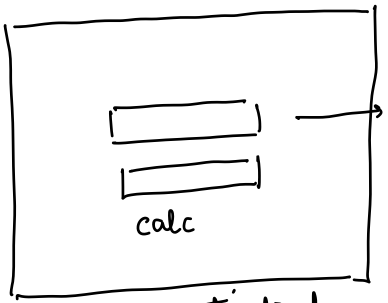
Today's content -

- generics
- Thread API in Java



4 classes

calculateAvg()

Interfaces → instantiated → no constructors

- All methods of interfaces are abstract.
- all methods are public.
- No objects → No instance variables

  all of variables are static and final in nature

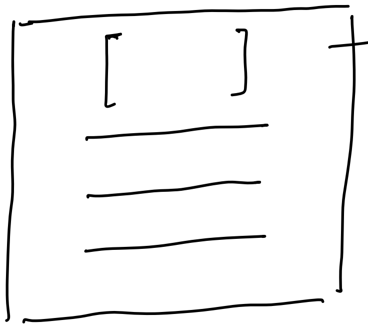
  ↓

  shared constants

- interfaces help us with multiple inheritance in java

  multiple inheritance causes → diamond problem.

- Abstract classes
  - It can be subclassed, it can't be instantiated
  - if one or more methods are abstract, then the class itself must be declared abstract.
  - subclass must implement all methods or declare itself abstract
  - AC → abstract and concrete methods.
  - variables can be final, non final, static, non static.



→ share constants → static final

ensure

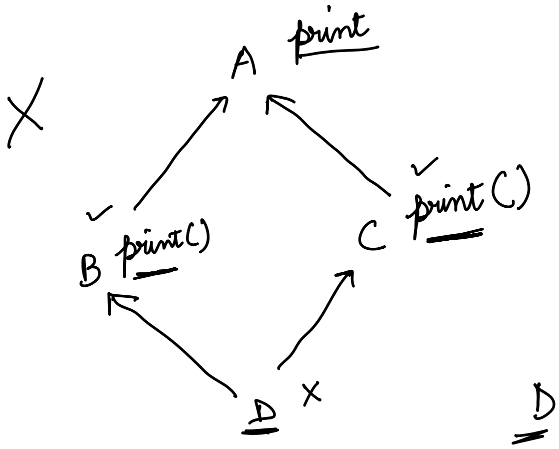
✓ C1 . ✓ C2 ✓ C3

• interfaces — multiple inheritance ✗

↳ share resources

C1 imp I1, I2, I3

Diamond problems



B extends A

C extends A

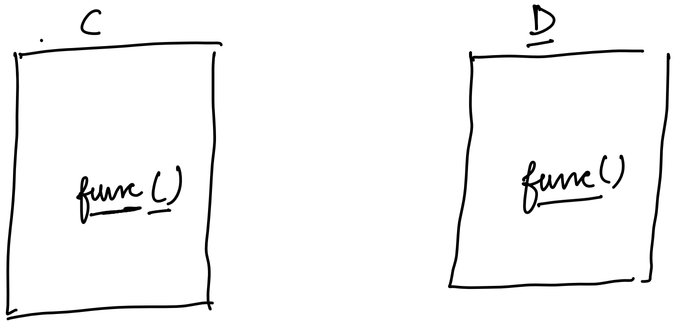
D extends B, C ✓

D d = new D()

d. print()

