OOP3

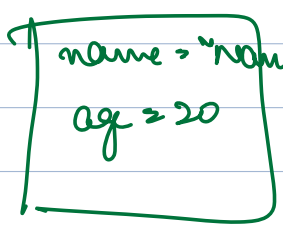
- ⇒ Copy Constructors ✓
  - ↳ Deep Copy V/S Shallow Copy ✓
  - ↳ Pass By Value V/S Pass By Ref ✓
- ⇒ Inheritance
  - ↳ Constructor Chaining

COPY CONSTRUCTOR

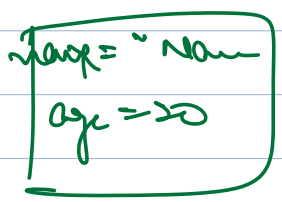
```
class Student {
    private String name;
            int age;
            String gender;
            double psp;

}
```

→ we have an object (X) of a student ⇒ X

→ we want to create another object of Student (Y). The values of the new student should be exactly same as prev student



X

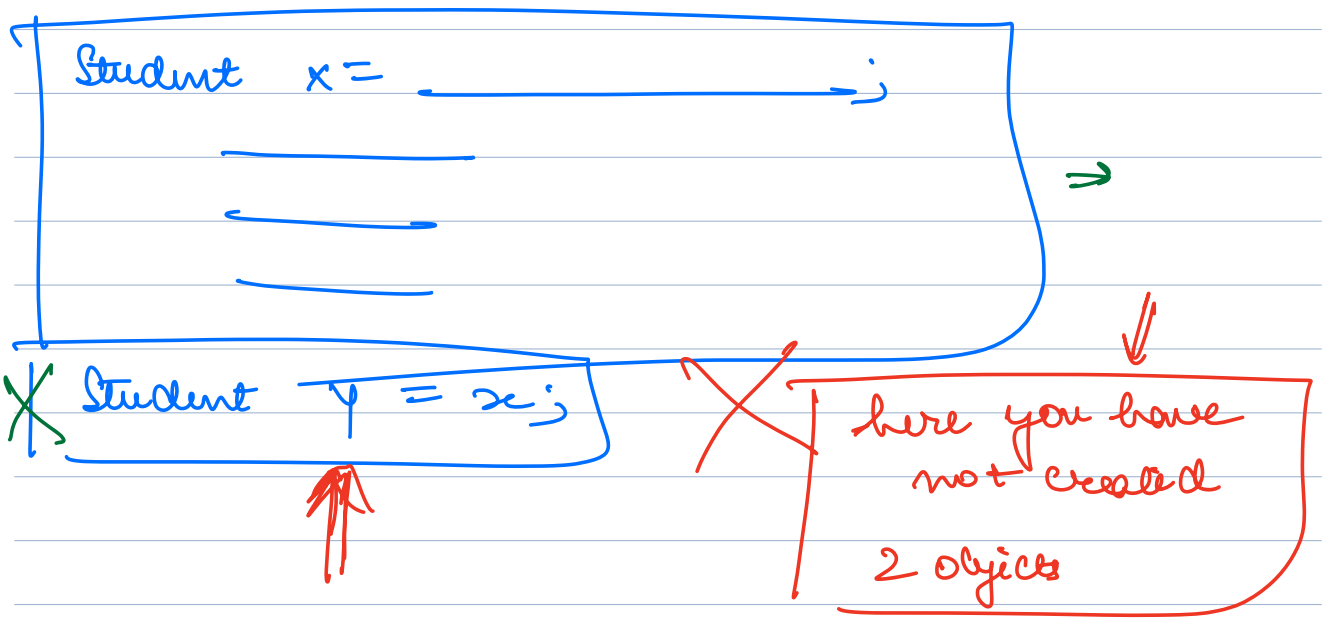


Y

Soln

