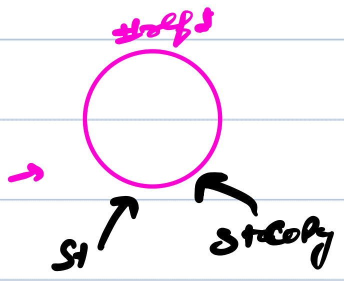
Today's agenda

↳ Prototype design Pattern

★ Given an object of a class, we need to create multiple copies of the given object.

//idea 0

```
↳ Student St = new Student();
  Student stCopy = St;
```

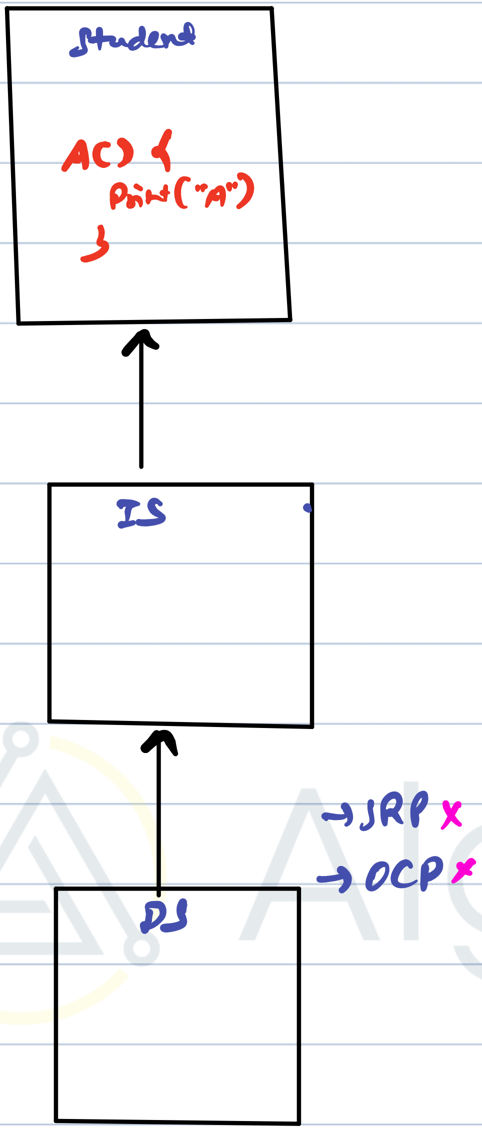


//idea 1

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

    stCopy.name = St.name;
    stCopy.age = St.age;
    stCopy.weight = St.weight;
           .
           .
           .
```

Cons:

1) Client needs to Know all the internal details of Student Class.



```
Student St = new Student() / IS();
        ↓
Student StCopy = ________

if (instance of St == Student) {
        StCopy = new Student();
}
else if (instance of St == IS) {
        StCopy = new IS();
}
else if (instance of DS == IS) {

}
```

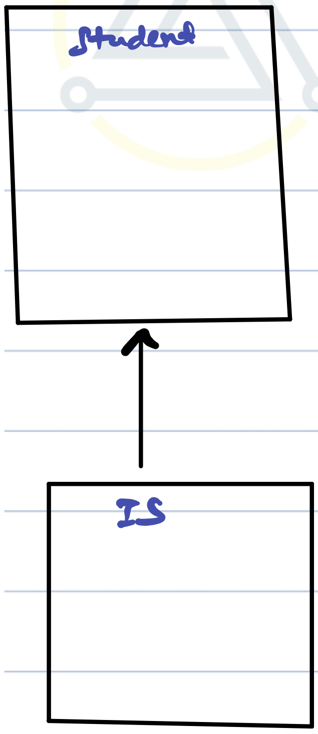
//idea2

↳ copy Constructor

→ Now Client doesn't need to know all the internal details of class.

```
Class Student {
  -
  -
  -
  Student (Student original) {
      this.name = original.name;
      this.age = original.age;
        :
  }
}
```

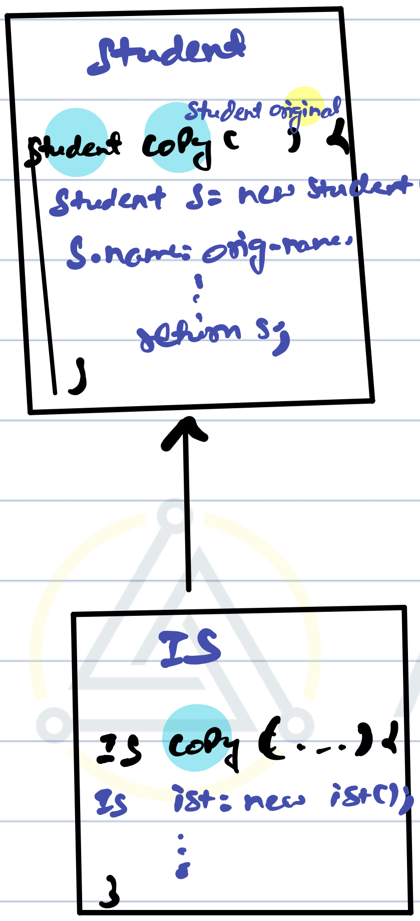
```
Student St = new Student()/IS();
        ↓
Student StCopy = ________

