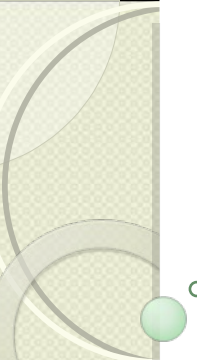




Design Pattern

Behavioral & Structural
Patterns



		Purpose		
		Creational (process of) object creation	Structural (composition of classes or objects)	Behavioral (how classes interacts and distribute responsibility)
Scope	Class (relationship between classes & subclasses, fixed at compile time)	Factory Method	Adapter	Interpreter Template Method
	Object (can be changed at run time and more dynamic)	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor



Behavioral Patterns

- Iterator
- Template Method
- Visitor
- Memento
- Strategy
- etc...

Iterator DP

- Tujuan

- Menyediakan sarana untuk mengakses elemen-elemen sebuah agregasi tanpa bergantung kepada representasi strukturalnya

- Konteks

- Ketika akan mengakses agregasi objek tanpa mengetahui representasi internal
- untuk mendukung pengurutan(transversal) dari objek agregate dari banyak cara
- Untuk menyediakan cara yang seragam untuk mengakses berbagai aggregate objek

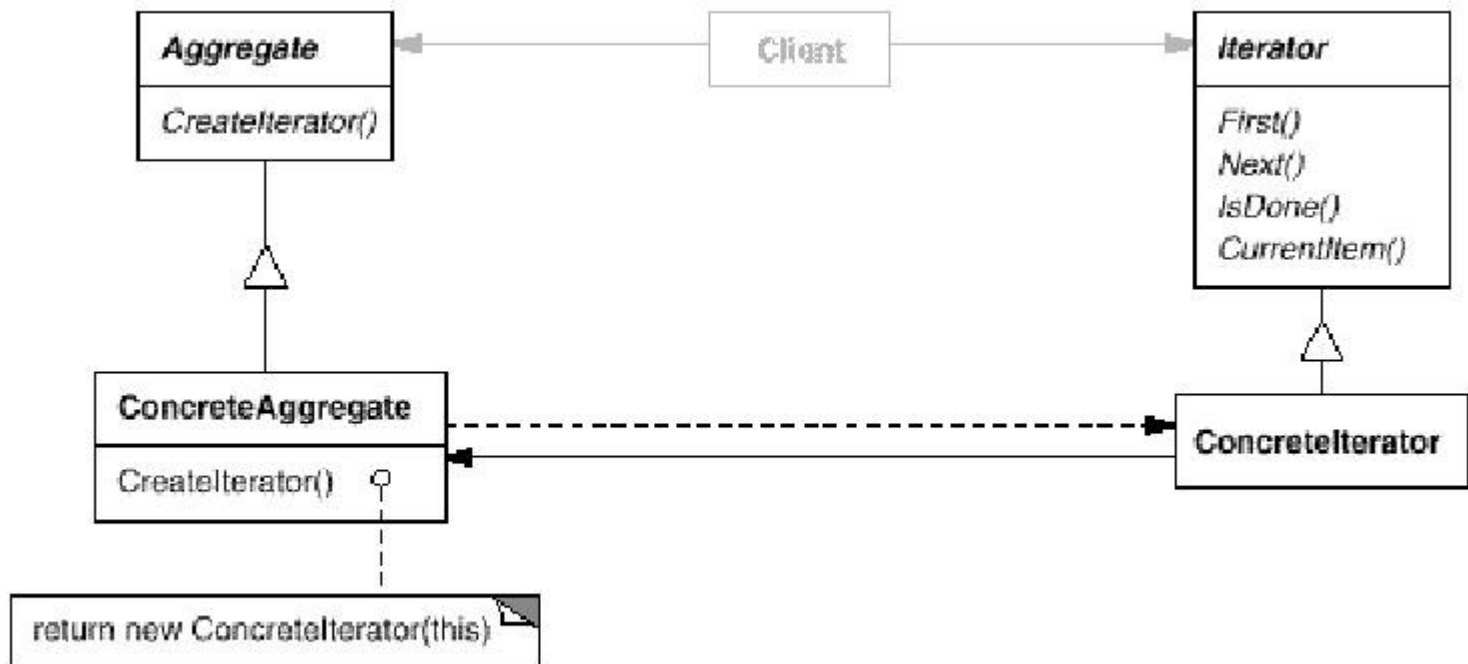


Iterator DP

- Actors:
 - Iterator (abstract)
 - ConcreteIterator
 - Agregate
 - ConcreteAggregate

