Agenda

- → Copy Constructor
- → Pass By Value V/s Pass By Reference.
- → Destructors
- → Inheritance (Pillars of OOP)
- → Polymorphism

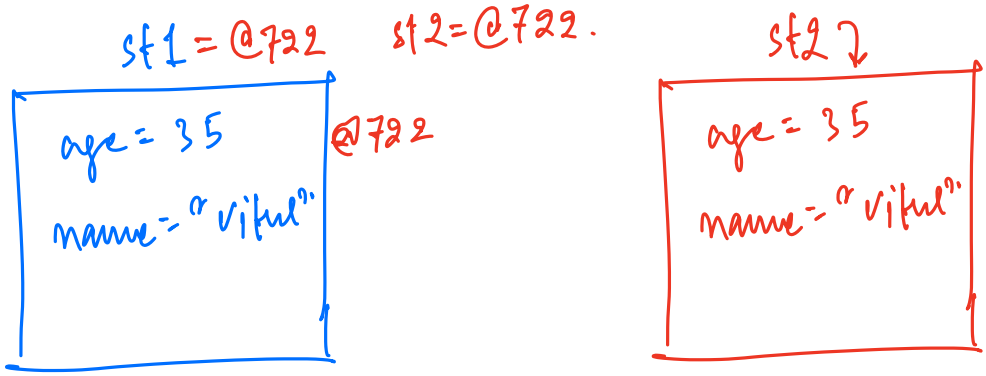
Small announcement

```
int age;
String name;

Student (int age, String name) {
    this.name = name;
    this.age = age;
}
```

Copy Constructor

Student st1 = new Student();



Student st2 = st1; → Wrong ✗
=
↳ Is this creating a new object? → No.

```
Student {
    private int age;
    private String name;
    Student(){
    }
    Student(Student old){
        this.name = old.name;
        this.age = old.age;
    }
}
```

↳ equivalent to writing

st2.name = ——
st2.age = ——

```
Student st1 = new Student();
st1.age = 35;
st1.name = "Vitul";
Student st2 = new Student(st1);
```

st1

st2

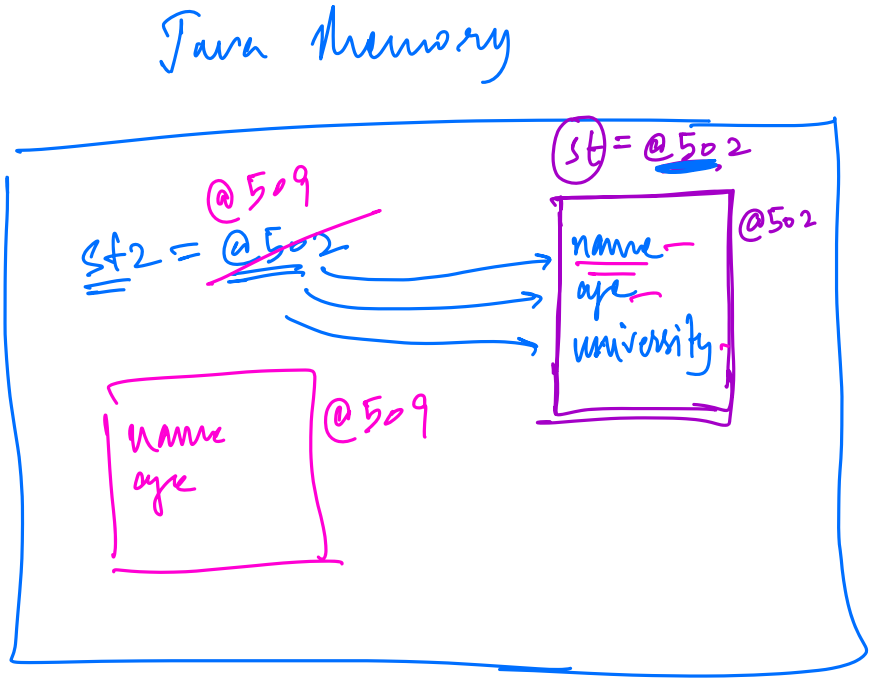
name = "Vitul"
age = 35

How to write copy constructor → Yes.
We also understand "this" key word → Yes.

