Pola Desain Perangkat Lunak

[Week] 6 – Creational Pattern, Builder

Prepared by: Tifanny Nabarian

Design Patterns Category

Creational Patterns	Creational patterns prescribe the way that objects are created.
Structural Patterns	• Structural patterns are concerned with how classes and objects are composed to form larger structures
Behavioral Patterns	• Behavioral patterns are concerned with algorithms and the assignment of responsibilities between objects.
Concurrency Patterns	• Concurrency patterns prescribe the way access to shared resources is coordinated or sequenced

Design Patterns Scope

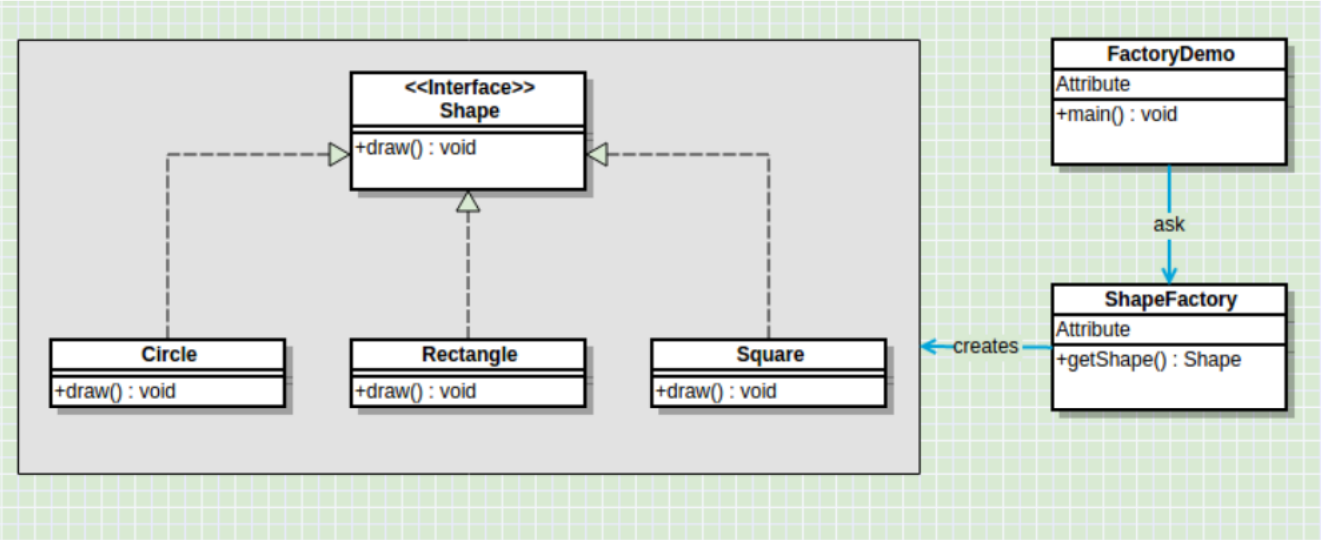
		Purpose		
		Creational	Structural	Behavioral
Scope	Class	• Factory method	• Adapter	• Interpreter • Template method
	Object	• Abstract factory • Builder • Prototype • Singleton	• Adapter • Bridge • Composite • Decorator • Fasad • Flyweight • Proxy	• Chain of responsibility • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor

