- Summary
- Today's content
  - ✓ 4 combinations of making objects using Base and child class
  - ✓ Overloading and overriding
  - ✓ final keyword
  - generics
  - Collections
- Ask as many doubts as possible, if I miss your doubt then please put it in questions tab.

- 4 combinations of Base and Child Class

✓ Vehicle v = new Vehicle ( )

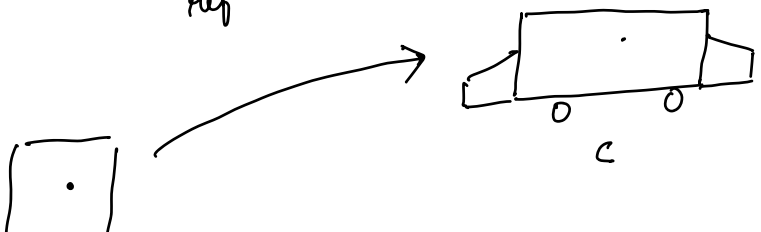
✓ Car c = new Car ( )

✓ Vehicle v = new Car ( ) → actual object creation.

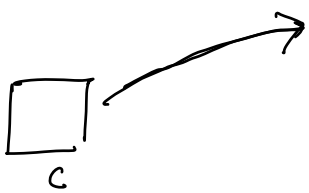
v → reference

✗ Car c = new Vehicle ( ) → actual

ref



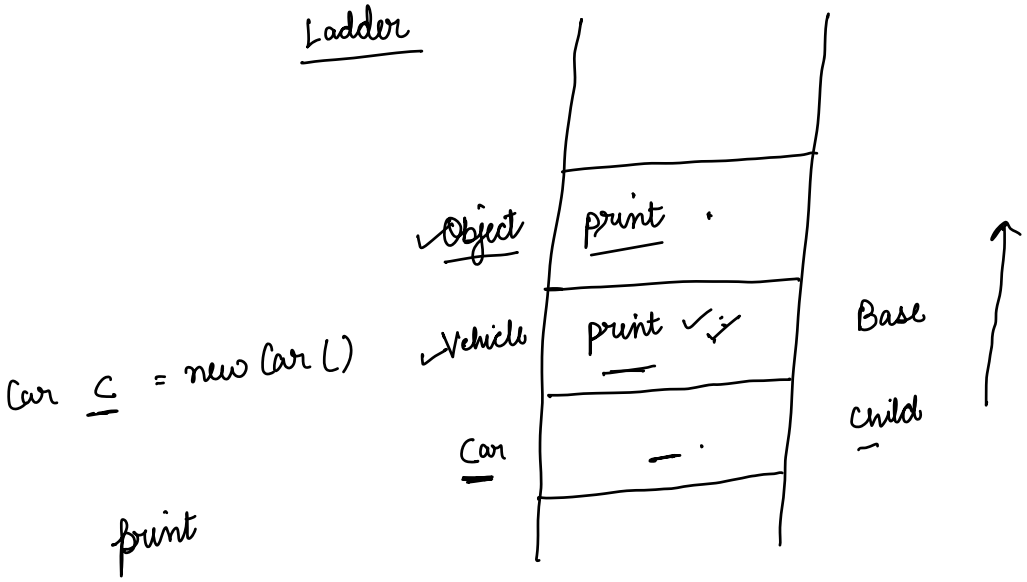
Vehicle → Car every car is a vehicle.

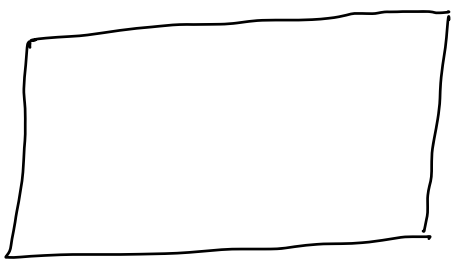


cycle
truck
Car
Scooter

✗ every vehicle is car

Ladder





B3
↑
B2
↑
Base!
↑
Child

- same package

private • (within class)
public ✓
def • ✓
pro ✓

Vehicle v1 = new Car()

