

PROTOTYPE & REGISTRY

* Agenda:-

- ⊙ Introduction
 - ⊙ Problem Statement
- ⊙ Prototype Design Pattern (approaches)
- ⊙ Real use-case of Prototype Design Pattern
- ⊙ Example of Prototype Design Pattern

* Prototype Design Pattern :-

* Definition



Helps us to solve Problems
where we need to create a
copy of a given Object.

* Problem Statement:-

① Given an object of a particular
Class

② we need to create a copy
of that object




new ← memory location. { Create a new
object with
exact same
attributes of
the given object.

```
class Client {
```

```
    perm() {
```

Student or
child of
student.

```
        Student originalStudent = ;  
        Student copyStudent = new Student();
```

```
            cs.name = os.name;
```

```
            cs.age = os.age;
```

```
            .  
            .  
            .
```

```
        }
```

```
    }
```

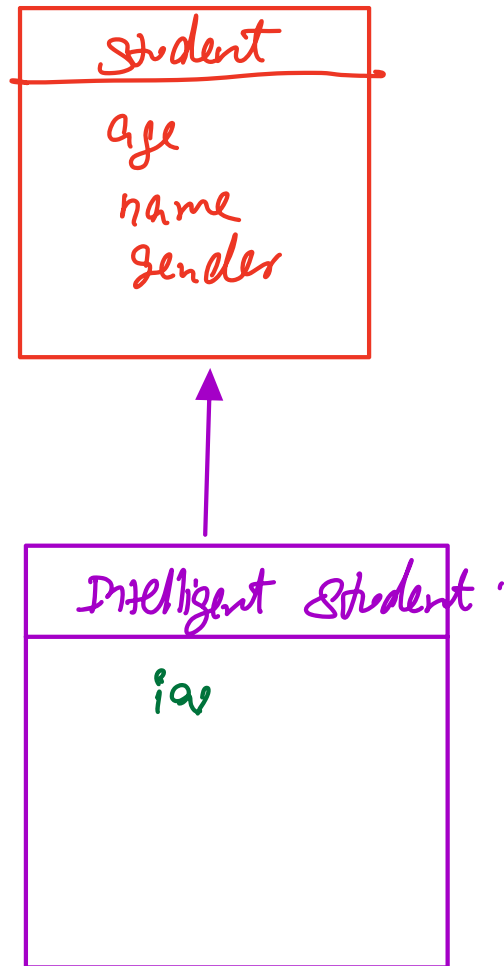
Disadvantages:-

① Client will need to know all the attributes & implementation details of the given object.

↓
(Tightly coupled.)


②- Object might have private attributes

②



Client

Student s = intelligent Student();
Student copy = new Student();



```

if ( OS instanceof Student ) {
    copy = new Student();
}
else if ( OS instanceof IntelligentStudent ) {
    copy = new IntelligentStudent();
}
else if ( _____ ) {
    :
    :
}

```

OCP \Rightarrow too many if-else!

It is hard to create a
 copy of student when student
 has different types / specifications.

(prone to errors)

② Copy constructor

student {

Student () ? }


```

student ( student o ) {
    : // copies data
}

```

Client

Student os = new Student();
Student cs = _____;



```
new Student (os);
```

```

Student ( Student o✓ ) {
    if ( o instanceof IS ) {
        o.in = _____;
    }
    else _____ }
}

```

COMS

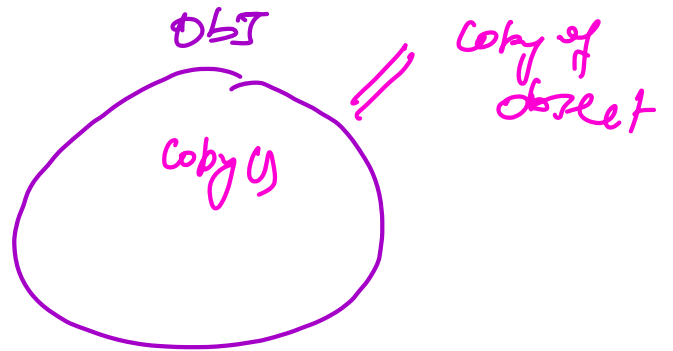
OCP

X



if a client wants to create a copy of an object, having the logic to create the copy within client class \Rightarrow prone to errors.