Iterator DP

- Structure



Visitor DP

- Tujuan

- memungkinkan adanya penambahan operasi yang dapat dilakukan pada sebuah struktur tanpa perubahan terhadap struktur tersebut

- Konteks

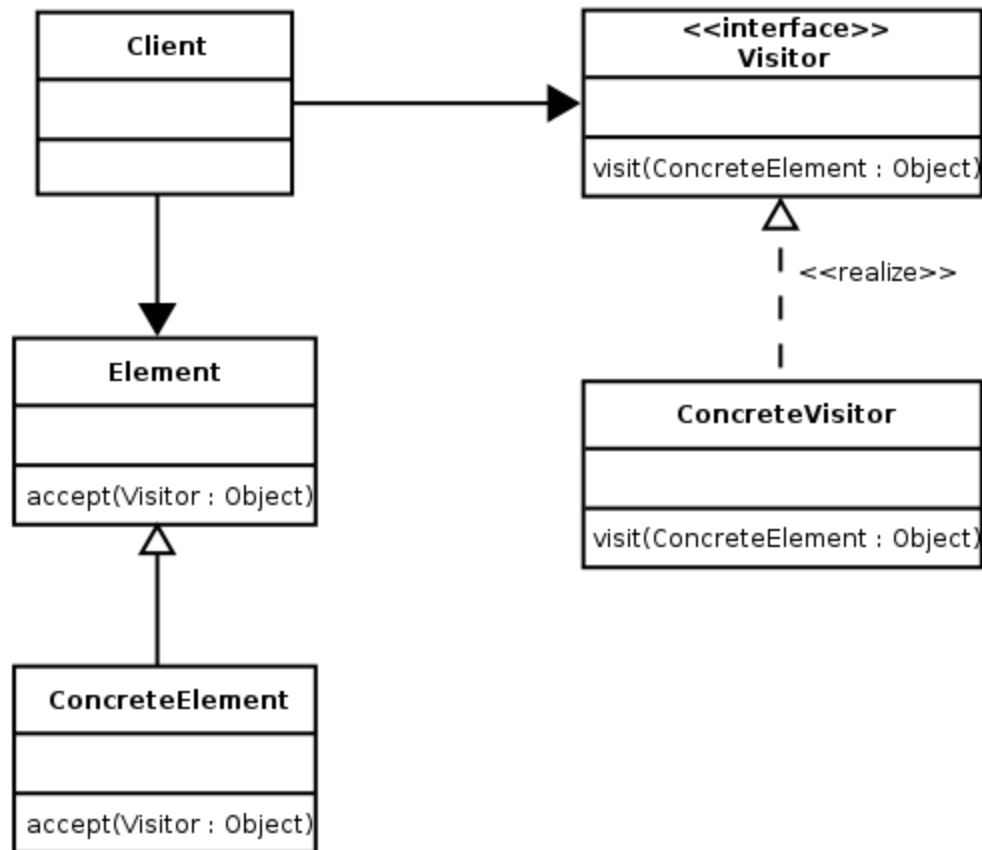
- Sebuah struktur terdiri dari banyak obyek dan ingin dilakukan operasi terhadap struktur tersebut
- Kita ingin menghindari adanya “polusi” operasi-operasi baru terhadap sebuah struktur
- Struktur objek jarang berubah, namun banyak operasi-operasi baru yang perlu didefinisikan

Visitor DP

- Actors
 - Visitor
 - ConcreteVisitor
 - Element
 - ConcreteElement

Visitor DP

- Structure



Strategy DP

- Tujuan

- Mendefinisikan sebuah kumpulan algoritma, kemudian mengenkapsulasi setiap algoritma dan membuat mereka dapat saling ditukarkan

- Konteks

- Ada banyak kelas yang saling terhubung dan berbeda dalam implementasi operasinya
- Kita memerlukan banyak variants dari sebuah algoritma

