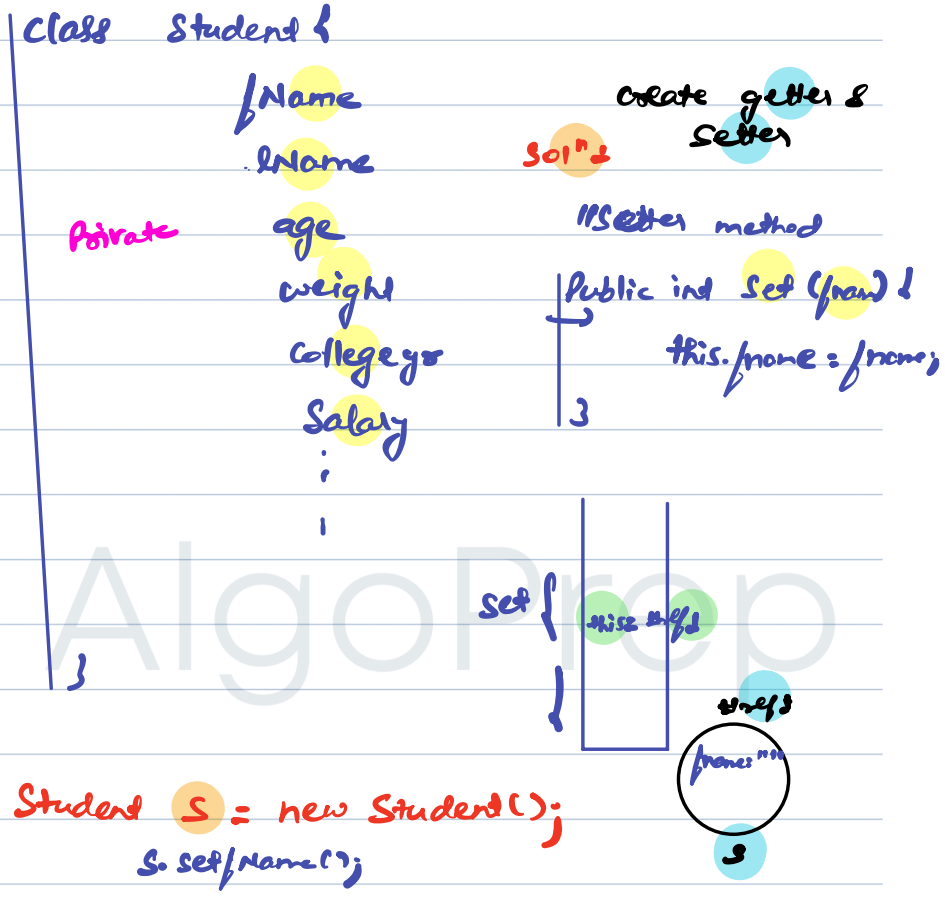
Today's agenda

↳ Builder design Pattern

① Create an object of a class with a lot of attr.



① The attributes are immutable.

↳ if getter & setter are Public, I can always Change the value of attributes.

Soln 2

↳ Create a Parametrized Cons of the class. Pass the values for each attributes.

```
Class Student {
    private fName → 2
            lName → 2
            age → 2
            weight → 2
            collegeyr → 2
            Salary →
            :
            :
}
```

```
Student ( fName, lName, age, weight,
          collegeyr, Salary . . . . ) {
    this.fName = fName;
    this.lName = lName;
    .
    .
}
```

Student s = new Student ("Subbesh", "", 24, 80, 4,)

↳ Not understandable as well as bug prone.

→ Not all attributes are mandatory.

↳ I will create all the Possible Combination of constructors.

2^6 Possible Constructors

```
class Student {
    String fName → 2
    String lName → 2
    private int age → 2
        int weight → 2
            collegeyr → 2
            Salary →
            :
            :
}
```

```
Student (String fName, int age){
    this.fName = fName;
    this.age = age;
}
Student (String lName, int weight){
    this.lName = lName;
    this.weight = weight;
}
```

→ Too many possible combination of constructors.

