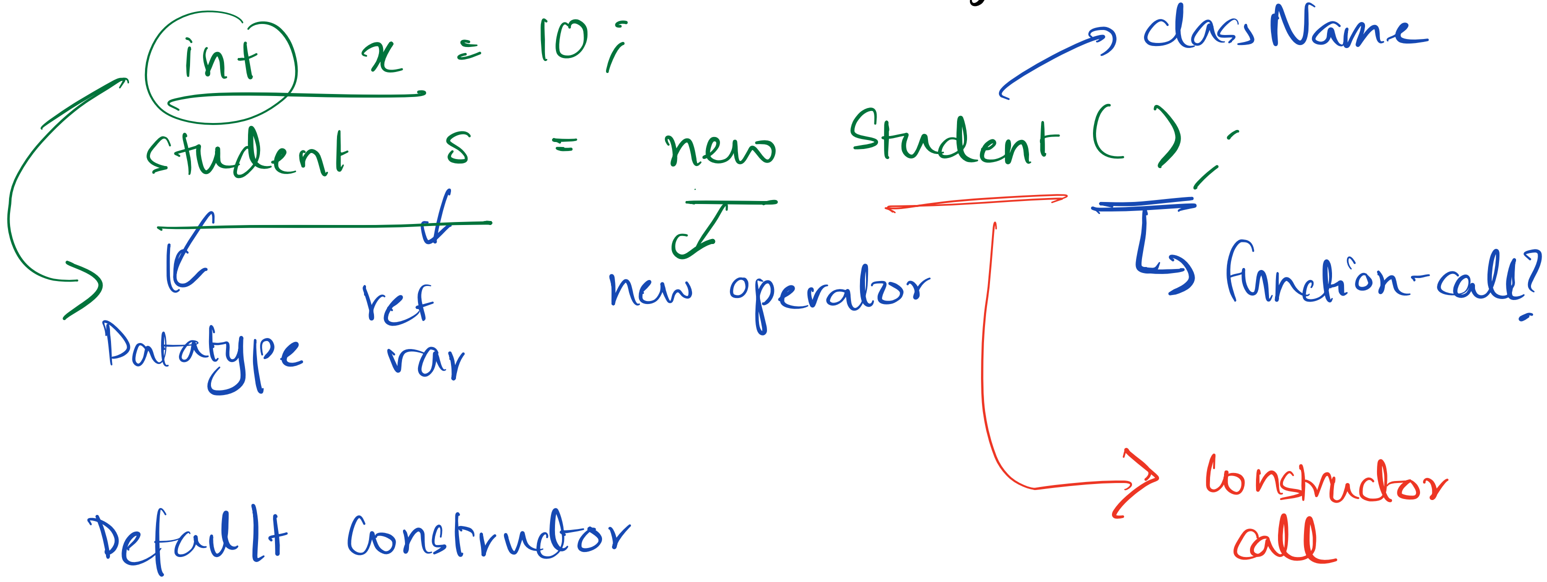
Agenda :

1. Constructors
   - Default
   - Parameterised
   - Copy
2. Shallow / Deep copy
3. Inheritence

Constructors :

Class : Blueprint of an object



Default Constructor

```
Class Student {

  int age;
  String name;
  String batch;

   Student () {
      age = 0;
      name =     null;
      batch =    null;
   }

}
```

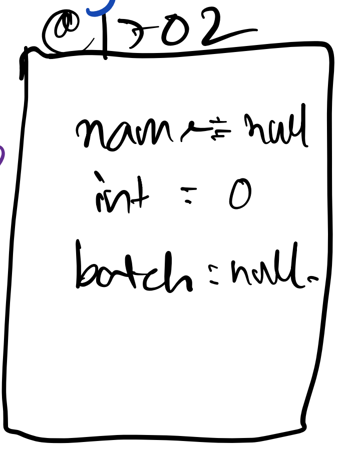
We can't see this code

this default constructor added by Java, not us.

Default Constructor :

1. Added by java when no other constructor is added by us.
   (java → compiler)
2. No return type (Always returns class type)
3. Assigns every attribute with default values.
4. Constructor name should be exactly class Name,

```
Student s = new Student();
s.name = "Akash";
```



Manual Constructors

→ Parametrised

→ Unparametrised.

```
Student( ) {
    name = "Default name";
    age = 10;
    batch = "Some batch";
}
```

```
Student ( ) {
    name = "Akash";
}
```

$\hookrightarrow$

```
name = "Akash"
age = 0 // default
batch = null;
```

Parameterised Constructor

```
Student ( ) {
    name = "Default name";
    age = 10;
    batch = "Some batch";
}

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

