Builder

mutable
↓

Student Student = Student.builder() → values are default
↓
immutable

.id()
.name()
.phone()
.build()

← Prototype and Registry DP →

⇒ Prototype :- early version of something, and we create multiple iterations of it, until finalised.
The end product is nothing but replica of the final prototype

Prototype DP is used to create replica/clones of existing objects.

→ Classmate Notebooks

→ backPageInfo

→ frontPage Design

→ **Notebook**

- noOfPages	**10k copies**
- gsm	→ 120
- height	→ 1
- width	→ 10cm
- weight	→ 8cm
- typeOfPage	→ 100gms
- mrp	→ ruled
- backPageInfo	→ 30
- frontPageDesign	→
	→

Start. manufacturing 10k copies, without

→ backPageInfo

→ frontPageDesign

we have to create 100 objects, 8 out of 10 attributes are same values !

i) pass all the attribute values to all objects

ii) keep creating clone of 1 object, and just pass values of 1/2 attribute.

better way

```
Class Student {

        private String batch;
        private Date joinYear,
        private String moduleName;
        private int psp;
}       private String paymentPartner;  → no getter/setter
```

```
Student s1 = new Student("May23", 2023, "LLD", 99);
              ↓
Student S2 = new Student();

        S2.setBatch(S1.getBatch());
        S2.setYear( S1.getYear());
        S2.setModule( S1.getModule());
        S2.setPSP( S1.getPSP());
```

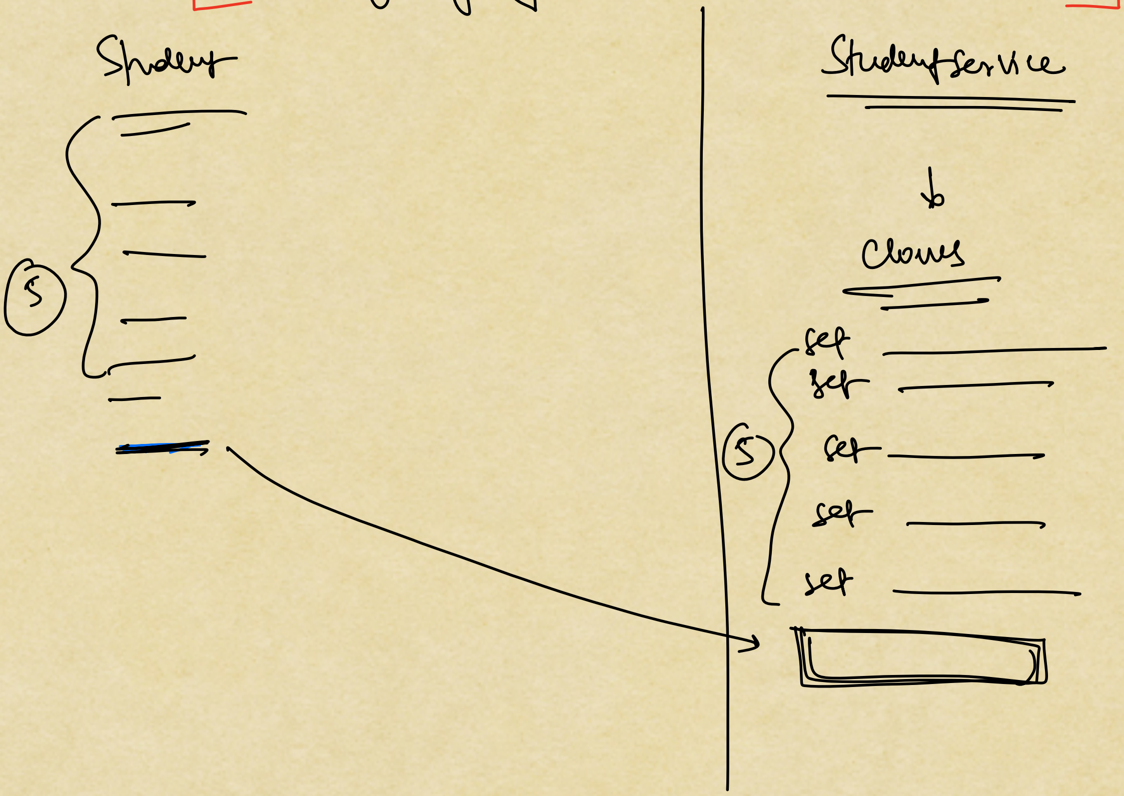
~~S2.setPaymentPartner(S1.getPaymentPartner()~~);

