→ Builder Design Pattern

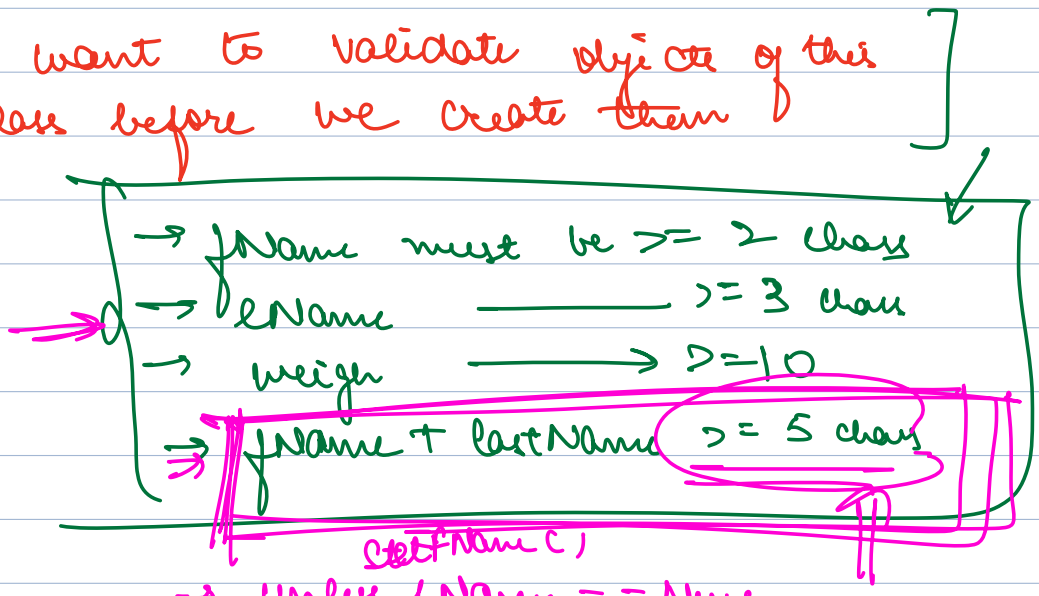
→ Factory Design Pattern

---

① Class that has a lot of attrs

```
class Student {
    fName
    lName
    age
    weight
    psp
    salary
    universityName
    phoneNumber
}
```

② We want to validate objects of this class before we create them



VI Class that has a lot of data

How to create obj and set prop

Class Student {

private
- fName
- lName
- age
- weight
- psp
- salary
- uniName
- phone Number

// getter Method
public int getPSP() {
return psp;
}

// Setter Method
public void setPsp(psp) {
this.psp = psp
}

}

Soln 1
1. Create an object using default cons.
2. Set value of attrs using getter / setter

Problems
1. Code will be big.

```
Student st = new Student();
st.setAge(12);
st.setFirstName(Naman);
st.
st.
st.
```

② How to validate obj ?
(especially validations involving multiple attrs)

③ Obj is immutable ⇒ can't change value of attrs of an object once it is created

[→ If getter and setter are public, I can always mutate.]

Sol^n 2

① Create a parameterized const of the class
② Pass the value of attr via const.
③ No getters setters

```
class Student {
    fName
    lName
    age
    weight
    psp
    salary
    uniName
    phone Number
}
```

```
Student ( String fName,
    String lName, int
    age, double weight,
    double psp, double
    salary, String uniName
    String phNo) {
    this.fName = fName
    validate ( );
}
```

(1) If class has 50000 many attr, code will be dgf to maintain.