if (instance of St == Student) {
        StCopy = new Student(St);
}
else if (instance of St == IS) {
        StCopy = new IS( St );
}
```



Note: All the child Class must have the copy Constructor.

//idea 3

↳ instead of having copy constructor, have a copy method and return the object.



```
Student
Student copy ( Student original ) {
    Student S= new Student();
    S.name= orig.name;
        :
      return S;
}
```

```
IS
IS copy (...) {
IS   ist= new iS();
       :
}
```

Student St = new Student()/IS();

Student StCopy = St.Copy();

→ OCP is not violated ✓

→ Now client doesn't need to know all the internal details of class.

→ To enforce copy method use interface.

→ Search API

AlgoPrep.in / Course-dashboad / Systemdesign / query

$$\begin{tikzcd} \text{query} \\ \text{accessing notes} \arrow[u] \arrow[r] & \text{accessing video} \\ & \text{accessing codes} \arrow[u] \end{tikzcd}$$

```
SearchAPI {
    url= . . . . _
    Path = _  __
    auth: _ _ _
     query: _  ~
}
```

→ there are scenarios where we don't want to create object from scratch, rather use copy of existing object and change few required attributes.

```
Class Student {
    name
    age
    weight
    batch
    instructor
}
```

```
Student Obj = new Student();
    obj.name = Preetham;
    obj.name = 27;
    obj.weight = 72;
    obj.batch = SD2
    obj.instructor = "Abhishek"
```

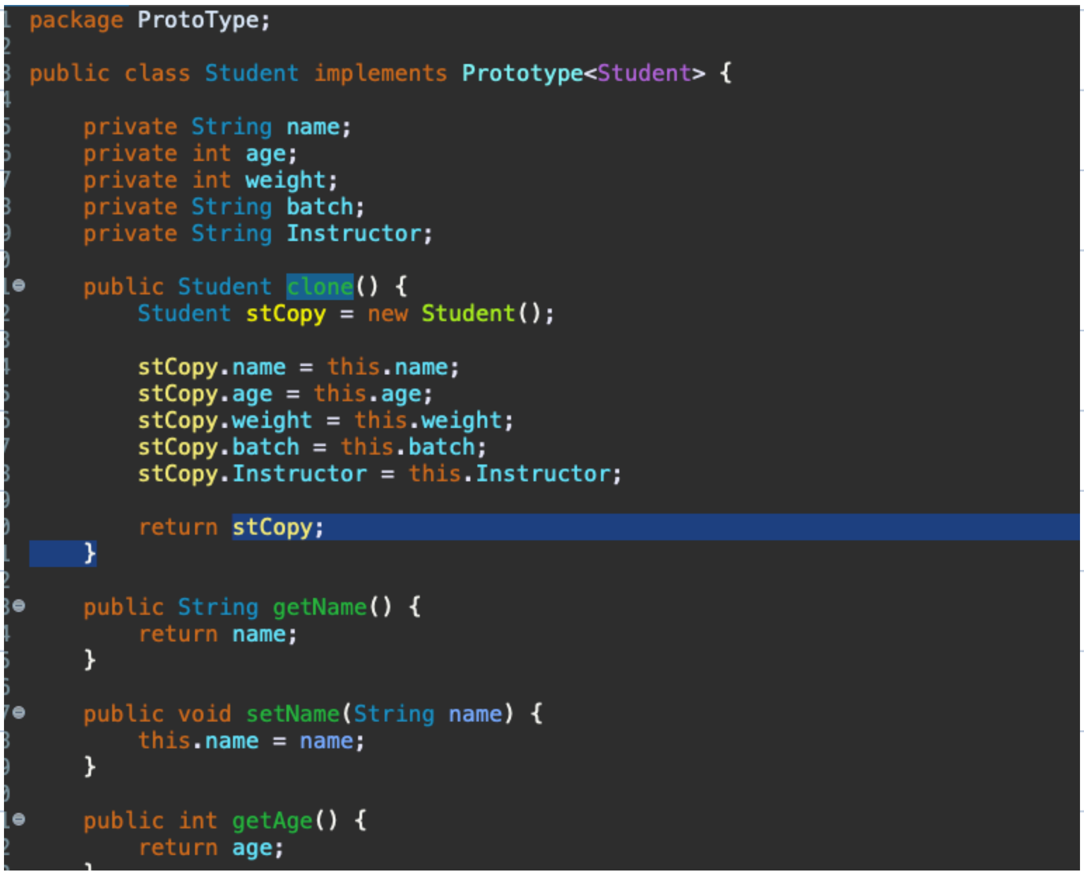
SD2	→ original obj 1
SD3	→ original obj 2
DSA1	→ original obj 3