Creational Pattern

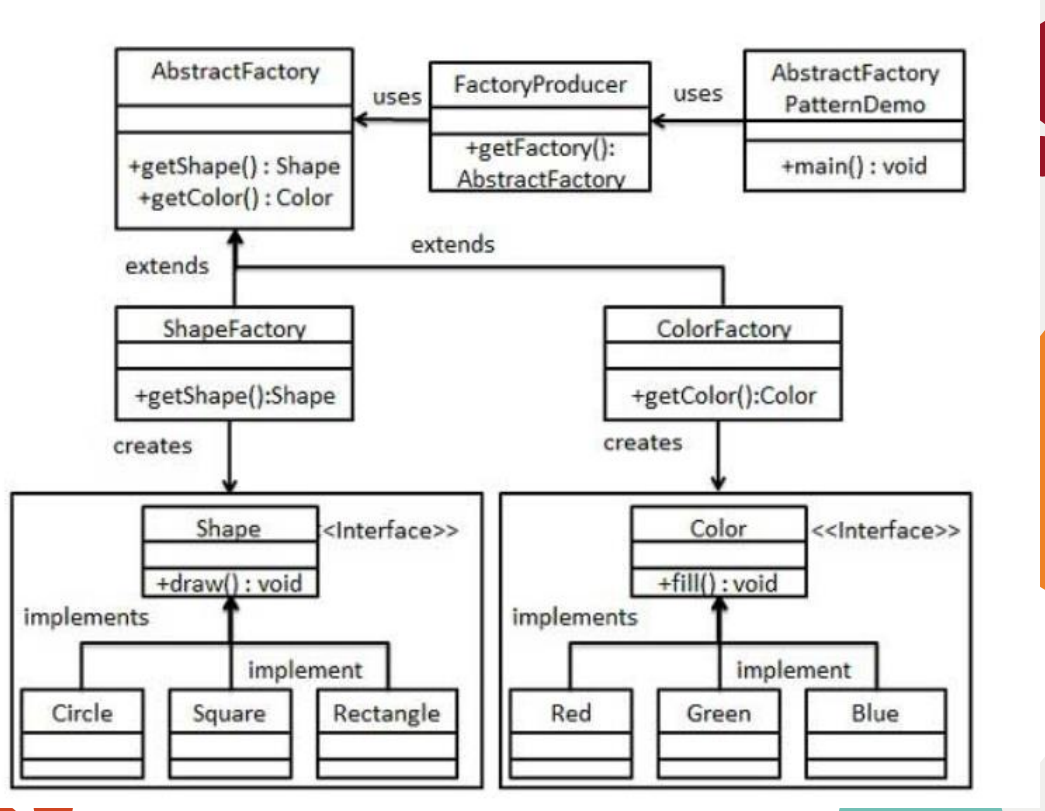
Review Factory Method and Abstract Factory

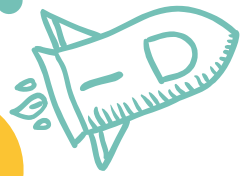
Factory Method

Example



Abstract Factory





Creational Pattern

Builder

Builder Concept

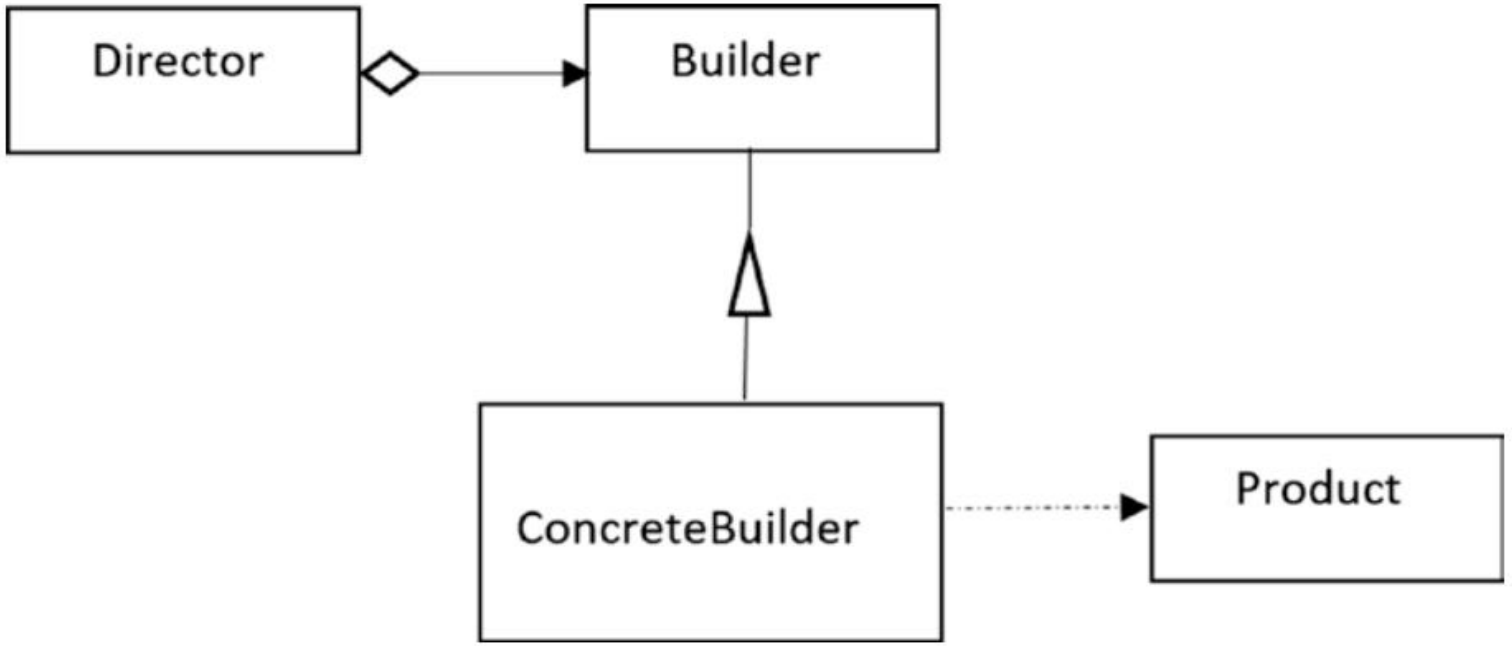
Definisi GoF

- Separate the construction of a complex object from its representation so that the **same construction processes** can **create different representations.**