D

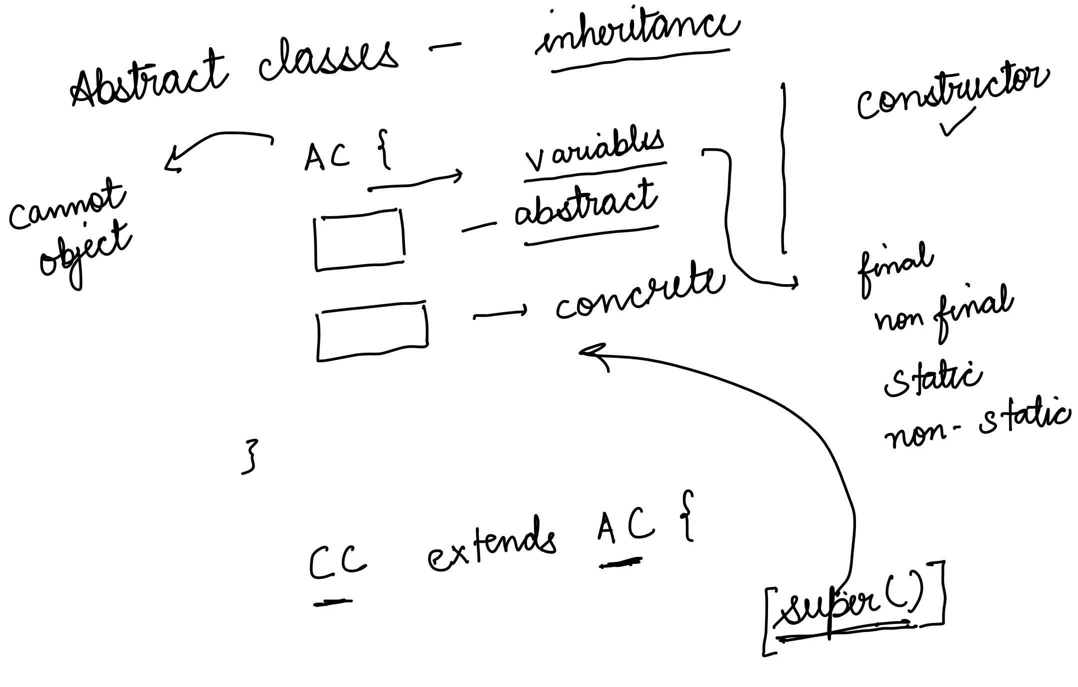
d. print()



```
A  implement C, D {

    func() {

    }
}
```



Abstract classes – inheritance

cannot object

```
AC {
    variables
    abstract
    concrete
}

CC extends AC {

}
```

[super()]

