3rd Sept 2023 ⇒ Sunday ⇒ 8:30 PM

Extra session

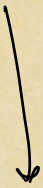
⇒ Builder Design Pattern → Creational

⇒ Imagine a class with lots of attributes.

```
class Student {
    - name
    - age
    - psp
    - batch
    - id
    - univName → can't be null
    - gradYear → can't be null (<= 2022)
    - email
    - phoneNumber
}
```

Creating a Student object,

Validation ⇒ they should have atleast 1 yr of exp.
they should have same graduation year



validate the Student object that got created

[object creation needs both memory and processing]

validation fails

Somehow validate the attributes before the object is created.

makes the code more readable and intuitive

make things immutable

Immutable

↓

Once the object is created, it can't be changed

nothing

↑

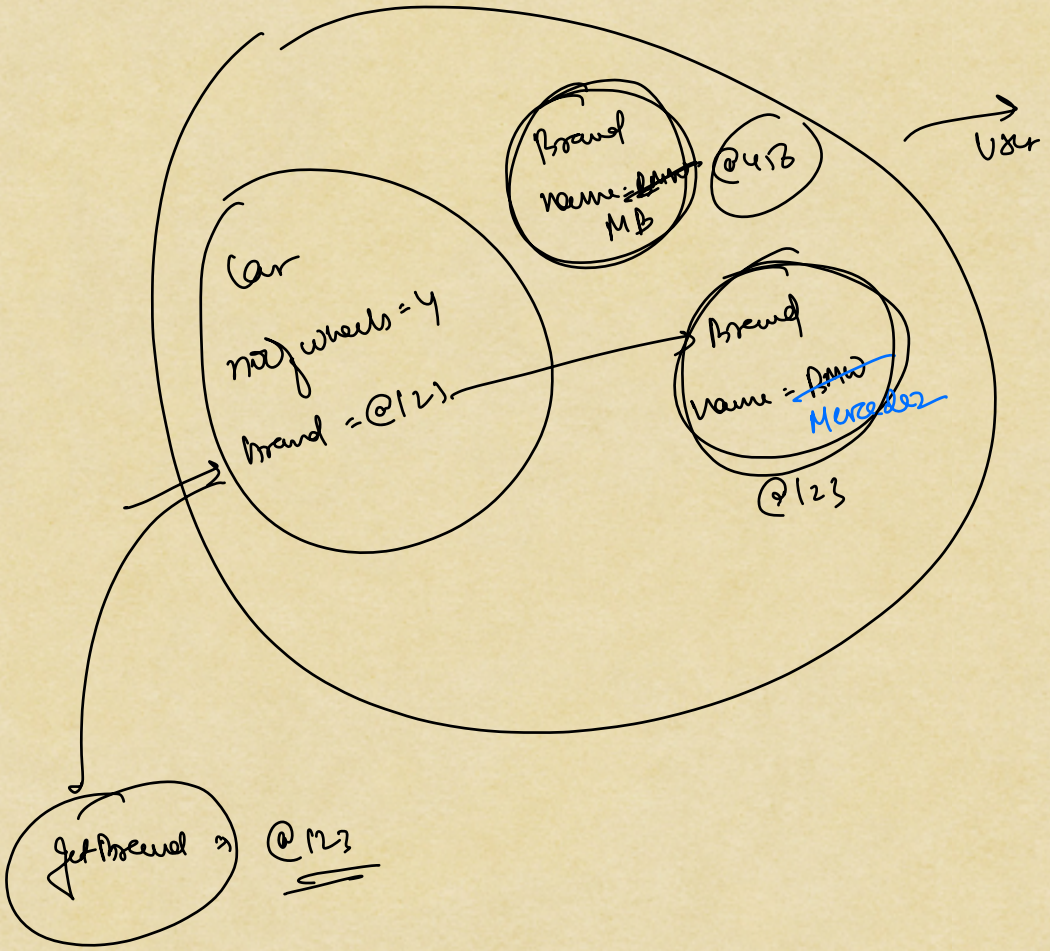
no value

immutable

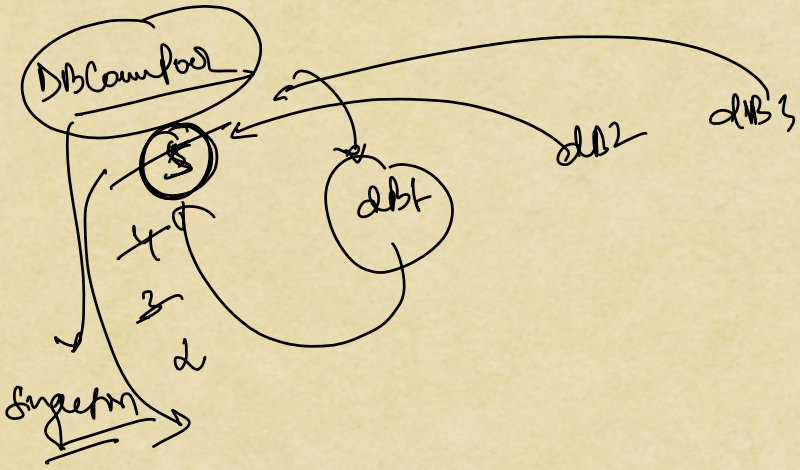
```
class Car {

    private final int noOfWheels;

    public Car(int noOfWheels) {
        this.noOfWheels = no.ofWheels;
    }

    public int getNoOfWheels() {
        return noOfWheels;
    }

}
```

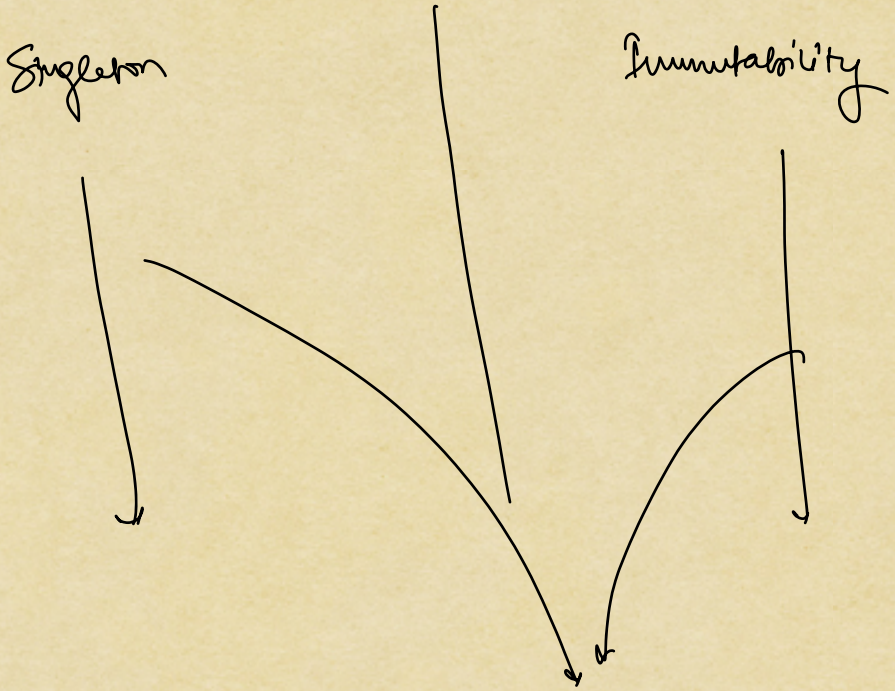
non-immutable

```
final class Car {
    private final int noOfWheels;
    private final Brand brand;

    public Car(int noOfWheels, Brand brand) {
        this.noOfWheels = noOfWheels;
        this.brand = brand;
    }

    public int getNoOfWheels() {
        return noOfWheels;
    }

