Agenda.

→ Prototype

→ Registry.

PROTOTYPE DESIGN PATTERN. → Creational.

Problem Statement

→ Given an object of a class.

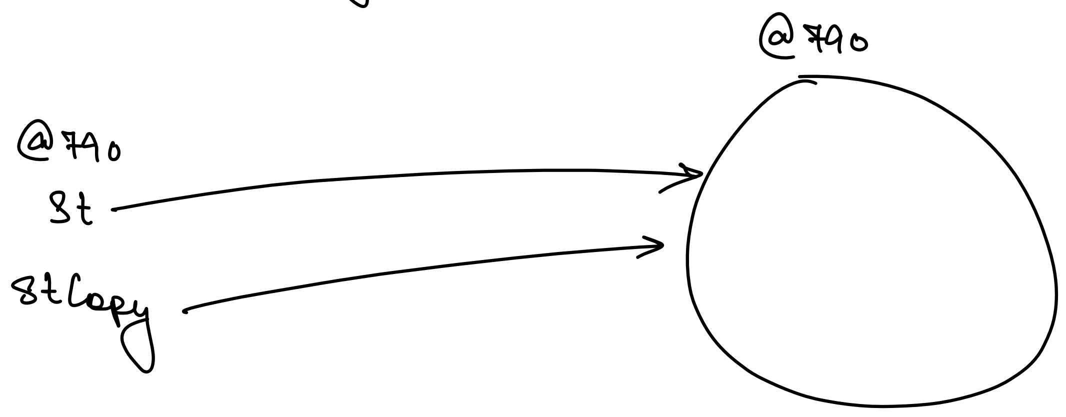
→ We need to create a copy of that object.

{Create a new object with exact same attrs}.

```
psvm(){
        Student st = new Student();
        St.name = ___
        St.age = ___
        St.psp = ___
        ___
        ___

      { Student stCopy = new Student();
        StCopy.name = st.name;
        StCopy.age = st.age;
        ___
        ___
        ___
}
```

Student stCopy = st ; ~~X~~

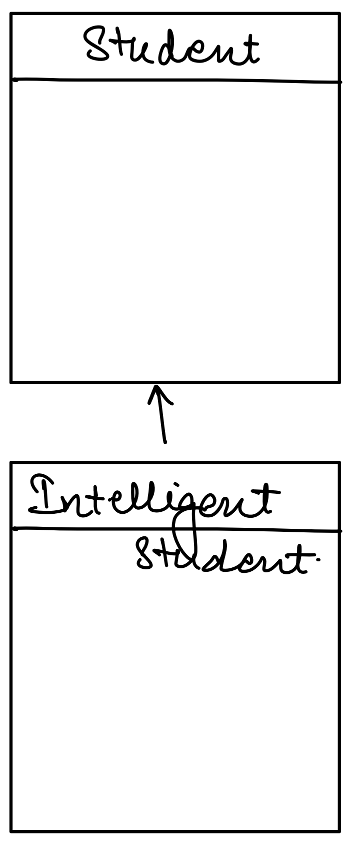


Issue

→ Client needs to know the complete details of student object.

→ Student might have some private attrs that client might not be able to access.

→



Student original = ______

Student ↓ IS ↓

Student copy = ?

```
if (original.instanceOf(Student)){
        copy = new Student();
}
else {
        copy = new IS();
}
```

⇒ OCP. Violation.

Copy Constructor.

```
Class Student {

    Student ( Student st ) {
        this.name = st.name;
        this.age = st.age;
    }
}

Student st = new Student();
Student stCopy = new Student(st);
```

```
⇒ Class IntelligentStudent extends Student {
      IntelligentStudent (IS st) {
          this.name = st.name;
          this.age = st.age;
      }
  }
```

⇒

```
                              Student
Student original = ________ <- IS().

Student copy = ?

if (original.instanceOf(Student)){
        copy = new Student(original);
}
else {
        copy = new IS(original);
}                                          } OCP X.
```

⇒ If client wants to create a copy of an object then having the copy logic on client side is prone to Errors.

⇒ Ideal Sol^n can be that if client outsources the copy logic to the object itself

⇒
```
    Student {

        Student Copy() {
            Student stCopy = new Student();
            stCopy.name = this.name;

            return copy;
        }
}
```

```
main(){
      Student original = new Student();
      Student copy = original.copy();
}
```

⇒ Intelligent Student :

```
Class IntelligentStudent extends Student{
                        Clone.
    Student Copy() {
        IS StCopy = new IS();
        StCopy.name = this.name;

        return copy;
    }


}
```

```
                                  IntelligentStudent
                                  ↓
Student original = ________________;

Student copy = original.copy();
                     IS
```

⇒ All the child class must also override the copy method.

Student PS()

```
Student original = ________ ;
Student copy = original.copy()
```

Prototype Design Pattern.

↓

Sample.