(1) Create a new student and manually copy all values — correct, but DON'T DO

(2) Student y = x ✗



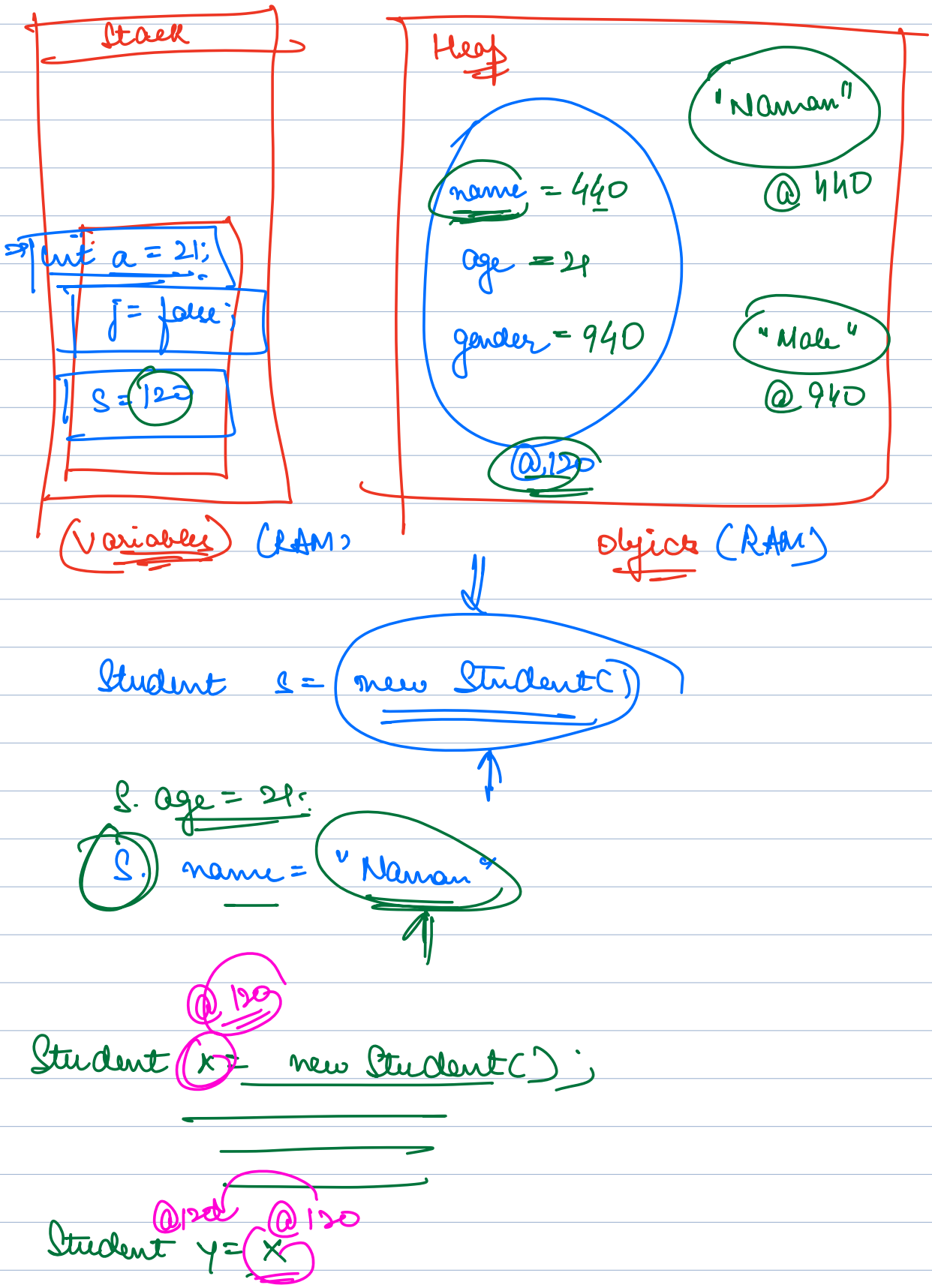
Java Memory

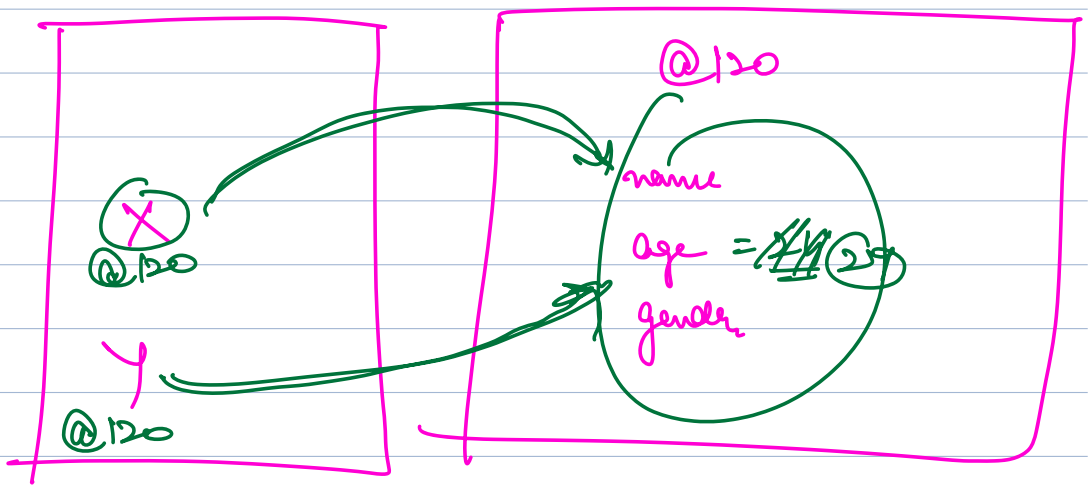
int a = 2;

Integer i = 23;

Types of data types

- Primitive (int, boolean, float)
- Object (formed from a class)


Stack
Heap
int a = 21;
f = false;
s = @120
"Naman"
@440
name = 440
age = 21
gender = 940
"Male"
@940
@120
(Variables) (RAM)
Objects (RAM)
Student s = new Student()
s.age = 21;
s.name = "Naman"
@120
Student x = new Student();
@120
@120
Student y = x



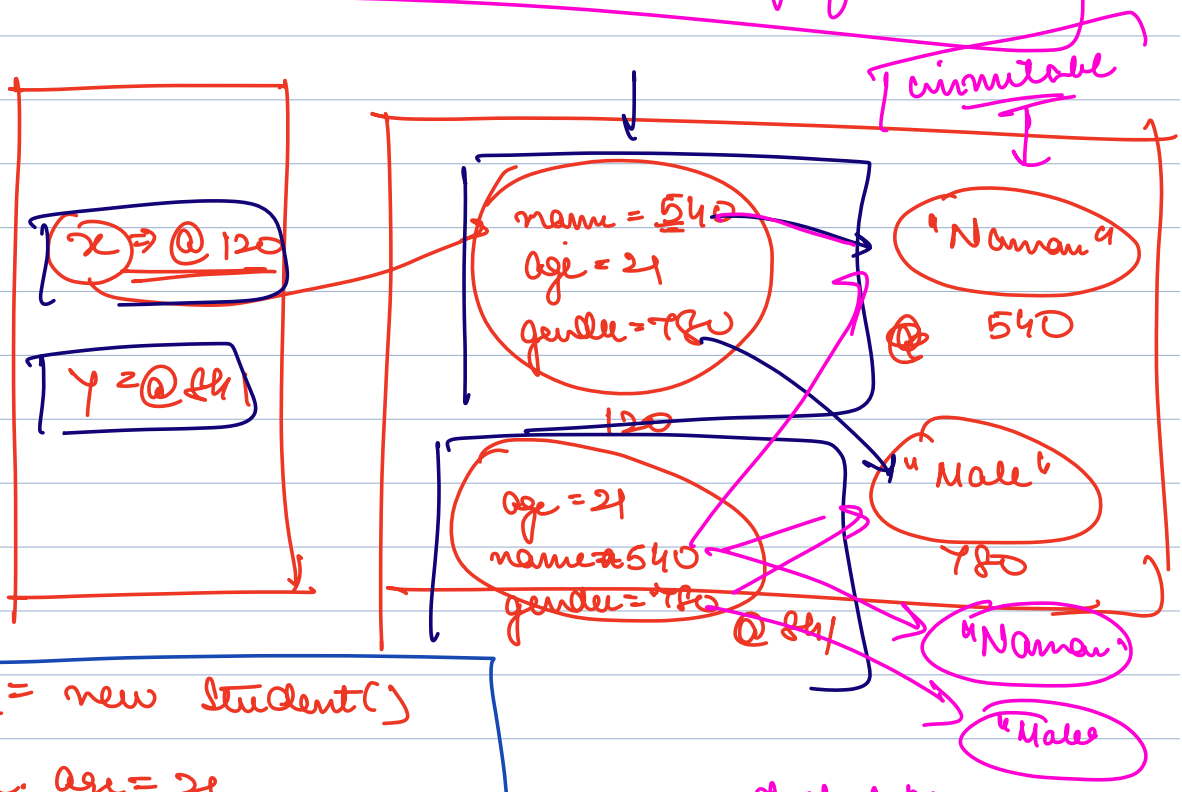
x.age = 24
Student y = x
y.age = 27
Sout (x.age) ⇒ 27