constructor ✓

final
non final
static
non-static

default → Shared constants

static

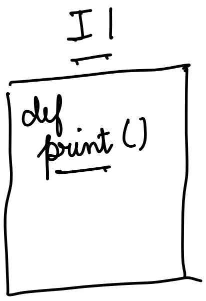
private

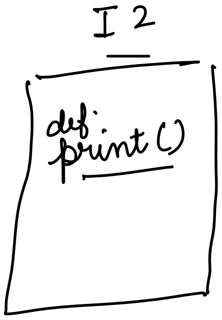
private static

method → print() ↓ same definition

Shared Constants

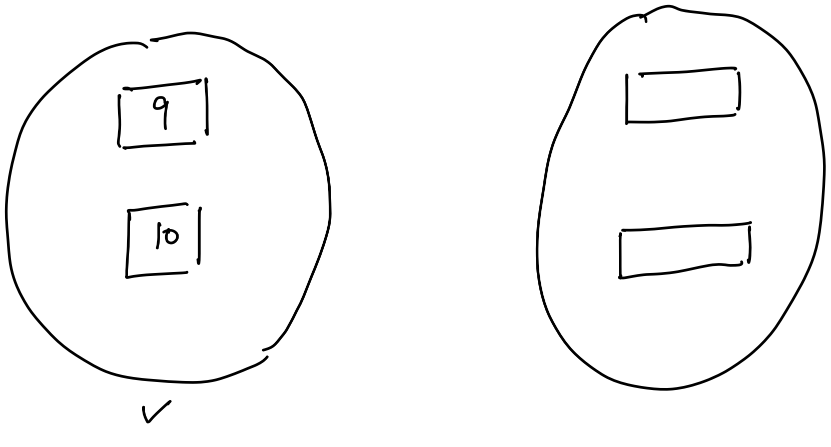
interfaces C1 C2 C3





S imp I1, I2

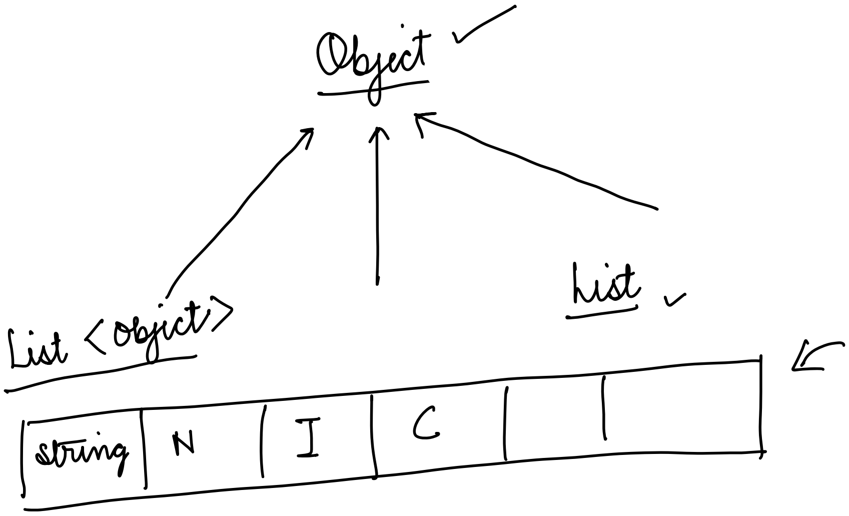
CTE

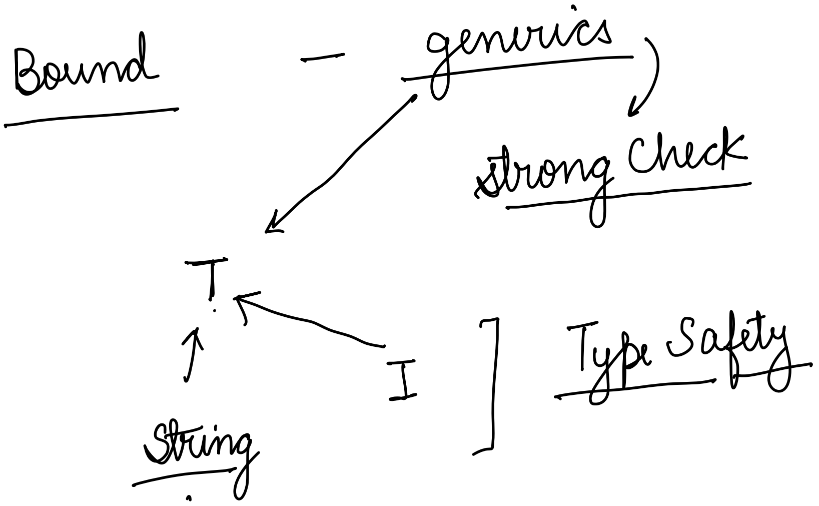


Pair — Generic types

```
Pair < T, Q >  {
      T
      Q


}
```





generics - reusability

Class →

List → S

└→ I

└→

List - In

└ List String

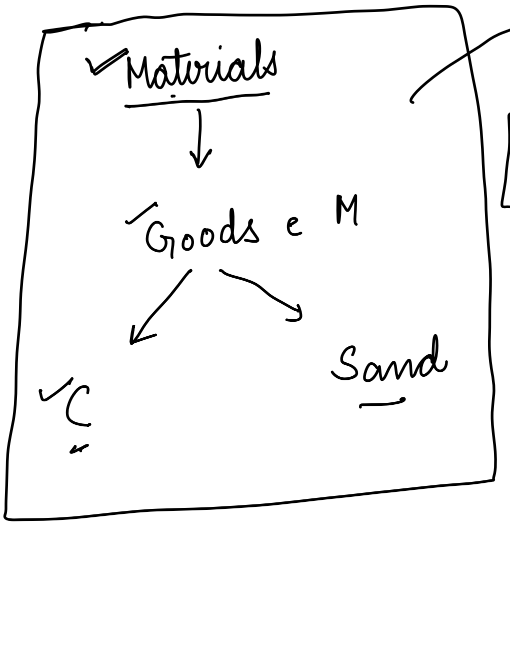
List → userdefined DT

⇒ generics - make DS possible.