problem

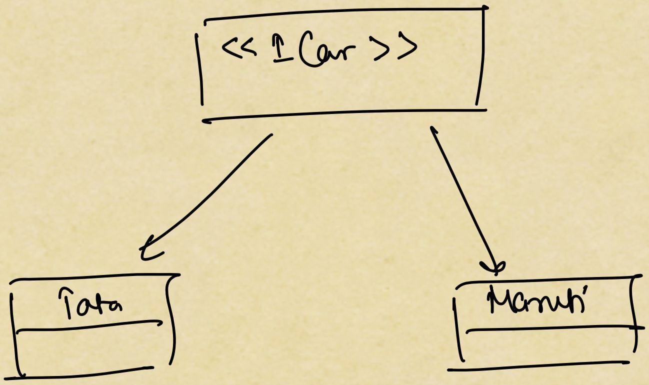
→ Dependent on the implementation of Student class

→ tightly coupled with Student

→ If anything is hidden it can't be clones.



⇒ Clone via copy constructor :-



```
public void sellCars(List<ICar> car) {
        clones
}
```

```
for( Car c : cars) {
        if( c is instanceof Tata)
                new Tata(c)
        else
                new Maruti(c)
}
```

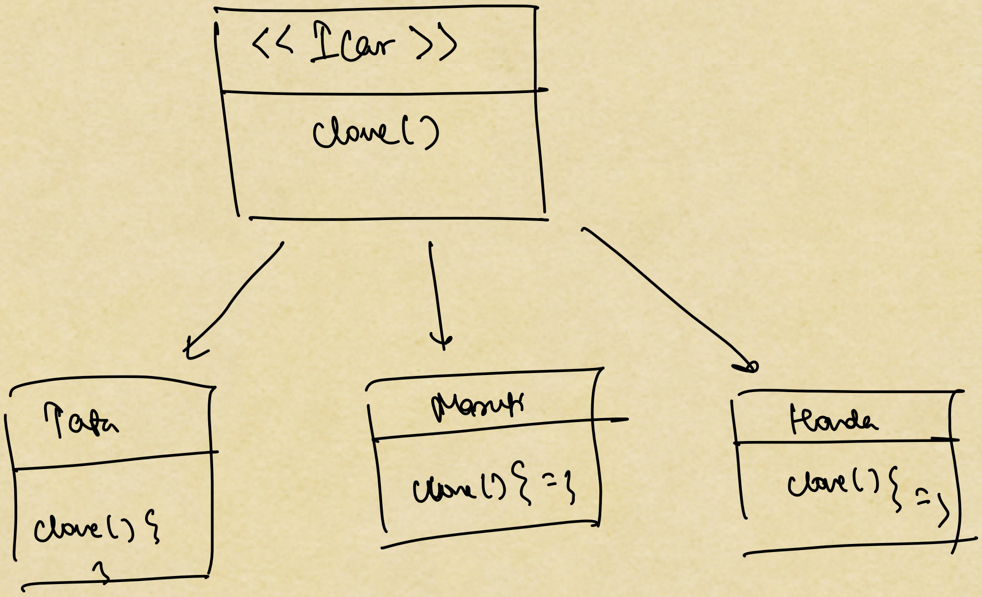
→ c is instance Hyundai

Honda

BMW

,

T

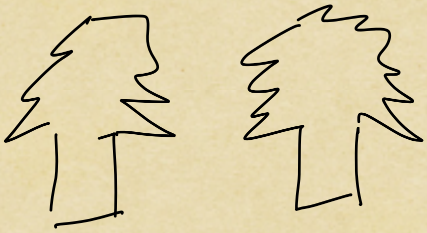


```
for(ICar c : cars) {
        c.clone();
}
```

{Problem Statement}

→ Clone objects,

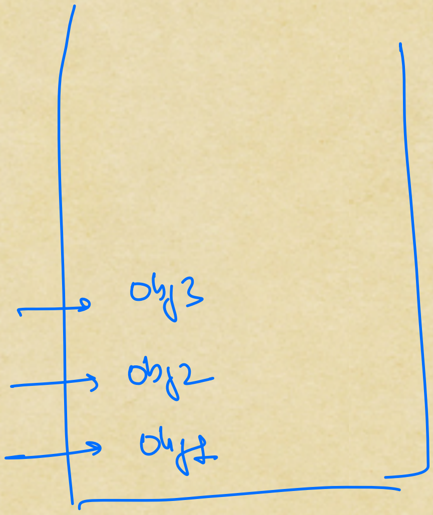
scenarios where majority attributes remain the same, and some minor changes are present.



