AGENDA

Builder design Pattern

- what
- why
- How to create BDP

start by 9:05 PM IST

*) BUILDER DESIGN PATTERN :

class with Many attributes

Student {

- name
- b-id
- phone-No
- age
- roll-No
- email
- Psp
- Grad-yr

}

creating Object of student : looks lik

```
Student st = new student();
st.setName("....")
st.setAge(19); .....
```

*) PROBLEM:

1.) lot of attributes

2.) if- want to validate object creation of class before creating Object

eg:

·) validation for age / phone-No etc

↓

Eventually - you do not want Object to be created unless validation Passes

MOST NAIVE SOLUTION:

#1) CONSTRUCTOR: who allows to create Objects.

↳ All validations should Go inside constructor ...

create constructor:

student(email, phone-no, grad-yr, batch-id,
name, age, univ ...) {

.....

}

PROBLEMS:

① lot of values. mistakes can happen

② code NOT readable

③ look at func to pass params again & again.

** ④ Passing Many 'null' → doesn't look Good.

new student("abc@gmail.com", null, 29, null,
null, 98262999, null ...)

⑤ if New attribute is added/ deleted →
constructor needs to be updated

*) SOLUTIONS:

I.) Overload constructor

↳ ① Lead to too many constructors

② Constructor with same signature cannot exist

student (string name , double psp)

student (string email , double psp)

NOT ALLOWED

The (1st) situation is actually called Telescoping constructors.

·> Have Many constructors

·> Internally → Telescoped together

① student(name){
this.name = name
}

② student (psp){
this . psp = psp
}

③ student (name, psp){
this . name = name
this. psp = psp
}

Internally → student(name, psp){

```
this(name)
this(psp)
}
```

→ avoids code duplicacy.

II:) SOLUTION-2

•) Need constructor → Perform validation
•) Don't want many args

→ SOLUTION:

→ Rather than many Params → let's Pass a <K,V> (Map)

datatype of key: String
datatype of value: Object

→ <String, Object>

can be anything.

Now, do we need many constructors → ✗

& No many args → ✗

eg: Map:

Code:

```
student (Map<String, Object> map) {
    name = (String) map.get("name");
    age = (int) map.get("age");
    :
    :
    // Perform validations
}
```

*) PROBLEMS:

1.) lot of type casting
2.) client typo's : nama instead of name

NOT the best SOLUTION. but would work.

III.) BEST / IDEAL SOLUTION:

-> Main issue was - too many params

we want: + compile time check for keys
+ has key and values.

Objects → attribute : key
value : values.

·) cannot pass 19 as Age

REVISE:

-> we have student class
·) Many attributes
·) validations
→ Map won't work.

Instead:

We can create a Helper class & pass that as Param.

```
class HelperStudent {
    // add exact same attributes
    // in this class
}
```

Object of Helper : `Helper h = new Helper();`

Student Object creation :

```
student s = new student(h);
```

```
student(Helper h) {
    if( h.grad-yr > 2024)        | validations
        Throw Exception();
    .
    .
    .
this.name=                       | ASSIGN
this.age=                        | values.
this.email=
}
```

LEARNINGS:

① Helper class - helps to create Object of student / building student

↳ Hence BUILDER DP

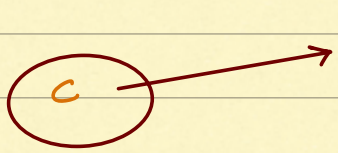
② Allows to create object where

-) we need to validate object creation
-) too many attributes.

validation would happen in student constructor itself.

DEMO.

*] IMPLEMENTING BUILDER IN PRODUCTION



client want to create object of student

↓

Ideally student class should tell how to create object....

*] STEPS:

- Expose method in student - getBuilder()
- In order to call this → should be static
- Move capability of Helper to student class
  - → create build() in student
  - → Eventually:
    - Get Builder from student
    - set attributes on builder
    - call build();
- All the setters should return Builder datatype.
- Make constructor of student as private as NO one should call. Every client should use Builder.