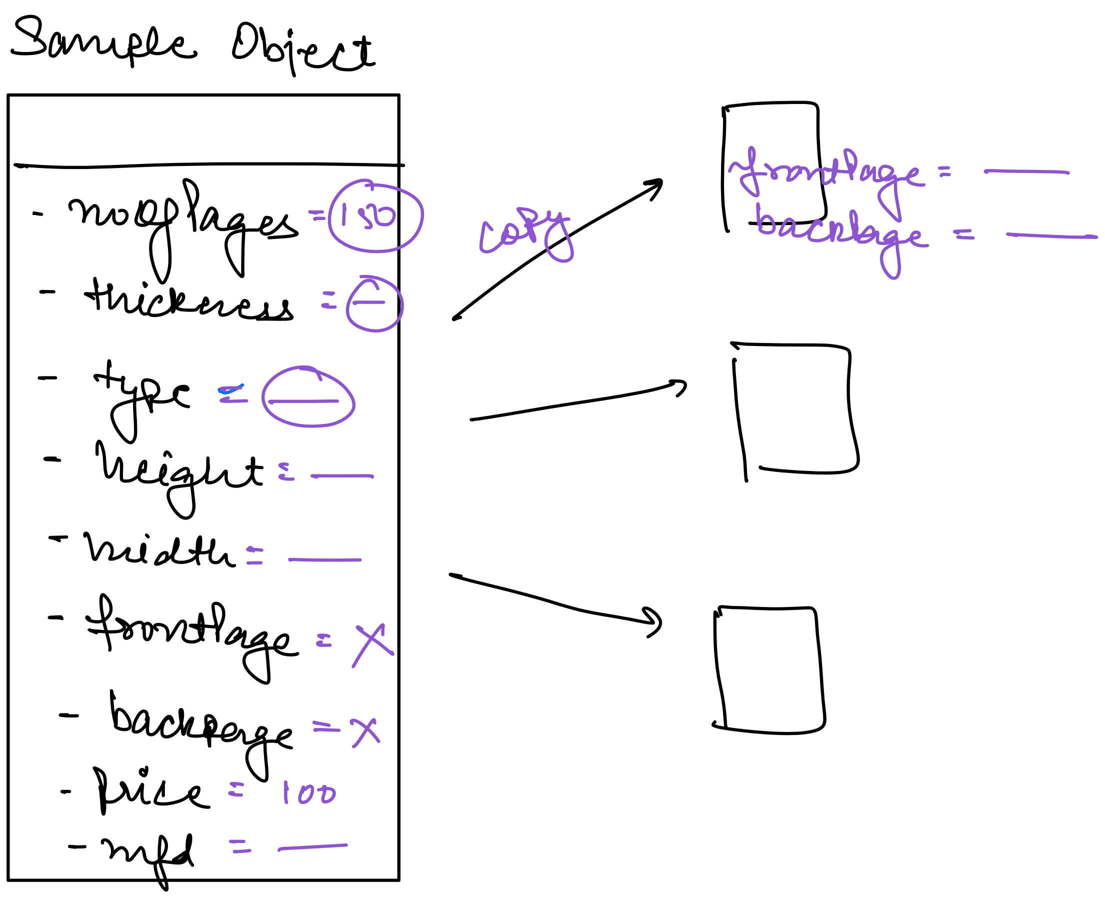
⇒ Classmate Notebooks.

Notebook

- noOfPages ✓
- thickness ✓
- type ✓
- height ✓
- width ✓
- frontPage ✗
- backPage ✗
- price ✓
- mfd ✓

Requirement:

Create 10000 notebooks of type plain & with 150 pages each.



Prototype Design Pattern.

⇒ Often there are scenarios where we don't want to create an object from scratch, rather we prefer creating an object from some existing template & changing only the required fields.

```
⇒ Class  Student {
            name
            age
            batch
            Psp
            avgBatchPsp
            Contest
            address
            CompanyName;
   }

Student  vishnu  =  new  Student();
Vishnu. name  =  ______
Vishnu. age  =  _____
Vishnu. batch = "Mar22 MWF"
Vishnu. Psp =  80
Vishnu. avgBatchPsp  =  75.0
  ____
  ____
```

Way1

```
Student  umes  =  new  Student();
umes. name  =  ______
umes. age  =  _____
umes. batch = "Mar22 MWF"
umes. Psp =  90
umes. avgBatchPsp  =  75.0
```

Way 2

Registry

```
Student prototype = ________
prototype.avgBatchPsp = 75.0;
prototype.batch = "Mar22 MWF";
prototype.contest = "______";
```

```
Student umes = prototype.copy();
umes.name = ________
umes.age = ______
umes.psp = 90
```

⇒ Steps.

1. In the class for which we want to create prototypes, declare a method copy() | clone() to create copy object.

Note: All the child class should override the copy method

2. Create the prototype objects & store them in Registry

↳ Map< String, Object >

3. Create the copy object from prototypes stored in Registry whenever required.

Registry

→ Class to store prototype objects.