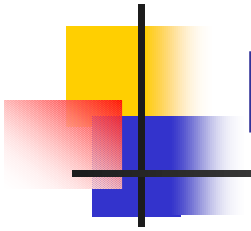
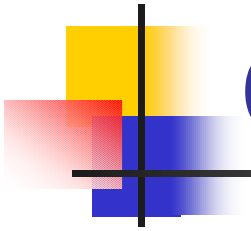


Die Schablonenmethode

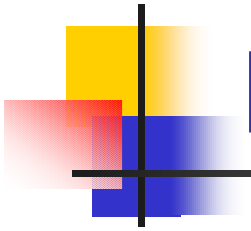
Thomas Heidinger

14.12.2001



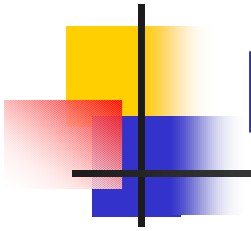
Gliederung

- Motivation
- Anwendbarkeit
- Struktur
- Teilnehmer
- Konsequenzen
- Implementierungsaspekte
- Beispiel
- Verwendung
- Verwandte Muster



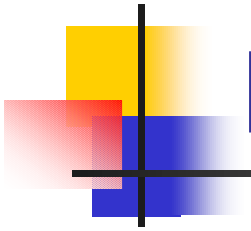
Motivation

- Bsp. Sortieralgorithmus
  - Struktur bleibt gleich, egal ob aufsteigend oder absteigend sortieren.
  - Zweimal vollständige Implementierung sicherlich nicht sinnvoll.



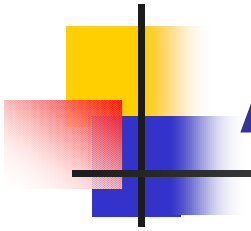
Motivation

- Wiederverwendung von Code
- Herausfaktorisierung von gemeinsamen Verhalten
- Lediglich das Skelett eines Algorithmus definieren, einzelne Schritte an Unterklassen delegieren



Motivation

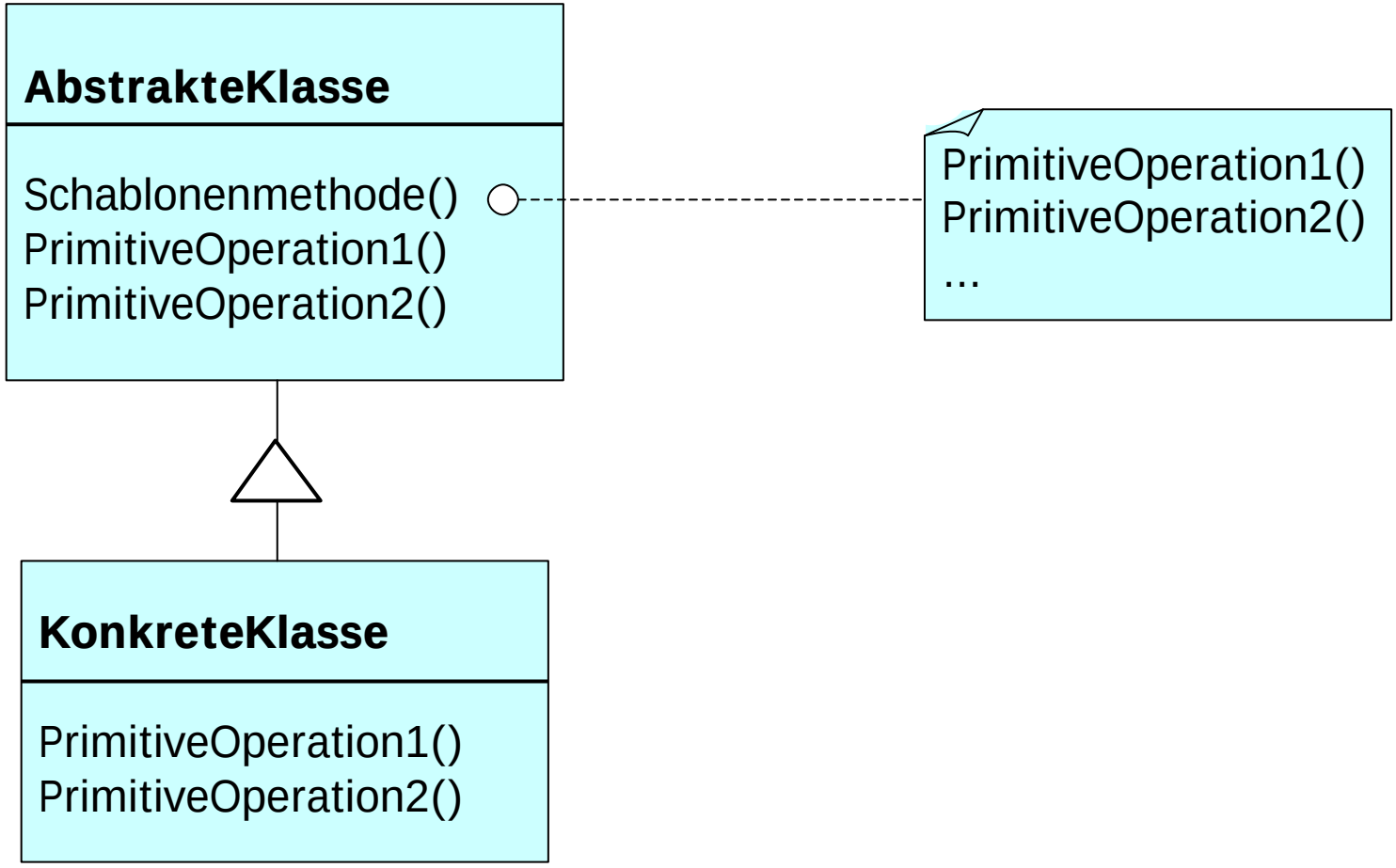
- Eine Schablonenmethode verwendet abstrakte Operationen, um einen Algorithmus zu definieren.
- Unterklassen überschreiben diese, um ein konkretes Verhalten zu ermöglichen.

