Builder Design Pattern

I got clarity why builder design pattern is used.

Earlier I used to think why not just pass null args to places where field value is not given.

```
class ItemSolutionBuildWithDate{
    ItemSolution itemSolution;
    EDDDetails eddDetails;
    ...
}
```

When I was working on it on V4, I understood, how complicated, it was and how builder design pattern helps in the PR not getting too long.

```
class Car{
    private String engine;
    private int wheels;
    private int seats;
    private boolean isManual;
    ...
    Notice how only getters are here.
    public String getEngine(){
        return engine;
    }
    public int getWheels(){
        return wheels;
    }
    public int getSeats(){
        return seats;
    }
    public boolean getIsManual(){
        return isManual;
    }
}
```

```
{private Car(CarBuilder carBuilder){
    this.engine = carBuilder.engine;
    this.wheels = carBuilder.wheels;
    this.seats = carBuilder.seats;
    this.isManual = carBuilder.isManual;
    ...
}
```

Private Constructor,

Static Inner Class instantiation we learnt in Bill Pugh's method

```
static class CarBuilder{
    private String engine;
    private int wheels;
    private int seats;
    private boolean isManual;

    public CarBuilder setEngine(String engine){
        this.engine = engine
        return this;
    }
    public CarBuilder setWheels(int wheels){
        this.wheels = wheels.
        return this;
    }
    public CarBuilder setSeats(int seats){
        this.seats = seats;
        return this;
    }
    public CarBuilder setManual(...){
        this.isManual = isManual;
        return this;
    }
    Final Build.
    public Car build(){
        return new Car(this);
    }
}
```

Builder Design Pattern.

Call from Main Method.

```
Car.CarBuilder builder = new Car.Carbuilder();
```

→ Normal new Key word initialisation.

OR

```
Car car = (Car.CarBuilder.setEngine()
          .setWheels()
          ⋮
          .build();
```

→ Static class' method is called so JVM internally initialises it.

```
private Car(Carbuilder carbuilder){
    this.engine = carbuilder.engine;
    this.wheels = carbuilder.wheels;
    this.seats = carbuilder.seats;
    this.isManual = carbuilder.isManual;
    ⋮
}
```

→ V.Imp Private Constructor,

Static Inner class instantiation we learnt in Bill Pughs method

```
static class CarBuilder{
    private String engine;
    private int wheels;
    private int seats;
    private boolean isManual;

    public CarBuilder setEngine(String engine){
        this.engine = engine
        return this;
    }

    public CarBuilder setWheels(int wheels){
        this.wheels = wheels.
        return this;
    }
    public CarBuilder setSeats(int seats){
        this.seats = seats;
        return this;
    }

    public CarBuilder setManual(....){
        this.isManual = isManual;
        return this;
    }

    Final Build.
    public Car build(){
        return new Car(this);
    }
}
```