Student x = new Student();
x.name =
x.age =

Client.java

Student y = new Student()
y.name = x.name
y.age = x.age
y.gender = x.gender

Cons : (1) Some attrs may be private
(2) Abstraction is not being followed




```
x = new Student()
x.age = 24.
x.name = "Naman"
x.gender = "Male"

y = new Student()
y.age = x.age
y.name = x.name
y.gender =
```

Deep copy

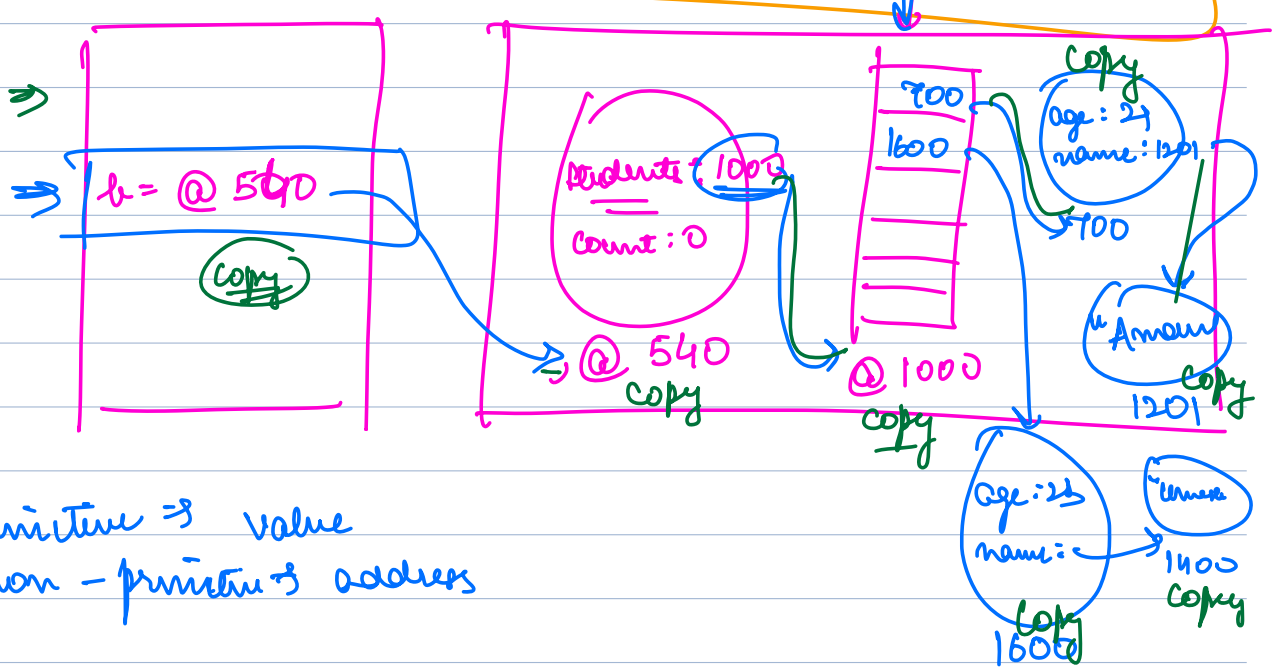
While we have created a copy, both are not completely separate
⇒ after a point, both are sharing same attr.
⇒ Shallow Copy

Deep copy ⇒ a copy of an object where no instance attr is shared anywhere

---

```
class Batch {
    List<Student> students;
    int strength;
}
```

```
Batch b = new Batch();
b.addStudent(new Student("Aman"));
b.addStudent(new Student("Umesh"));
```