Why we need copy constructor?

```
Student st1 = new Student();
Sop(st1.name) → "Vitul";

Student st2 = st1
st2.name = "Alok";
Sop(st1.name) → "Alok";
```



copy constructor of Student Class.

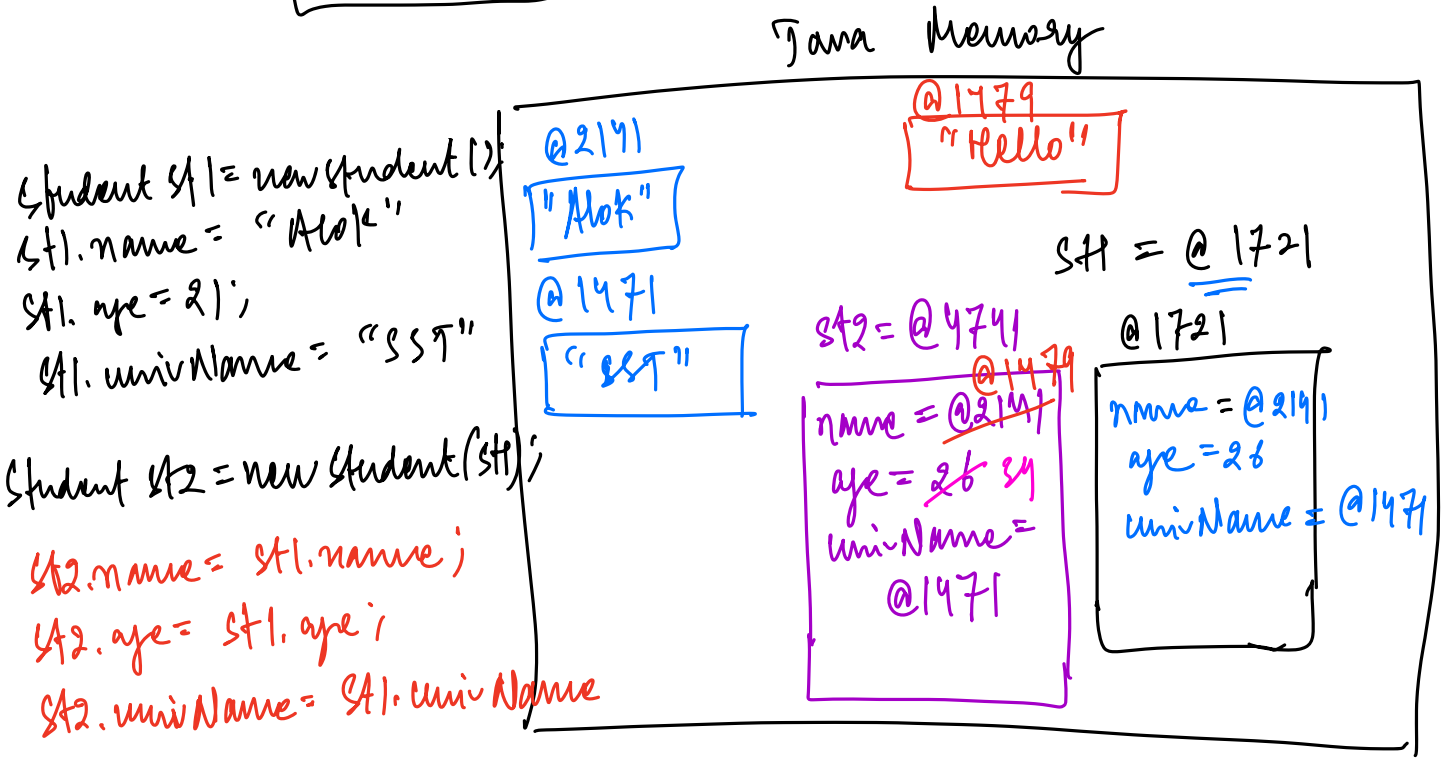
```
public Student (Student oldStudent) {
    this.name = oldStudent.name;
    this.age = oldStudent.age;
    this.psp = oldStudent.psp;
    this.isPlaced = oldStudent.isPlaced;
}

Student st2 = new Student (st1);
```

Types of Data Types:-

1) Primitive Types:- int, double, float } they are stored as it is.
No Java memory is utilised.

