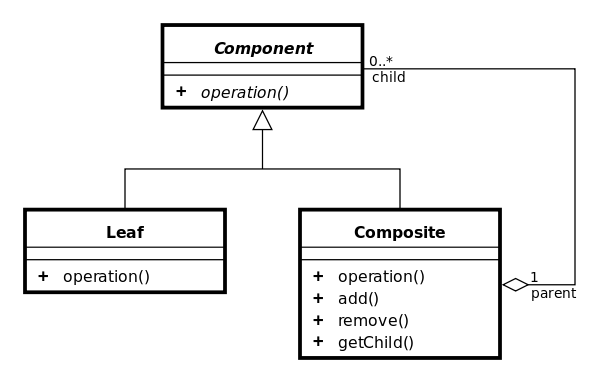
# http://www.dofactory.com/Patterns/Diagrams/strategy.gifStrategy Pattern

## Vorteile

* Es wird eine Familie von Algorithmen definiert.
* Es wird die Auswahl aus verschiedenen Implementierungen ermöglicht und dadurch erhöhen sich die Flexibilität und die Wiederverwendbarkeit.
* Es können Mehrfachverzweigungen vermieden werden und dies erhöht die Übersicht des Codes.
* Strategien bieten eine Alternative zur Unterklassenbildung der Kontexte.

## Nachteile

* Man muss die verschiedenen Strategien um den Context initialisieren zu können.
* Gegenüber der Implementierung der Algorithmen im Kontext entsteht hier ein zusätzlicher Kommunikationsaufwand zwischen Strategie und Kontext.
* Die Anzahl von Objekten wird erhöht.

# Composite Pattern

## Vorteile

* Einheitliche Behandlung von Primitiven und Kompositionen
* Leichte Erweiterbarkeit um neue Blatt- oder Container-Klassen

## Nachteile

* Ein zu allgemeiner Entwurf erschwert es, Kompositionen auf bestimmte Klassen (und damit zumeist Typen) zu beschränken.

# Observer Pattern

## Vorteile

* Subjekte und Observer können unabhängig variiert werden. Subjekt und Observer sind auf abstrakte und minimale Art lose gekoppelt. Das beobachtete Objekt braucht keine Kenntnis über die Struktur seiner Observer zu besitzen, sondern kennt diese nur über die Observer-Schnittstelle. Ein abhängiges Objekt erhält die Änderungen automatisch.

## Nachteile

* Änderungen am Objekt führen bei großer Observeranzahl zu hohen Änderungskosten. Einerseits informiert das Subjekt jeden Observer, auch wenn dieser die Änderungsinformation nicht benötigt. Zusätzlich können die Änderungen weitere Änderungen nach sich ziehen und so einen unerwartet hohen Aufwand haben.
* Der Mechanismus liefert keine Information darüber, was sich geändert hat. Die daraus resultierende Unabhängigkeit der Komponenten kann sich allerdings auch als Vorteil herausstellen.

# Adapter Pattern

## Vorteile

* Die Vorteile eines Klassenadapters bestehen darin, dass er sich genau einer Zielklasse anpasst und dadurch nur das Verhalten der Zielklasse überschreiben kann. Der Objektadapter kann auch Unterklassen mit anpassen

## Nachteile

* Nachteilig wirkt sich aus, dass ein Klassenadapter nicht zur automatischen Anpassung von Unterklassen verwendet werden kann.

# Façade Pattern

## Vorteile

* Die Fassade fördert die lose Kopplung, weil mehrere Schnittstellen zu einer zusammengefasst werden, und senkt die Komplexität, da sie das zugrunde liegende Subsystem versteckt. Außerdem kann das Subsystem durch die lose Kopplung leichter erweitert werden.

## Nachteile

* Der Nachteil besteht darin, dass eine zusätzliche Indirektionsstufe eingeführt wird.

# Factory Method Pattern

## Vorteile

* Fabrikmethoden entkoppeln ihre Aufrufer von Implementierungen konkreter Produkt-Klassen. Das ist insbesondere wertvoll, wenn Frameworks sich während der Lebenszeit einer Applikation weiterentwickeln – so können zu einem späteren Zeitpunkt Instanzen anderer Klassen erzeugt werden, ohne dass sich die Applikation ändern muss.

## Nachteile

* Die Verwendung dieses Erzeugungsmusters läuft auf Unterklassenbildung hinaus. Es muss eine eigene Klasse vorhanden sein, die die Klassen-Methode zur Erzeugung aufnehmen kann.

# Template Pattern

## Vorteile

* Kein Doppelter Code in Klassen

## Nachteile

* Grosse Abhängigkeiten zur Superclass (🡪Implementation neuer Funktionalität allenfalls Aufwändig.)

# State Pattern

## Vorteile

* Ein Vorteil des Systems ist, dass komplexe und schwer zu lesende Bedingungsanweisungen vermieden werden können.
* Es können neue Zustände und neues Verhalten auf einfache Weise hinzugefügt werden. Die Wartbarkeit wird erhöht und Zustandsobjekte können wiederverwendet werden.

## Nachteile

* Auf der anderen Seite rechtfertigt der Nutzen bei sehr einfachem zustandsbehaftetem Verhalten unter Umständen nicht den teils beträchtlichen Implementierungsaufwand. Kann das Objekt sehr viele Zustände annehmen, in denen jeweils nur sehr wenige Aktionen erlaubt sind, muss dennoch jeder Zustand für jede Aktion der anderen Zustände Code enthalten, um die Schnittstelle korrekt zu implementieren, auch wenn in diesen jeweils nur eine Ausnahmebehandlung stattfindet. In einer großen Bedingungsanweisung ließe sich die Ausnahmebehandlung hingegen in einem gemeinsamen "sonst"-Zweig vereinen.

# http://www.johnloomis.org/ece538/notes/Horstmann/ch05/iterator/Ch5_un03.jpgIterator Pattern

## Vorteile

* Die Implementierung der zu Grunde liegenden Datenstruktur bleibt verborgen

## Nachteile

* Wenn der Iterator sein Aggregat kennt und/oder das Aggregat über seine Iteratoren Buch führt, verteuern sich vor allem das Erzeugen und Vernichten von Aggregaten. (Im Gegenzug erhält man eine höhere Sicherheit)

# Proxy Pattern

Vorteile

?

Nachteile

?