```
                                  int     String    String
Student S = new Student (10, "Akash", "Apr23");
            new Student (3); ✗  — Error in Java
            new Student ( );
```

In Case we have written only param. constructor

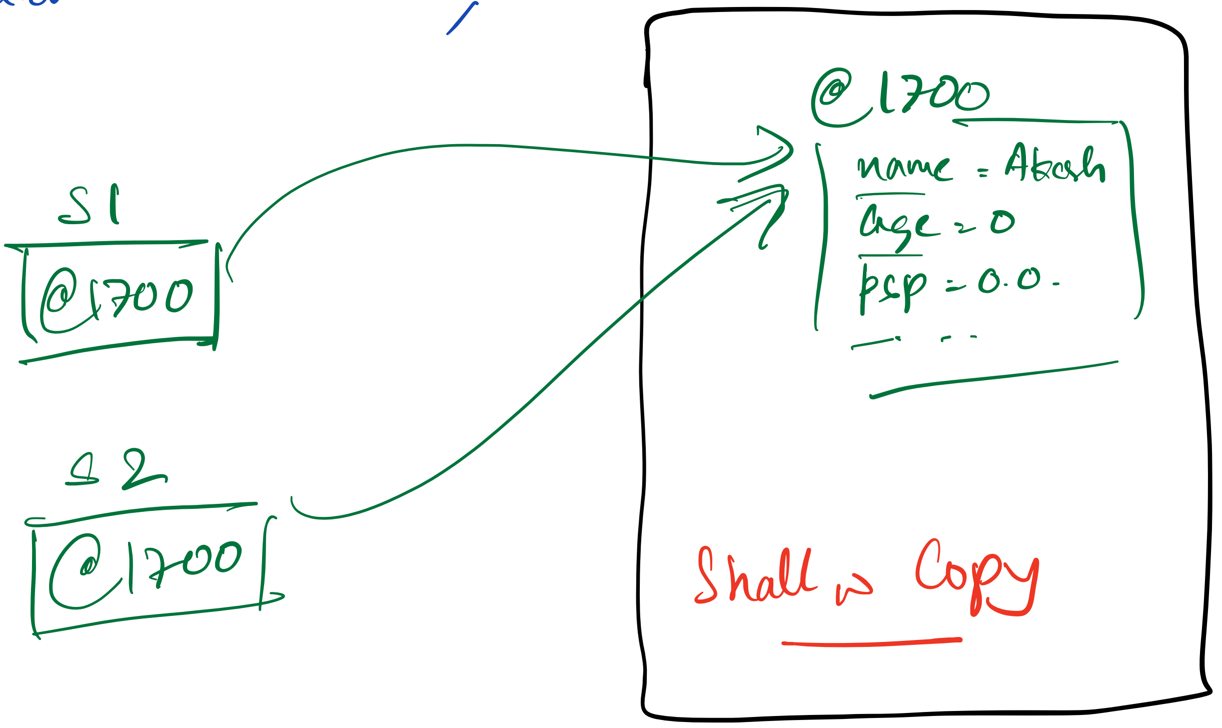
```
↳ new Student (10, "Akash", "Apr23");
  new Student (3); ✗ → Error in Java
  new Student ( ); ✗ → Error in Java
```

```
Student (int age)
{   if(age > 20) {
        this.age = age;
    }
}
```

Copy Constructor :

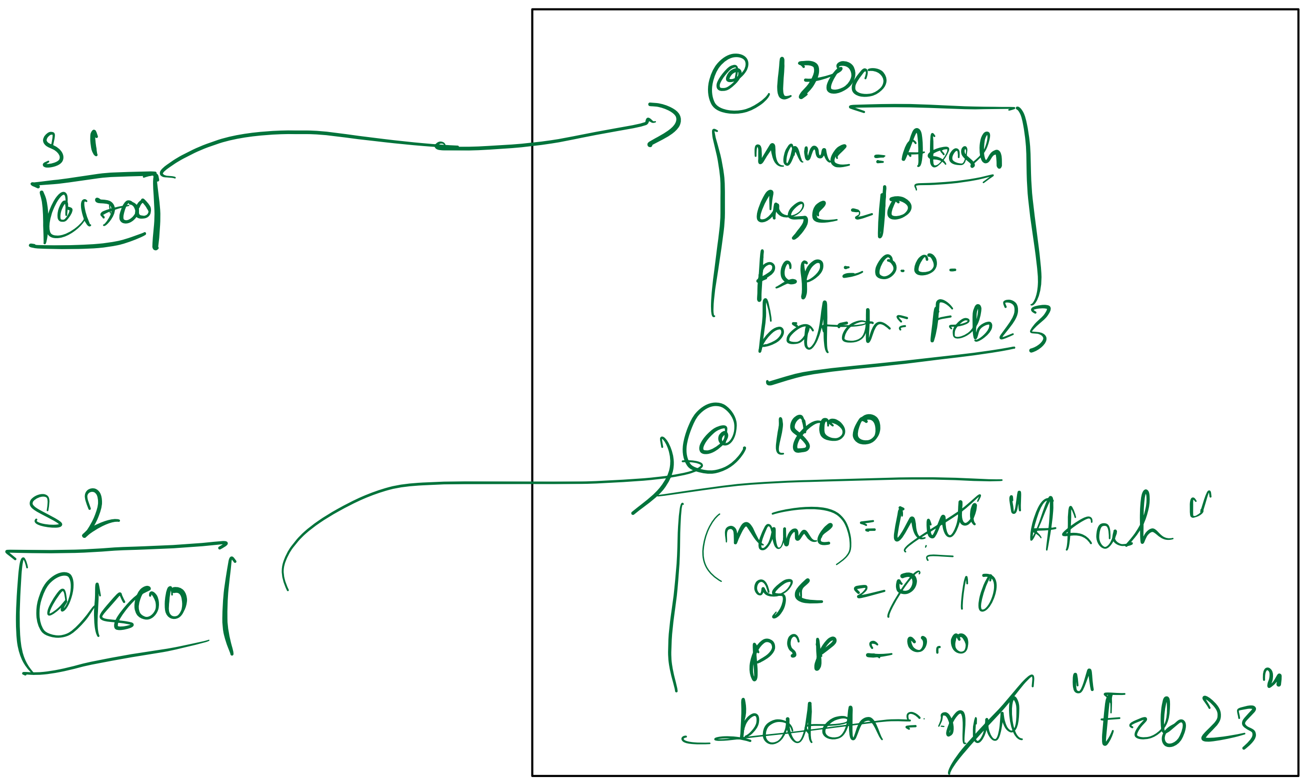
```
Student s1 = new Student("Akash");

