Builder Design Pattern

Builder Design Pattern is a part of creational design pattern.

- **It will be helpful when you create an object.**

```
public class Phone
{
private String os;
private String processor;
private Double screenSize;
private int battery;
private int camera;

public Phone(String os, String processor, Double screenSize, int battery, int camera)   //To create an object we need to pass 5 values. We can use setters or constructors.

                {
                super();
                this.os=os;
                this.processor=processor;
                this.screenSize=screenSize;
                this.battery=battery;
                this.camera=camera;
                }

@override
public String toString()
{
return “Phone[os=“+os+”,Processor=“+processor
+”,ScreenSize=“+screenSize+”,Battery=“+batterry+”,Camera=“+camera+”]”;
}
}
```

```
public class Shop
{
public static void main(String args[])
{
Phone p=new Phone(“Android”, “Qualcom”,5.5,3100,13);
System.out.println(p);
}
}
```

```
Phone p1=new Phone(“Android”,5.5,”Snapdragon”); // not possible
```

```
public class Phonebuilder
{
		private String os;
		private String processor;
		private Double screenSize;
		private int battery;
		private int camera;

		public PhoneBuilder setOs(String os) {
				this.os = os;
				return this;
		}
		public PhoneBuilder setProcessor(String processor) {
				this.processor = processor;
				return this;
		}
		public PhoneBuilder setScreenSize(double screenSize) {
				this.screenSize = screenSize;
				return this;
		}
		public PhoneBuilder setBattery(int battery) {
				this.battery = battery;
				return this;
		}
		public PhoneBuilder setCamera(int camera) {
				this.camera = camera;
				return this;
		}
		public Phone getPhone()
		{
				return new Phone(os, processor, screenSize, battery, camera):
		}
}
```

```
public class Shop
{
public static void main(String args[])
{
Phone p=new PhoneBuilder(). setOs(“Android”). setScreenSize(5.5).getPhone();
System.out.println(p);
}
}
```

- So for creating object we have to pass the parameters in sequence and we have to set all the parameters.
- But suppose we want to set only os and screensize and don't want to set all the parameters.
- Some users don't concern about all the parameters only they want a phone.
- Issue-

1. We don't want to set all the parameters.
2. Even you want to pass all the parameters you should know the sequence of all the parameters.

Hence we use Builder Design Pattern.

TO fill a bottle of CocaCola-

1. Sanitize
2. Filled with liquid
3. Cap is attached
4. Label is applied.

Like that

There will be a

1. getPhone() method that creates a phone object.
2. Various setters are there that will set the attributes with some value individually.
3. It is not compulsory to follow a fixed sequence while initializing the object.

So you will implement this thing when you will have multiple parameters in the constructor.

Advantage of Builder Design Pattern

- The main advantages of Builder Pattern are as follows:
  - It provides clear separation between the construction and representation of an object.
  - It provides better control over construction process.
  - It supports to change the internal representation of objects.