#575 OOAD: Assignment 2 – Group 1

1. **How Factory & Builder patterns improve the design of your system?**

FACTORY DESIGN PATTERN :

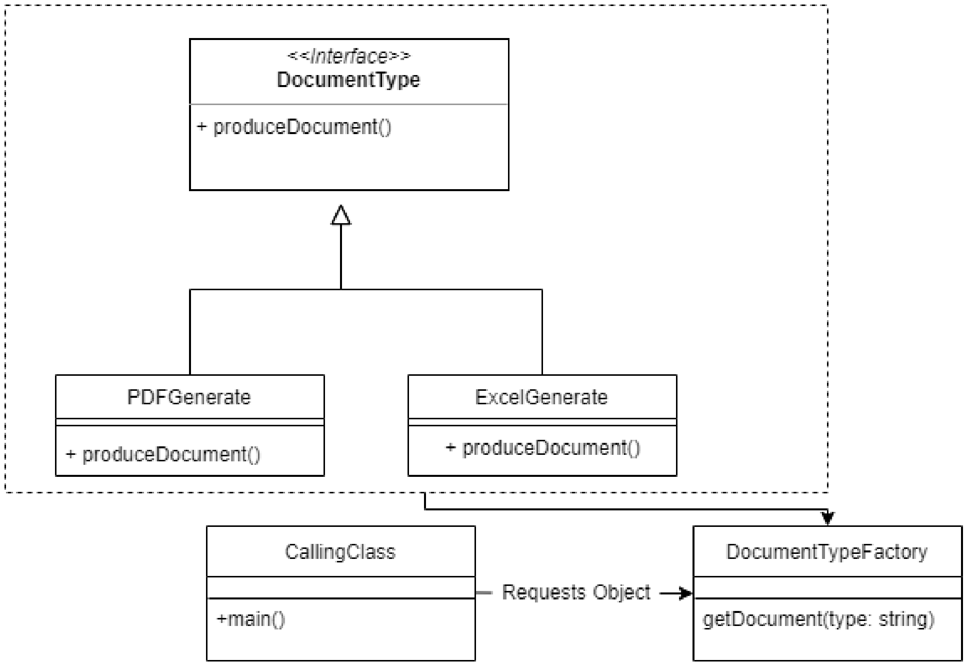
- The factory design pattern's objective is to return one of the sub-classes based on the input, when there is a superclass with multiple sub-classes.
- The factory design pattern comes under the creational design pattern, and this class provides abstraction between implementation & client classes.
- It removes the instantiation of client classes from client code
- In factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.
- In our project, first you need to define a document type interface. Next, we shall implement this document type into concrete classes. Then, we need to create a document Factory method that supplies document type based on the input.
- The idea here is that the caller requests the factory for a type of document type. The factory then supplies an appropriate document type object.

BUILDER DESIGN PATTERN:

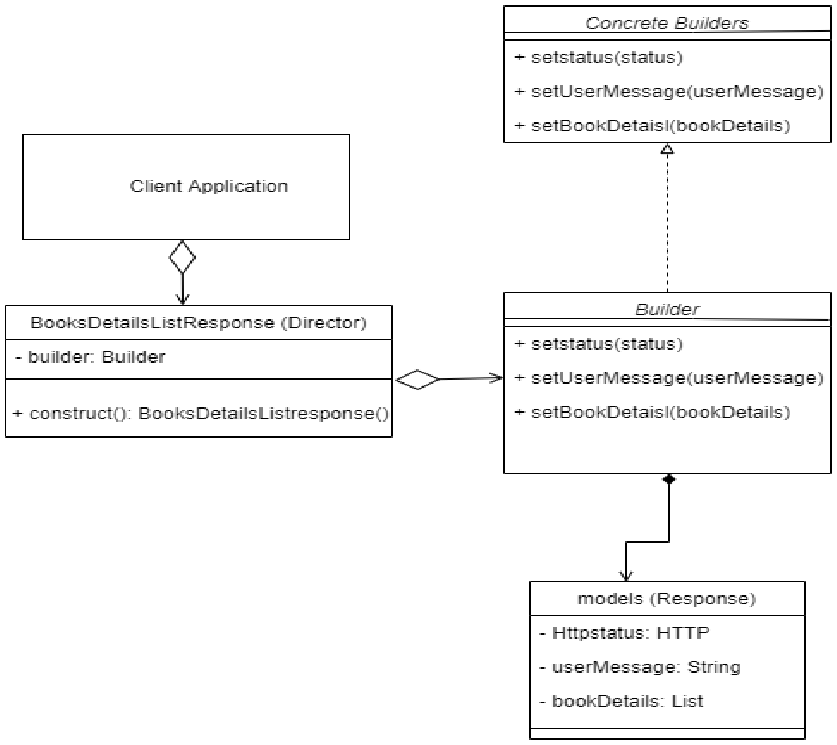
- The Builder design pattern also comes under the creational design pattern and provides one of the best ways to create an object.
- One of the main reasons of using builder design pattern is that it allows you to vary the product's internal representation, it encapsulates the code for construction and representation and provides control over the steps of construction process.
- Builder design pattern also helps in minimizing the number of parameters in the constructor and thus there is no need to pass in null for optional parameters to the constructor.
- In our project, we took BookDetailsResponse class and implemented the builder pattern, create static anonymous inner class named Builder to the pojo. We use static because we want to return / Use the current object.
- Add same fields to it from pojo. Also ADD setter of each field with return type of Builder Class.
- And finally ADD method build which will return the new BookDetailsResponse object instance.