Student s2 = s1;
```



```
Student s1 = new Student(10, "Akash",
                         "Feb23");

Student s2 = new Student();
  s2.name = s1.name;
  s2.age = s1.age;
  s2.batch = s1.batch;
  s2.psp = s1.psp;
```



Copy Constructor

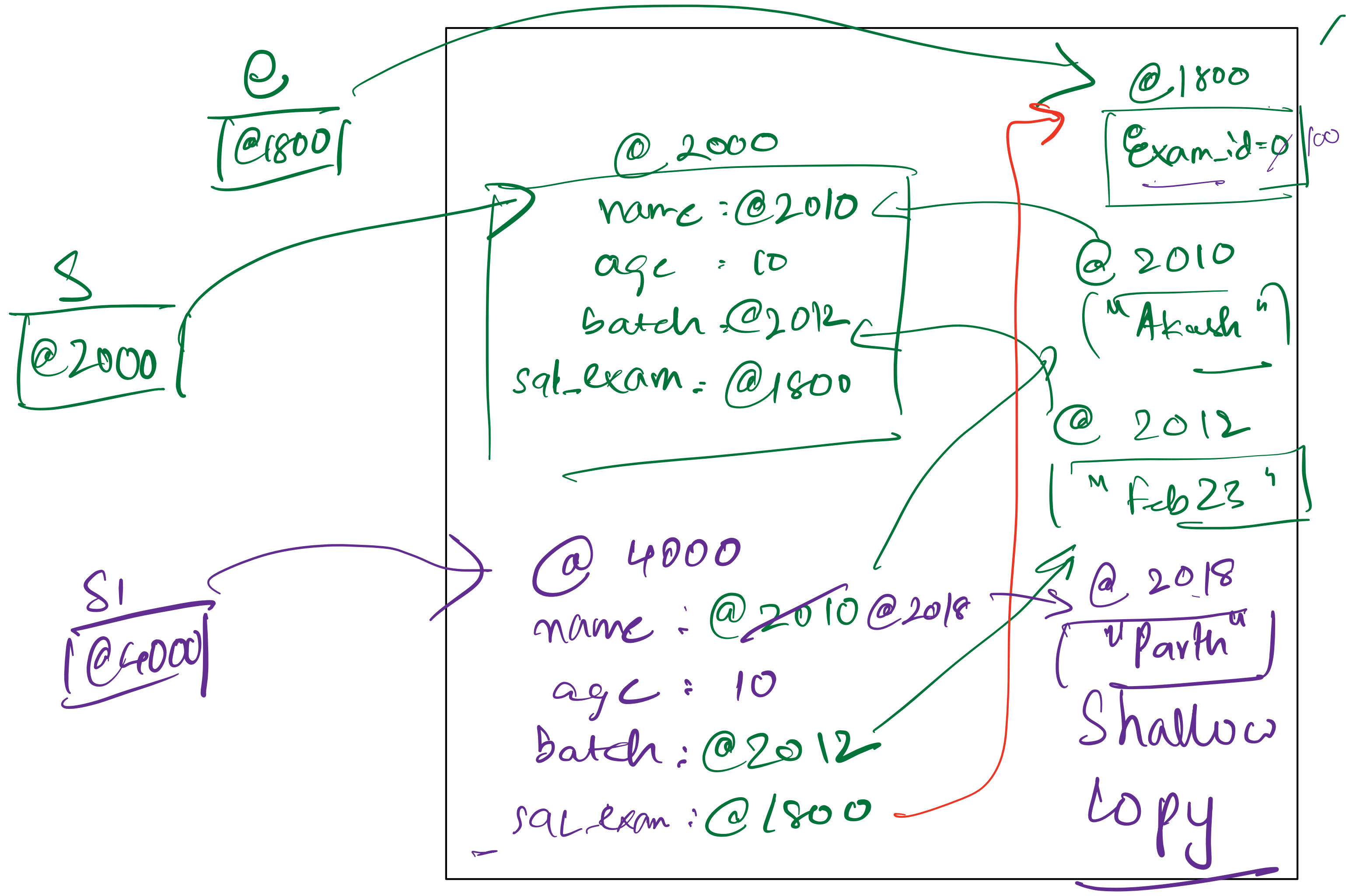
```
Student (Student s ){
    this. name = s.name ;
    this. age = s. age ;
    this. psp = s. psp ;
    this . batch = s. batch ;
}   this . exam = s. exam ;