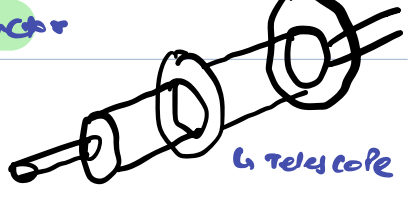
→ not all constructor may be possible to create.

```
Student (fName, age){ this.fName=fName; this.age=age }
Student (fName, age, weight){ this.fName=fName; this.age=age; this.weight=weight; }
Student (fName, lName, age, weight){ this.fName=fName; this.age=age; this.weight=weight; this.lName=lName; }
```

Code duplication?!

↓

Telescoping constructor



↳ Telescope

```
Student (fname, age) { this.fname = fname
                       this.age = age }

Student (fname, age, weight) { this(fname, age);
                               this.weight = weight; }

Student (fname, lname, age, weight) { this(fname, age, weight);
                                      this.lname = lname; }
```

→ Ideally these should be 1 constructor

```
Class Student {
    String fName
    String lName
    Private int age
        int weight
            collegeyr
            Salary
            :
            :
}
```

```
                                    any type can be passed
                                    ↓
Map<String, Object> val

Student ( ) {
    if (val.containsKey(fName)) {
        this.fName = (String)val.get(fName);
    }
    if (val.containsKey(age)) {
        this.age = (Integer)val.get(age)
    }
}
```

Q) find some data structure that allows having multiple values within it.

Map<String, Object>

key	value
fName	Sukhesh
age	24
weight	80
:	:
.	.

→ Client {

map < , > {

.Put (fname, Subhash);

.Put (aeg, 24);

}

}

tough to debug.

→ you might map number to a String type

Cin (string)

(fname, 181)

fname : "181"

↳ This issue should be caught.

12

Final folⁿ

```
Class helper {
    String fName
    String lName
    int age
    int weight
    CollegeYr
    Salary
    :
    :
}
```

?: →

```
Student (helper val) {
    this.fName = val.fName;
    this.lName = val.lName;
    :
    :
    :
}
```



```
Client {
    helper h = new helper();
    h.fName = "Subhesh";
    h.age = 24;
    h.weight = 80;

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

h.fName = 131?!

↓ Caught at compile time.

Break till 9:23 Pm

↳ The helper class doesn't need to be immutable.

→ when to use Builder design Pattern.

↳ Class with a lot of attributes.

↳ immutable attributes in class.

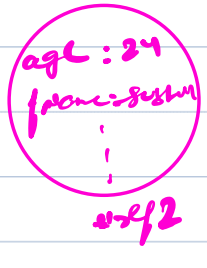
```
1 package Builder_Pattern;
2
3
4 public class client {
5
6     public static void main(String[] args) {
7         Builder builder = Student.createBuild();
8         builder.setAge(24);
9         builder.setFname("Subhesh");
10        builder.setLname(null);
11        builder.setWeight(80);
12
13
14
15        Student s1 = builder.build();
16
17        System.out.println(s1.getFname());
18    }
19
20 }
21
```

```
public class Builder {
    private int age;
    private String fname;
    private String lname;
    private double weight;

    public Student build() {
        return new Student(this);
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
    public String getFname() {
    public void setFname(String fname) {
    public String getLname() {
    public void setLname(String lname) {
    public double getWeight() {
    public void setWeight(double weight) {