1. Compile - reference
2. Run time

⇒ v1.print() ✓

ref → Vehicle

CTE ×

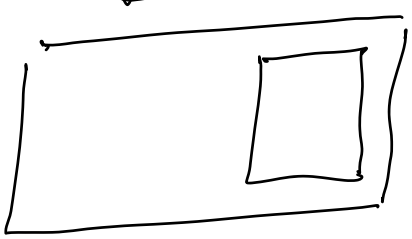
Vehicle

✓ getmaxspeed

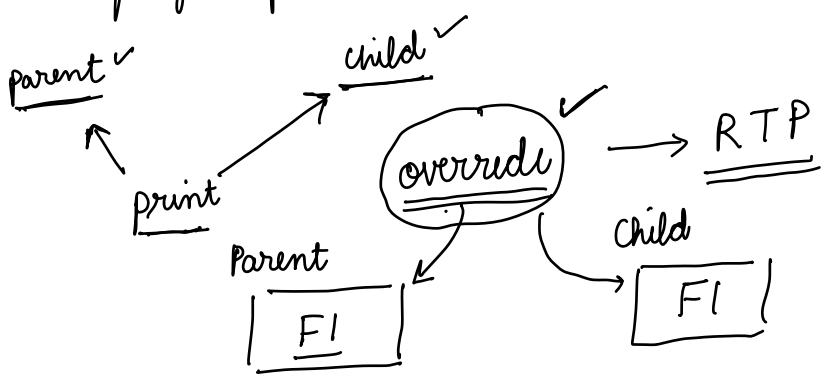
RTE ×

Car object

✓ getmaxspeed



Run time polymorphism



- RTP — when we get to know at runtime

same name, many forms

common methods.

Static methods

↓

override

static → Parent class

↓

polymorphism

static → overridden ✗

static → overloaded ✓

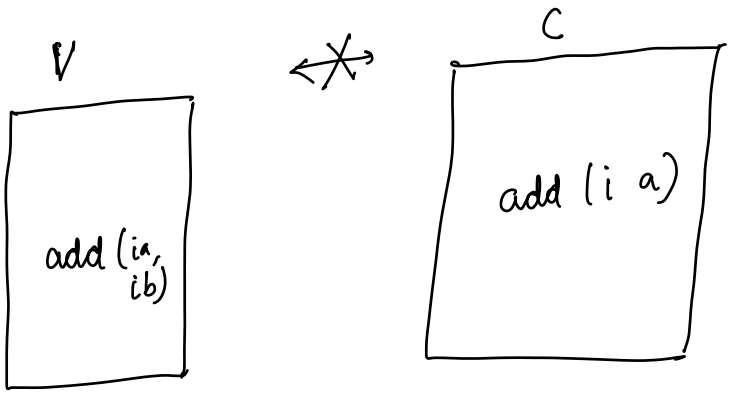
overriding – RTP – common P C

overloading – CTP – same name fnc in same class

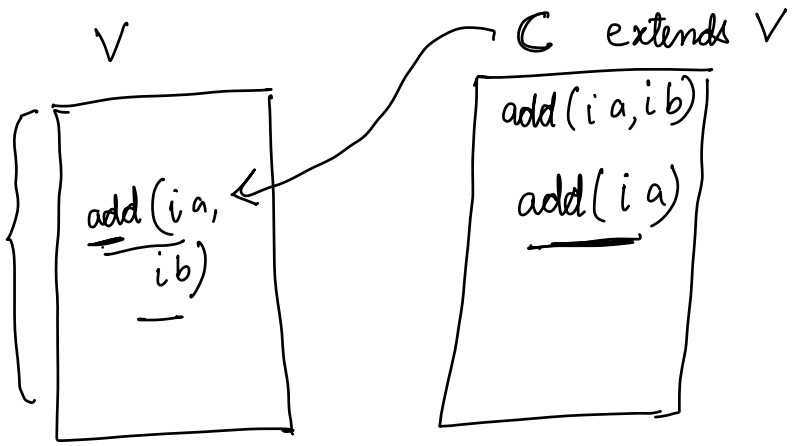
Parent
- add
- add

Child
- add
- add

Case 1



Case 2



P
P
d

Car c = new Car()

c.add(4) ✓

c.add(4,5) ✓

overloading

Vehicle v = new Car()