(2) Non understandable as well as big prone code.

```
Student st = new Student ("ABC", "DEF", 12, 70, 70,
                          110, "KH2", 1234);
```

(ABC, DEF, 70, 12, 70,)

(3) Not all attrs are mandatory

```
Student st = new Student("ABC", null, null,
                         null, 123, null, null)
```

Class Student {

fName -

lName -

age -

weight -

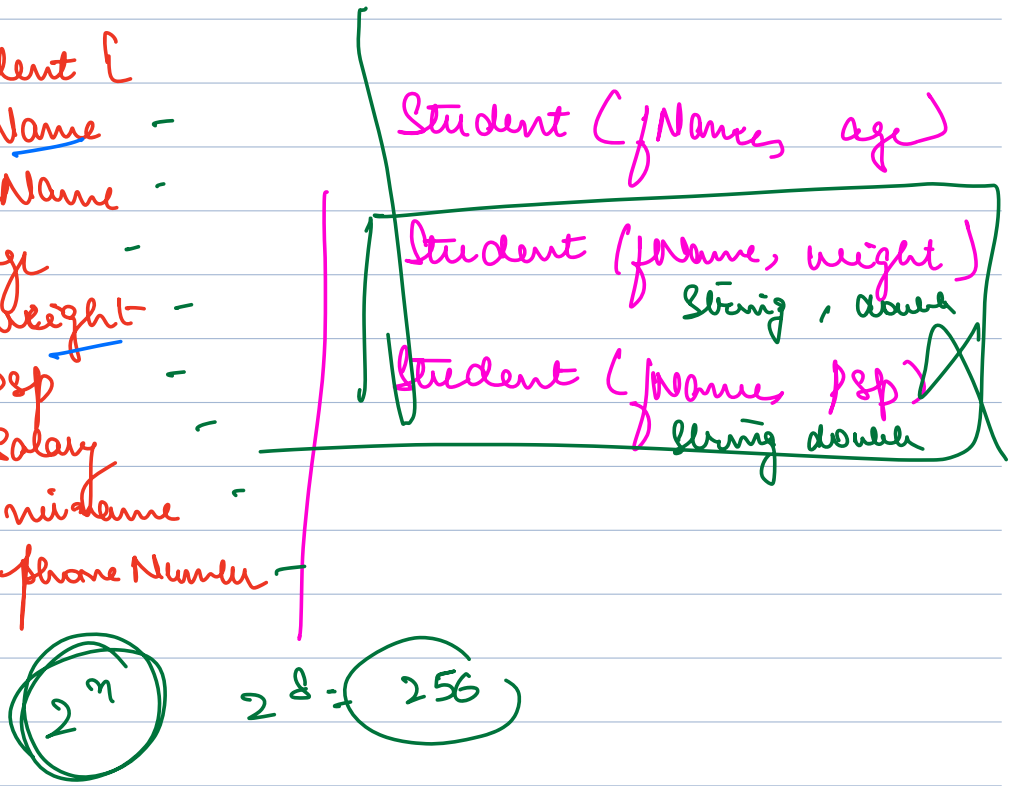
psp -

Salary -

universalName -

phone Number -

}



① Too many cons possibility

② Not all const may be possible to create

6 Cons

Student (fName, age)

this.fName = —

age = —

Student (fName, age, weight)