2) Objects:- stored in memory =
Variable store address of this memory location.



① st2.age = 34

② st2.name = "Hello" → In Java, new memory will be allocated for "Hello".
⇓
st2.name = new String("Hello"); Because Strings are immutable in Java

③

Deep and Shallow Copy.

Shallow Copy.

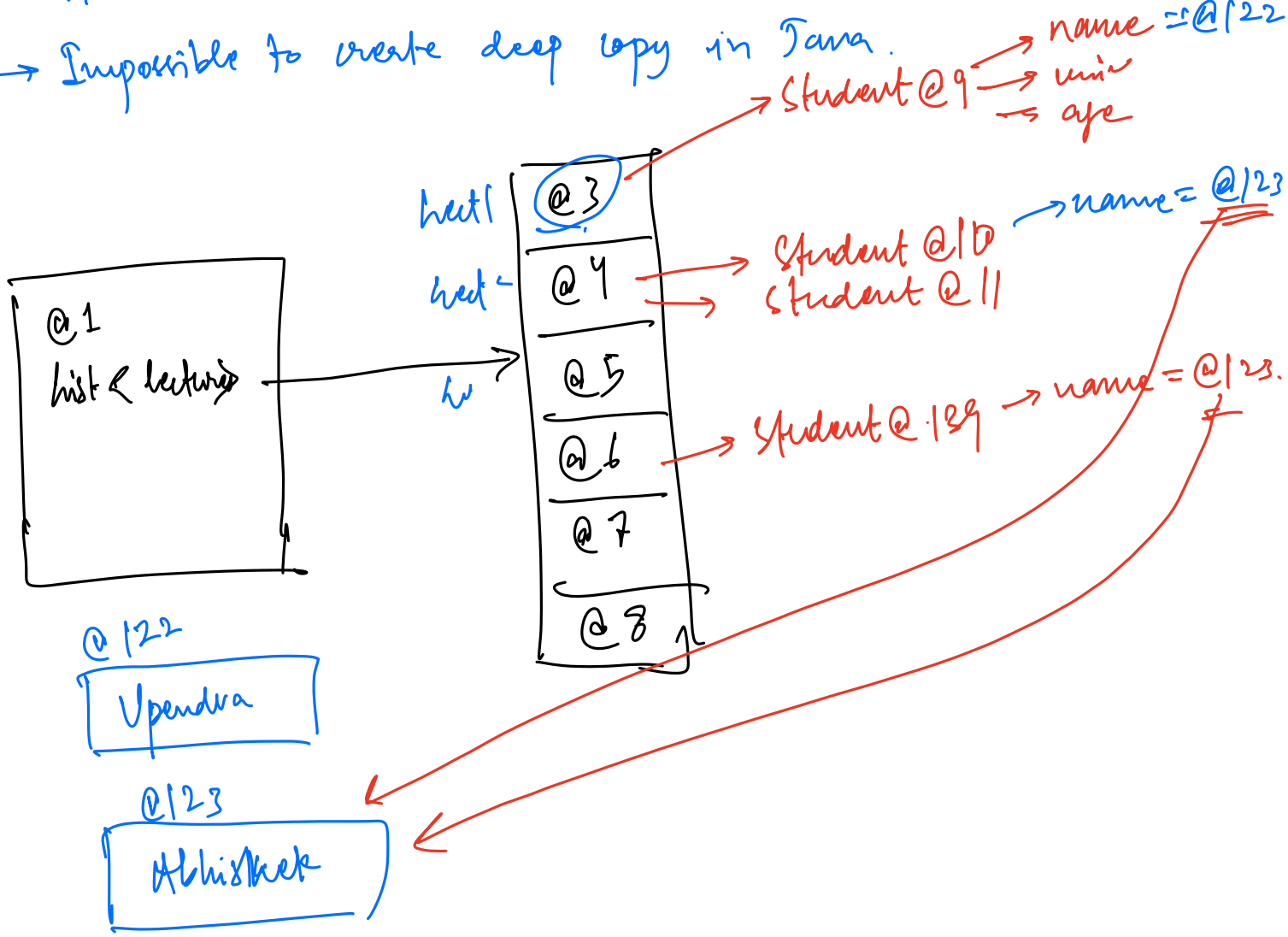
When you create a new object but behind the scenes it refers to attributes of the old object.

```
Student st1 = new Student();
Student st2 = st1;   → Shallow Copy.
```

Deep Copy

→ No shared data

→ Impossible to create deep copy in Java.



