Builder Design Pattern

- When creating objects of classes.Sometimes you don't need to pass all the values to the object.
- For example you have created a Laptop class with variables like brand,model,price,ram,VGA,HDD,SSD are the variables.
- But when creating the objects some laptops don’t have the VGA,SSD or HDD.
- In such cases you would have to pass null values when creating objects.
- To avoid this issue.You can use builder design patterns.
- After implementing according to the builder design pattern you don’t need to pass null values or 0 values.
- Example code below.

```
public class Phone  5 usages  VimukthiWaththegama
{ private String brand;
    private String model;  3 usages
    private String color;  3 usages
    private double screenSize;  3 usages
    private double price;  3 usages
    private String processorName;  3 usages

    public Phone( String brand, String color, String model, double price, String processorName, double screenSize )
    {
        this.brand = brand;
        this.color = color;
        this.model = model;
        this.price = price;
        this.processorName = processorName;
        this.screenSize = screenSize;
    }

    public Phone( String brand, double screenSize, String processorName, String model, String color, double price )
    {
        this.brand = brand;
        this.screenSize = screenSize;
        this.processorName = processorName;
        this.model = model;
        this.color = color;
        this.price = price;
    }
```

Class with variables and the constructor

```
public class PhoneBuilder  9 usages  VimukthiWaththegama
{
    private String brand;  2 usages
    private String model;  2 usages
    private String color;  2 usages
    private double screenSize;  2 usages
    private double price;  2 usages
    private String processorName;  2 usages
```

Create another class(builder class) and define the values again here.

```java
public PhoneBuilder setBrand( String brand )  1 usage  VimukthiWaththegama
{
    this.brand = brand;
    return this;
}

public PhoneBuilder setScreenSize( double screenSize )  no usages  VimukthiWaththegama
{
    this.screenSize = screenSize;
    return this;
}

public PhoneBuilder setProcessorName( String processorName )  no usages  VimukthiWaththegama
{
    this.processorName = processorName;
    return this;
}

public PhoneBuilder setPrice( double price )  1 usage  VimukthiWaththegama
{
    this.price = price;
    return this;
}

public PhoneBuilder setModel( String model )  no usages  VimukthiWaththegama
{
    this.model = model;
    return this;
}

public PhoneBuilder setColor( String color )  1 usage  VimukthiWaththegama
{
    this.color = color;
    return this;
}
```

Create a class(builder class) and create setters.but return type is not ‘void’.The return type should be the class type(builder class type)

```
public Phone getPhone(){  2 usages  VimukthiWaththegama
    return new Phone( brand,color,model,price,processorName,screenSize );
}
```

And create a get method inside the builder class and return the class that we need to create according to the builder design pattern.
Return a new object of the class and pass the values.Those values are the values we have defined in the builder class

```
//Builder design pattern
Phone phone = new PhoneBuilder().setBrand( "Samsung" ).setColor( "Black" ).getPhone();
System.out.println(phone);

Phone phone1 = new PhoneBuilder().setPrice( 100000.00 ).getPhone();
System.out.println(phone1);
```

Now you can create objects without passing unnecessary values,