List <T> ✓

Bounds → Class <T>

method [T]



T

anything

<? extends T>

Material

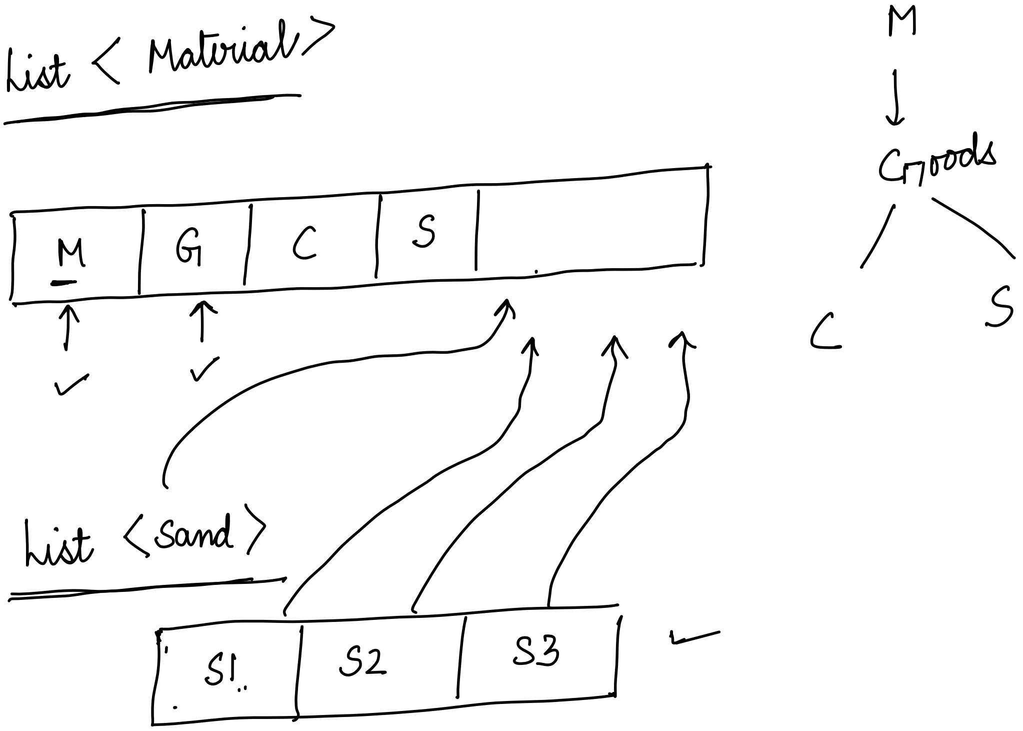
? extends Material

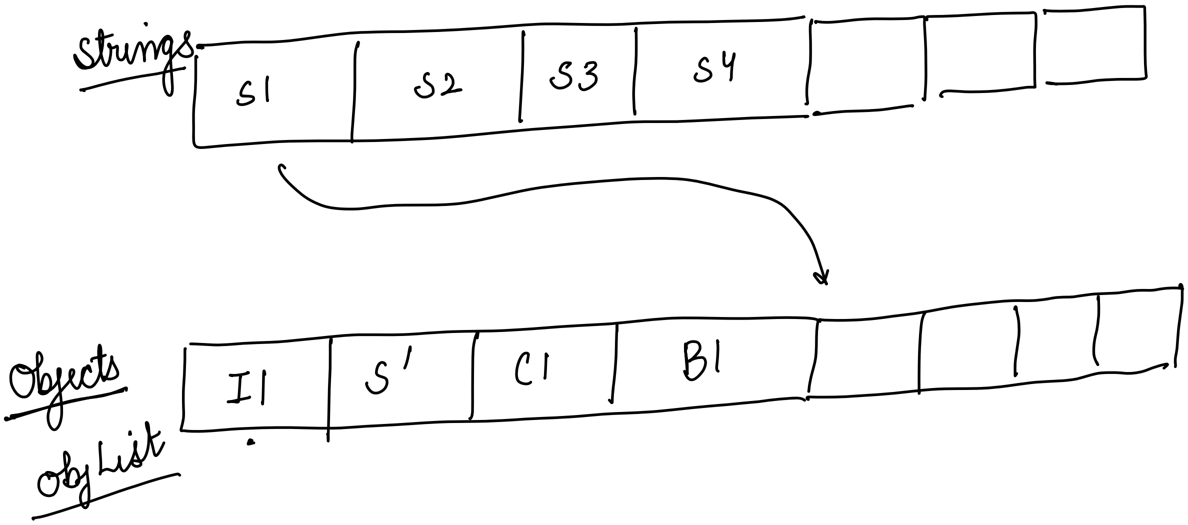
- Invariance
- Covariance
- Contravaruaince