primitive ⇒ value
non-primitive ⇒ address

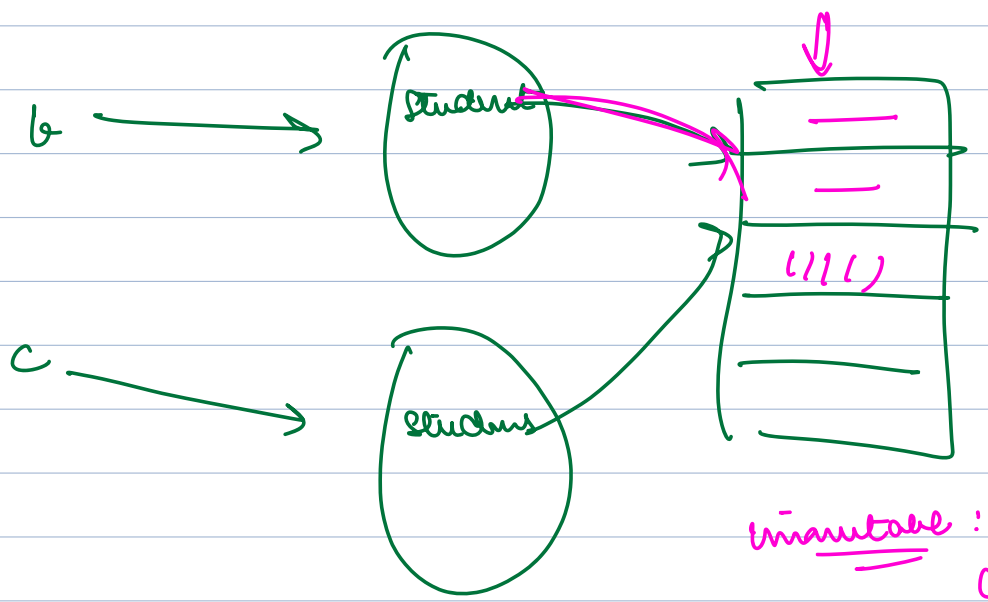
Creating a deep copy can be very inefficient and even may be difficult
⇒ recursively create a copy of every attr

Batch b = new Batch()
b.addStudent(Aman)
b.addStudent(Ajay)

Batch c = new Batch()
c.students = b.students;

c.addStudent(Atul)

⇒ sout(b.students.size()) ⇒ 3

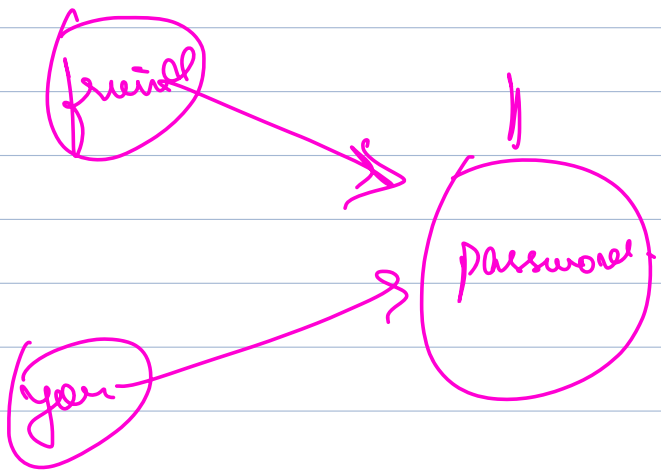


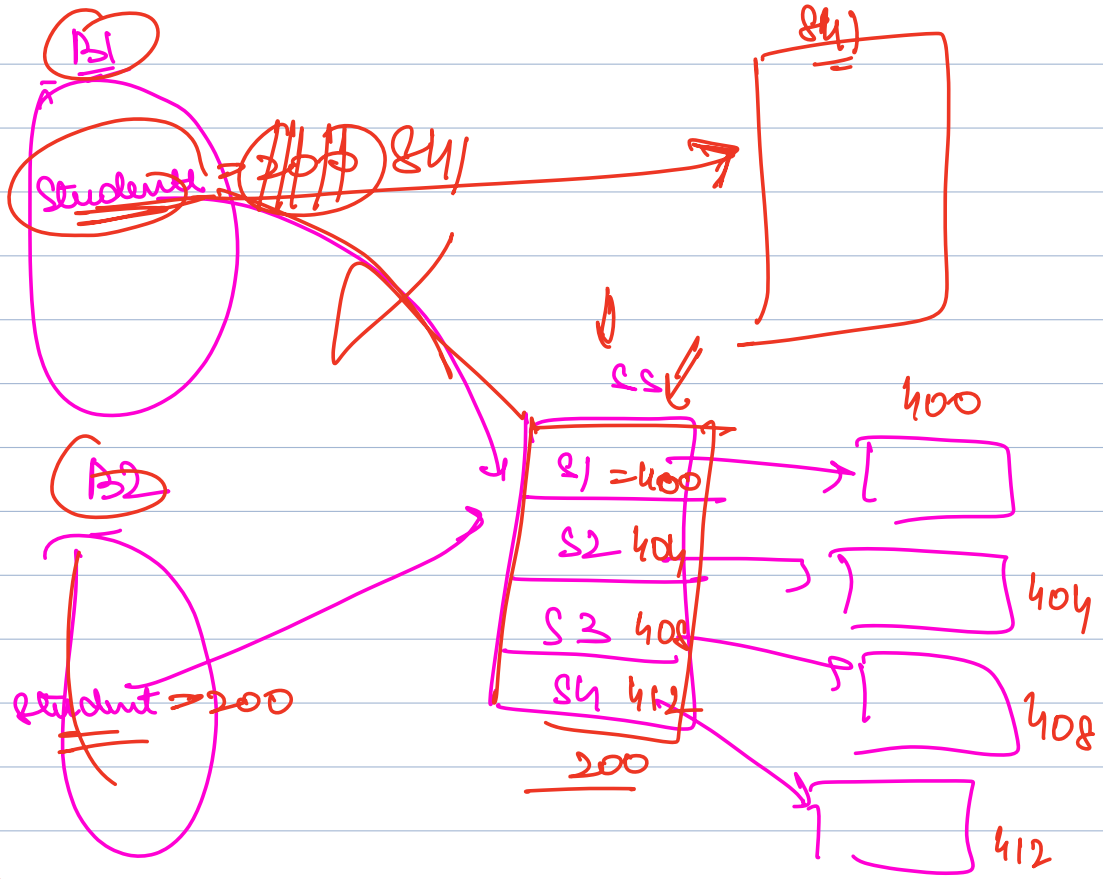
immutable: can't change once created

c.addStudent()

⇒ Be careful when creating shallow copies. unless shared obj are immutable

Immutable: Object that can't be modified once created.