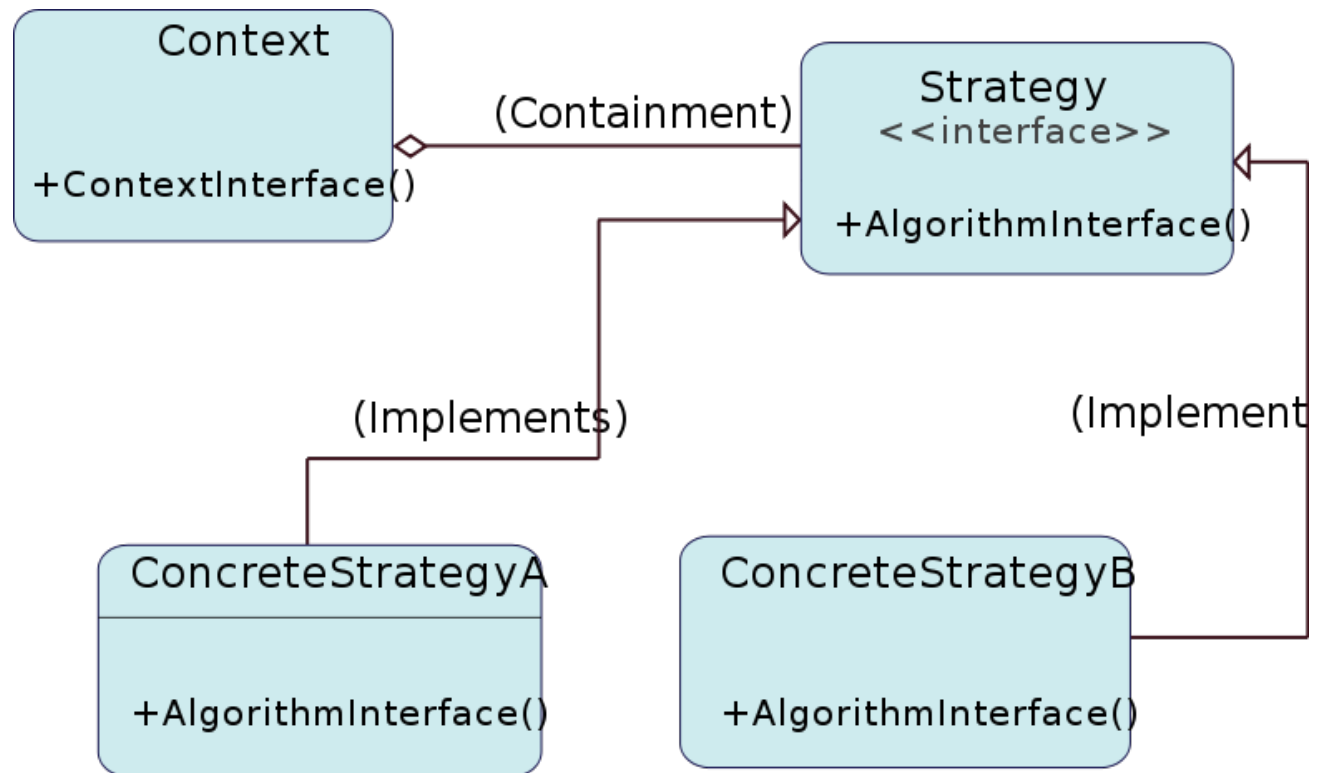


Strategy DP

- Actors
 - Strategy
 - ConcreteStrategy
 - Context

Strategy DP

- Structure





Template Method DP

- Tujuan

- Mendefinisikan kerangka dari sebuah algoritma dalam sebuah metode, dan menunda pendefinisian setiap langkah dalam subclass

- Konteks

- Kita ingin mendefinisikan bagian yang invarian dari sebuah algoritma, tanpa menentukan bagian spesifiknya
- Kita ingin mengontrol pendefinisian metode pada subclass

