

# OOP - II

## Encapsulation

- Constructors
- types of construction
- access modifiers

## Inheritance

- types of inheritance
- Advantages & Disadvantages

- Diamond problem

- JAVA

---

Small Talk

---

Encapsulation

① a single unit → class/object

② Info. hiding

---

A a = new A ( )



Construction  
ctor

Bank Account {



}

Syntax

function

access

Class Name

parameters

(            )

---

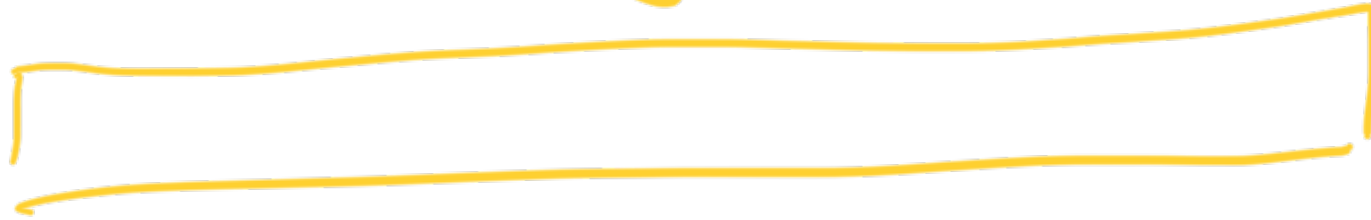
```
class Student {
```

```
    String name;
```

```
    Integer age;
```

String batchName;

}



Type → Default constructor

---

Compiler → a zero-argument  
constructor

public

Student () {}

}

}

Student a = new Student();

{  
a.name = "EK"  
a.email = ekadronain

a.name = Trump;

① unsecure

"\_A\_"

② name → .trim

---

no entry point for custom  
logic

---

Letter → setter methods

- get Name ()

- Set Name () — .trim ()

---

Student → psp — 0 — 100

— 110

then → entry point

Setter

→ fail object creation

Constructor

A()  
↳ a. b = 1

public

↳ Setters

↳ a



• A()

↳



in secure

↳

setters

secure

- 
- secure
  - validation ~~Sanitization~~
  - allows to fail object creation

A() - zero  
- nullary

Parametrised

↳ ctor + arguments

```
class Student {  
- int age }  
- string name }
```

~~Student (age, name)~~

```
{ ( this.age = age / 100;  
  this.name = name.trim()  
  throw exception()  
}
```

}

ctor

① [ compiler creates an instance

② assign values to fields

③ [ compiler returns the instance

```
Student s1 = new Student("EK", 1);  
Student s2 = new Student(s1);
```

s1.name = "Do" | s1.age = 20  
s2.name => "EK"





heap

String name  
name.append()

S2.name  
= "0"



modifies the accessibility of a class  
method  
field  
ctor

modifiers — non-access

access

→ final  
→ static  
→ synchronized

→ public  
→ private  
→ protected

default X = A

→ default

↑  
B

	Class	Package	Subclass	Global
<u>public</u>	Y	Y	X	<del>Y</del>
<u>private</u>	Y	X	<del>X</del>	X
<u>default</u> (package protected)	X	Y	<del>X</del>	X
<u>protected</u>	Y	Y	Y	<del>X</del>