Student S1 = new Student (10, "Akash",
                         "Feb23 ") ;

Student s2 = new Student (S1) ;

Exam e = new Exam() ;
Student s = new Student(10, "Ab", "Feb", e)
Student S1 = new Student (S) ;

                         this.sql_exam = e;
```



S1 . sql_exam . exam_id = 100

S . sql_exam . exam_id

S1.name = "Parth"

String a = "Akash";

internally $\longrightarrow$ new String("Akash");

a = "Parth";

internally : ↳ new String("Parth");

Break Till

$\boxed{9:07}$

Inheritence :

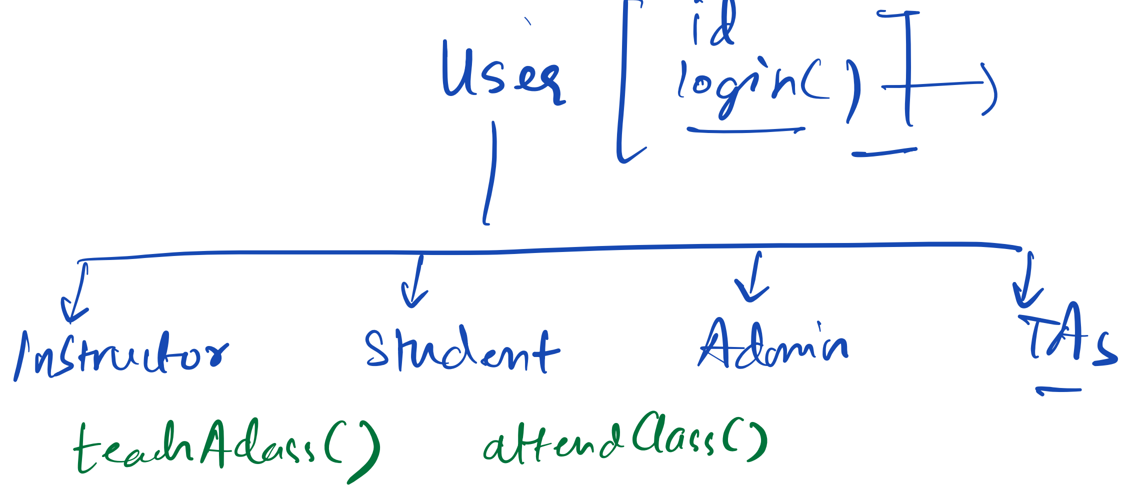
$$\begin{tikzcd}
 & & \text{Animal} \arrow[dl] \arrow[d] \arrow[dr] \arrow[r] & \text{breatne()} \\
 & \text{Reptiles} & \text{Mammals} \arrow[dl] \arrow[d] \arrow[dr] & \text{Aquatic} \\
 & \text{Dog} \arrow[dl] \arrow[d] \arrow[dr] & \text{Cat} & \text{Humans} \\
\text{GS} & \text{Bulldog} & \text{Lab} \arrow[r] & \boxed{\text{Share attributes \& behaviours}}
\end{tikzcd}$$

child
birth()

$$\begin{tikzcd}
\text{oven} \arrow[r, leftrightarrow, bend left] & \text{car}
\end{tikzcd}$$

start(); start();

$\rightarrow$ $\boxed{\text{logical relationship between the child \& parent classes}}$



- Does the child class have access to all inherited attributes & behaviours?

```
Class User {
  int user_id

}

class Student extends User {
      string name;

  }
```