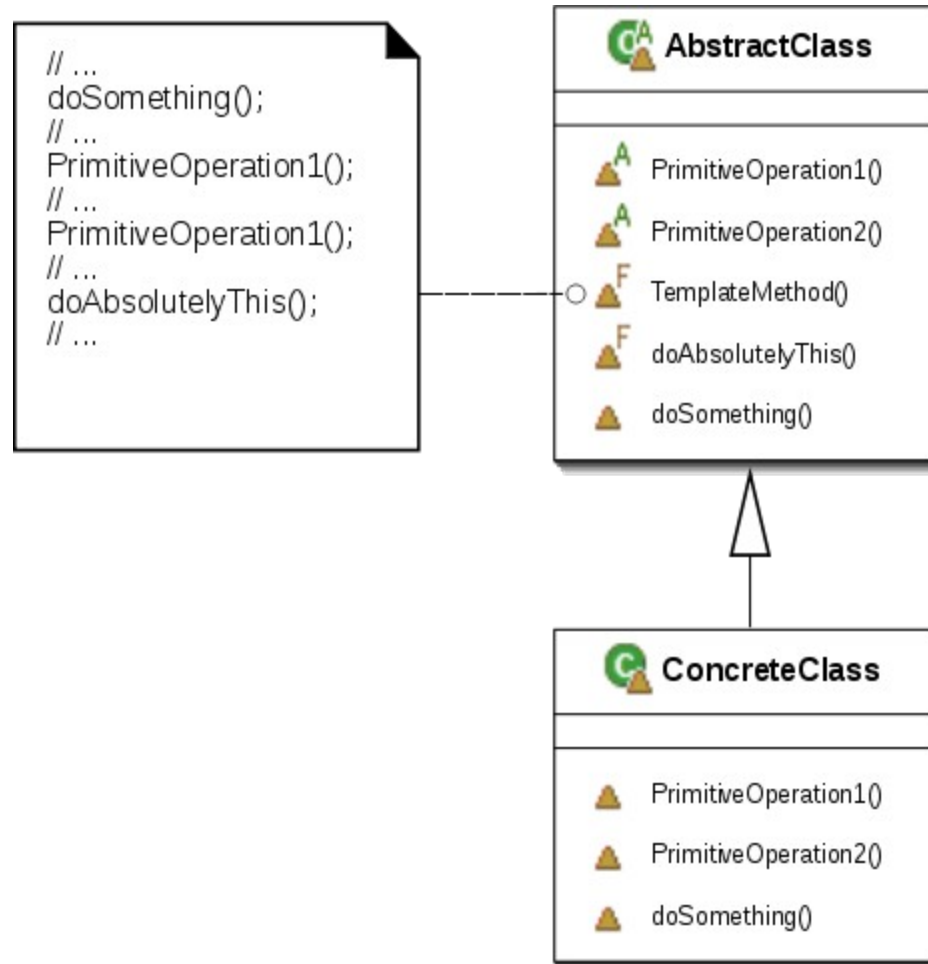


Template Method DP

- Actors
 - Abstract Class
 - Concrete Class

Template Method DP

- Structure





Structural DP

- Tujuan utama:
 - Memudahkan identifikasi relasi dan komunikasi antar struktur-struktur objek
- Some popular structural DPs:
 - Adapter
 - Bridge
 - Composite
 - Decorator
 - Facade
 - Flyweight
 - Proxy

Adapter DP

- Tujuan

- Mengubah interface sebuah kelas menjadi interface lain yang diinginkan

- Konteks:

