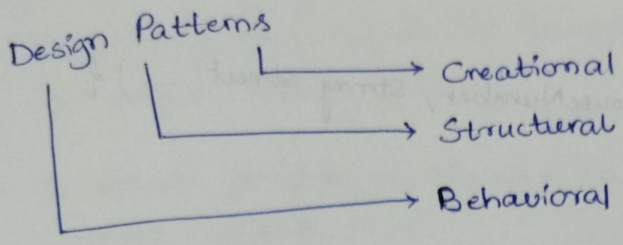
DESIGN PATTERNS



Creational

deals with the process of creation of objects of classes.

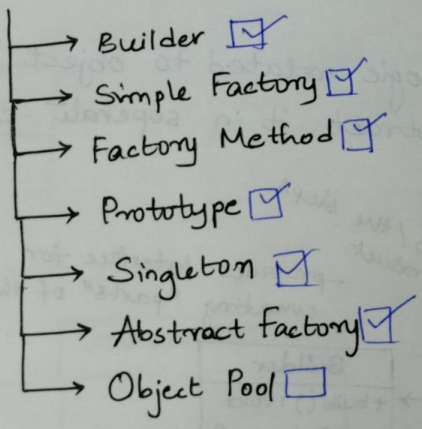
Structural

deals with how classes and objects are arranged to or composed

Behavioral

these patterns describe how classes and objects interact and communicate with each other.

Creational Patterns



What pblm does Builder Design Pattern Solves?

Objects that need other objects or "parts" to construct them.

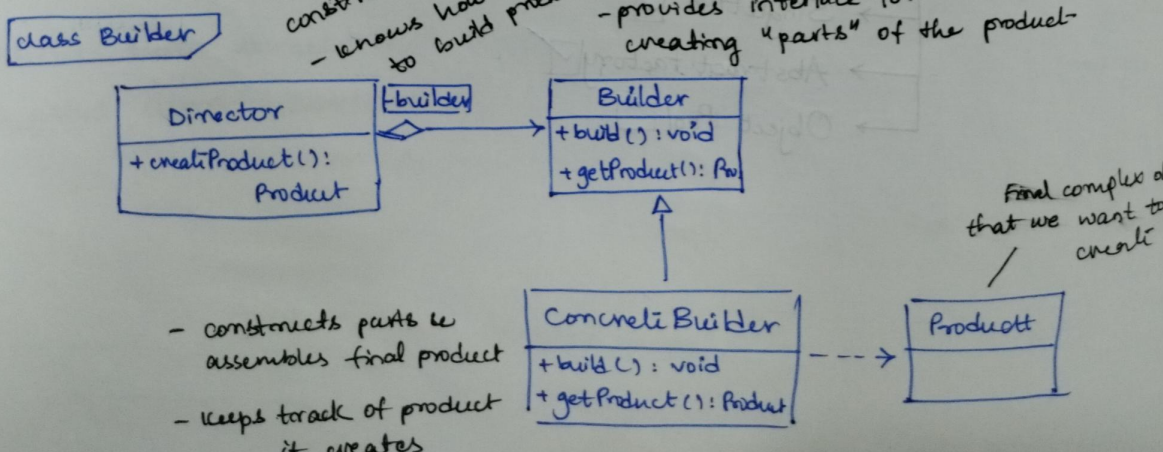
```
class Address {
    public Address (string houseNumber, string street, ...) {
        // initialise
    }
}

class User {
    public User (string Name, Address address, ....) {
        // initialise
    }
    //other code
}
```

What is Builder?

- We have a complex process to construct an object involving multiple steps, then builder design pattern can help us.
- In builder we remove the logic related to object construction from "client" code & abstract it in seperate classes.

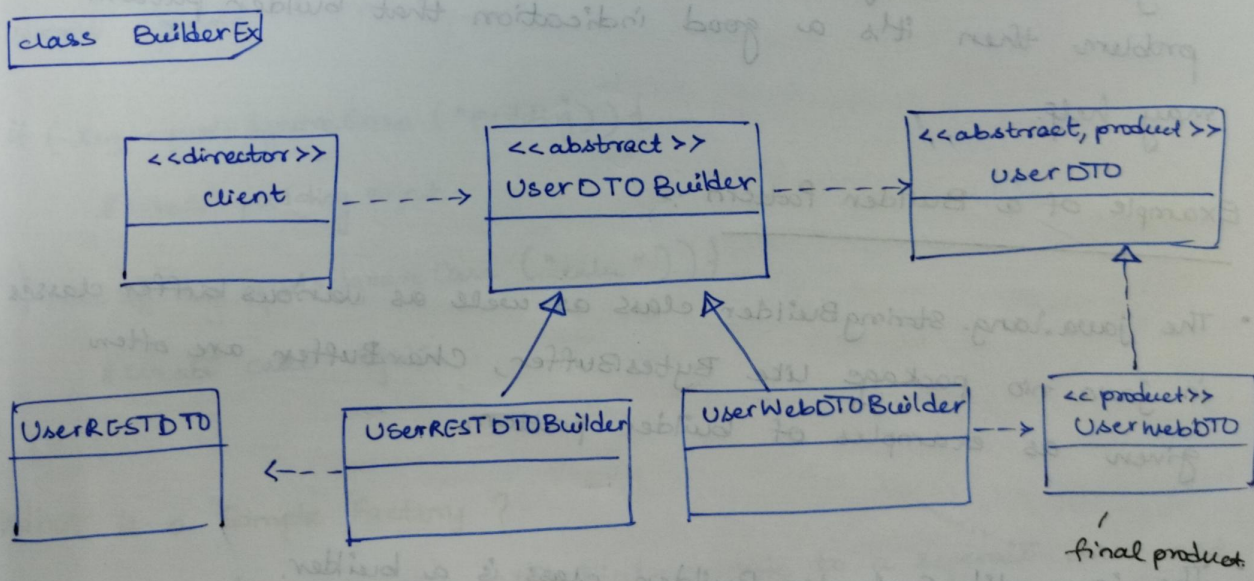
UML :-



Implement a Builder

- We start by creating a builder
  - Identifying the "parts" of the product & provide methods to create those parts.
  - It should provide a method to "assemble" or build the product / object.
  - It must provide a way / method to get fully built object out. Optionally builder can keep reference to an product it has built so the same can be returned again in future.
- A director can be a seperate class or client can play the role of director.

Example UML :-



Implementation Considerations

- You can easily create an immutable class by implementing builder as an inner static class. You'll find this type of implementation used quite frequently even if immutability is not a main course/ concern.

Design Conciderations

- The director role is rarely implemented as seperate class, typically the consumer of the object instance or the client handles that role.

- Abstract builder is also not required if "product" itself is not part of any inheritance hierarchy. You can directly create concrete builder.

- If you are running into a "too many constructor arguments" problem then it's a good indication that builder pattern may help.

Example of a Builder Pattern :-

- The java.lang.StringBuilder class as well as various buffer classes in java.nio package like BytesBuffer, CharBuffer are often given as examples of builder pattern.

- The java.util.Calender.Builder class is a builder.

↓

Java 8

Pitfalls :-

1.) A littlebit complex for beginners mainly because of 'method chaining', where builder methods return builder object itself.

2.) Possibility of partially initialised object ; user code can set only a few or none of properties using with xxx methods and call build(). If required properties are missing, build method should provide suitable defaults or throw exception.