Interface – contract

Interface → Extend

```
I1 → Interface
public int compareTo(T o);
public int fun1();
public int fun2();
```

Class → Implement → class have to provide body for every method of interface

Class → Person { } → Implement → Comparable → compareTo

object → a →

object → b

}

* How to decide weightage parameter of 'a' and 'b'.

a.compareTo(b)

value – Integer

-ve → a is smaller

+ve → b is smaller

0 → a is equal to b.

```
public int compareTo(Person o){
    return this.age - o.age;
}
```

int → wt value

character → ASCII

String – lexicographically → Dictionary order

a < b →

a > b →

a == b =

a → factor from which we can decide weightage of class object

b

Decide for Person class →

1. a is greater if age is greater than b
2. a is smaller if age is smaller than b
3. if age is equal, Decision factor is Height

```
public int compareTo(Person other){
    if(this.age > other.age){
        return 1;
    } else if (this.age < other.age){
        return -1;
    } else{
        return this.ht - other.ht;
    }
}
```

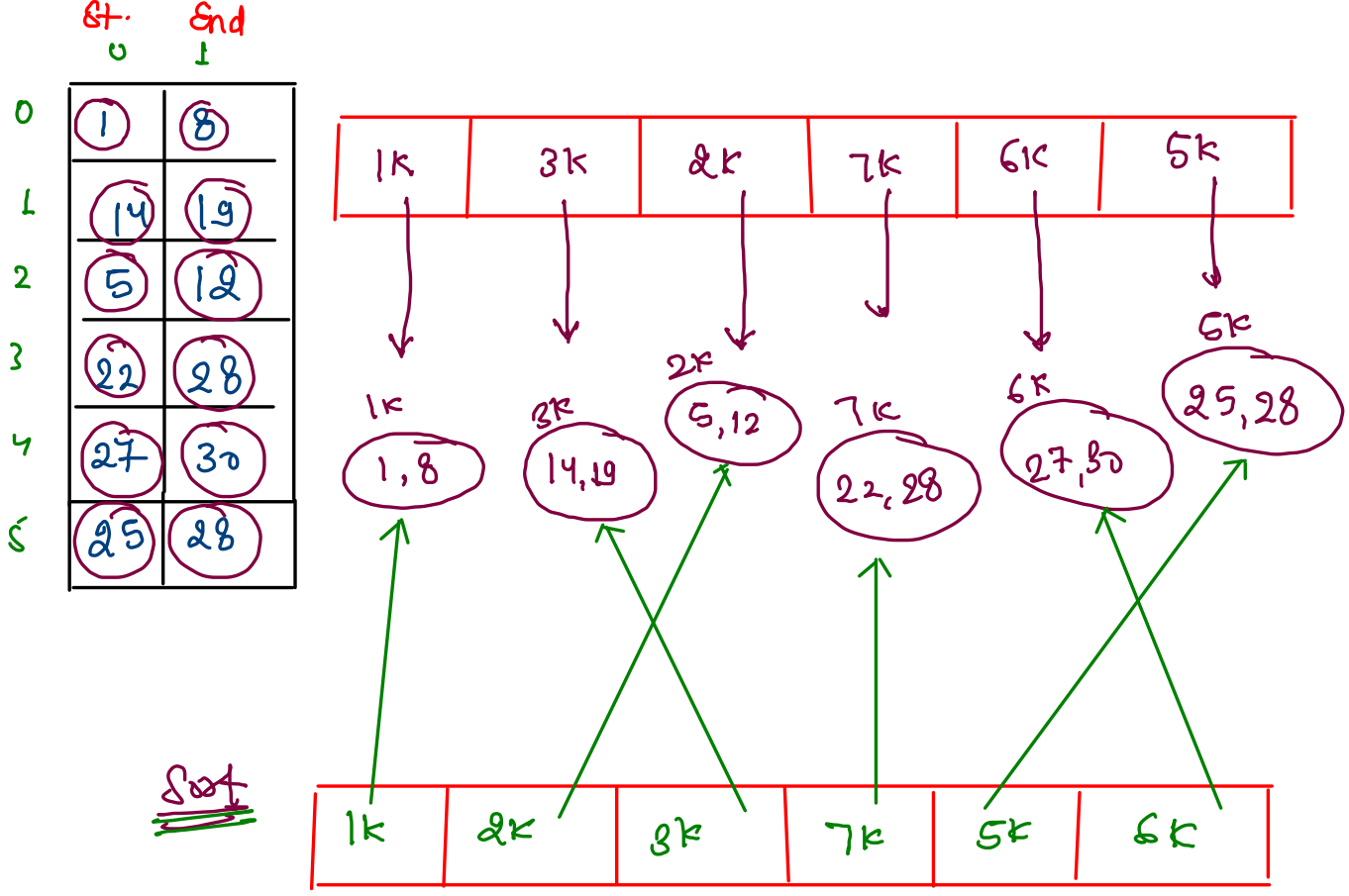
Interface

Class → implement

Comparable → method compareTo

working.

implementation




```java
Pair[] pairs = new Pair[arr.length];

for(int i = 0; i < pairs.length; i++) {
    int st = arr[i][0];
    int end = arr[i][1];

    pairs[i] = new Pair(st, end);
}

Arrays.sort(pairs);
// stack
```

Smallest Number following pattern:

pattern → i i d d

string

constraints → * pattern.length = 8

* Number can't repeat

1 – 9

1 3 2 5 4

i d i d ⇒ 13254

5 4 3 2 1

d d d d ⟶ 54321

Smallest possible Number

1 2 5 4 3 } smallest possible Number

i i d d

1 2 → 12

i

2 d 1 → 21

3 d 2 d 1 → 321

1 2 3 4

i i i → 1234

3 d 2 d 1 i 6 d 5 d 4 → 321654

1 2 → 12

i

1 2 3 → 123

i i

$^{2}d^{1}$ → 21

$^{3}d^{2}d^{1}$ → 321

d i d d i i d

2 1 | 5 4 3 | 6 | 8 7

length of pattern = n

count digit in res = n+1

pattern → d

→ i

why it is smallest

→ on most significant digit, we are trying to put least weightage digit according to pattern.