X. v.add(4)

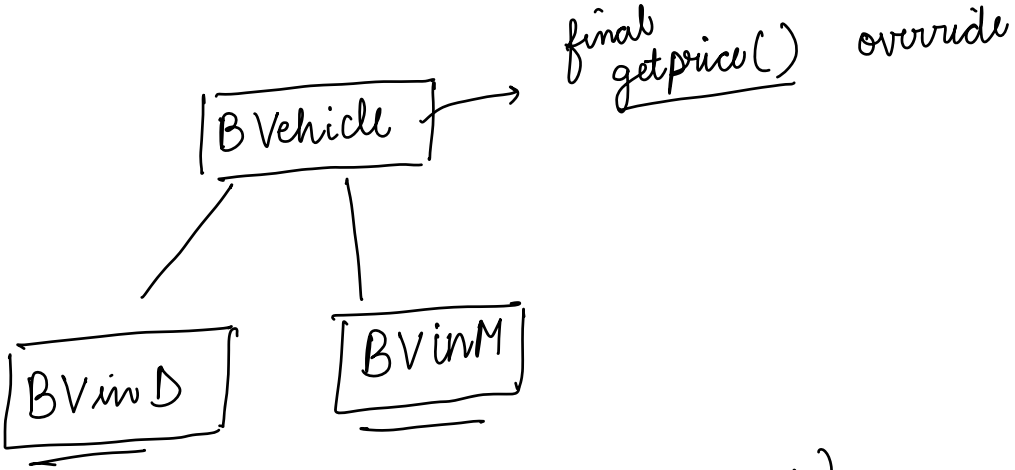
v.add(4,5)

~~overridding~~ X

~~overloading~~ X

final variable — initialized once

→ ① defining
② constructor



- String — final (cannot be extended)
  ↓
  To Do

- break : 10:36.

generics.
collections

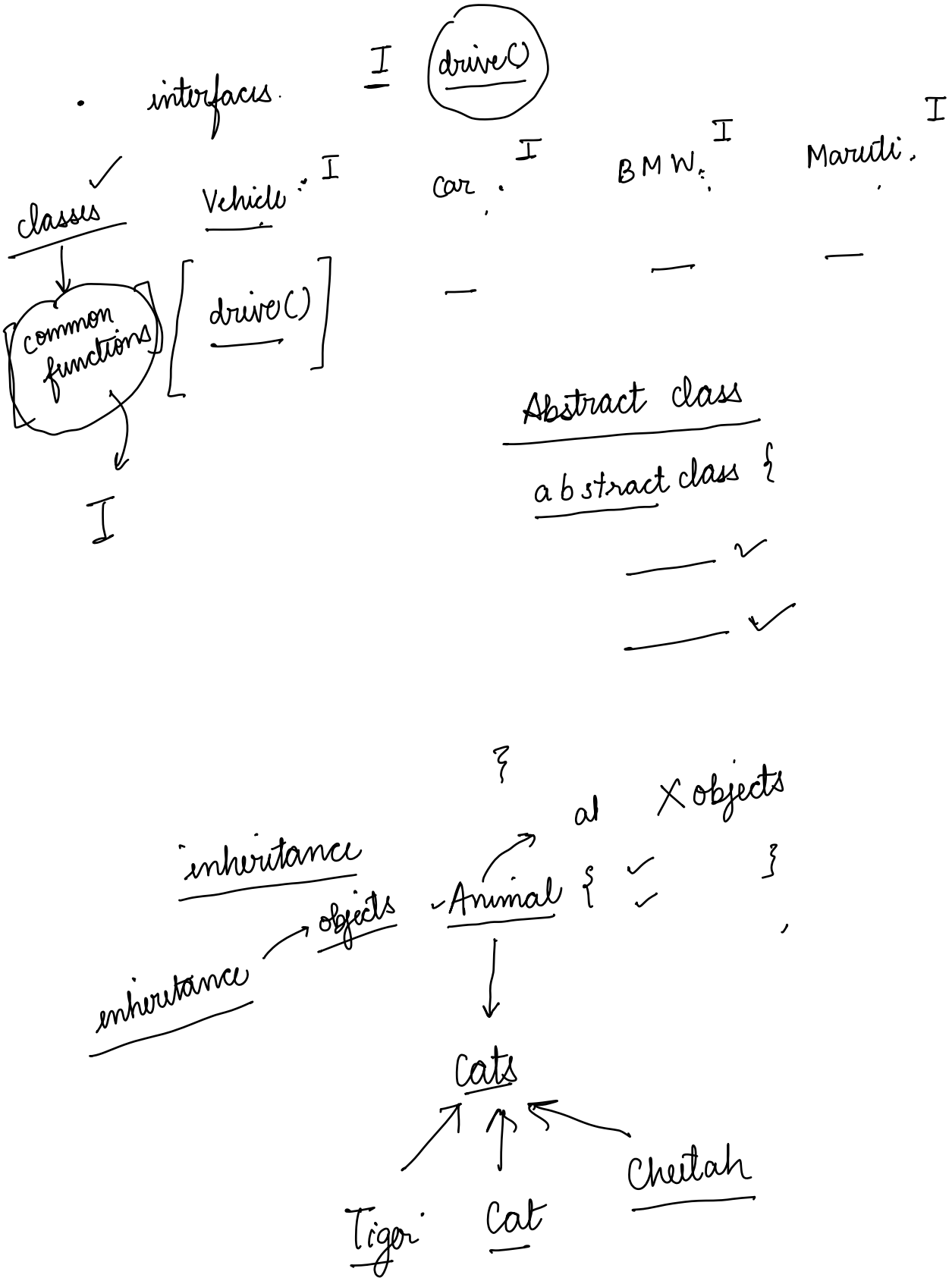
Collections :

