Agenda : ① Prototype & Registry design pattern

Class starts at 9:05

Problem statement ⇒ Given an object of the class, I want to create copy of that object with the exact same attributes.

```
Client {
    psvm ( ) {
        Student st = ⇓ ;
⇒       Student copy = new Student ( );
        copy.name = st.name;
        copy.age = st.age;
    }
}
```

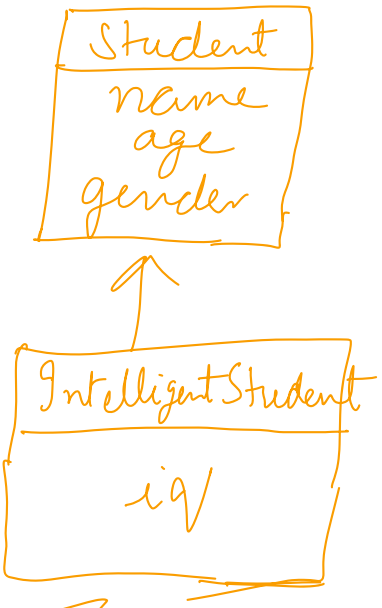
Cons:

① Client will need to know all the

internal details of Student

(2) What if there are private attributes in the Student class.

(3)



St can be equal to Student object or IntelligentStudent object.

```
if  typeof St == Student
      copy = new Student();
if  typeof St == IntelligentStudent
      copy = new IntelligentStudent();
```

→ violation of OCP

⇒ So, it looks like client might not be the best decider for creation of objects.

⇒ Copy Constructor

```
Student {
    Student (Student original) {
    }
}
```

```
Client {
    psvm {
        getObject();
        Student original = ⇑
        Student copy = ⇓
    }
}
```

again there would be if-else to decide which copy constructor to call.
(still a violation of OCP.

⟹ The responsibility of creating copy of an object should be outsourced to the object itself.

Client :
```
✓ Student st = ____________;
✓ Student copy = st.copy();
```

Benefits (1) No tight coupling between Client and Student class, client doesn't need to know internal details of it.

② No OCP violations.

```
Student {
    public Student copy() {
        Student copy = new Student();
        copy.name = this.name;
        copy.age = this.age;
        return copy;
    }
}
```

```
IntelligentStudent
    GS copy() {
        GS is = new GS();
        is.Name = this.name;
        is.age = this.age;
        is.iq = this.iq;
        return is;
    }
```

① St = new Student();

② St = new IntelligentStudent();

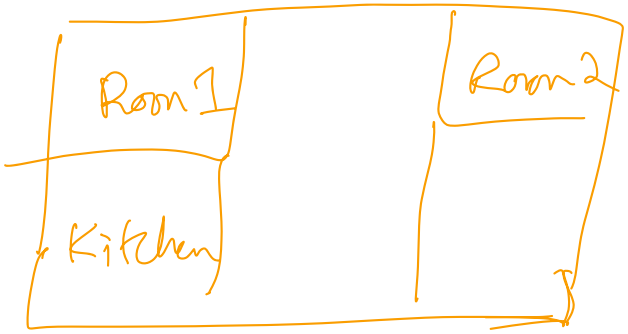
based on ① or ②

③ st.copy();

⇒ Therefore, always ensure that all child classes override the copy method. Otherwise it would be a surprise for

the client.