$^{3}d\ ^{2}d\ ^{1}i\ ^{6}d\ ^{5}d^{4}$

↳ 321654

$^{2}d^{1}i^{4}d^{3}i^{5}d^{6}$ → 214365

d d i d d i

3 2 1 6 5 4

out of the loop push → stack print

count = ~~1 2 3 4 5 6~~ 7

d → push
cont++;
i++;

i → push
cont++;
i++.
print stack
remove

d i d d i d

2 1 | 5 4 3 | 7 6

i i d d i

1 | 2 | 5 4 3 | 6 →

pattern → d d i i d i d d

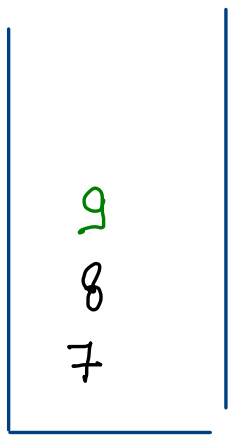
3 2 1 4 6 5 9 8 7

print out of the loop

count = ~~1 2 3 4 5 6 7 8~~ 9

d → push
count++;
i++;

i → push
count++
print stack
i++;



out of the loop

① → push count

② Stack print

capacity → 7

top of stack // peek → index

0	1	2	3	4	5	6
10	20	30	60	55	65	57

tos

pop() / peek() ← stack

push (10) underflow

push (20)

push (30)

peek() → 30

push (60)

push (55)

push (90)

push (75)

push (66) → stack overflow

size → tos ⇒ 7

pop() → 75

pop() → 90

push (55)

push(57);

pop(); → 57 data

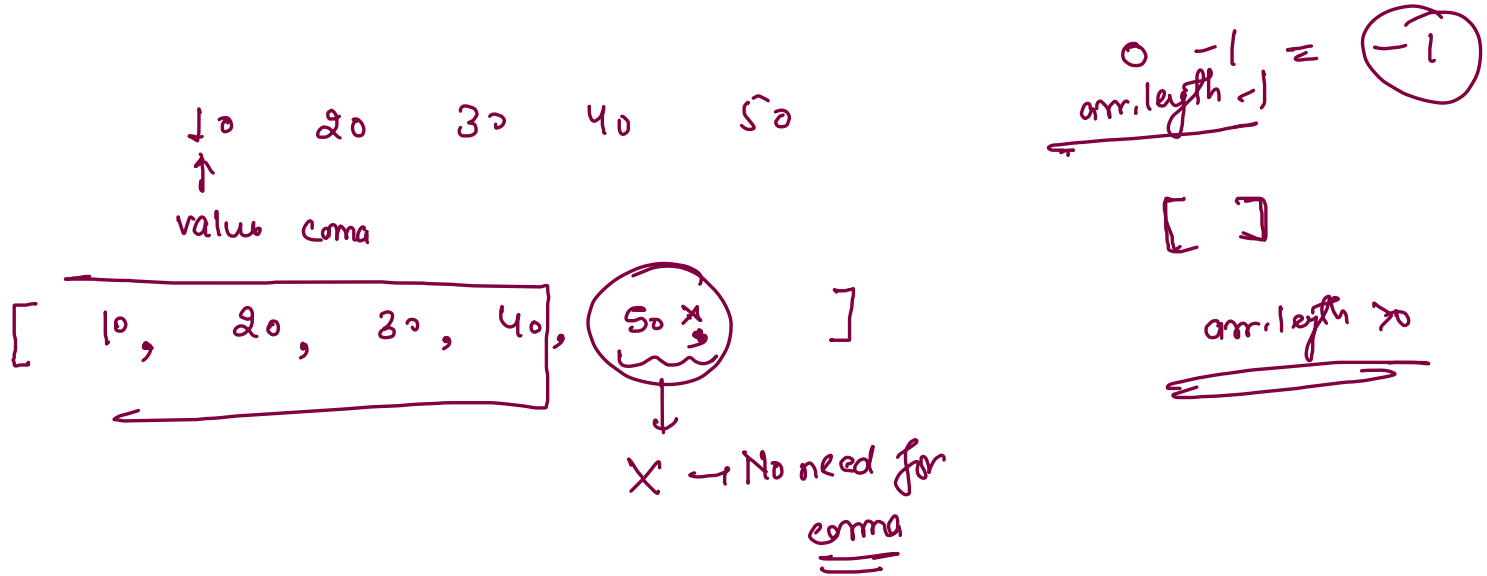
Sysо(57)

[10, 20, 30, 60, 55, 65, 75]

Stack

1. push
2. pop
3. peek
4. size
5. isEmpty

additional 6. Display



Dynamic Stack ⟶ ① Push ⟶ Stack overflow

↳ ① cap * 2

② Arrage all data in new array

③ Pointer Arrage