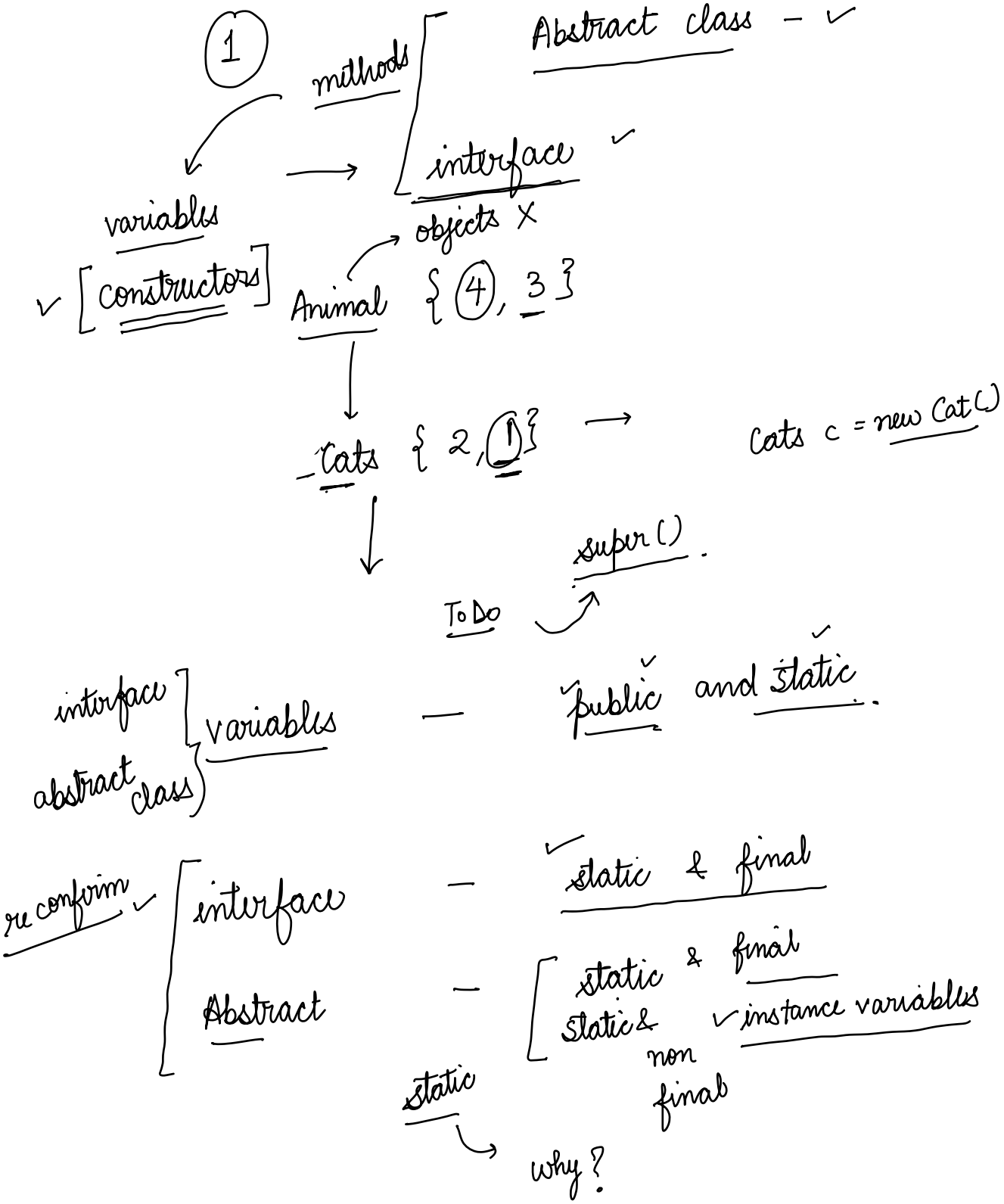
Iterable
↳ Collection
↳ List

↳ Queue

↳ set

interface
contract
implement
No objects


interfaces
I
drive()
classes
Vehicle
I
car
I
BMW
I
Maruti
I
common functions
drive()
I
Abstract class
abstract class {
}
inheritance
objects
Animal
{
}
inheritance
objects
Cats
Tiger
Cat
Cheetah


