BUILDER.

1) Class with a lot of attributes.

```
Class Student {
    - Name
    - age
    - batch
    - Psp
    - univName
    - gradYear
    - PhoneNo
}
```

```
Student st = new Student();

st.setName(—)
st.setAge(—)
st.setBatch(—)
st.setPsp(—)
______
______
```

2) We want to Validate the Student object before its creation.

Validations

→ gradYear <= 2023

→ Phone no. should be valid.

→ ____

→ ____

No Student object should be created if any of the validation fails.

```
Class  Student {
        - Name
        - age
        - batch
        - Rep
        - univName
        - gradYear
        - PhoneNo

        Student (String name, int age, String batch,
                  double psp, String univName, . . . ){
            if (age > 20) {
                    throw ———
            }
            if (phone is invalid){
                    throw ———
            }
            ———
            ———
            ———
            this.name = name
            this.age = age
            ———
            ———
            ———
        }
}
```

```
PSVM () {

    Student st = new Student("Raj", 20, "Eve",
                             90.0, ------);

        ↳ Prone to Errors.
        ↳ Difficult to understand.
}

Class Student {
      - Name
      - age
      - batch
      - Rep
      - univName
      - gradYear
      - phoneNo
      Student (String name) {
            this.name = name;
      }

      Student (String name, int age)
            this.name = name;
            this.age = age;
      }
```

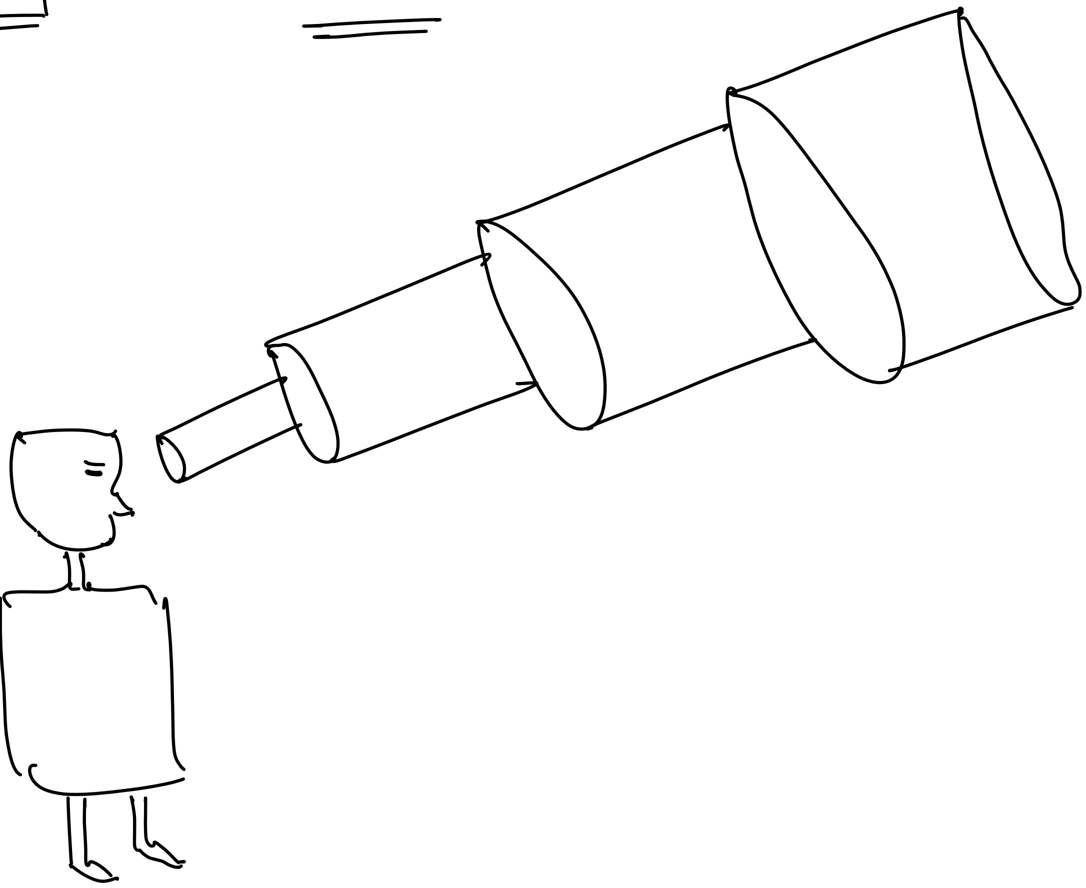
① ⇒ N attrs. : # of Constructors = 2^N

```
Student (String univ, int age){

}
```

② Some of the constructors might not be feasible as well because of same signature.

Telescopic Constructors.



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

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

```
Student(name, age, batch)
        this(name, age)
        this.batch = batch
}
```

Usecase

$$\text{Student}(\underbrace{\langle\text{Data Structure}\rangle}_{\downarrow})$$

Some DS that can allow us to pass multiple attributes of different datatypes.

"name" : ____

"age" : ____

"batch" : ____

____

____

$\Bigg\}$ Map

Map<String, Object>

```
Class  Student {
        - Name
        - age
        - batch
        - Rep
        - univName
        - gradYear
        - PhoneNo
        Student (Map<String, Object>  map) {
            this.name = (String)map.get("name");
            this.age = (Integer)map.get("age");
            =
        }
```

```
map.put("nama", —);
map.put("age", "Scaler");
```

"Scaler"

Runtime Exception.

Issues

1) Typo. in keys.
2) Type mismatch issue.

⇒ What we need?

Something like map, that allows to store different values each with a different name. Also it should provide a compile time check over the values & keys.

ds.nama ×

ds.age = "—";

```
Class Helper {
    - Name
    - age
    - batch
    - Rep
    - univName
    - gradYear
    - phoneNo
}


Student {
    - Name
    - age
    - batch
    - Rep
    - univName
    - gradYear
    - phoneNo

    Student (Helper helper){
        //Validations
        this.name = helper.name
        this.age = helper.age
        ____
        ____
        ____
    }
}
```

```
psvm() {
    Helper helper = new Helper();
    helper.setName(—);
    helper.setAge(—);  => Compile Time Check.
    helper.setBatch(——);
  { helper.name = ——
  {
    Student st = new Student(helper);
}
```