Announcement → This Weekend

Blog Website → ① Why is it good to have separate key with mapping tables?

② Why not possible to have deep copy?

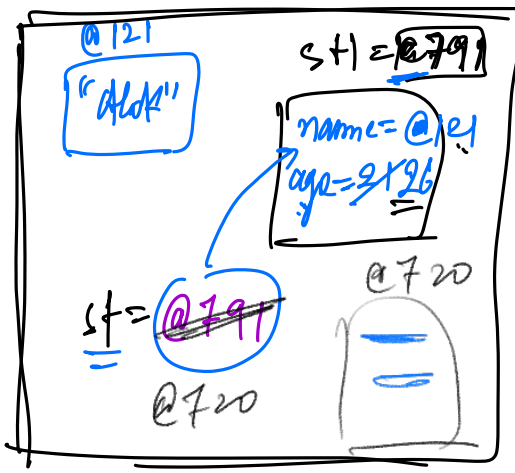
Pass By Value Vs Pass By Reference.

Java is always Pass By Value → 100%

@791 → Is this student here passed by val or passed by ref.

```
doSomething(Student st) {
    st.name = "Alok";
    st.age = 26;
    st = new Student();
    st.name = "Aalok";
}

main() {                    @791
    Student st1 = new Student();
    doSomething(st1);
    Sopln(st1.name);  → "Alok";
                  @791
}
```



Address ≠ Reference.

int a = 21;
Student st1 = @721;

Pass By Reference

B1 = → slaps B1

B2 → Feels the slap

Pass By Value

B1 = @721

Burn B1's house

B2 = @721

cousins.

→ Once all the work related to that object is done. the object is garbage collected in Java.

* How garbage collector works in Java → Please read. Blog

→ In languages like C++ garbage collector is not available, we use destructor.

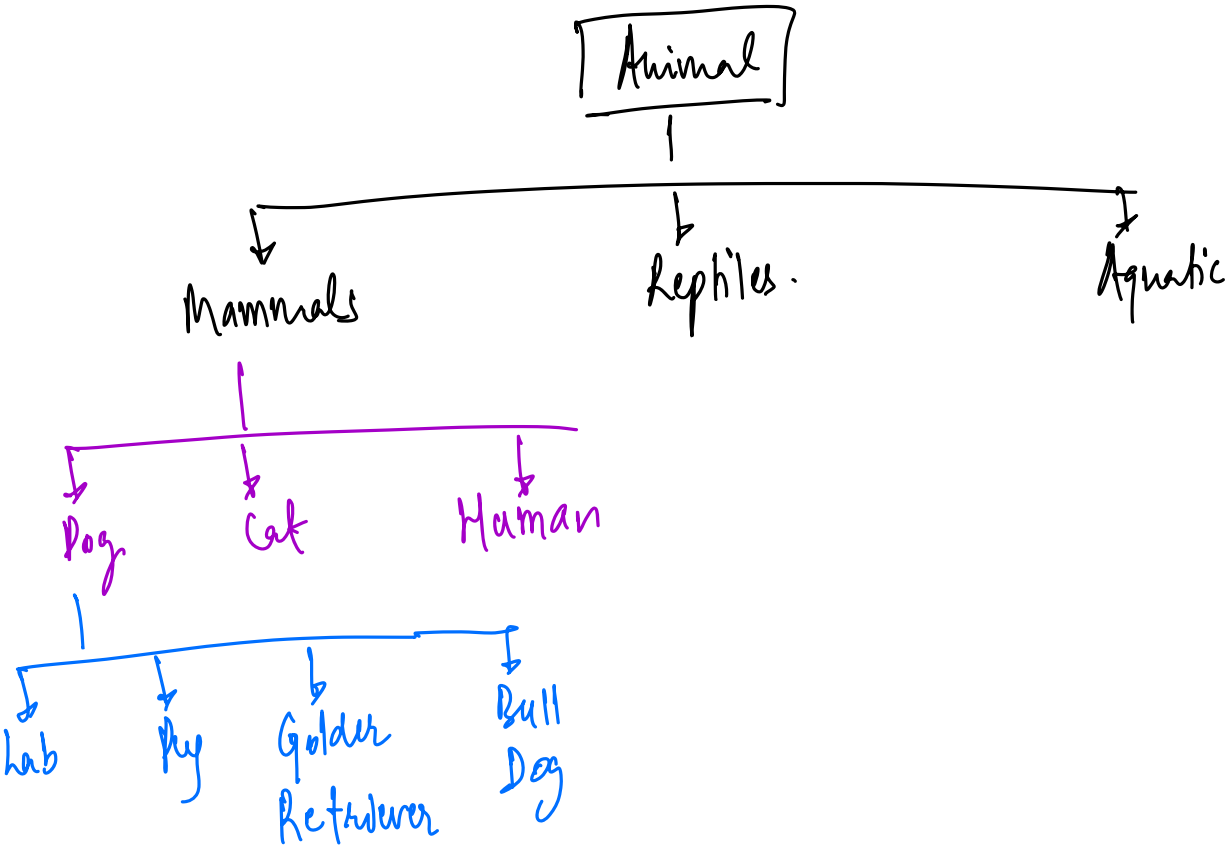
Destructors → (Optional)