    public Brand getBrand() {
        deepcopy of brand
        return deepcopy
    }
}

class Brand {
    private String name;

    getter / setter
}
```




```
class Bike {

    private int engineSize;
    private static Bike instance;
    private Bike() {

    }

    getter/setter engineSize;

    public static Bike getInstance() {
```


DBConnPool
DB3
DB2
DB1
4
3
2
Singleton


Singleton
Immutability

=> Validations:

i) no student should have grad year ≥ 2023

ii) phone number must be valid

iii) age >= 18

iv) univ. name should not be null.

* No student object should get created until these validations are done.

```
class Student {
    -name
    -age
    -psp
    -batch
    -id
    -univName
    -gradYear
    -univ
    -phoneNumber
}
```

parameterised constructor

setters

→ validations in parameterised constructor :-

- messy / confusing code
- multiple / duplicate validation code incase of constructor overloading
- object created the moment we enter the constructor.

→ validations in setter methods :-

- Can't use setter methods until Object is created.
- Object remains in heap if validation fails.
- messy / confusing code

Solution

I/p values on a hashmap

id — 1

name ——

age ——

email ——

validate (HashMap) {

→ throw an exception

}

class Builder {

- name
- age
- PSP
- batch
- id
- univName
- gradYear
- email
- phoneNumber

}

→ create the builder object

→ validate everything

→ use this obj. to create a student obj