Opn 1

@ @41

B1. Students = new ArrayList();

Sout (B2. Students. Size()); ⇒ 4

(B1. Student. Size() → 0

Opn 2

B1. Students. add (new Student)

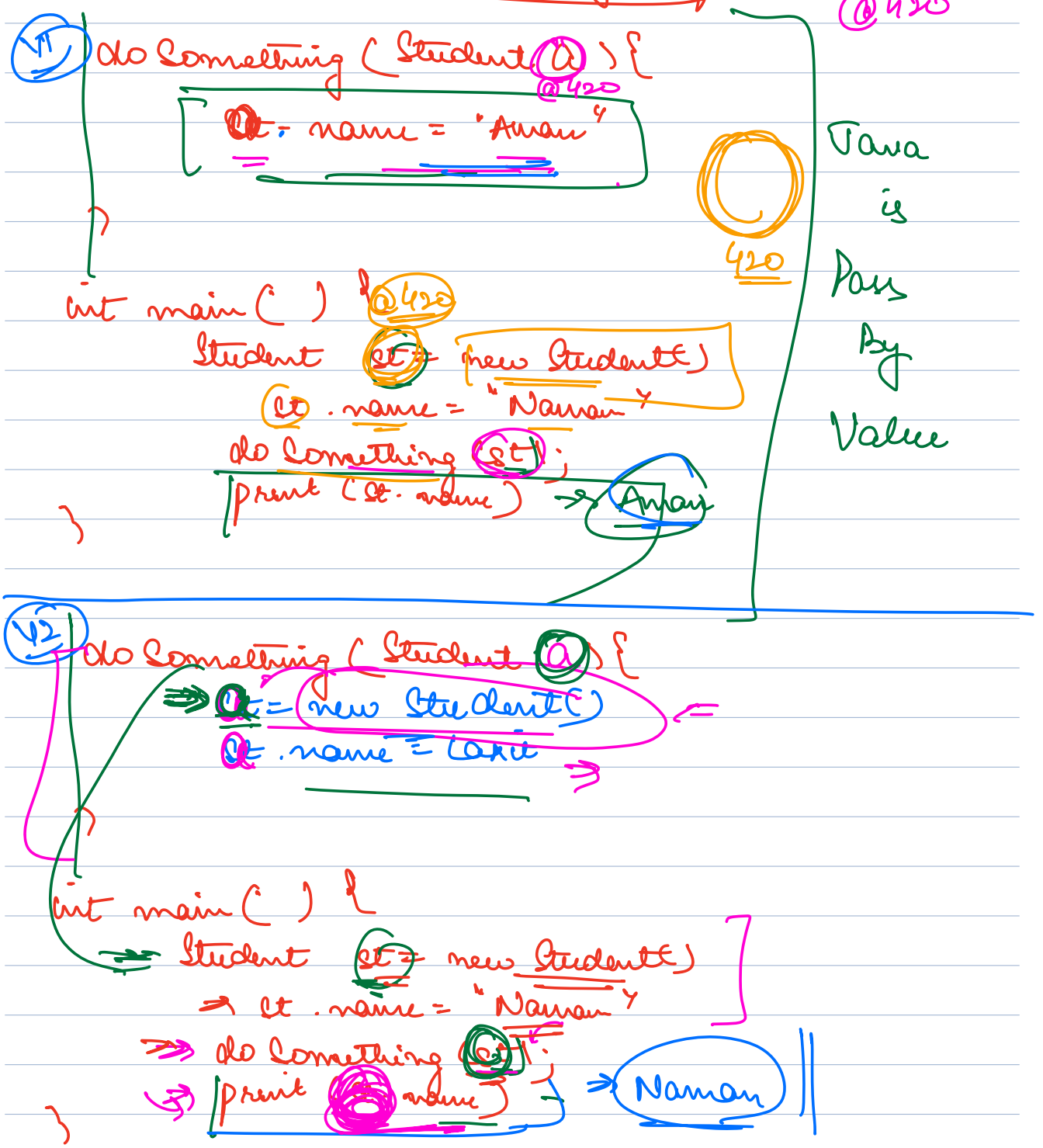
Student @ 120

S. name = "Naman"

@ 120

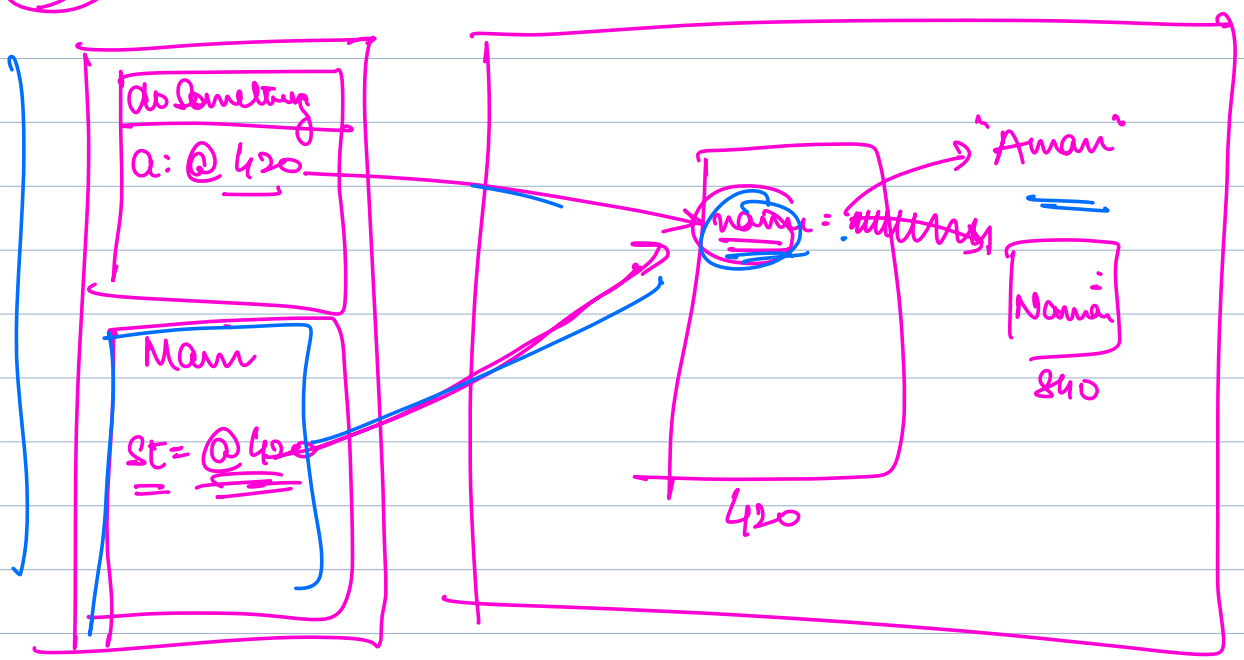
"Naman"