subset

Student

Sage

---

## Inheritance

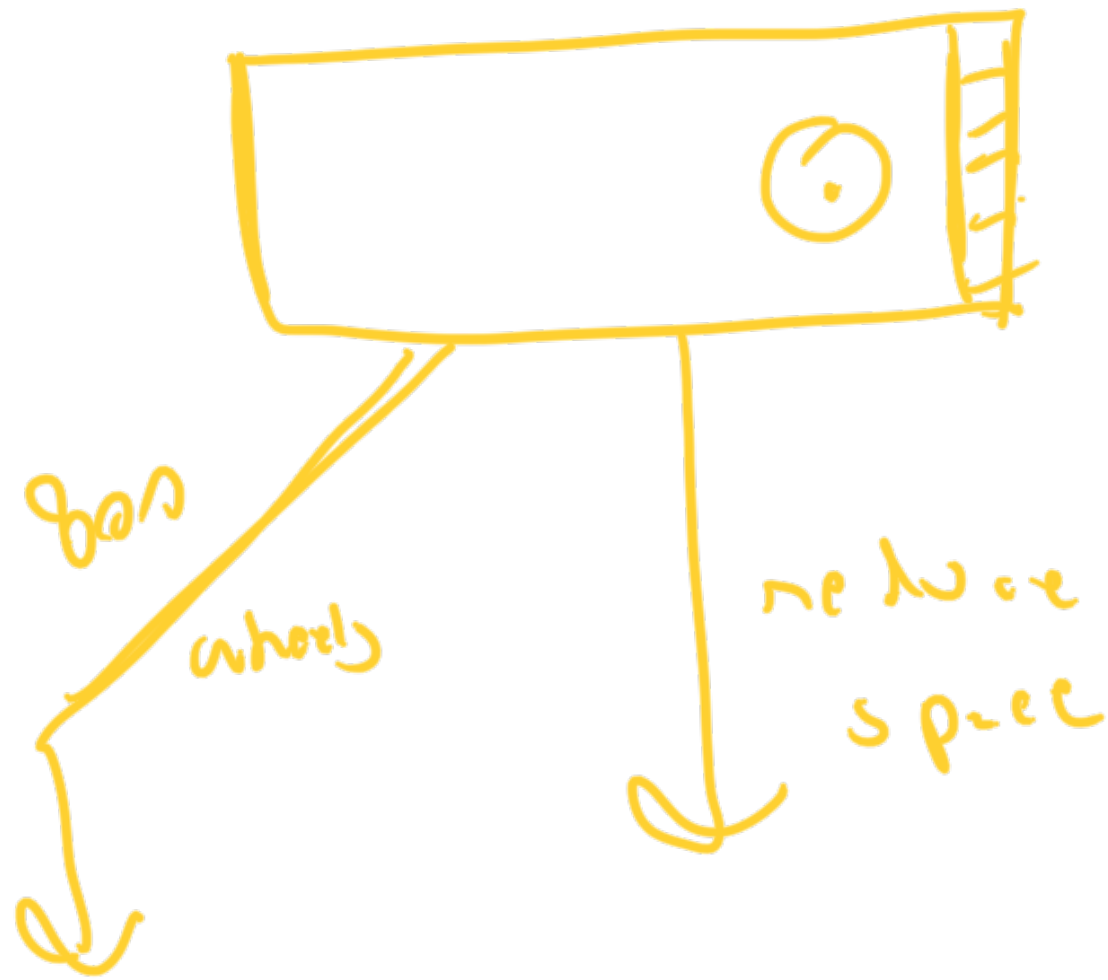


A + 0

Ritz

Manvi + 800





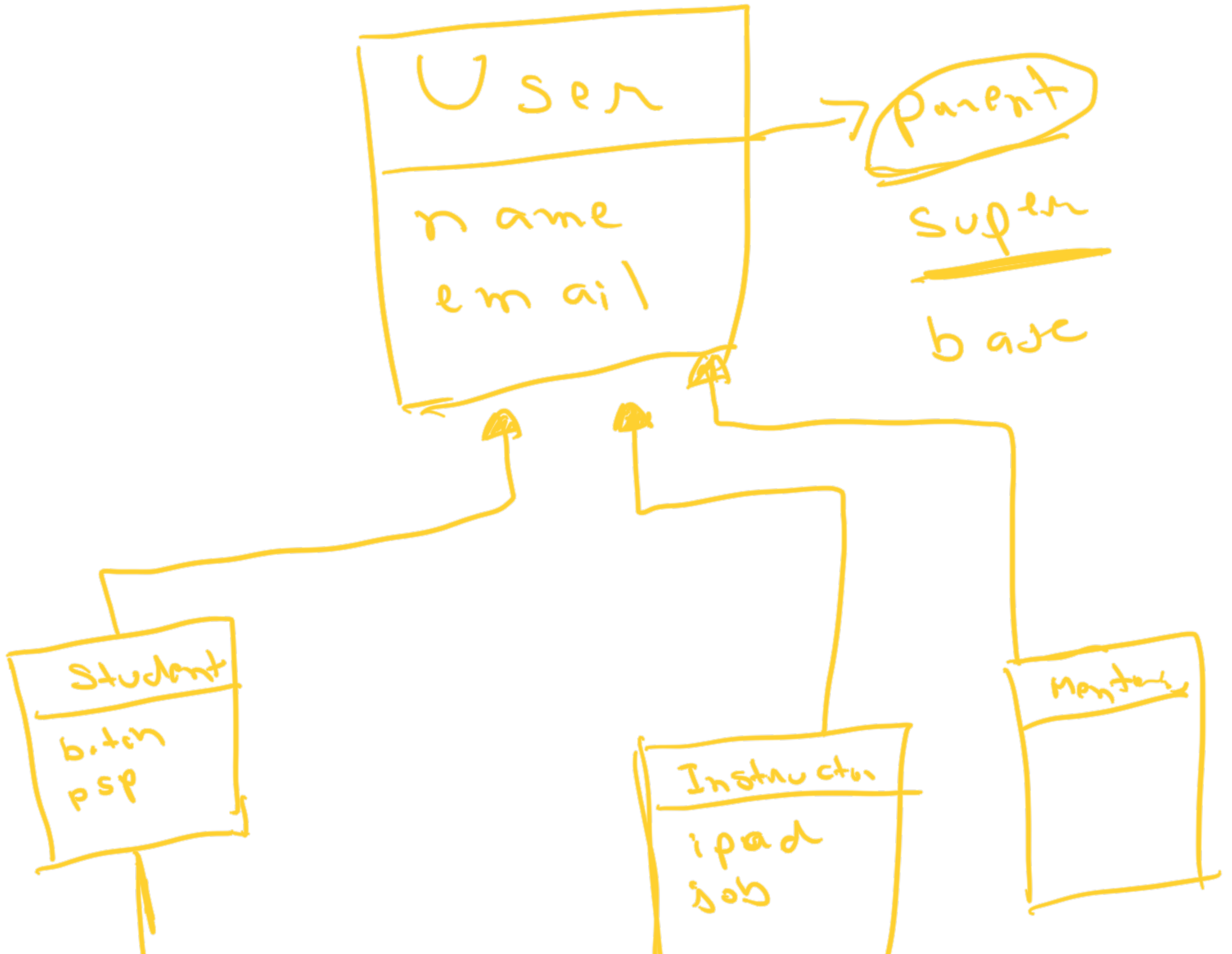
→ Student

→ Instructor

→ Mentor

name  
email  
address  
phone

duplication





- ① child class
- ② derived
- ③ Sub class

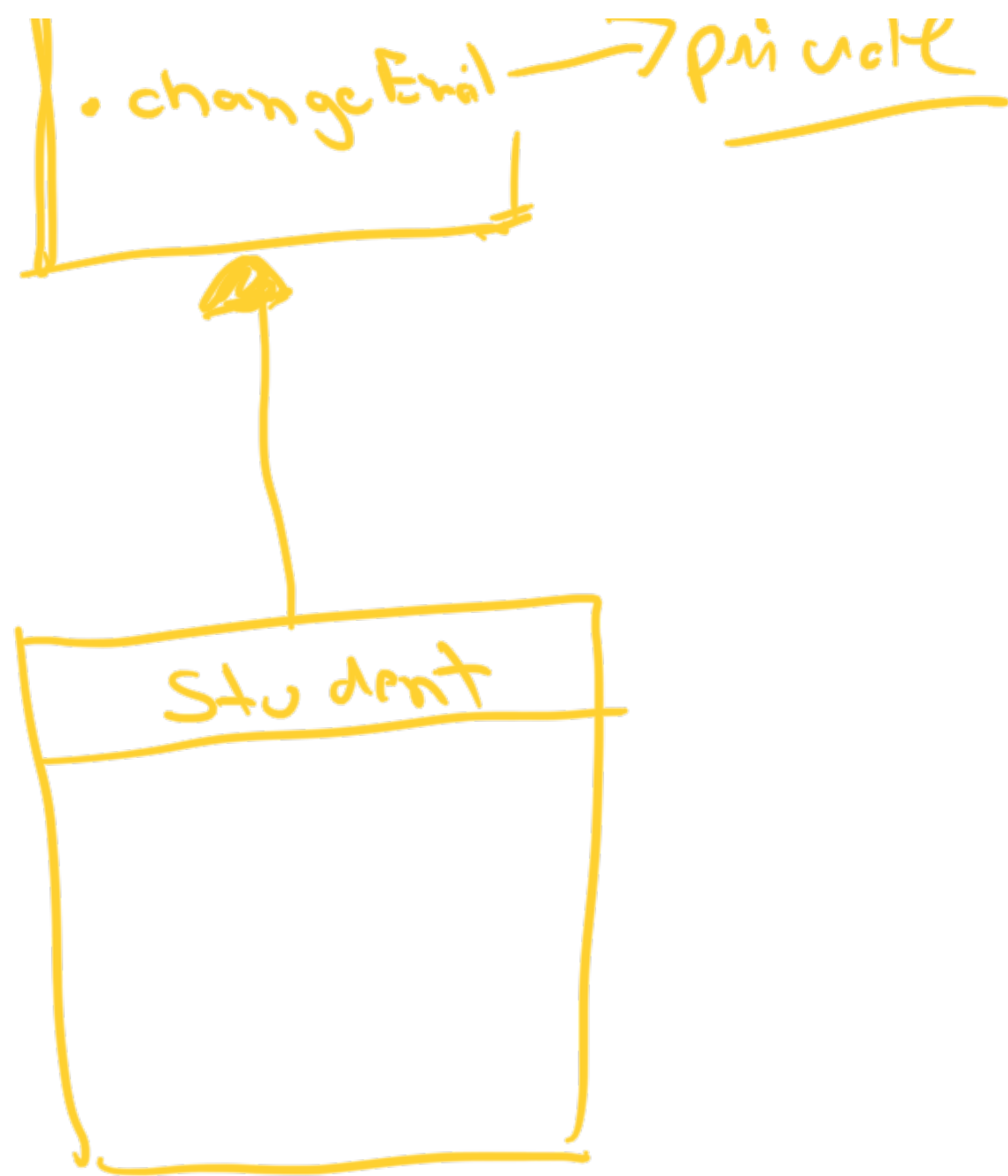
reusability —

• extends



class Student  
extends  
User

✓



Student

Private

# S. Change Email()

①

Single

Animal

Database

Dog

MySQL

Husky

Aurora

MariaDb

②

Multilevel

③

Hierarchical

Dea

Vodafone

husky — ~~husky~~  
dog  
|  
husky

④

Multiple

Mammal

User

Member

Java does not support multiple inheritance

---

Diamond problem