Ideal Solution:-



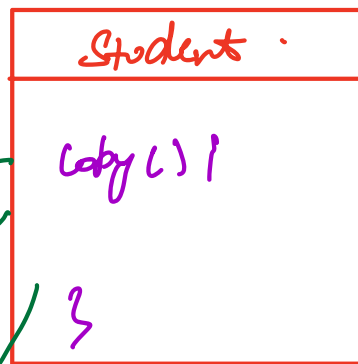
Client=

Student s = _____ ;
Student copy = s. copy();

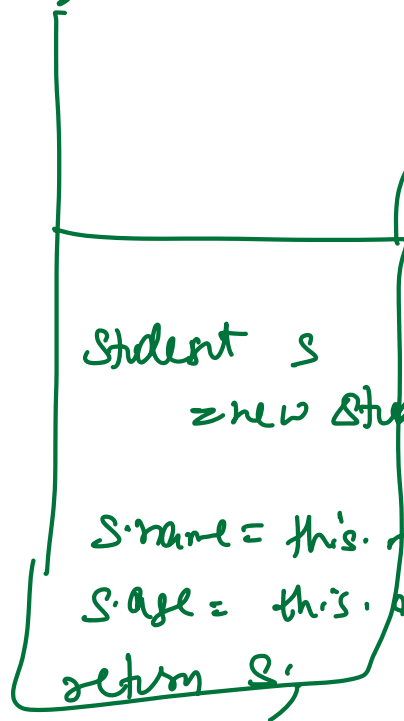
Advantages :-

① No Tight coupling.

② OCP ✓



new object

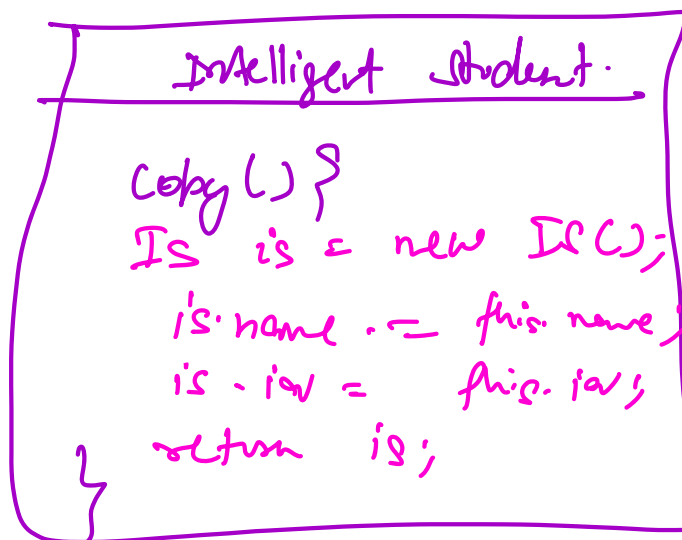


} new object.