Pass by value v/s Pass By Reference

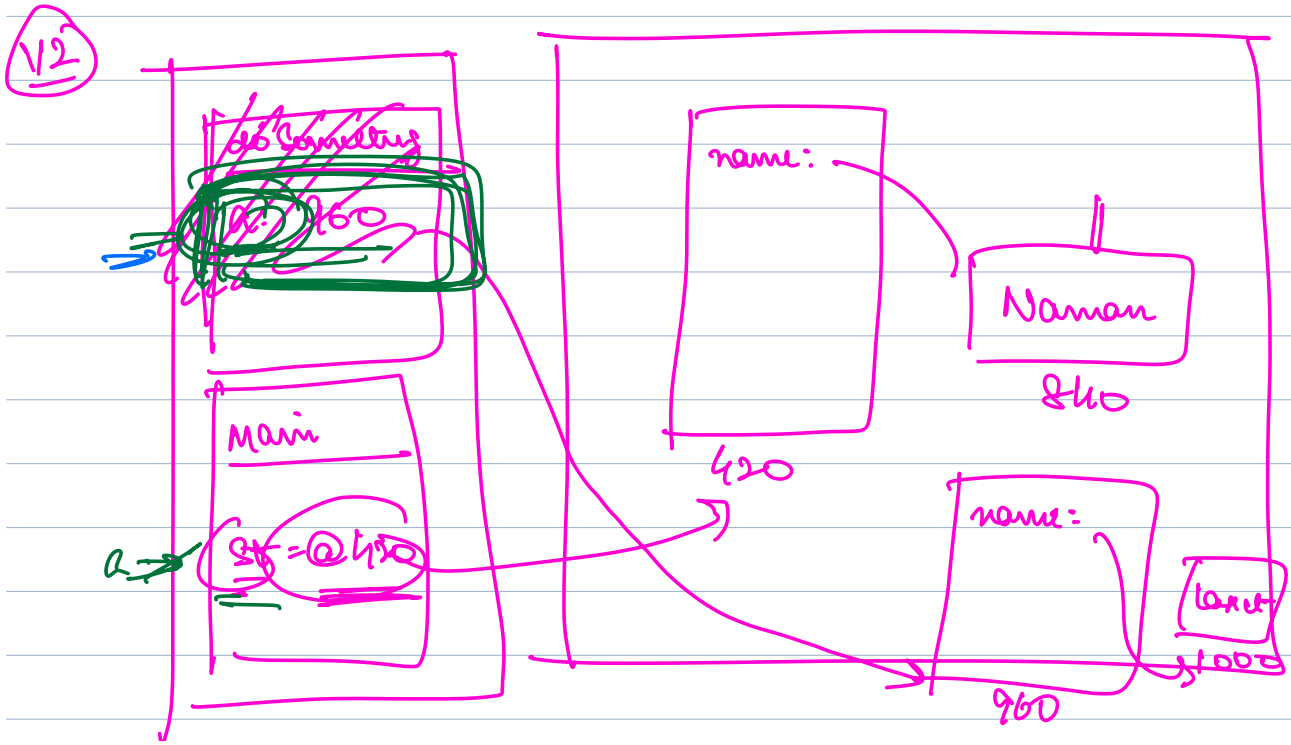


Pass By value ⇒ when passing to fn, a copy of the param is passed

V1

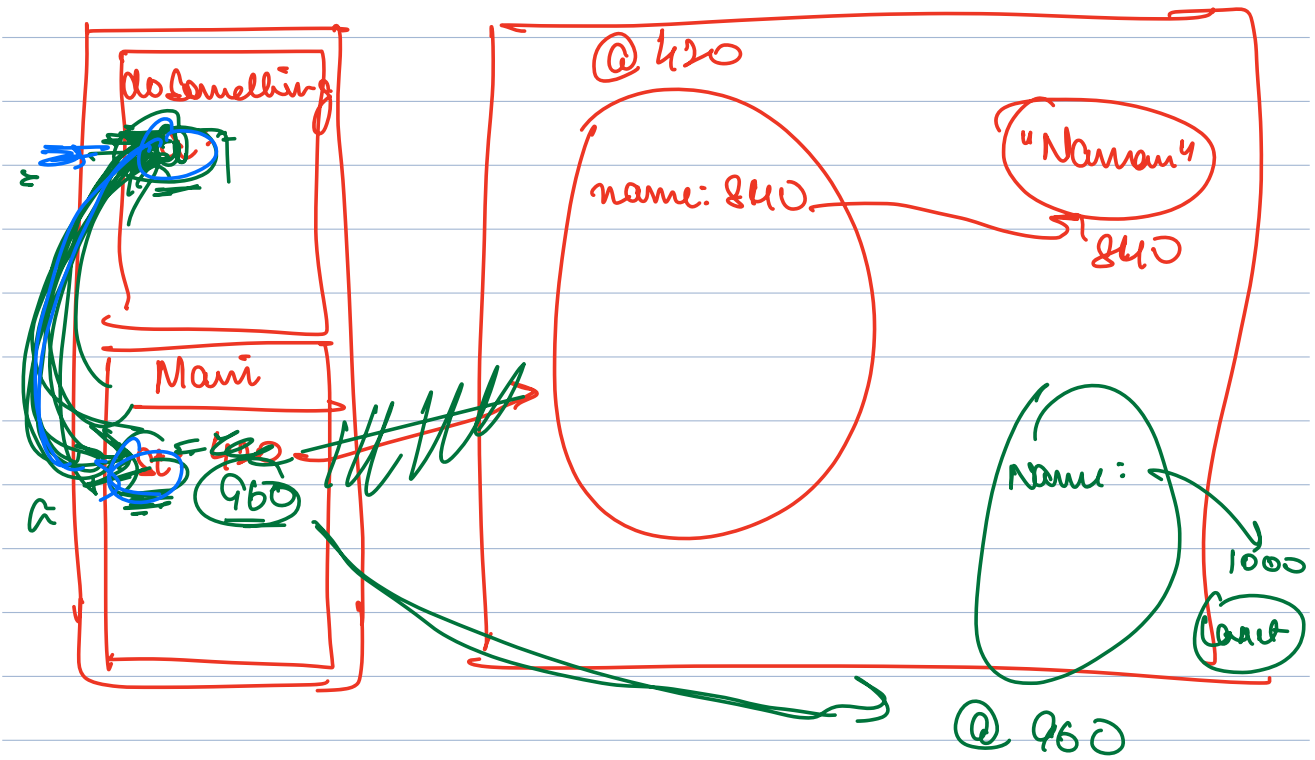


V2



Pass By Ref as 2 very close lovers

"Pain happens to someone other also gets hurt"



Pass By Ref

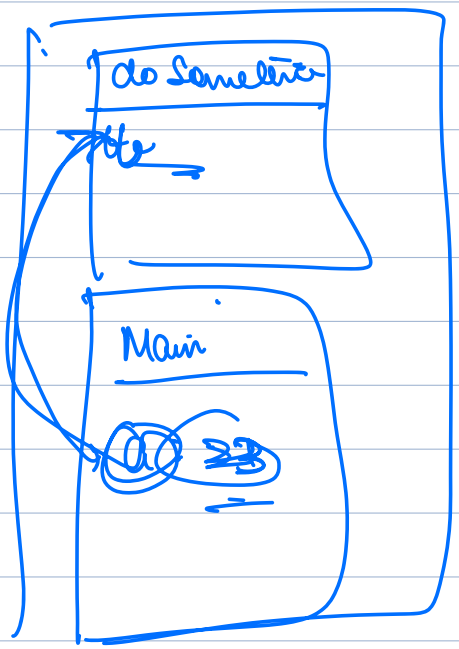
Ravi ⇒ Aman ⇒ Someone Beats you

Pass By Value

Ravi

Aman ⇒ Broke his leg

Pass By Ref



doSomething(int a) {

a = 23

int main() {

int a = 21

doSomething(a)

print(a)

}

PBV = 21 PBR = 23

4200

doSomething(Student) {

Students = null

}

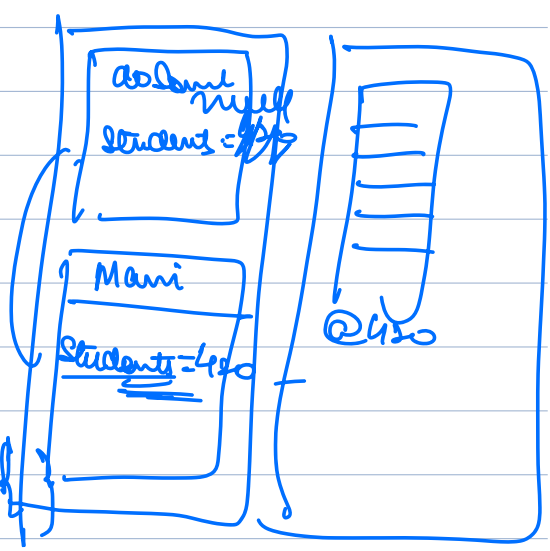
main() {

List<Student> students = {}

doSomething(students)

print(students)

}