1
methods
Abstract class - ✓
interface ✓
variables
constructors
objects X
Animal {4, 3}
Cats {2, 1}
Cats c = new Cat()
super()
ToDo
interface
abstract class
variables — public and static.
reconfirm
interface — static & final
Abstract — static & final
static & non final instance variables
static
why?

→ Animal {

[int noofLegs;] ← non static

ab eat()

ab scream()

ab sleep

run() {

4 × 20 = 80

5 × 20 = 100

}

}

Cats extends Animal {

}

iterable — foreach

↳ Collection

↳ set

→ Queue

→ List

↳ ArrayList
- simplest list
- indexed access

LinkedList
- direct acc — not req
- faster insert

Vector

↳ thread safe

↳ Stack extends Vector

- thread safe

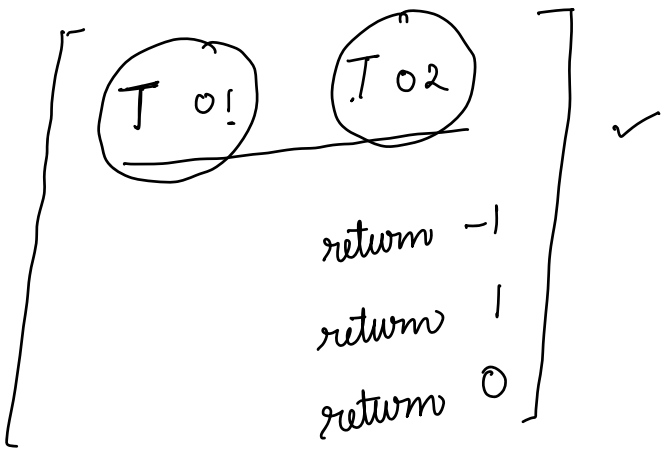
LIFO

Push pop

Set (unique element)

- HashSet — unique
  7 6 5
  | 5 |
  | 6 |
  | 7 |
- LinkedHashSet — unique
  → 5 6 7 insertion order
- TreeSet · thread safe
  Comparable · order — desc ✓
  asc ✓

comparable
comparator



Queue [FIFO]

- LinkedList — ✓ L in Q, L
- ArrayDeque → in → both ends, rr → both ends
- PriorityQueue → ordered
  min heap ✓
  → max

Map (Key, Value) pairs

↳ HashMap → unordered

TreeMap - sorted

LinkedHashMap - insertion

- generics
  - normal · List [ ]
  - bound ✓
    ↳ CV
    IV
    Co

✓ generics

✓ Multi threading

Doubts

Boolean b → Objects

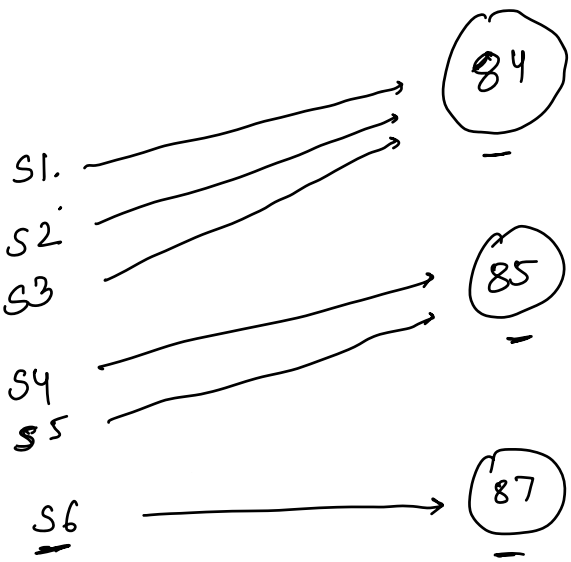
- True
- false
- Null

boolean

- true
- false

- Wrapper

Heap



Vehicle (v1) = [new Car()] →

List l = [new ArrayList]

l = [new LinkedList]

l = [new Vector]