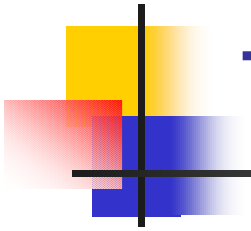


Anwendbarkeit

- invariante Teile eines Algorithmus einmal festlegen; Unterklassen implementieren variierendes Verhalten
- Verallgemeinerung: gemeinsames Verhalten von Unterklassen herausfaktorisieren und in allgemeiner Klasse plazieren

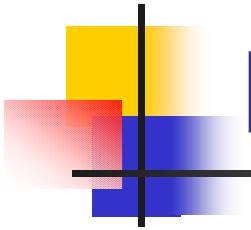
Struktur


AbstrakteKlasse
Schablonenmethode()
PrimitiveOperation1()
PrimitiveOperation2()
PrimitiveOperation1()
PrimitiveOperation2()
...
KonkreteKlasse
PrimitiveOperation1()
PrimitiveOperation2()



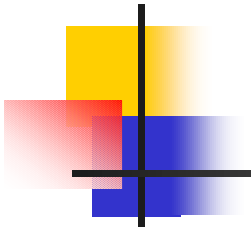
Teilnehmer

- AbstrakteKlasse
  - definiert abstrakte primitive Operationen
    - unterklassenspezifische Schritte des Algorithmus
  - implementiert eine Schablonenmethode
    - definiert Skelett eines Algorithmus
    - nutzt die primitiven Operationen
- KonkreteKlasse
  - implementiert die primitiven Operationen



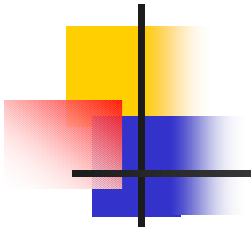
Konsequenzen

- invertierter Kontrollfluß
  “Hollywood-Prinzip“
- Einschubmethoden möglich
  - machen per Default nichts
  - sie **dürfen** bei Bedarf von Unterklassen überschrieben werden, die abstrakten primitiven Methoden dagegen **müssen** überschrieben werden