COPY CONSTRUCTOR

Constructor that taken an object of the same class as a parameter and returns a new obj with same values as passed obj

```
Class Student {
    private int age;
    private String name;
    private String gender;

    Student ( ) { }

    Student ( Student other ) {
        this . name = other . name
        this . age = other age
        this . gender = other . gender .
    }
}
```

other: age = 23, Name @410, @420

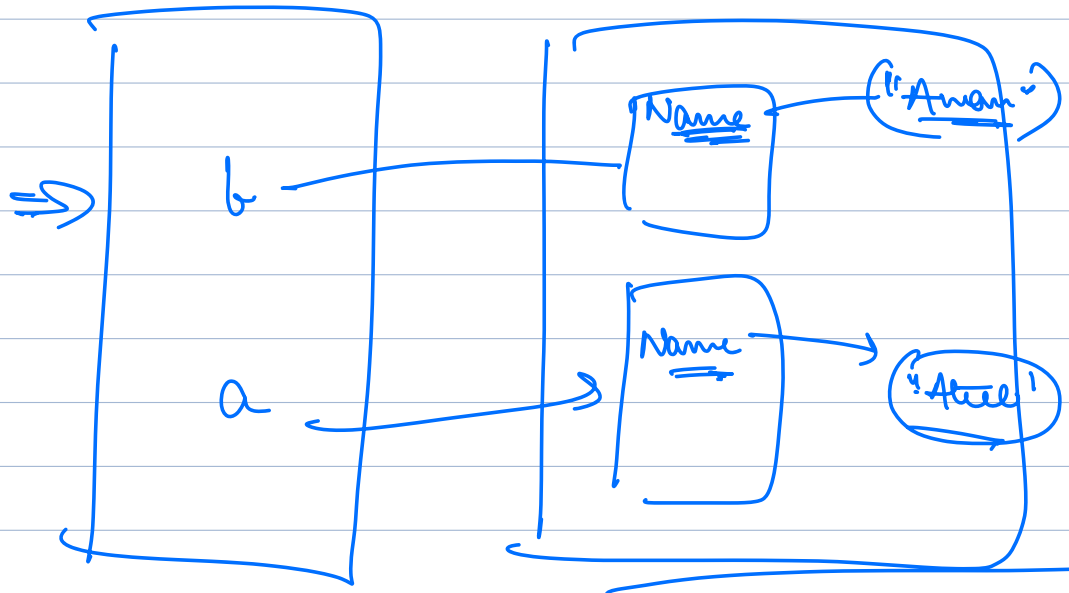
this: age = 23, name = @410, @440

Same

Shallow →

Abstraction

```
Client {
    psvm() {
        Student a = ____ ;
        a. name = Akil
        Student b = new Student(a)
        b. name = Aman
        sout (a. name) → Akil
    }
}
```