Prototype :



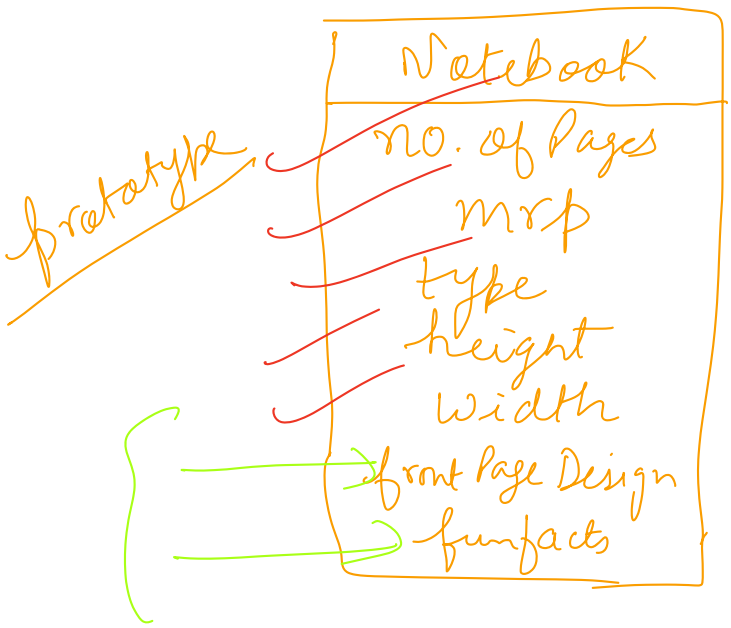
→ Blueprint of an object

⟹ Classmate example

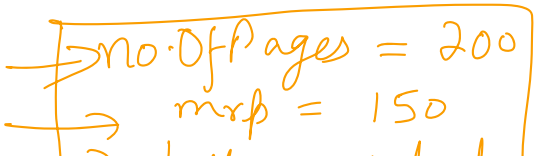
① ⟹ Fun facts at the last

② ⟹ Front page design.

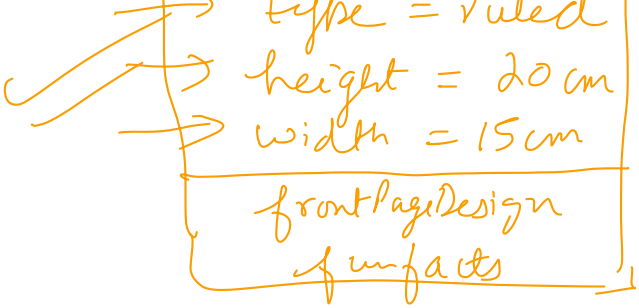
Classmate factory ⟹ Factory Management System



⟹ I have to print 10,000 copies of A4 size & ruled & 200 page notebook.



