collision

e3.equals(e1) ⟶ True
⟶ false

hashcode(E4) → 5

Note:-

1) If two objects are same then their hashcode will be same.
2) If two objects hashcode is same it doesn't means that objects are also same. It can be verified by calling equals function.

toString() :—

```
Student s = new Student();
Arraylist<Student> a = new Arraylist<>();
a.add(s);
System.out.println(a);  ⟶  a.toString();
```

↓

address

Array class & Collection class :-

Student → static f1();

→ f2()

Student.f1(s);

Student s;

s.f2();

s1.equals(s2)

equals(s1, s2);

Arrays.sort(a); ⟶ primitive datatype → Quick Sort

Collections.sort(e); ⟶ Objects → Merge sort.

Arrays.toString(a); ⟶ it will Convert primitive array to string.

Comparator & Comparable

(1)

```
public class Emp implements Comparable<Emp>
{
    public int compareTo(Emp e1)
    {
        return this.id - e1.id;
    }
}
```

1 - 1 = 0

1 - 2 = -ve (1<2)

2 - 1 = +ve (2>1)

(2)

Comparator ⟶ compare(emp e1, emp e2) ⟶ id / name

Comparable ⟶ compareTo(e2); ⟶ id "or" name

↓

Collections.sort(a);

Collections.sort(a, new id());

Collections.sort(a, new name());