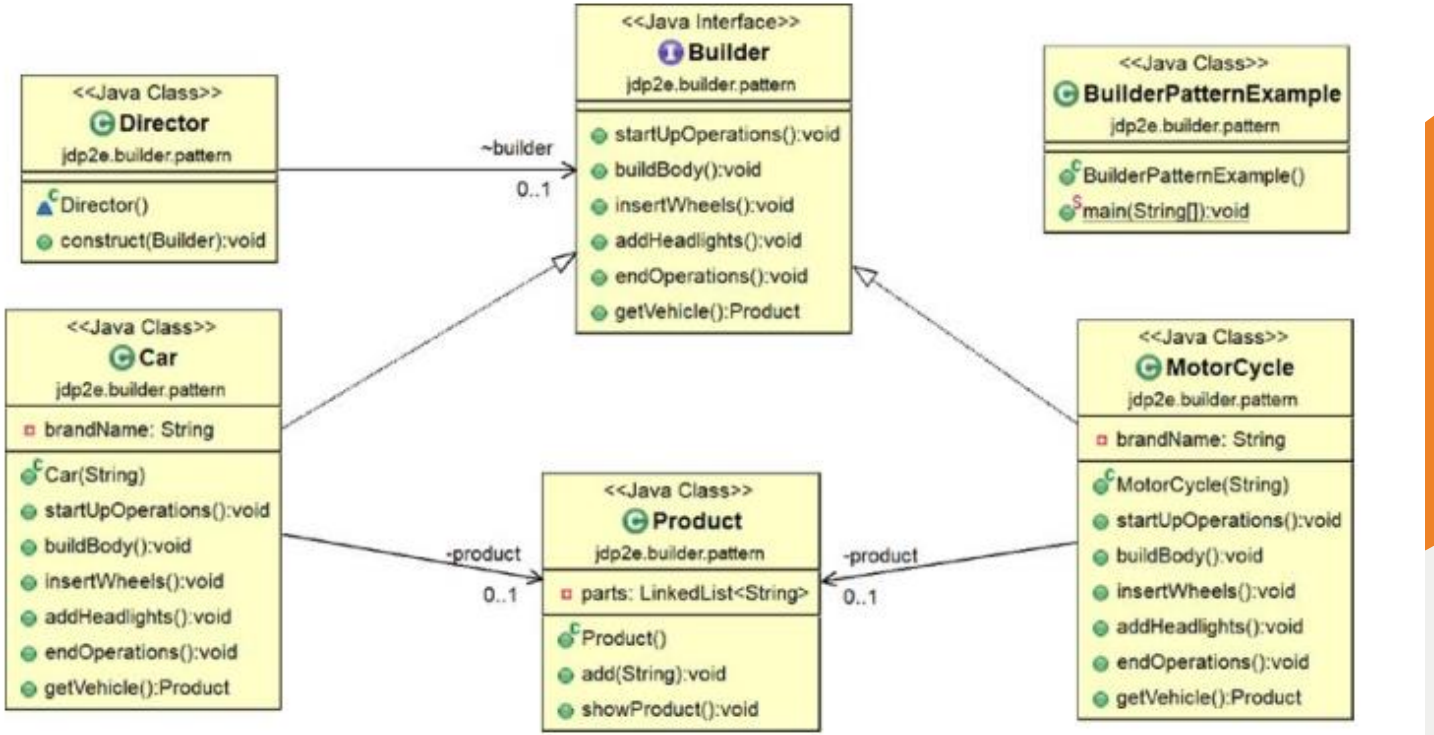
Real World Example

- To complete an order for a computer, different parts are assembled based on customer preferences (e.g., one customer can opt for a 500 GB hard disk with an Intel processor, and another customer can choose a 250 GB hard disk with an AMD processor). -> "**PC Rakitan"**

Builder Concept



Builder Example



Builder Example (Illustration)

In this example, we have the following participants: Builder, Car, MotorCycle, Product, and Director. The first three are very straightforward; Car and MotorCycle are concrete classes and they implement the Builder interface. Builder is used to create parts of the Product object, where Product represents the complex object under construction.

