



Today's agenda

↳ Builder design Pattern



AlgoPrep



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

```
class Student {  
    // Name  
    private String name;  
    private int age;  
    private double weight;  
    private String college;  
    private double salary;  
    // Setter method  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

```
Student s = new Student();  
s.setName("S");
```

name
s

② The attributes are immutable.

↳ if getter & setter are public, I can always change the value of attributes.



Soln2

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

```
class Student {  
    fName → 2  
    lName → 2  
    Private age → 2  
    weight → 2  
    college → 2  
    Salary →  
    ;  
    ;  
}
```

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

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

↳ Not understandable as well as bug Poone.

→ 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

College → 2

Salary →

;

;

}

Student (String ^{String} fName, int ^{int} age) {
 this.fName = fName;
 this.age = age;
}

Student (String ^{String} lName, int ^{int} weight) {
 this.lName = lName;
 this.weight = weight;
}

- Too many possible combination of constructors.
- Not all constructors may be possible to create.

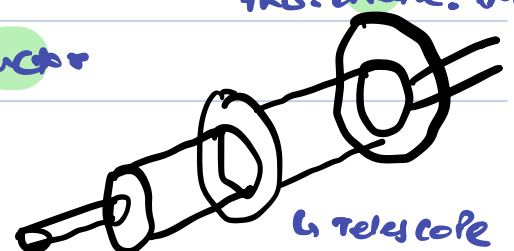
Student (fName, age) {
 this.fName = fName;
 this.age = age;
}

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





```
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;  
}
```



AlgoPrep



→ Ideally these should be 1 constructor

```
class Student {
```

```
    String fName
```

```
    String lName
```

```
    private int age
```

```
    int weight
```

```
    college yr
```

```
    salary
```

```
    ;
```

```
    ;
```

```
}
```

↓ any type can be placed
map<String, Object> val

```
Student ( ) {
```

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

```
    if (val.containsKey(age)) {  
        this.age = (int) val.get(age);  
    }
```

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

map<String, Object>

key	value
fName	Subhesh
age	24
weight	80
.	.
.	.

req-24



→ Client {

Tough to debug

map < > {

.put (fname, Subhash);

.put (age, 24);

}

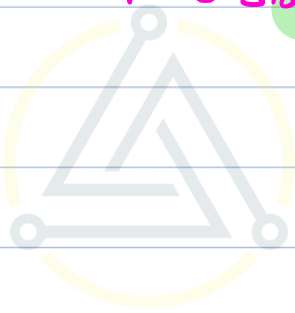
→ you might map number
to a string type

(in long string)

(fname, 131)

fname = "131"

↳ This issue should be caught.



AlgoPrep

Final Bol

class helper {

String fName

String lName

?: →

int age

int weight

College

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;

h.fName: !31?;

↓
Caught at compile time.

Student s = new Student(h);

}

Bolok 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        Student s1 = builder.build();
14        System.out.println(s1.getFname());
15    }
16 }
17
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) {}
}
```

```
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     public Student(Builder builder) {
10        this.age = builder.getAge();
11        this.fname = builder.getFname();
12        this.lname = builder.getLname();
13        this.weight = builder.getWeight();
14    }
15
16    public int getAge() {
17        return age;
18    }
19
20    public String getFname() {
21        return fname;
22    }
23
24 }
```

age: 24
fname: Subhesh
lname: null
weight: 80

age: 24
fname: Subhesh
lname: null
weight: 80



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
10    private 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    }
25
26    public String getLname() {
27        return lname;
28    }
29
30    public double getWeight() {
31        return weight;
32    }
33
34 }
```



```
4 public static Builder createBuild() {
5     return new Builder();
6 }
7
8 public static class Builder {
9     private int age;
10    private String fname;
11    private String lname;
12    private double weight;
13
14    public Student build() {
15        return new Student(this);
16    }
17    public int getAge() {
18        return age;
19    }
20    public Builder setAge(int age) {
21        this.age = age;
22        return this;
23    }
24    public String getFname() {
25        return fname;
26    }
27    public Builder setFname(String fname) {
28        this.fname = fname;
29        return this;
30    }
31    public String getLname() {
32        return lname;
33    }
34    public Builder setLname(String lname) {
35        this.lname = lname;
36        return this;
37    }
38    public double getWeight() {
39        return weight;
40    }
41    public Builder setWeight(double weight) {
42        this.weight = weight;
43    }
44 }
```

Console X