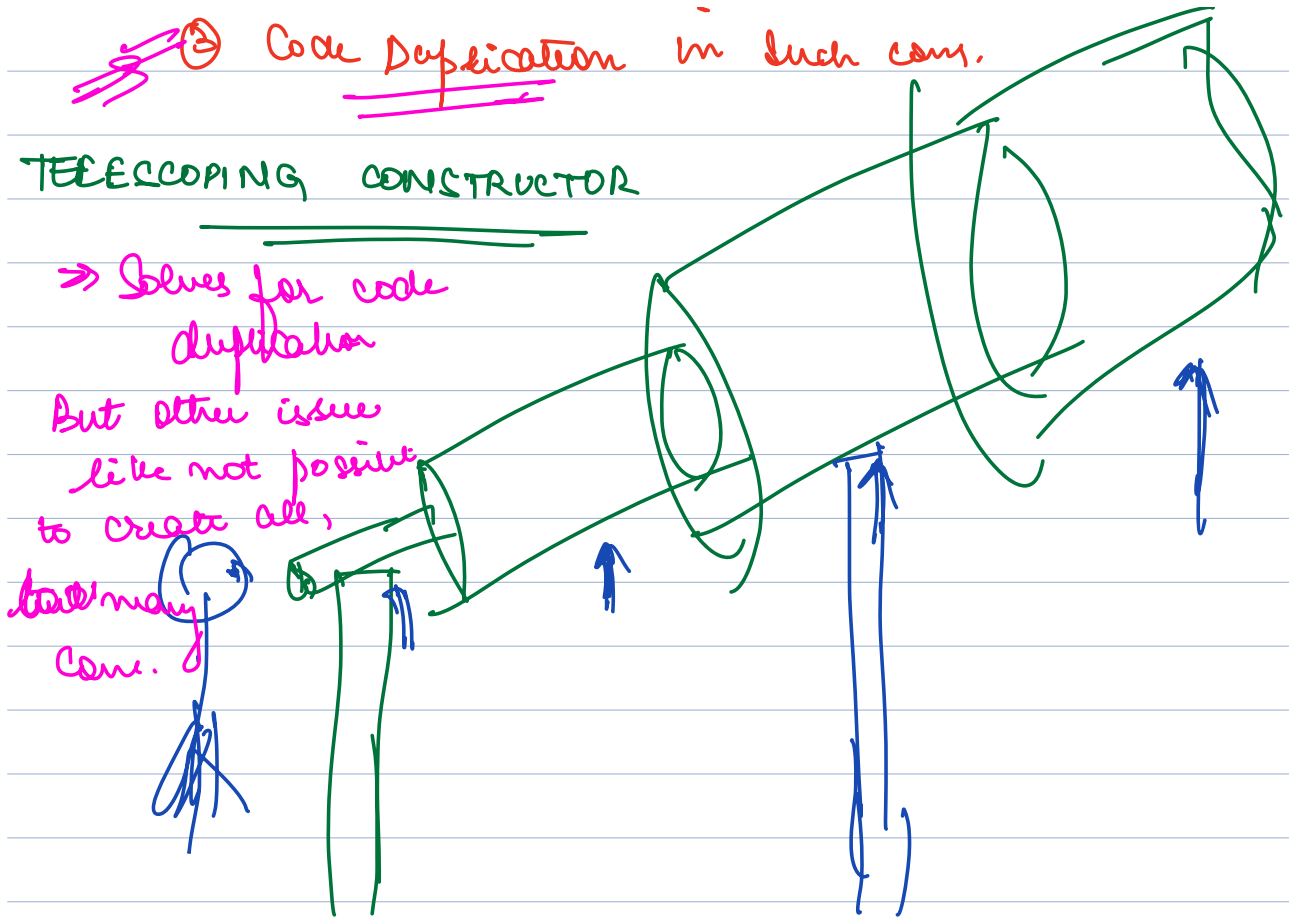
fName = —

Student (fName, lName, age, weight)

Student (fName, lName, age, weight, psp)

③ Code Duplication in such cons.

TELESCOPING CONSTRUCTOR



⇒ Solves for code duplication

But other issues like not possible to create all, too many cons.

Class Student {

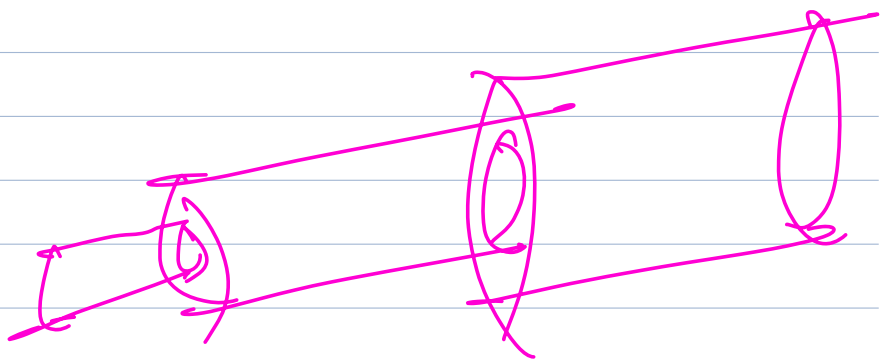
```
Student ( String fName) {
    this.fName = fName;
}

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

⇒ Too many cons

≠ Not all pos

}