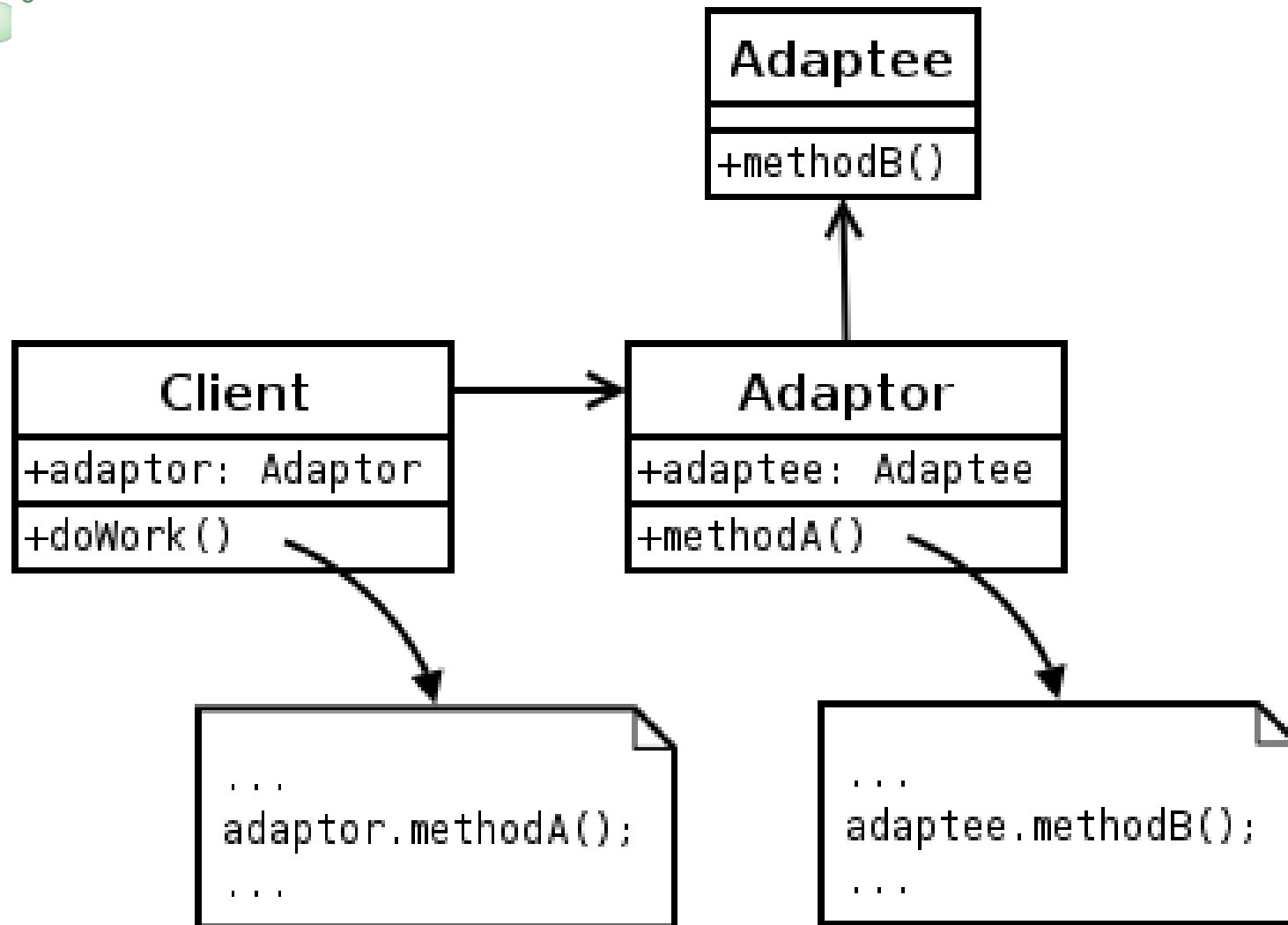
- Kita akan menggunakan sebuah kelas, akan tetapi kelas tersebut tidak memiliki interface dengan format/bentuk yang sesuai
- Kita ingin menggunakan kelas yang dapat digunakan kembali (*reusable*) dengan kelas-kelas lain yang belum diketahui sebelumnya



Adapter DP

- Actors
 - Client
 - Adaptor
 - Adaptee

Adaptor DP



Composite DP

- Tujuan

- Mengkomposisikan objek ke dalam struktur tree untuk merepresentasikan relasi *whole-part* (keseluruhan-bagian)/ komposisi
- Menyeragamkan interface untuk bagian dan keseluruhan (gabungan)

- Konteks

- Kita ingin merepresentasikan hirarki dari objek-objek.
- Kita ingin *client* untuk mengabaikan perbedaan antara komposisi dan bagian-bagiannya