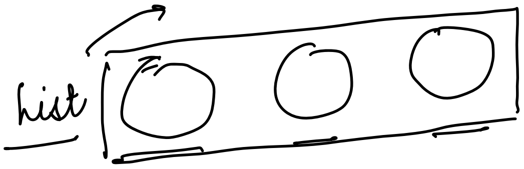
?
anything

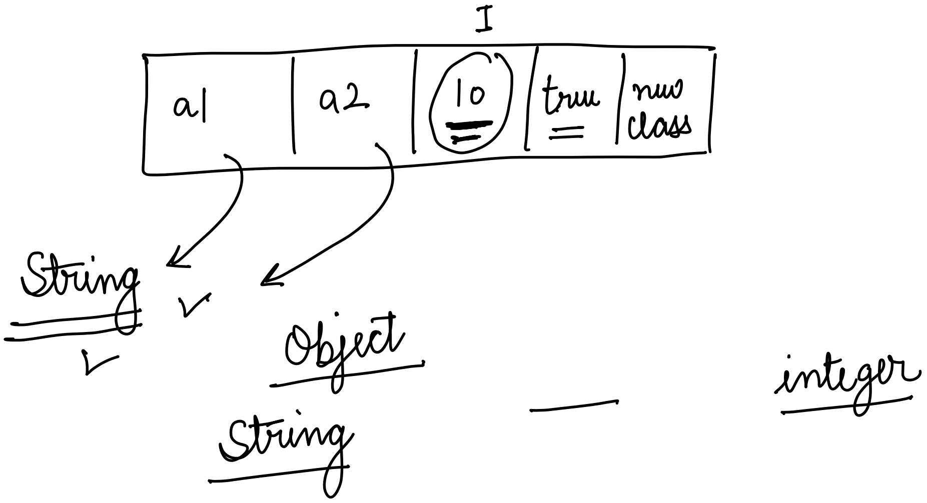
? extends Material
Bound
or

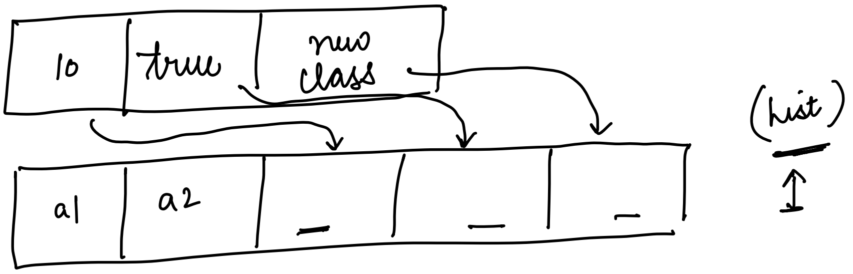
T extends Material
→ any child class




strings
S1
S2
S3
S4
Objects
objList
I1
S'
C1
B1


List


I
a1
a2
10
true
new class
String
Object
integer
String


10
true
new class
(List)
a1
a2
_
_
_

Type Safety → Compile time ✓ b=0 → 10/b

Type → Run time ✓

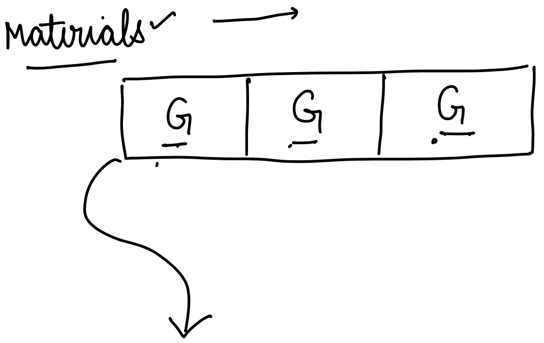
↓

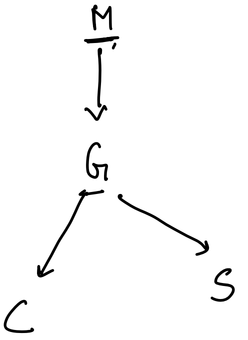
Type Erasure ✓

Invariance – accept type

covariance – M accepty subtypes