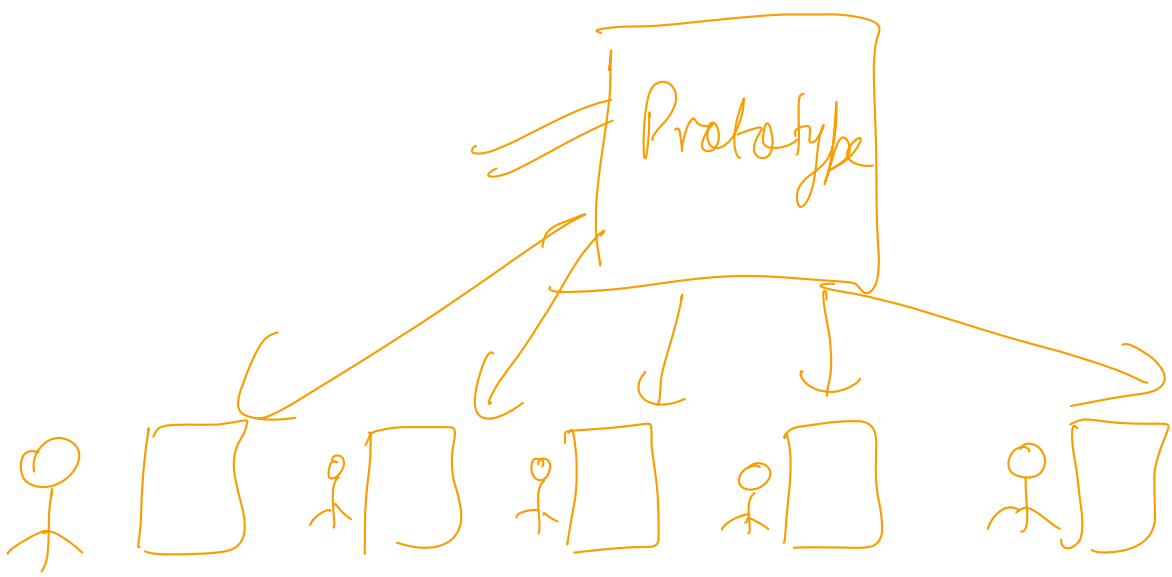
⟹ Prototype of



200 page, A4 size, ruled notebook,

(1) When the factory is working on above order it can just copy the prototype and it can put in values for FPD & FF, and it will be done.

(2) A lot of copies by copying the prototype & altering the variable values.



⇒ Prototype acts like a template.

eg: Let us say we have a SearchAPICall

all 3 are going to be same
- url = ____
- path = ____
- token = ____

this would be diff.
- query. = ____ → "Hello" / "Cricket" / "Movies"

Way 1

```
Client {
    SearchAPICall sapi = new SearchAPICall();
    sapi.url =
    sapi.path =
    sapi.auth =
    sapi.Query = "";
```

Way 2

```
Client {
```

① SearchAPI call sapi = getSearchPrototype();
② SearchAPI call copy = sapi.copy();
③ copy.query = "live scores";

}

⇒ There are scenarios where you dont want to create object from scratch. Then just adjust few attributes and you are done.

⇒ Dont change the template / prototype itself.

Eg: class Student {
- name
- age
- studentPsp.
- batch
- anyBatchPsp;
}

Way 1 =

Student abc = new Student ( );
abc.name = "abc";
abc.age = 21;
abc.studentPsp = 80.0;
abc.batch = "11d August";
abc.avgbatchpsp = "85.0";

Student xyz = new Student ( );
abc.name = "xyz";
abc.age = 21;
abc.studentPsp = 90.0;
abc.batch = "11d August";
abc.avgbatchpsp = "85.0";

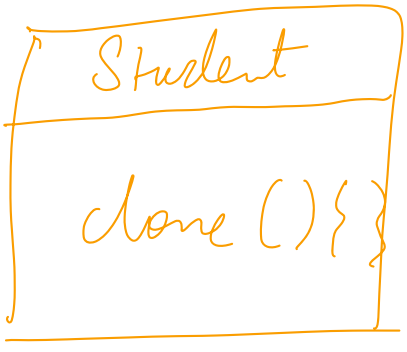
Way 2

Student augLLDStudent = new Student();
augLLDStudent.avgPsp = 80.0;
augLLDStudent.batch = "Aug LLD";
Registry.register("AugLLDStudent", augLLD Student);

Student abc = Registry.get("AugLLDStudet").copy();

```
abc.name = "abc";
abc.age = 21;
abc.studentRsp = 90.0;
```

Step 1 : In the class that you want to create prototype of, create a method called clone();



clone() ⇒ creates copy of object

Note: If you have a child class, always remember to override the clone method, otherwise it would lead to surprises for the client.

Step 2: Store the prototype in registry.

StudentRegistry ⇒ to store

Map<String, Student> map; Student prototypes

register(String key, Student student) { }

get(String key) { }

① often these prototypes are stored in registry, at the time application starts.

② Registers are singletons.