GC | "Mark and Sweep"

In java, as soon as an object becomes no longer needed ⇒ GC finds that obj. and removes that obj from mem

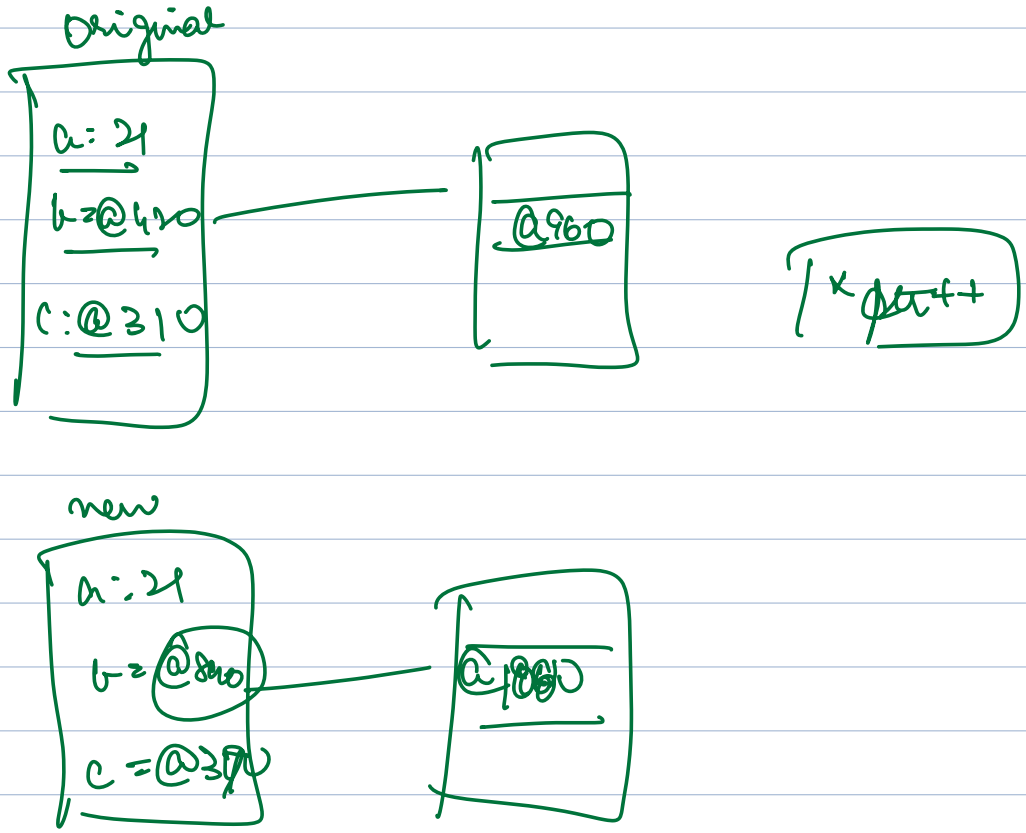
⇒ In languages that don't have GC ⇒ manually delete

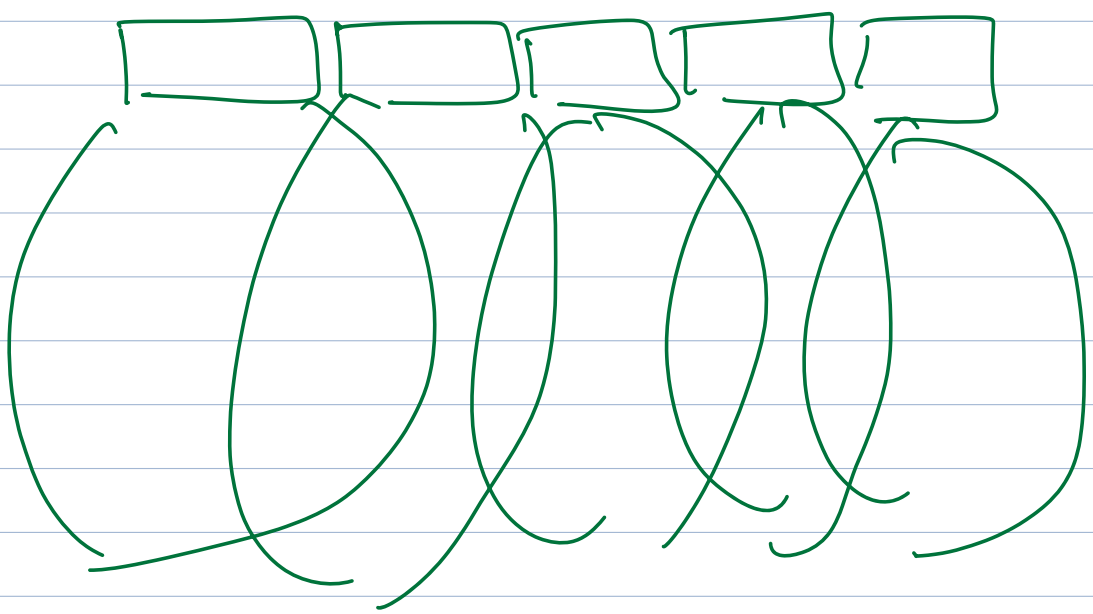
H/W

Build a garbage collector

(Mark and Sweep algo)

Inheritance





MS Word → MS Word Update