Contravariance – M! accept supertypes of Material.



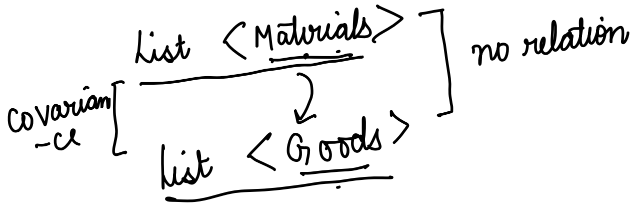


List <Goods>

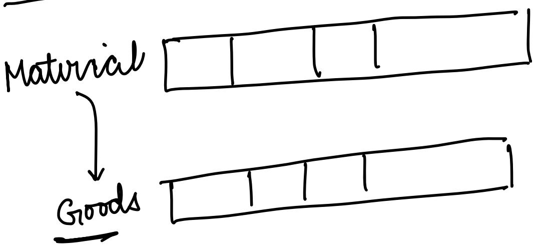
M

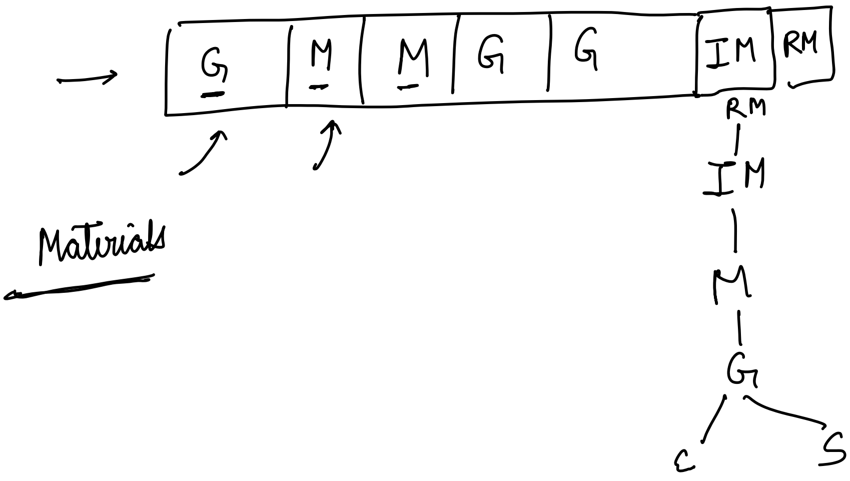
↓

G



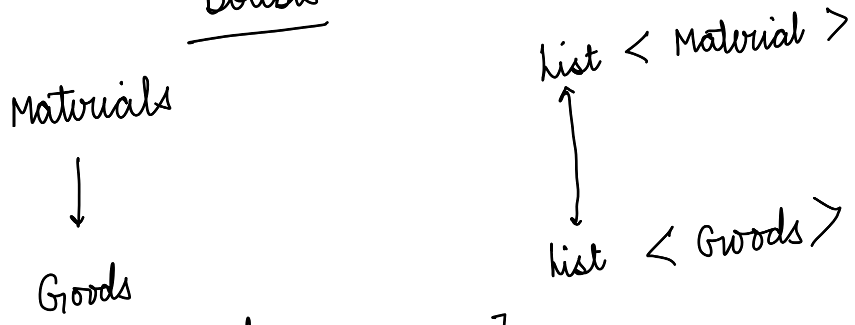
Contravariance – accept supertypes





Break → 11:00 pm

Doubts



list < ~~Materials~~ Goods > Mf

list <? extends Materials> Af

— covariance

list < Materials> Mf

list <? super Goods> Af

— Contravariance

Common Type

T	-	Type
E	-	Element
K	-	Key
V	-	Value
N	-	Number

✓ T

↓

needs definition

↓

generic type

Pro

and [?