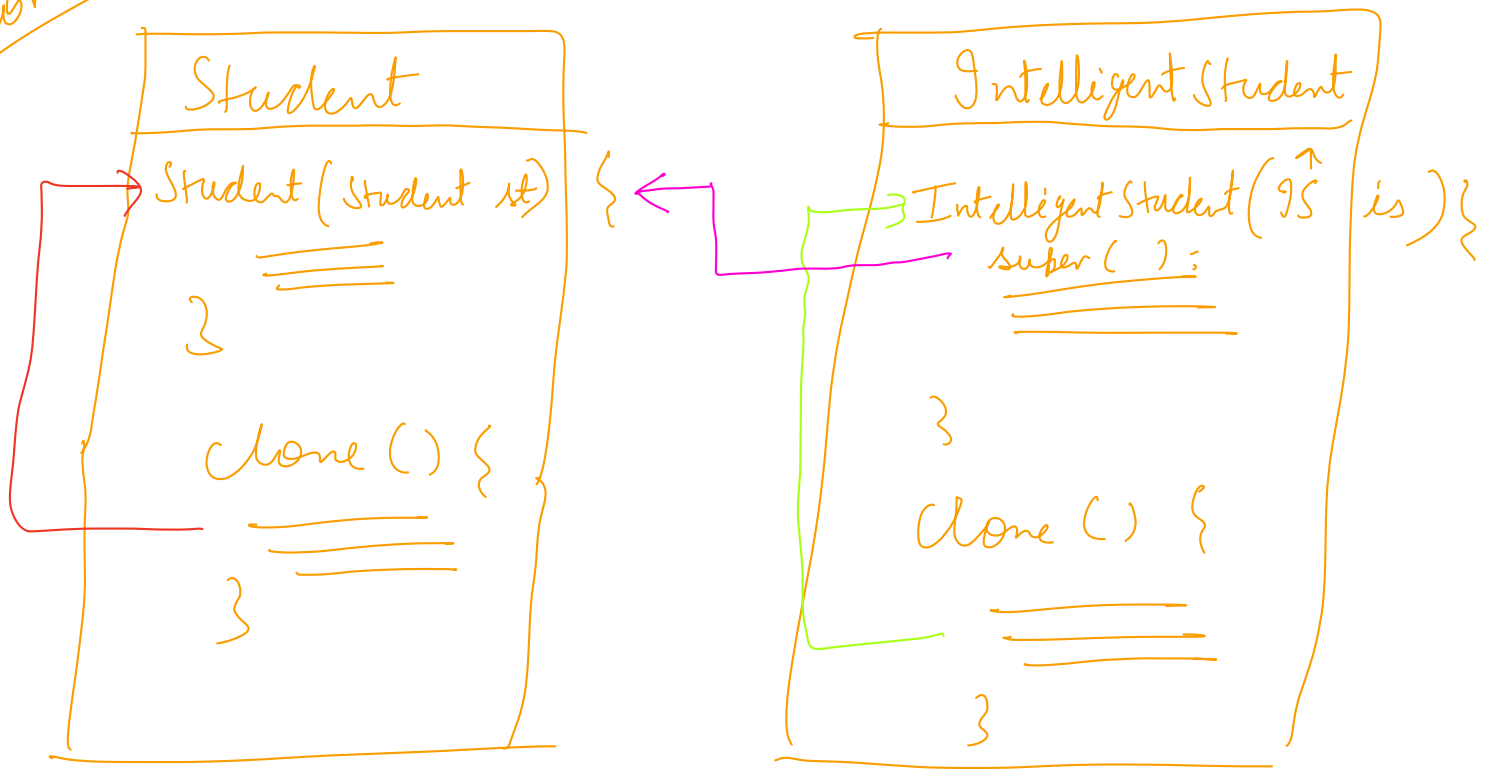
Step 3 : Client calls the registry to get prototype. It then creates a copy of that prototype and does it works on the copy

Prototype: ⇒ If you want to create a copy, outsource it to the object rather than copying yourself.

Registry: ⇒ If you need something again and again, store those

Springboot container

things in a registry.



⟹ We always need to over-ride in child class

⟹ If child class doesnt have access to private attrs. of parent class, it does a tricky thing it calls the copy constructor of own class which calls the copy constructor of parent class.

⟹ Why we cannot directly call copy constructor instead of clone

⟹ as it will again cause the

same (if - else problem .