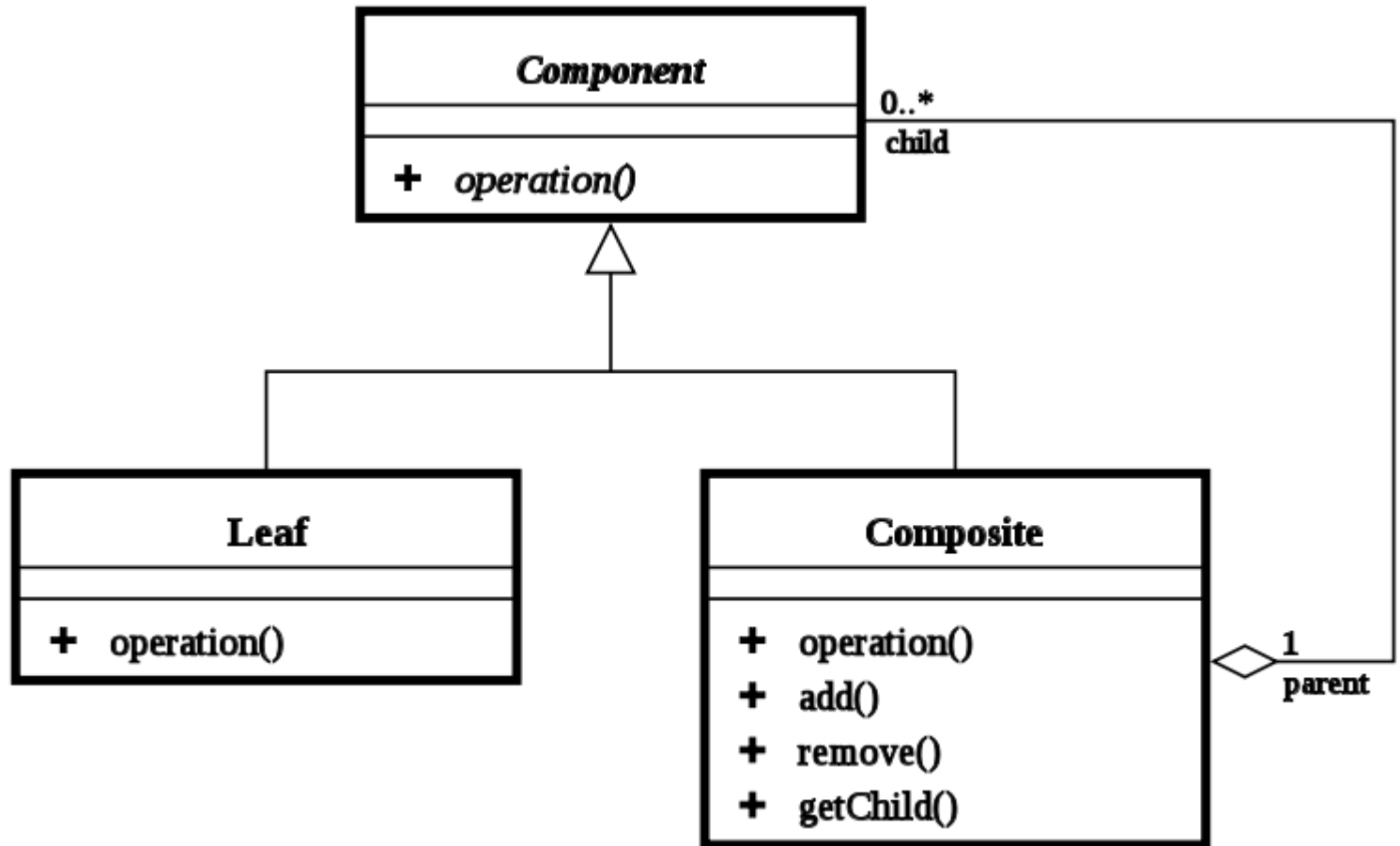


Composite DP

- Actors
 - Client
 - Component
 - Composite
 - Leaf

Composite DP

- Structure





Other Design Patterns

- Masih ada beberapa DP lain yang belum kita kaji
- Pada Buku Design Patterns: Elements... terdapat 23 Dps yang dikelompokkan ke dalam Creational, Behavioral dan Structural
- Umum dikaji juga, DP untuk Concurrency: Lock, Monitor, etc...
- Noteworthy also: architectural patterns: layers, MVC, MV-VM, peer-to-peer etc.