→ opposite of constructor
→ Mostly used in languages where we have pointers.

Inheritance

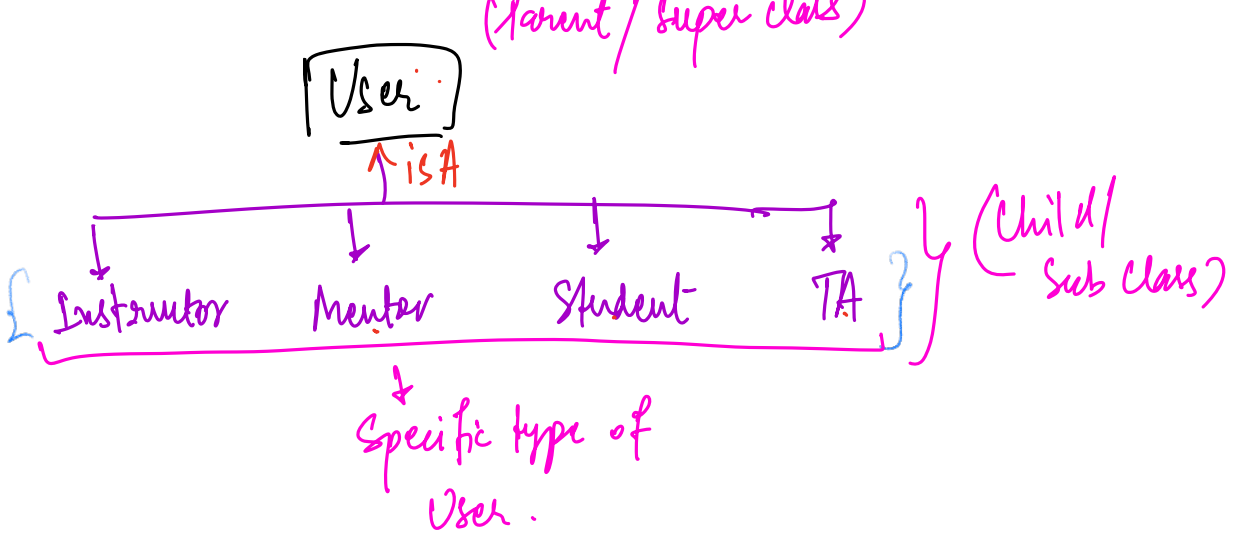
→ OOP is based on thinking about software systems.

→ OOP allows us to think in hierarchical way.