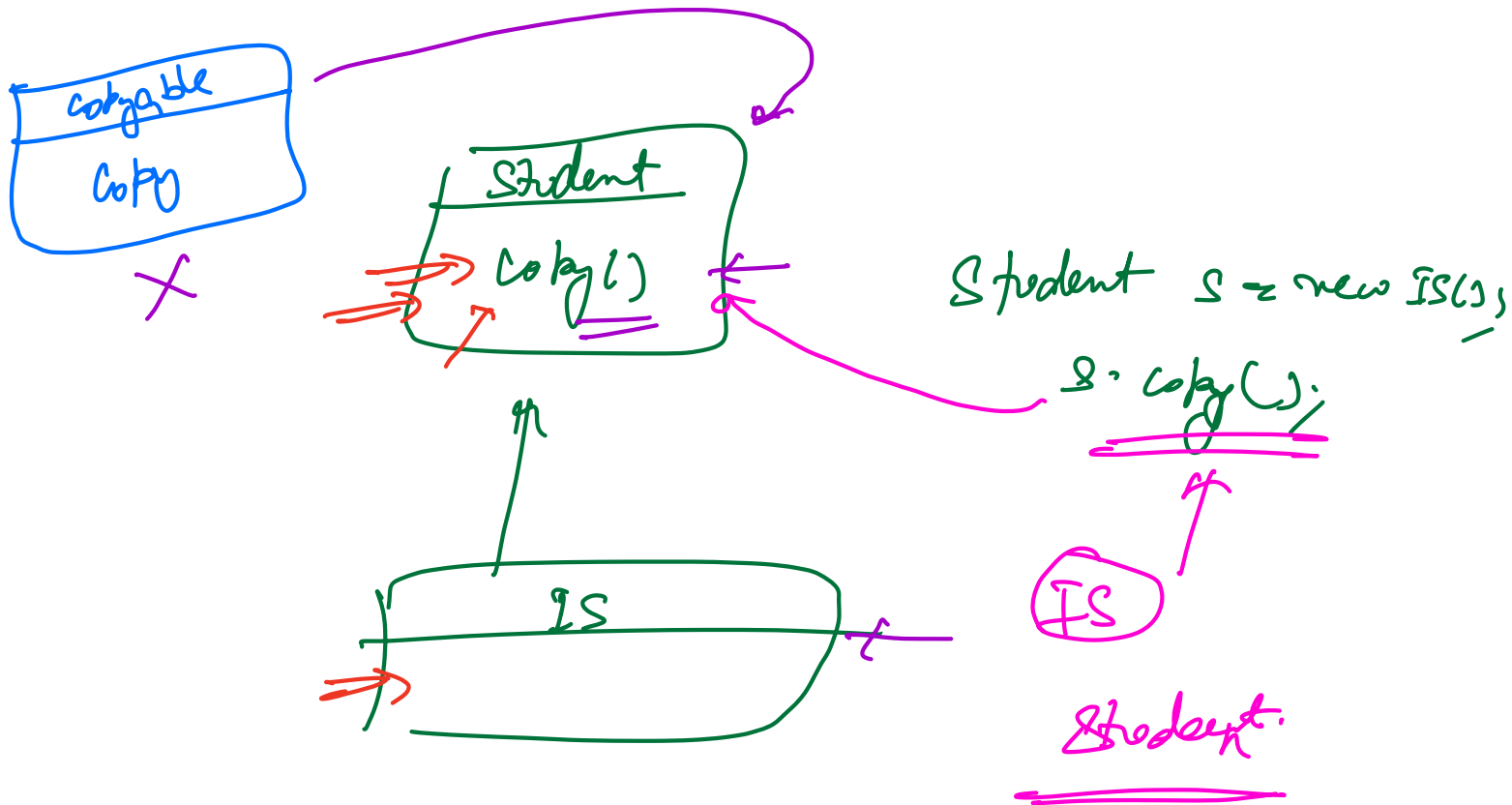
} Initialize.

} return.

Extends



~~new Student();~~
~~new IS();~~
 Student s = s. copy();



UNFORTUNATELY!

There is no way to enforce it

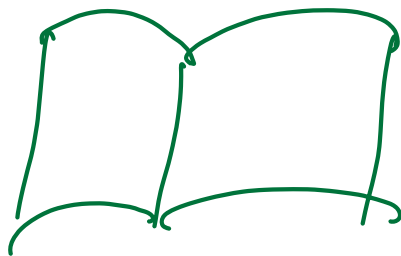
we need to caution \rightarrow that all the
 child classes override the copy()

↓
SURPRISE !

(*) Prototype Design Pattern :-

↳ Basic layout of an Object!

Classmate's Notebook !