↓

anything]

con

- **Concurrency**

one job at a time

1 CPU - CS

C

1 head

10 plates

P1 ⎤
P2 ⎥ Juggling
P3 ⎦
P4

Parallelism

multiple executions/jobs at a time

10 CPU - CS

P

10 heads

Eat 10 different plates

10

1 CPU

↓

4 cores

Context switching

C	C	C	C
A1	A2	A3	A4
2ns	2ns	2ns	2ns

1 CPU ✓ - 4 Tasks

10 CPU ✓ - P → CS

1

↓

1C → C

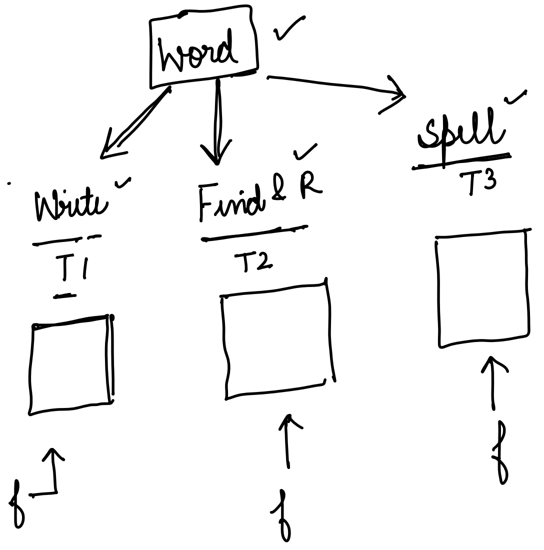
40 tasks

Threads :

- Program → file - .java
  .c
  .cpp
- Process → main memory
- Thread → Sub parts

CS - 8 cores

P ✓ C ✓

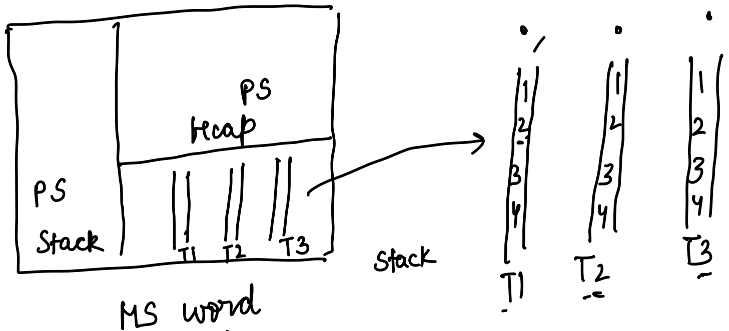


Logic
algo

Shared memory

Text ✓
Content ✓

memory map



1 CPU

⇒ T1.1
T1.2

T2.1
T2.2
T2.3

T3.1
T3.2

MS word
↓
2 ns

Uncontrolled scheduling

Static main() ⟶ main thread

t1 ⟵

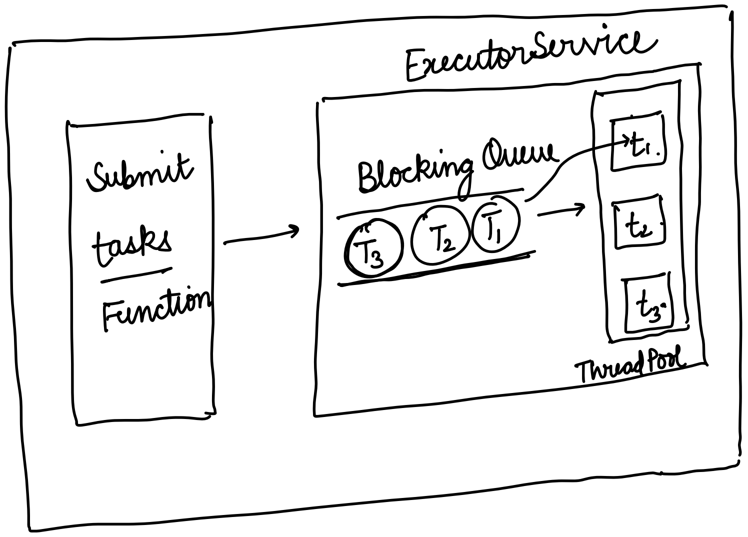
run

t1. start() ⟶ ① create a thread
② run()