Lab → Dog → Mammal → Animal

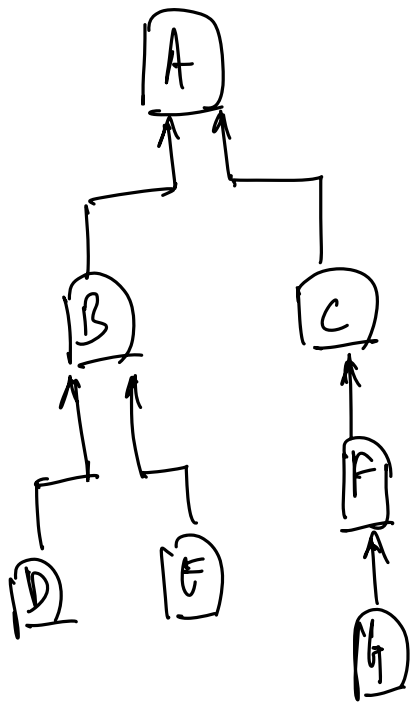
This representation of hierarchies in classes is known as Inheritance.



```
class User {
    String username;
    public void login() {
        |
    }
}
```

* A child class inherits all the members of the parent class and it may or may not have some of it's own as well.

Why Inheritance ? We will wait till Polymorphism



What is abstraction?
→ Yes, it's an idea.

Ques:- Which of the classes has highest level of abstraction?

A

Doubts

Student s2 = new Student(); → @800

Student s1 = new Student();

Student temp = s1;

s1 = s2

s2 = temp

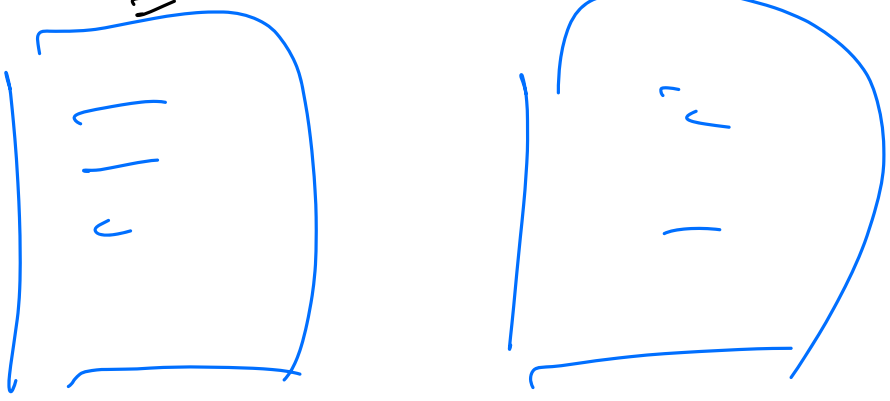
s1 = @700

temp = @700

s1 = @800

s2 = @700

doSomething(s1, s2);



doSomething(Stu s1, Stu s2) {

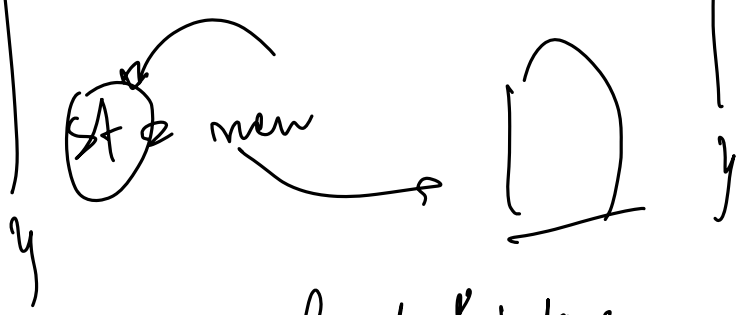
}

Student *stE = new

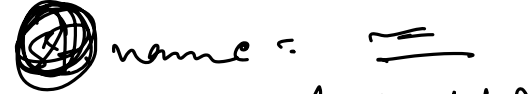
doSomething(*stE);