```
Student (fName, age, psp) {
    this (fName, age)
    this.psp = psp;
}
```



⇒ We have a class that has many attributes. How will you create an object of that class?

Follow up

1. Obj should be immutable
2. Creating an obj should be easy.

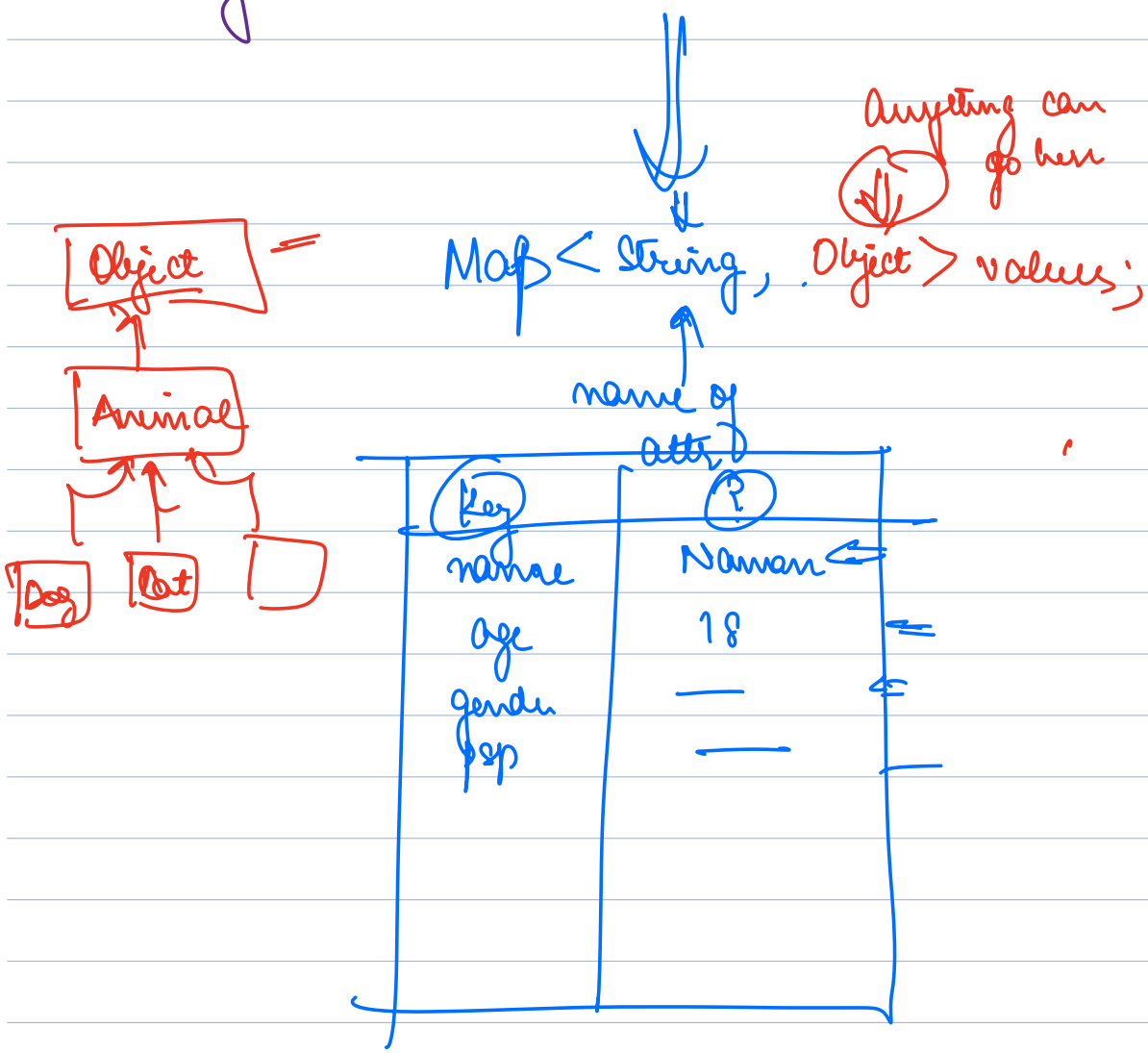
⇒ Whatever col^{n} has to come up from constructor.

- → There should be less (ideally one) constructor
- ⇒ Client code should be readable.

Student( )

→ we need to find some data structure that allows having multiple values within it

⇒ values inside DS should be recognizable by a name