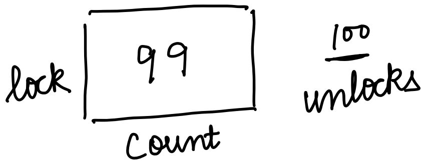
2 things

↳ Executor Service ⟹ 1 Thread
2
3
100 threads
1000 threads

↳ Race Condition



Race condition



count = count +1 => 3

R (Retrieve it and registor)

I Increment

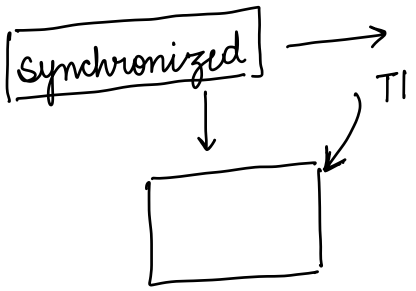
S Store it back

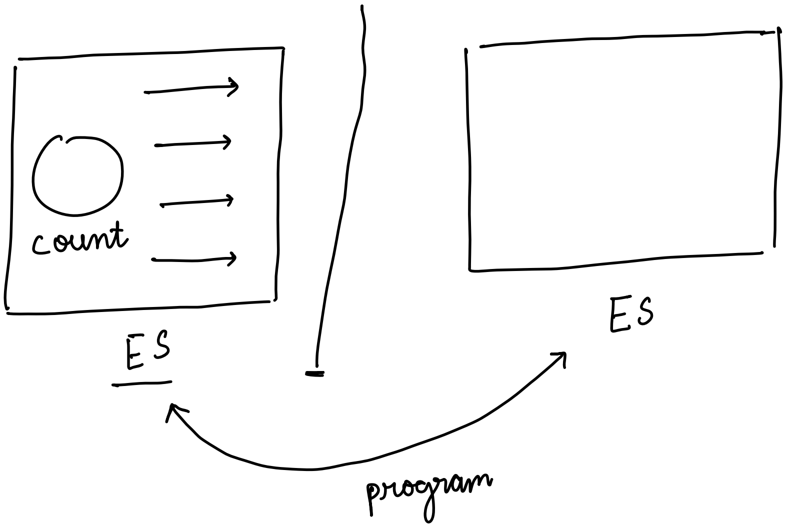
lock

	T1	T2.
R	99	99
I	100 (Regis)	100
S	100	100

1000

R I S → 1 step ✓

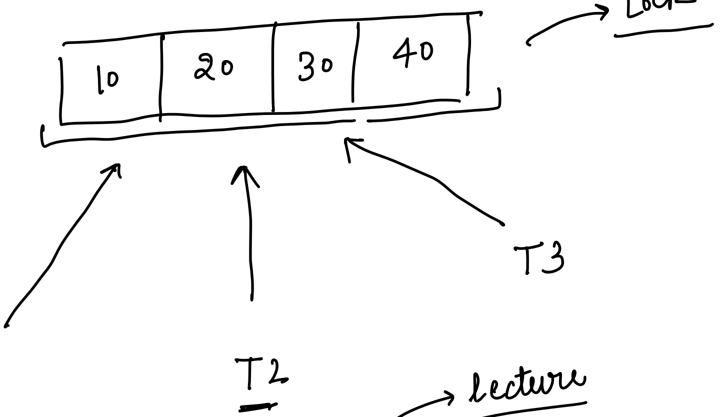




join()

Thread Safe Data Structure

AL ✗
LL ✗
Vector ✓
Stack ✓



⇒ Lecture - Atomic
Volatile
Reflection - ✓
→ lecture

100 MCQs - Wednesday
↓
32

Java1
J2
J3
J4