→ tees in games

file => object => changes → content

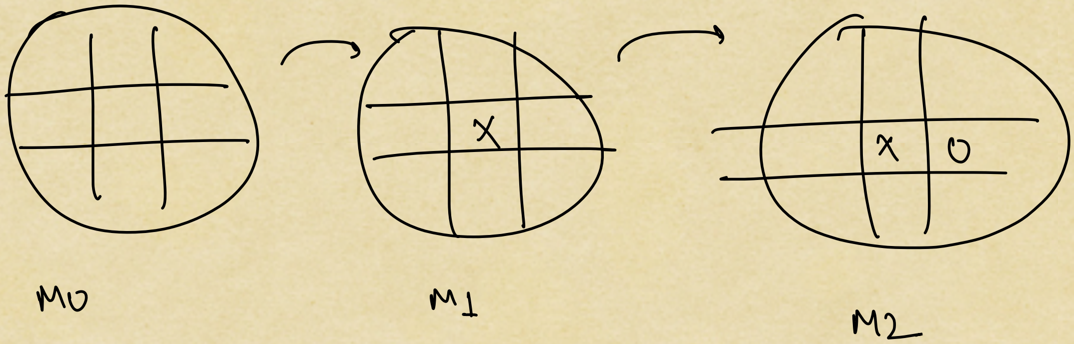
- → content → only changes
- → color
- → name
- → CreateDateTime
- → exportPDF

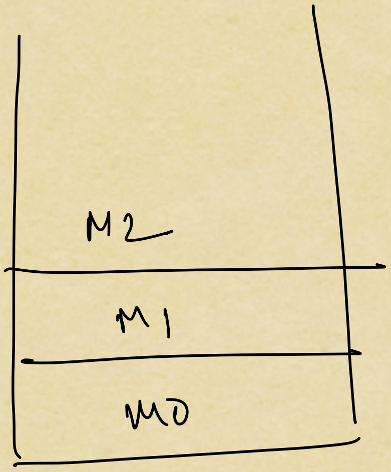


TicTacToe

↓

undo





→ Solution

i) getter/ setter

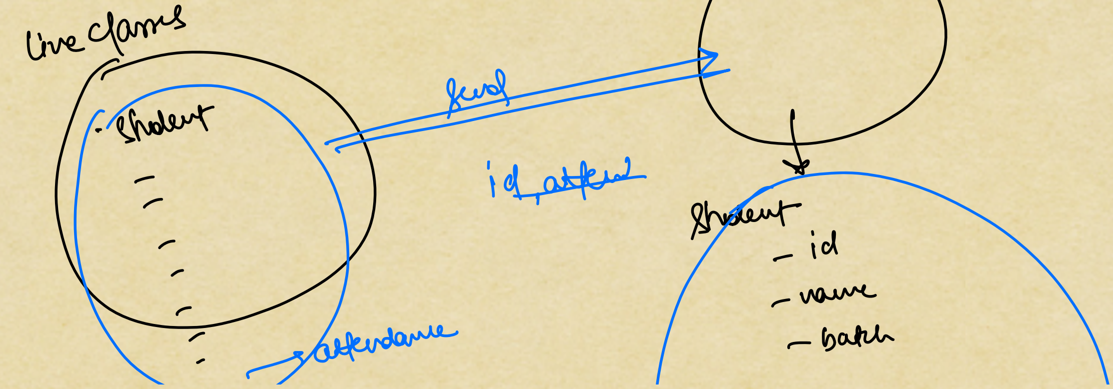
ii) deep copy constructor => fails when multiple children classes of any given type