Since Car and MotorCycle are the concrete implementations of the Builder interface, they need to implement the methods that are declared in the Builder interface. That's why they needed to supply the body for the `startUpOperations()`, `buildBody()`, `insertWheels()`, `addHeadlights()`, `endOperations()`, and `getVehicle()`methods. The first five methods are straightforward; they are used to perform an operation at the beginning (or end), build the body of the vehicle, insert the wheels, and add headlights. `getVehicle()` returns the final product. In this case, `Director` is responsible for constructing the final representation of these products using the `Builder` interface. (See the structure defined by the GoF). Notice that `Director` is calling the same `construct()` method to create different types of vehicles.

Builder

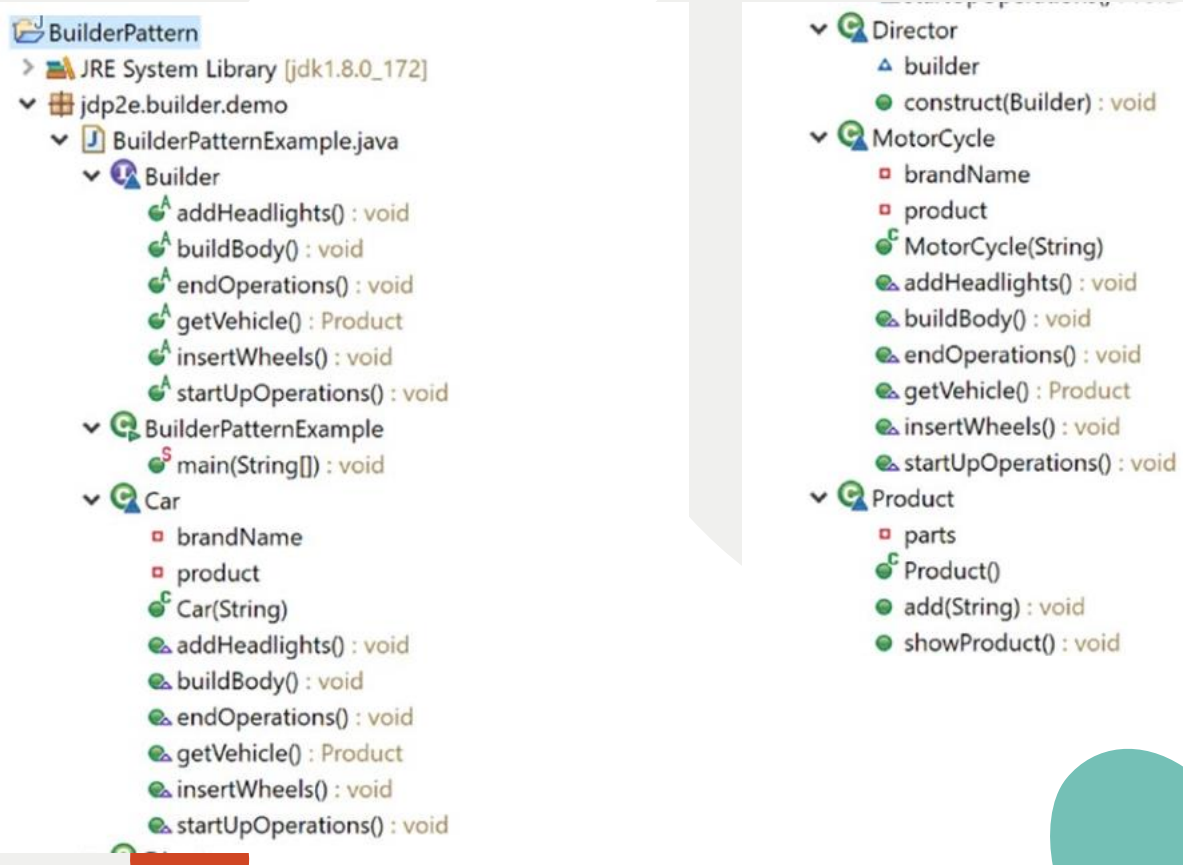
- Pada saat membangun suatu obyek bisa jadi terdapat lebih dari 1 kelas builder dengan implementasi yang berbeda-beda
- Setiap implementasi dari kelas builder menghasilkan representasi obyek yang berbeda-beda. Pemisahan ini mengurangi ukuran dari obyek
- Desain yang dihasilkan akan lebih modular, dan mudah untuk menambahkan builder baru
- Proses konstruksi objek menjadi independen dari komponen yang membentuk objek sehingga proses konstruksi objek lebih mudah dikontrol

What are the advantages of using a builder pattern?

- You can create a complex object, step by step, and vary the steps
- Using this pattern, the same construction process can produce different products.
- Since you can vary the construction steps, you can vary product's internal representation

Let's Practice!

Builder Example (Package Explorer)



Thank you!