}
```

this : obj1

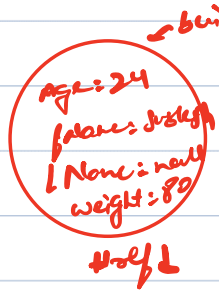
```
1  package Builder_Pattern;
2
3  public class Student {
4      private int age;
5      private String fname;
6      private String lname;
7      private double weight;
8
9
10     public Student(Builder builder) {
11         this.age = builder.getAge();
12         this.fname = builder.getFname();
13         this.lname = builder.getLname();
14         this.weight = builder.getWeight();
15     }
16
17
18     public int getAge() {
19         return age;
20     }
21
22     public String getFname() {
23         return fname;
24     }
```

obj2



obj2

builder

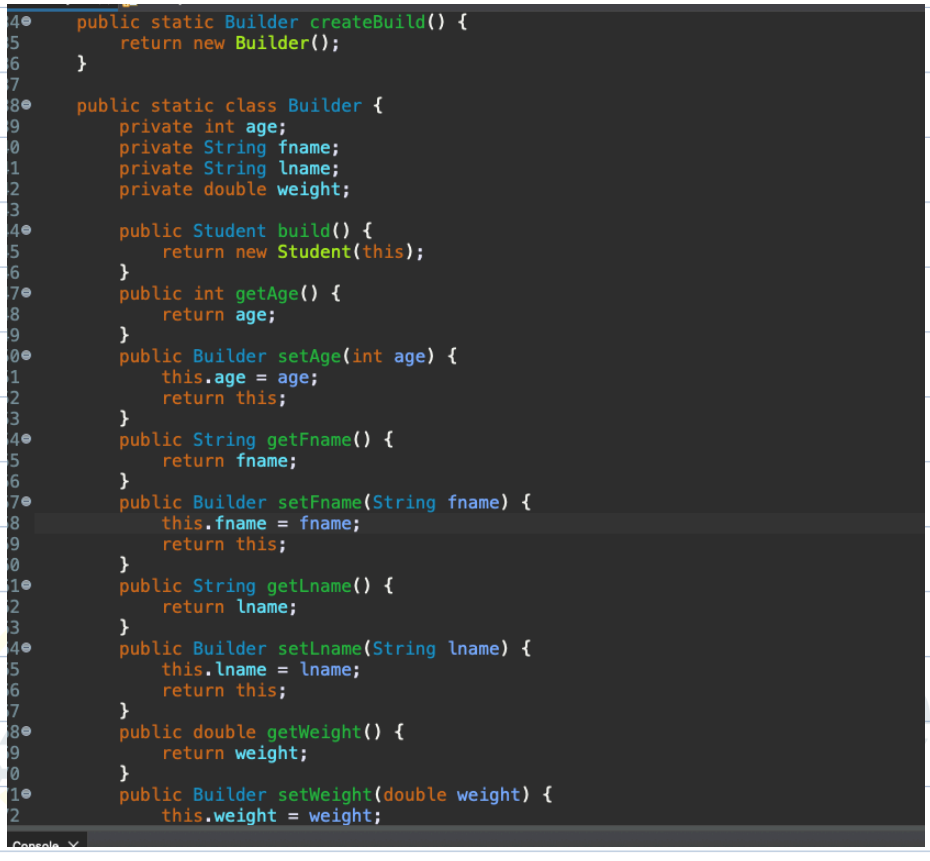


obj1

Final Soln

```
3
4  public class client {
5
6      public static void main(String[] args) {
7  //      Student.Builder builder = Student.createBuild();
8  //
9  //      builder.setAge(24);
10 //      builder.setFname("Subhesh");
11 //      builder.setLname(null);
12 //      builder.setWeight(80);
13 //
14 //      Student s1 = builder.build();
15
16
17         //Production ready code of Builder design Pattern
18         Student s2 = Student.createBuild()
19                 .setAge(24)
20                 .setFname("Subhesh")
21                 .setLname(null)
22                 .setWeight(80)
23                 .build();
24
25
26
27     }
28
29 }
30
```

```
3  public class Student {
4      private int age;
5      private String fname;
6      private String lname;
7      private double weight;
8
9
0      private Student(Builder builder) {
1          this.age = builder.getAge();
2          this.fname = builder.getFname();
3          this.lname = builder.getLname();
4          this.weight = builder.getWeight();
5      }
6
7
8      public int getAge() {
9          return age;
0      }
1
2      public String getFname() {
3          return fname;
4      }
5
6      public String getLname() {
7          return lname;
8      }
9
0      public double getWeight() {
1          return weight;
2      }
3
```



```
    public static Builder createBuild() {
        return new Builder();
    }

    public static class Builder {
        private int age;
        private String fname;
        private String lname;
        private double weight;

        public Student build() {
            return new Student(this);
        }
        public int getAge() {
            return age;
        }
        public Builder setAge(int age) {
            this.age = age;
            return this;
        }
        public String getFname() {
            return fname;
        }
        public Builder setFname(String fname) {
            this.fname = fname;
            return this;
        }
        public String getLname() {
            return lname;
        }
        public Builder setLname(String lname) {
            this.lname = lname;
            return this;
        }
        public double getWeight() {
            return weight;
        }
        public Builder setWeight(double weight) {
            this.weight = weight;
```