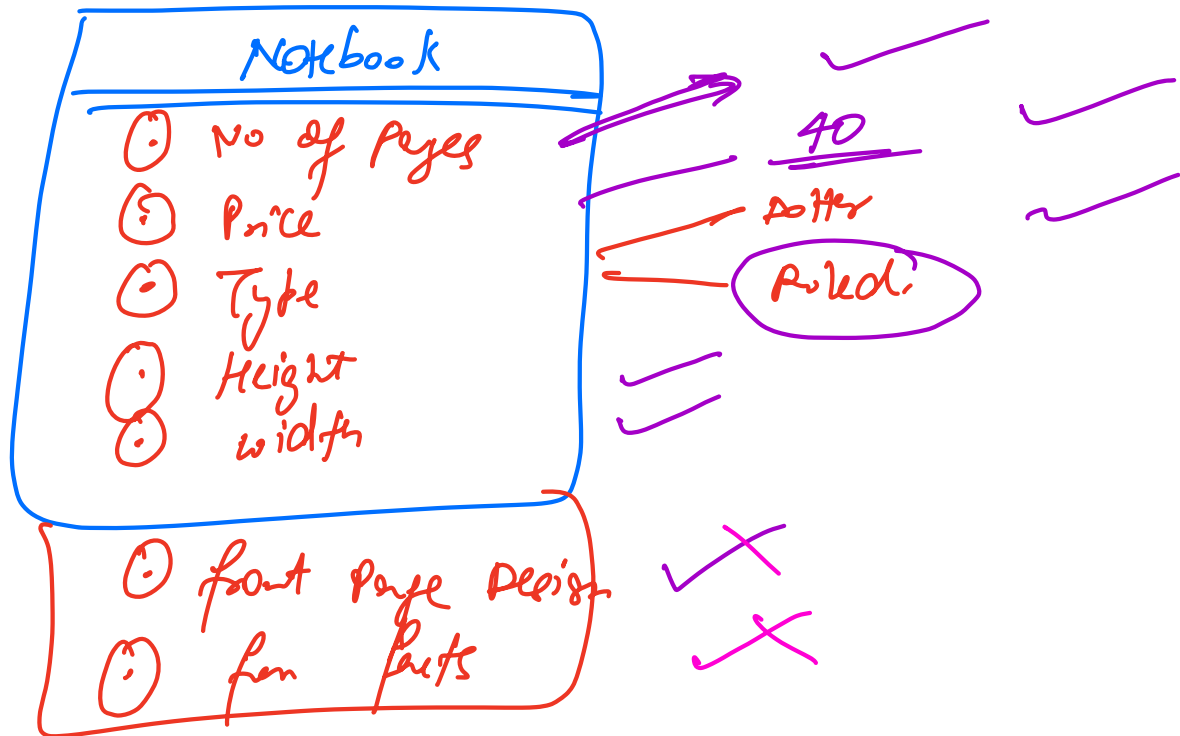
Design
Cover



Classmate Notebook Factory.



Factory Management System.



Classmate wants to create 10000 notebooks of A4 size with 120 page that are Ruled.

A4-120-Ruled

No. of Pages = 120

Price = 40

height = 120mm

width = 80mm

Type = Ruled.

funfacts = null

front page = null

when the factory works, it creates
a copy of the prototype object
and adds the front page &
funfacts in every notebook

⇓

DONE!

Real use case :-



Search Queries

Search API {

URL =

token =

⋮

query =

}

Approach 1 :-

Search API Sapi => new Search API(C);
 Google,

Sapi - url = _____;

Sapi - token = _____;

Sapi - query = _____;

⋮

Approach 2

SearchAPI prototype = get SearchAPI
Prototype();

SearchAPI obj1 = prototype . copy();

obj1 . array = _____ ;

SUMMARY :-

① we often face scenarios where we don't want to create the object from scratch.

② Rather, we prefer to create a copy from template
↓
Change the values!

