⇒ BUILDER DESIGN PATTERN.

# Class with lot of attributes.

```
Class Student {
        String name;
        int age;
        String batch
        double psp;
        String univName;
        int gradYear;
        String phoneNo;
}
```

```
Student s = new Student()
s.setName(——);
s.setAge(——);
s.setPsp(——)
```

# We want to validate the object before its creation.

Validations.

1. Students should have gradYear < 2020.
2. Phone no. should be valid.

⇒ No Student object should be created without checking validations.

```
Class Student {
        String name;
        int age;
        String batch
        double psp;
        String univName;
        int gradYear;
        String phoneNo;
}
```

```
Student(name, age,
        batch, psp,
        univName,
        gradYear, phoneNo){
    if(gradYear > 2021) {
        throw ____
    }
    ____
    ____
    ____
}
```

```
psvm(){
    Student st = new Student("Kailash", 25,
                   "Morning", 84.26, . . .
                   - - - - - - -);
}
```

Issues.

1. Difficult to understand.
2. Prone to Errors.

# Class Student {

    Student(name) {
        this.name = name
    }
    Student(name, age) {
        this.name = name
    }
    Student(name, univName) {
        this.~~name~~ = name
    }
    Student(name, batch) {
    }
}

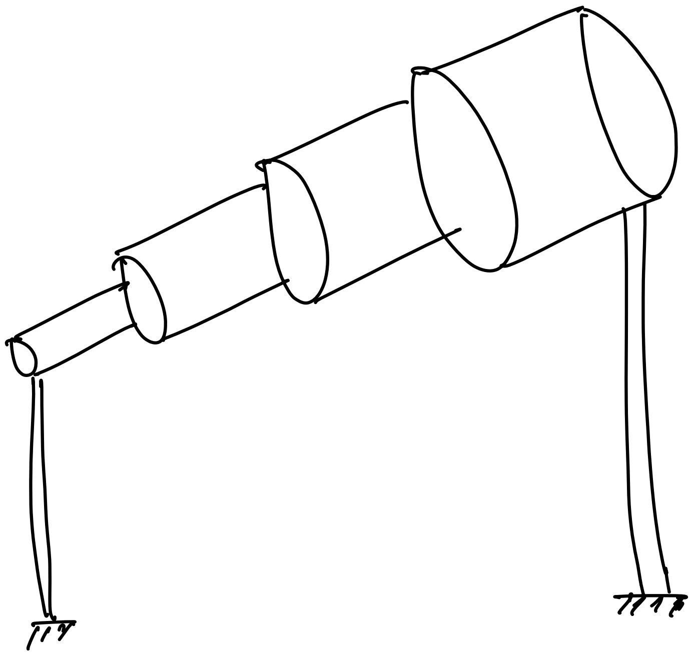
⇒ Issue :

Too many constructors.

⇒ 2^N

⇒ Too Constructors with same method signatures are not even allowed.

⇒ Telescopic Constructors.



```
Student (name){
    this.name = name;
}

Student (name, univName){
    this(name);
    this.univName = univName;
}
```

⇒ Class Student {

```
Student (param){ attrs ⇒
    ____
}
```

⇒ Map:

Some data structure that can allow us to pass multiple

name : ____

age : ____

batch : ____

```
HashMap< String, Object >. map.

Class  Student {

    ______
    ______
    ______
    ______
    ______
    ______

    Student ( Map< String, object >  map ){
        this.name  = (String) map.get("name");
        this.age = (Integer) map.get("age");
        ______
        ______
}
```

→ Type Casting

⇒ It can lead to some Runtime Exceptions.

```
psvm() {

    Map< String, Object >  map = ______________ ;
Typo.    map.put( "nama", "Shorabh");
    ______    ______
    ______    ______
    ______    ______
    Student  st = new Student (map);
}
```

→ No compile time check on ~~Attr~~ names.

# Something which is like a Map (it allows us to store attr names with their values) and also provides compile time safety over the attribute names & attribute type.

map.name = ~~75~~

map.nama = " ~~~~ "

```
Class ~~Helper~~ Builder {
    String name;
    int age;
    String batch
    double psp;
    String univName;
    int gradYear;
    String phoneNo;
}
```

```
Helper helper = new Helper();
helper.setName(__75__);
helper.setGradYear(____);
```

psvm().

```
Class Student {

    Student(Helper helper) {
        // Validations.
        this. name = helper. name;
        this. age = helper. age;

    }
}
```

→ No type casting.

→ helper.nama

⇒ BUILDER.

Allows us to create an object where we have

① Class with too many attributes.

② Validate before object creation.

———— * ————