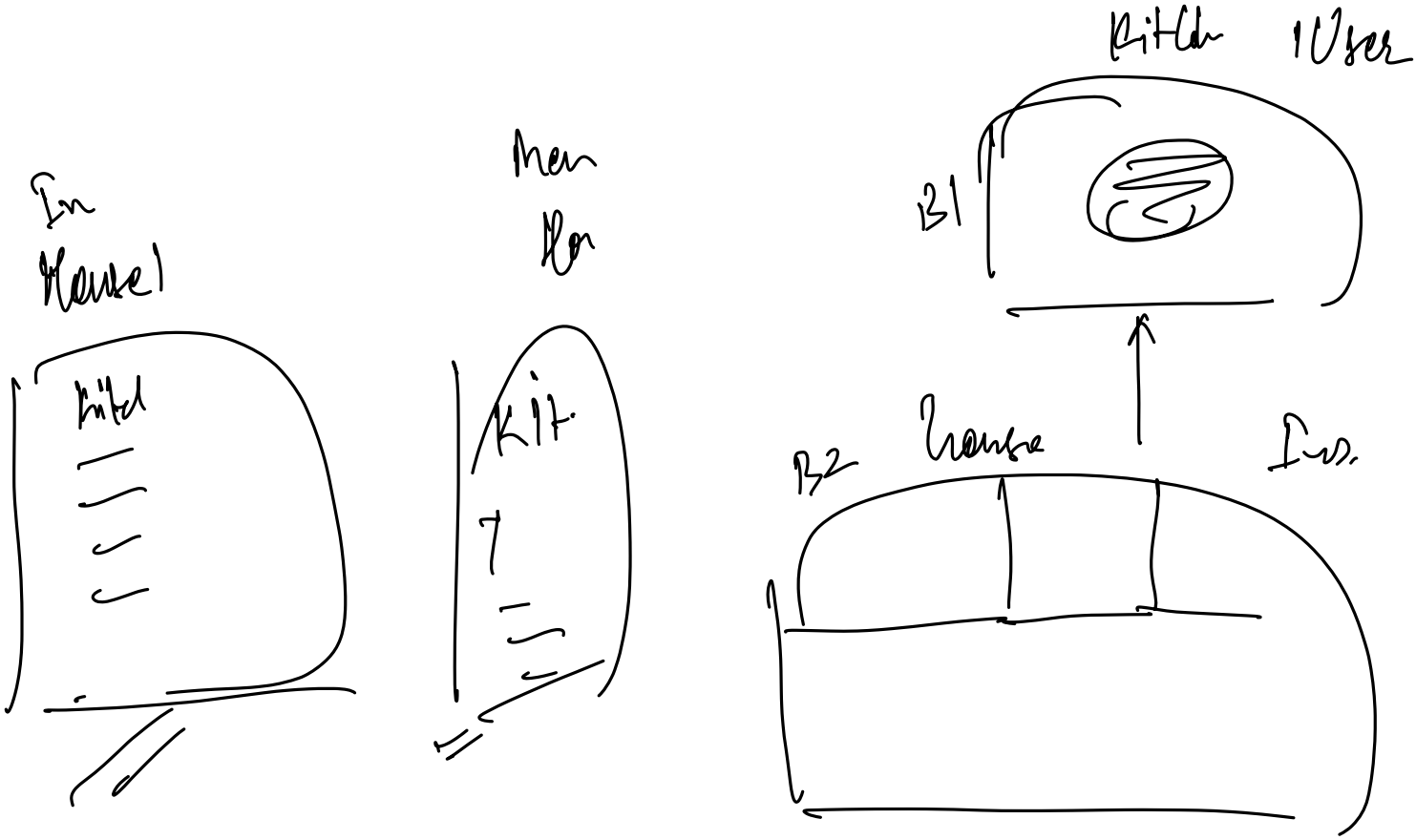
doSomething(Student stE) {

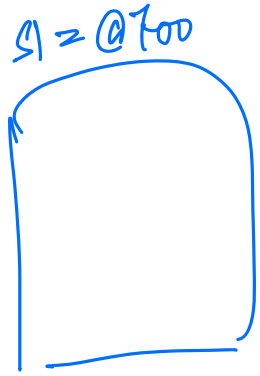
doSomething(s1 = 12)



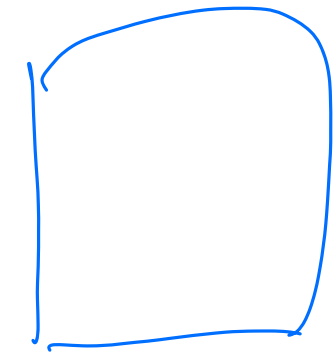
Read Pointers-







st1 = @800
st2 = @700




```
doSomething ( Student st1, Student st2){
    Student temp = s1;
    s1 = s2;
    s2 = temp;
}
```

st1 -> @800 st2 = @700

```
main() {
    Student s1 = ...;
    s2 -> @800 ...;
    doSomething( s1, s2);
    Sop ...
}
```

@700