How to implement!

```
class Student {
```

```
    String name;  
    int age;  
    int psp;  
    double avg psp of batch;  
    String batchName;  
    ;  
}
```

```
Student s = new Student();
```

```
s.name = "Phira";
```

```
s.age = 25
```

```
s.psp = 80
```

```
s.batchName = "June 2023"
```

```
s.avg psp of batch = 80;
```

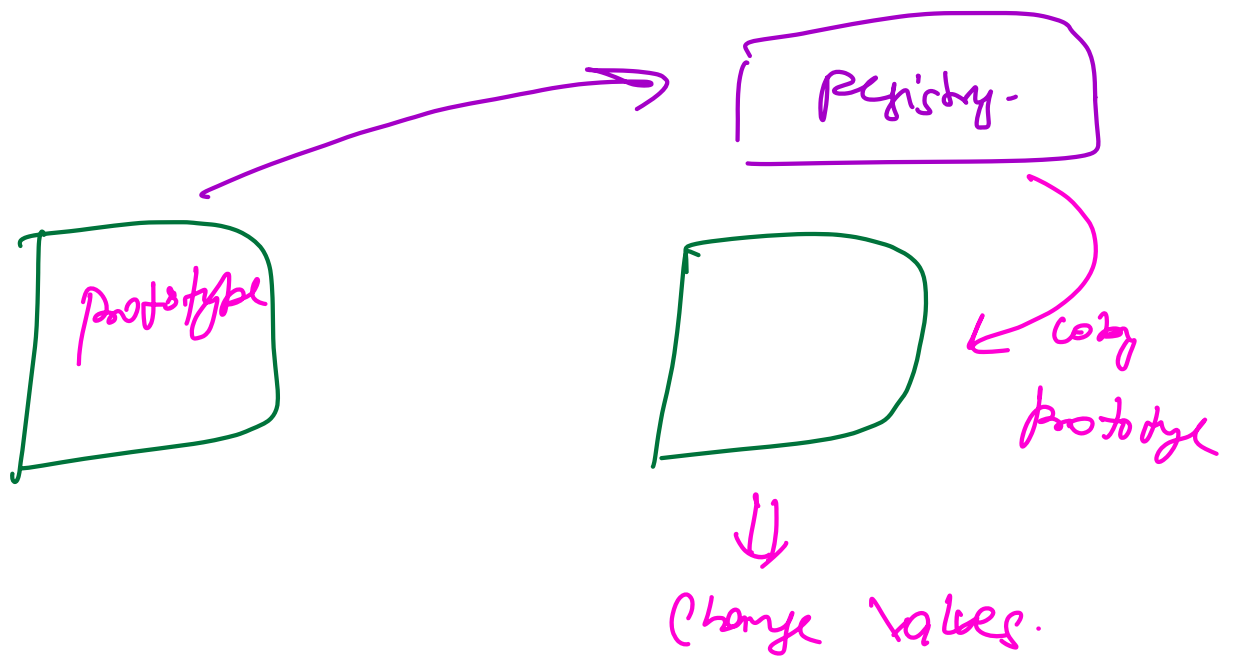
790.5

`Student mariya = Registry.get("Junction")`

`marrya.name = 'Marrya'`

variyel. zyl ≈ 25

!



Class Registry

```
class StudentRegistry {  
    Map<String, Student> registry;
```

```
    Student getPrototype ( String batch ) {  
        return registry.get ( batch );  
    }
```

```
    void setPrototype ( String batch,  
                        Student o ) {  
        registry.put ( batch, o );  
    }
```

①-

Clone ()

↳ If object has child
classes

↳ Must override
clone ().

2. Store the prototype in the registry.

3. Client \Rightarrow Registry \Rightarrow Prototype
modify values. \leftarrow Copy

1. Registry class are singleton;

2. Normally, when application starts
 \downarrow
all prototypes are stored in registry.

Summary:-

Prototype

if you want to create a copy of a particular object rather than copying yourself, the object should have the responsibility to copy itself

Registry

- ① if something is needed again & again,
⇒ store such things in registry.

Ex — Application Context.

Break till 10:06
PM IST

```

Student
{
  s(s obj) {
    :
  }
  copy() {
    new S(this);
  }
}

```

```

IS
{
  IS(is obj) {
    super
    inv = —
  }
  copy() {
    new IS(this);
  }
}

```

↓
law of reuse

2

3

1

Assignment :-

① Implement Prototype Registry Design Pattern

↓

Student, IS, Log Cons.

Student Registry, Client.

② Read about Prototype Registry Pattern.

↳ Benchmarking

↳ Refactoring. Java.