```
Registry
Map<String, Student> map;

    register (Key, Student) {
        map.put (Key, Student);
    }

Student get ( Key ) {
        return map.get(Key);
}
```

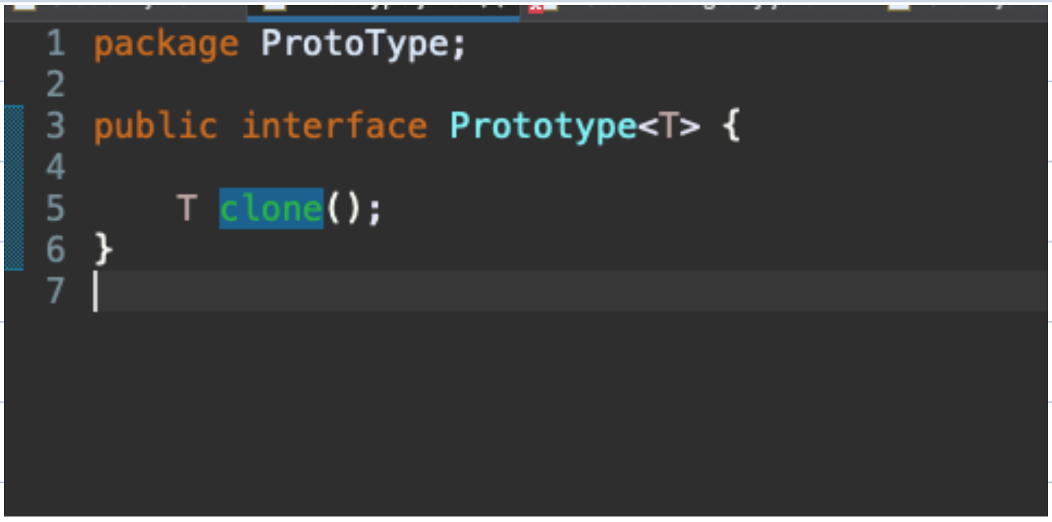
↳ client will first call registry and then Clone the objects.

Break till 9:20 Pm

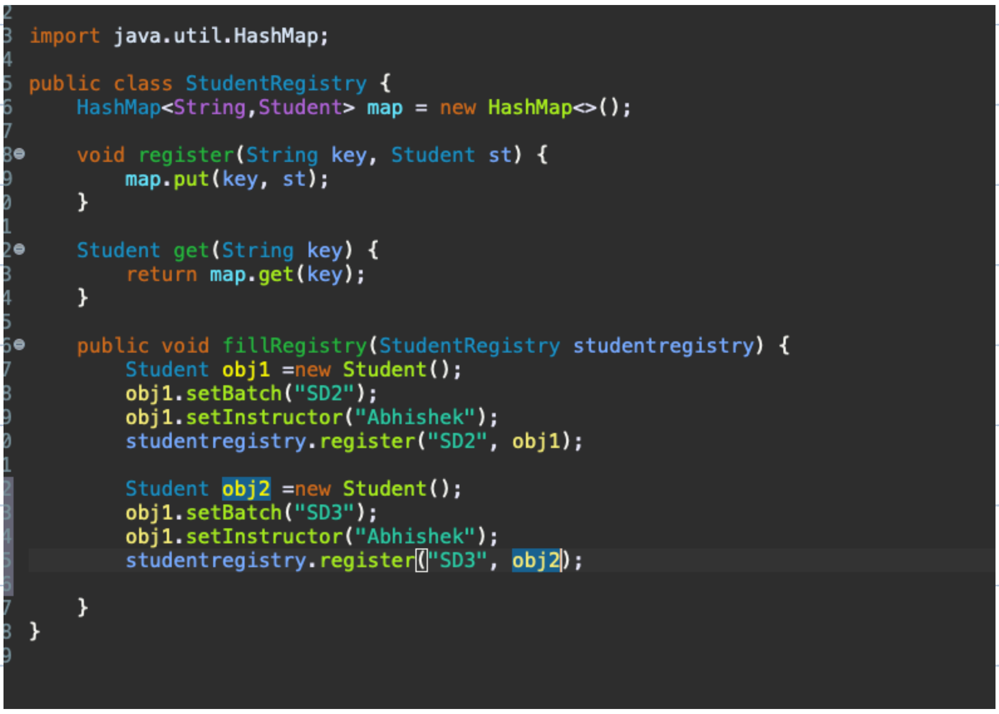
Code



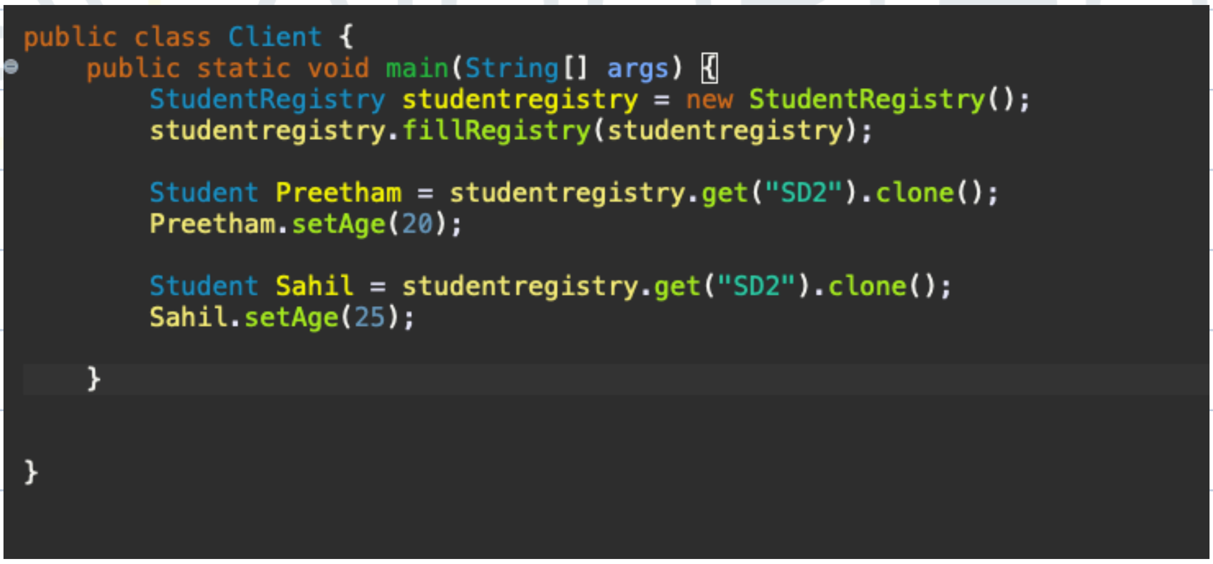
```java
package ProtoType;

