2. Builder Design Pattern

- Builder design pattern is used to construct an object step by step, when an object has lot of attributes.
- Builder design pattern can be used, when we want to ensure that object should be created only after the validations are completed

Problem

- We have a Student class with a lot of attributes

```
no usages  new *
public class Student {
    no usages
    private int id;
    no usages
    private String name;
    no usages
    private int age;
    no usages
    private double psp;
    no usages
    private String batch;
    no usages
    private String collegeName;
    no usages
    private int graduationYear;
    no usages
    private String email;
    no usages
    private String mobileNumber;
    no usages
    private double totalYearsOfExperience;
}
```

- Normally we can either have a huge constructor which takes all the attributes & have the validation inside the constructor, or we can create the object using default constructor & use setters to set all the attributes.

- But before the constructor code execution begins, the object would have already been created in the memory, constructor will just initialise the values for all the attributes
- So we want to validate the attributes of the class even before creating the object.
- The validations are,
    - Graduation year should be > 2023
    - Email should be proper
    - Phone number should be proper
    - Age should be > 0

What we want?

- No Student object should be created unless all the validations are performed.

Issues with Constructor chaining:

- Having all the attributes in the constructor is difficult to understand & it is prone to error, in case if certain attributes can be set later & not at the time of object creation. In this case, we have pass null or default values for that attribute in the constructor
- Having multiple constructors for different combinations of the attributes will lead to tons of constructors
- Having multiple constructors can lead to Telescoping constructors, which should be avoided.

Telescopic Constructor:

```
// Telescopic constructor
1 usage  new *
public Student(String name){
    this.name = name;
}
1 usage  new *
public Student(String name, int age){
    this(name);
    this.age = age;
}
no usages  new *
public Student(String name, int age, double psp){
    this(name, age);
    this.psp = psp;
}
```

Student Builder:

- Create another class called StudentBuilder with all the attributes of the Student class
- Version 1 - https://github.com/PraveenS1997/DesignPatterns/tree/main/Builder/v1
- Version 1 resolves the constructor issues & also the Student object will be created only after all the validations are done on the Student attributes
- Version 2 - https://github.com/PraveenS1997/DesignPatterns/tree/main/Builder/v2
- Version 3 - https://github.com/PraveenS1997/DesignPatterns/tree/main/Builder/v3