```
Student {
    int age;
    String fName;
    String lName;
    double psp;

    Student (Map<String, Object> values) {
        if values.contains("age")
            age = (int) values.get("age");
        if values.contains("fName")
            fName = (String) values.get
        validation logic
    }
}

Client {
    psvm() {
        Map<String, Object> values;
        values.put("age", 12);
        values.put("fName", 131);
    }
}
```

// Diff to debug behaviour

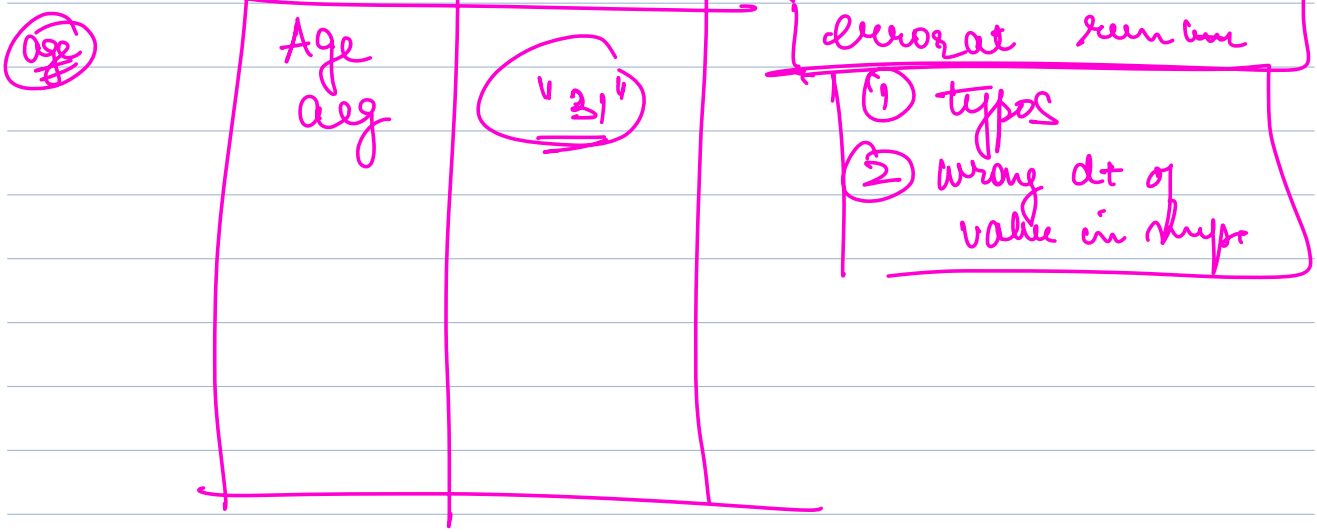
// Runtime Exc

OR.

// good codebase should find as many issues as possible at compile time

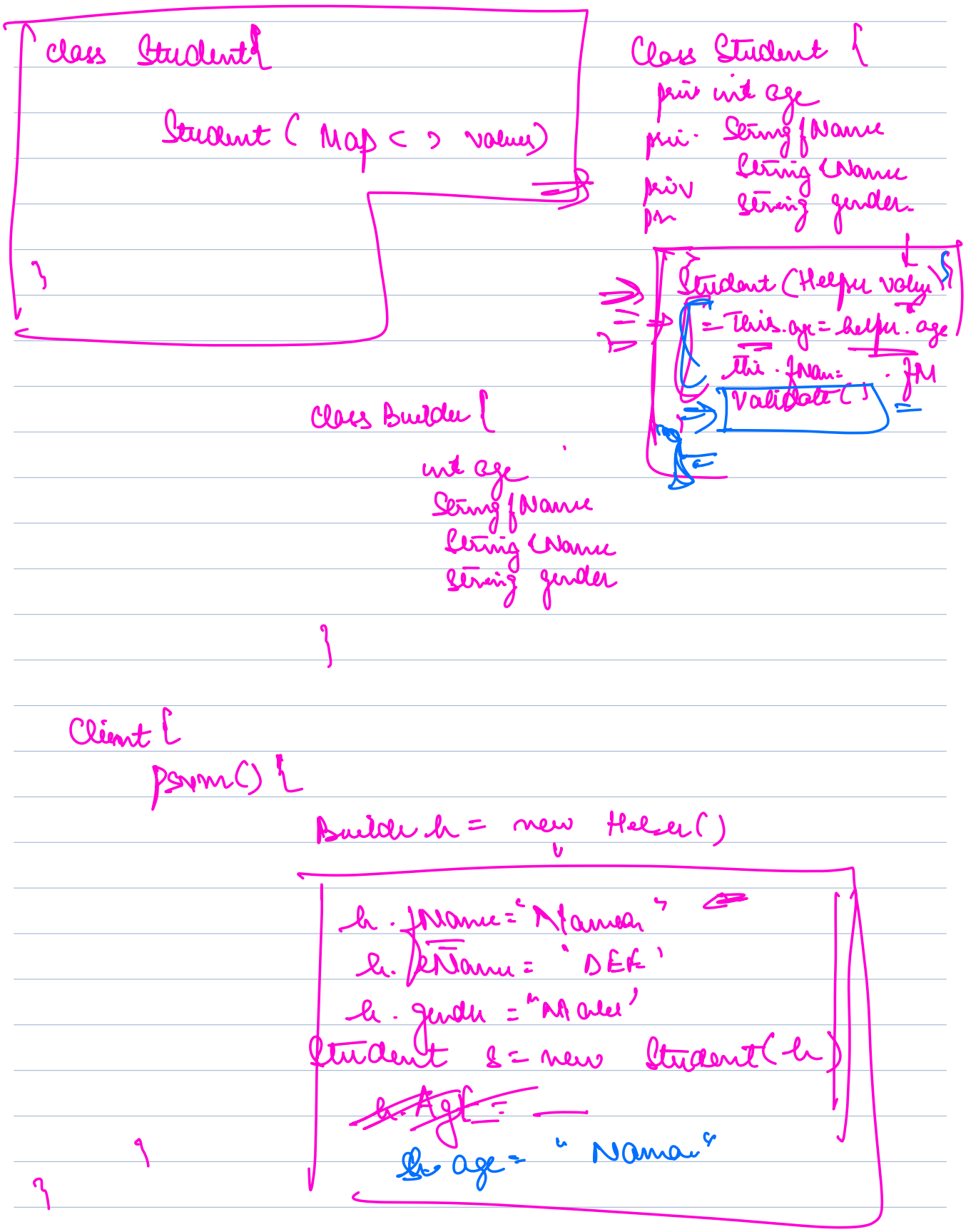
→ Good But we would expect type Safety
→ Avoid careless mistakes.

Map < String, Object >



Something

→ allows multiple values
→ every value has a defined data type
→ every value must have a defined name


class Student{
Student (Map < > values)
}
Class Student {
pri int age
pri String fName
priv String lName
pr String gender
Student (Helper valu
= This.age = helper.age
this.fNam = .fN
Validate ( )
Class Builder {
int age
String fName
String lName
String gender
}
Client {
psvm() {
Builder b = new Helper()
b.fName = "Naman"
b.lName = "DEK"
b.gender = "Male"
Student s = new Student(b)
s.Age =
s.age = "Naman"
}
}

When to use Builder

- → class with many attr
- → validate obj of a class
- → immutable obj of class with many attr