public class Student implements Prototype<Student> {

    private String name;
    private int age;
    private int weight;
    private String batch;
    private String Instructor;

    public Student clone() {
        Student stCopy = new Student();

        stCopy.name = this.name;
        stCopy.age = this.age;
        stCopy.weight = this.weight;
        stCopy.batch = this.batch;
        stCopy.Instructor = this.Instructor;

        return stCopy;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
```



```java
package ProtoType;

public interface Prototype<T> {

    T clone();
}

```



```
import java.util.HashMap;

public class StudentRegistry {
    HashMap<String,Student> map = new HashMap<>();

    void register(String key, Student st) {
        map.put(key, st);
    }

    Student get(String key) {
        return map.get(key);
    }

    public void fillRegistry(StudentRegistry studentregistry) {
        Student obj1 =new Student();
        obj1.setBatch("SD2");
        obj1.setInstructor("Abhishek");
        studentregistry.register("SD2", obj1);

        Student obj2 =new Student();
        obj1.setBatch("SD3");
        obj1.setInstructor("Abhishek");
        studentregistry.register("SD3", obj2);

    }
}
```



```
public class Client {
    public static void main(String[] args) {
        StudentRegistry studentregistry = new StudentRegistry();
        studentregistry.fillRegistry(studentregistry);

        Student Preetham = studentregistry.get("SD2").clone();
        Preetham.setAge(20);

        Student Sahil = studentregistry.get("SD2").clone();
        Sahil.setAge(25);

    }


}
```