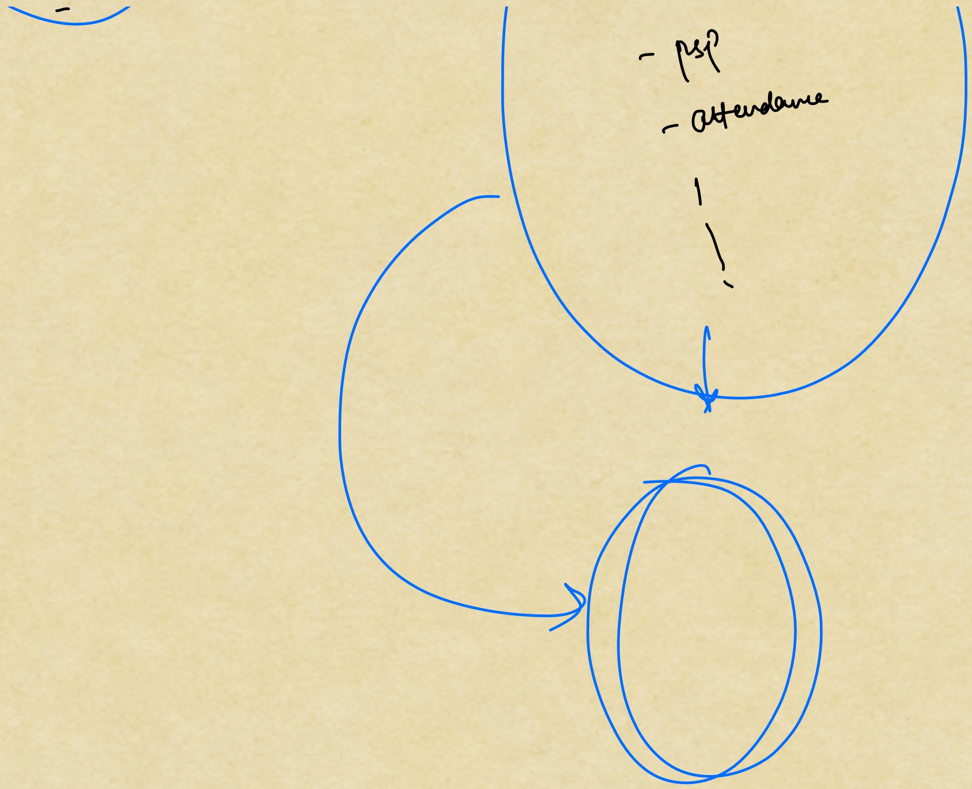
iii) Prototype DP

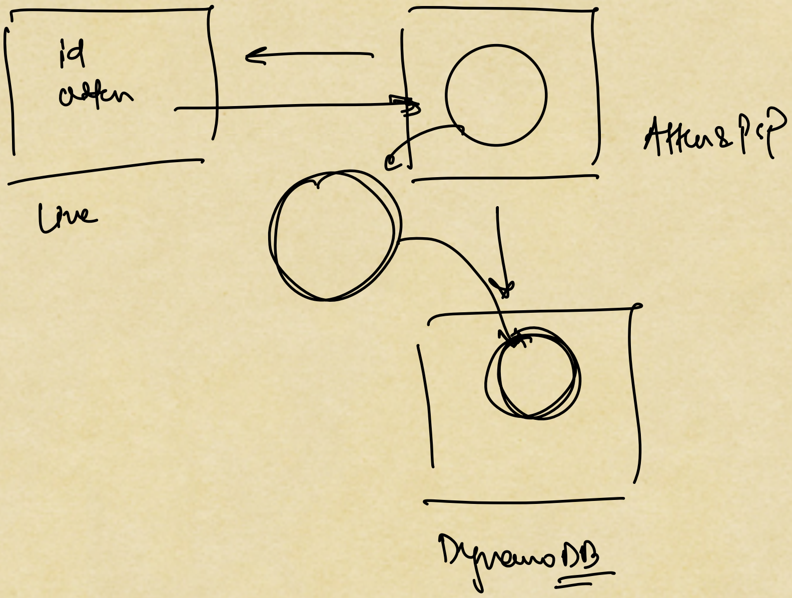
↓

```
interface Prototype<T> {

    T clone();

}
```

Microservices




- rsvp
- attendance


id
atten
Live
Atten & rsvp
DynamoDB