2. Class Diagrams

A. FACTORY DESIGN PATTERN



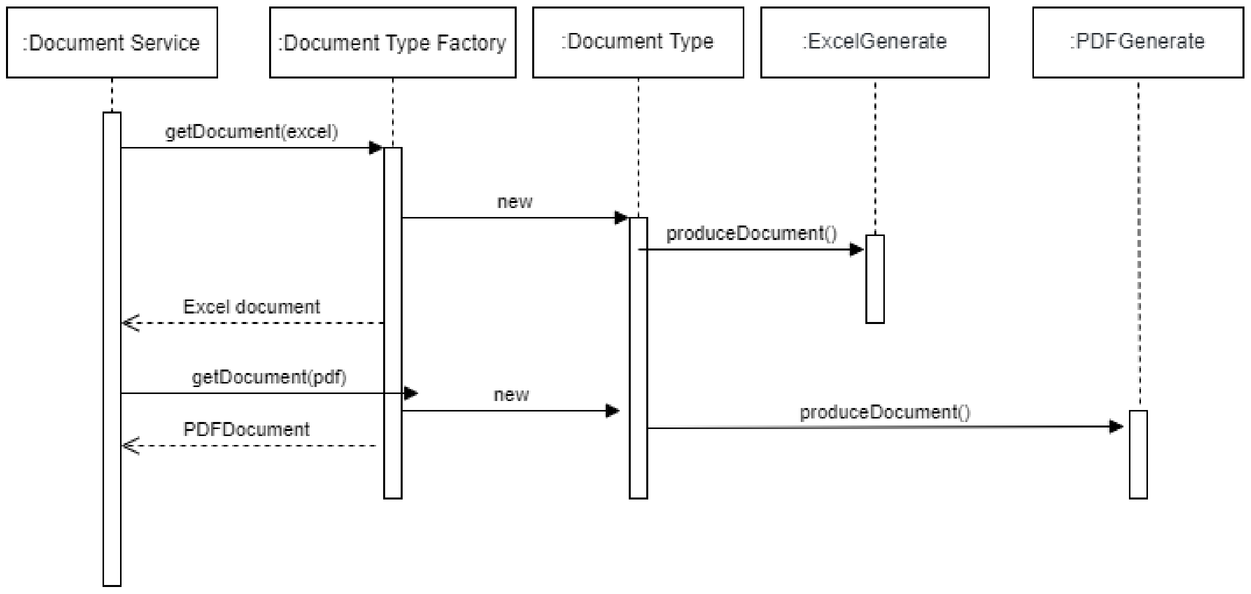
B. BUILDER DESIGN PATTERN



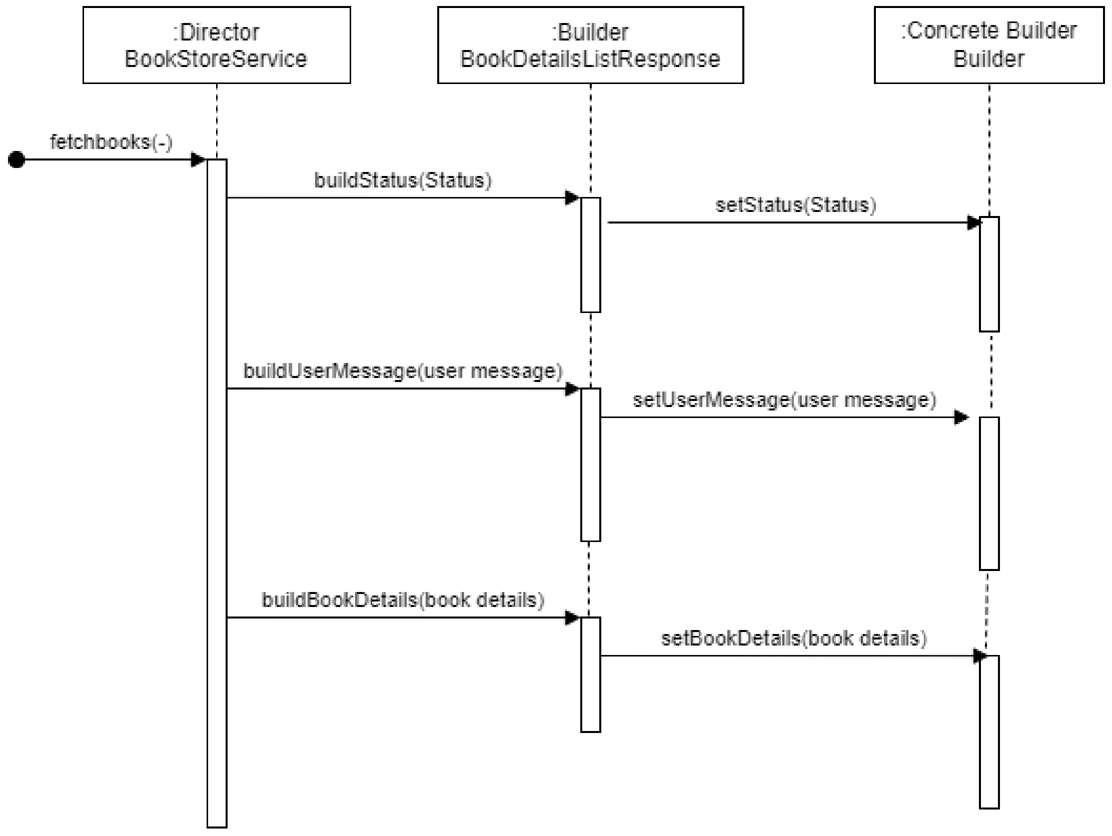
Concrete Builders
+ setstatus(status)
+ setUserMessage(userMessage)
+ setBookDetaisl(bookDetails)
Client Application
BooksDetailsListResponse (Director)
- builder: Builder
+ construct(): BooksDetailsListresponse()
Builder
+ setstatus(status)
+ setUserMessage(userMessage)
+ setBookDetaisl(bookDetails)
models (Response)
- Httpstatus: HTTP
- userMessage: String
- bookDetails: List

3. Sequence Diagrams

A. FACTORY DESIGN PATTERN



B. BUILDER DESIGN PATTERN


:Director BookStoreService
:Builder BookDetailsListResponse
:Concrete Builder Builder
fetchbooks(-)
buildStatus(Status)
setStatus(Status)
buildUserMessage(user message)
setUserMessage(user message)
buildBookDetails(book details)
setBookDetails(book details)