AGENDA

Builder design Patteren

- what
- WHY
- How to create BDP

start by 9:05 PM IST

*) BUILDER DESIGN PATTERN:

class with Many attributes

Student {

- name
- b-id
- phone-No
- age
- roll-No
- email
- Psp
- Grad-yr

}

creating Object of student: looks lik

```
Student st = new student();
st.setName("....")
st.setAge(19); .....
```

*) PROBLEM:

1.) lot of attributes

2.) if- want to validate object creation of class before creating Object

eg:

·) validation for age / phone-No etc

↓

Eventually - you do not want Object to be created unless validation Passes

MOST NAIVE SOLUTION:

#1) CONSTRUCTOR: who allows to create Objects.

↳ All validations should go inside constructor ...

create constructor:

```
student(email, phone-no, grad-yr, batch-id,
        name, age, univ ...){
    ......
}
```

PROBLEMS:

① lot of values. mistakes can happen

② code NOT readable

③ look at func to pass params again & again.

** ④ Passing many 'null' → doesn't look good.

```
new student("abc@gmail.com", null, 29, null,
            null, 98262999, null...)
```

⑤ if New attribute is added/ deleted →
constructor needs to be updated

*) SOLUTIONS:

I.) Overload Constructor

↳ ① Lead to too many constructors

② Constructor with same signature cannot exist

student (string name , double psp)
student (string email , double psp)

NOT ALLOWED

The (1st) situation is actually called Telescoping constructors.

·> Have Many Constructors
·> Internally → Telescoped together

① student(name){
this.name = name
}

② student (psp){
this . psp = psp
}

③ student (name, psp){
this . name = name
this. psp = psp
}

Internally → student(name, psp) {

```
this(name)
this(psp)
}
```

→ avoids code duplicacy.

II.) SOLUTION-2

•) Need constructor → Perform validation

•) Don't want many args

→ SOLUTION:

→ Rather than many params → let's pass a <K,V> (Map)

datatype of key: String

datatype of value: Object

→ <String, Object>

can be anything.

Now, do we need many constructors → ×

& No many args → x

Eg: Map:

Code:

```
student (Map<String, Object> map) {
    name = (String) map.get("name");
    age = (int) map.get("age");
    :
    :
    // Perform validations
}
```

*) PROBLEMS:

1.) lot of type casting
2.) client typo's : nama instead of name

Not the best solution. but would work.

III.) BEST / IDEAL SOLUTION:

-> Main issue was - too many params

we want: + compile time check for keys
+ has key and values.

Objects → attribute : key
value : values.

-) cannot pass 19 as Age

REVISE:

-> we have student class
-) Many attributes
-) validations
→ Map won't work.

Instead:

We can create a Helper class & pass that as Param.

```
class HelperStudent {
    // add exact same attributes
    // in this class
}
```

Object of Helper : `Helper h = new Helper();`

Student Object creation :

```
student s = new student(h);
```

```
student(Helper h) {
    if( h.grad-yr > 2024)
        Throw Exception();
    .
    .
    .
    this.name=
    this.age=
    this.email=
}
```

validations

ASSIGN values.

LEARNINGS:

1. Helper class - helps to create Object of student / building student
   → HENCE BUILDER DP

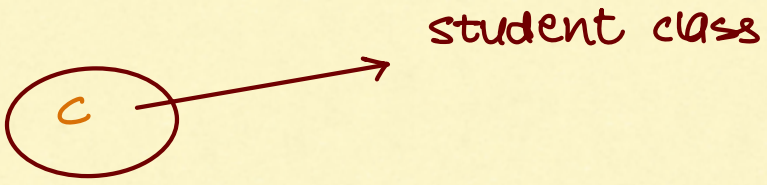
(2) Allows to create object where

-) we need to validate object creation
-) too many attributes.

validation would happen in student constructor itself.

DEMO.

*] IMPLEMENTING BUILDER IN PRODUCTION



client want to create object of student

↓

Ideally student class should tell how to create object....

*] STEPS:

- Expose method in student - getBuilder()
- In order to call this → should be static
- Move capability of Helper to student class
  - → create build() in student
  - → Eventually:
    - Get Builder from student
    - set attributes on builder
    - call build();
- All the setters should return Builder datatype.
- Make constructor of student as private as NO one should call. Every client should use Builder.