Implementierungsaspekte

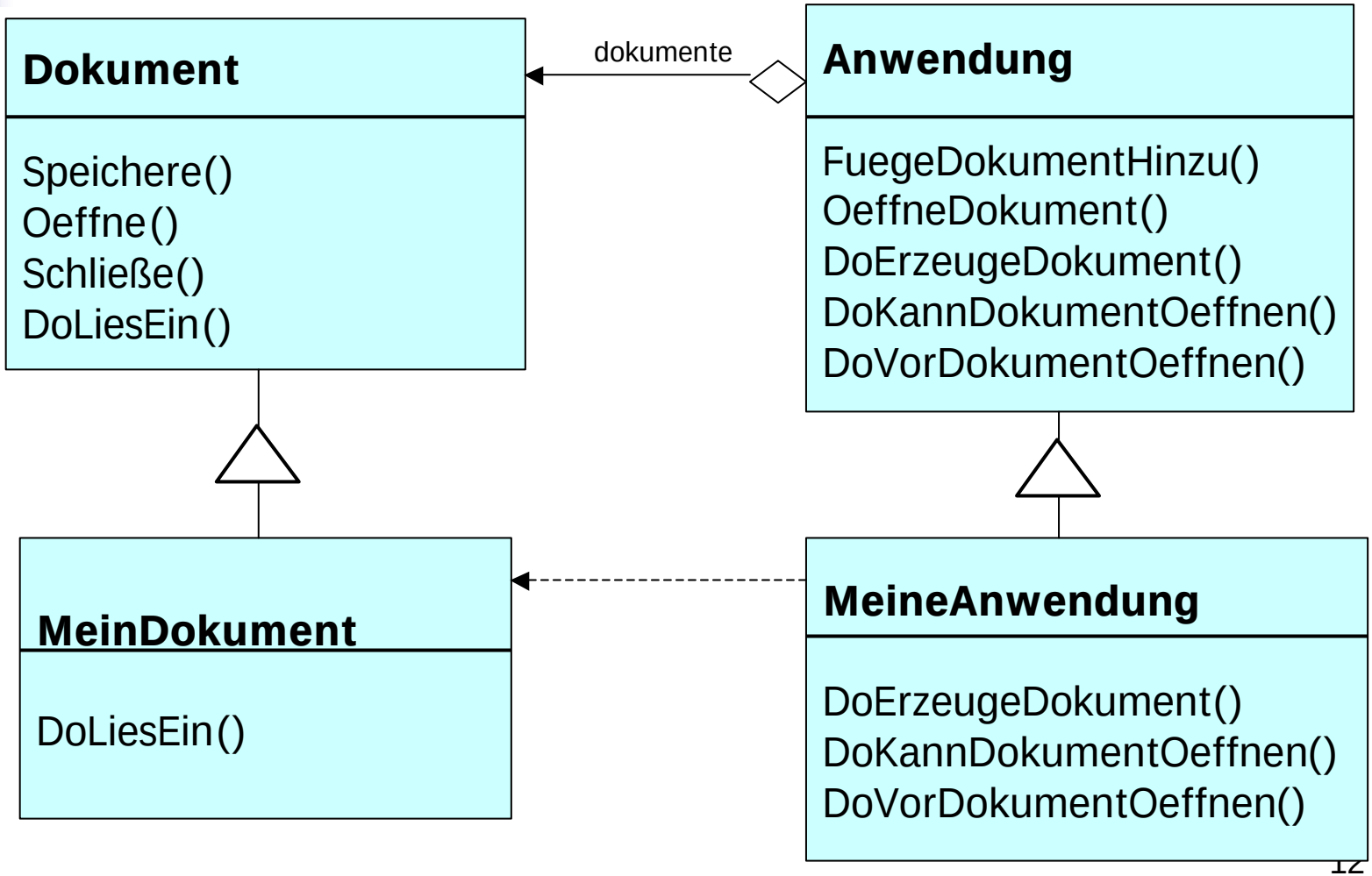
- Verwendung der C++ Zugriffskontrolle
  - Primitive Operationen als geschützt deklarieren, damit sie nur von Schablonenmethode aufgerufen werden können.
  - Primitive Operationen als virtuell deklarieren.
  - Schablonenmethode nicht überschreiben, daher nicht virtuell.

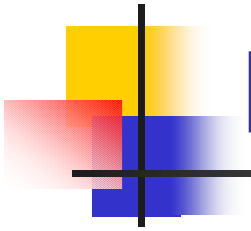


Implementierungsaspekte

- Minimierung primitiver Operationen
  - je mehr überschrieben werden muß, desto aufwendiger ist es
- Namenskonventionen
  - zu überschreibende Operationen durch Präfix kennzeichnen

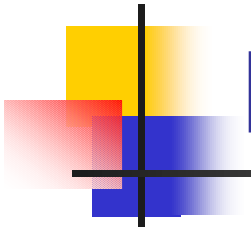
Beispiel


Dokument
Speichere()
Oeffne()
Schließe()
DoLiesEin()
dokumente
Anwendung
FuegeDokumentHinzu()
OeffneDokument()
DoErzeugeDokument()
DoKannDokumentOeffnen()
DoVorDokumentOeffnen()
MeinDokument
DoLiesEin()
MeineAnwendung
DoErzeugeDokument()
DoKannDokumentOeffnen()
DoVorDokumentOeffnen()



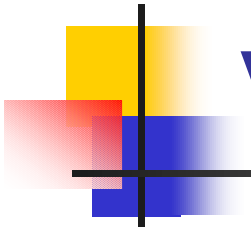
Beispiel

```
void Anwendung::OeffneDokument (const char* name) {
    if (!DoKannDokumentOeffnen(name)) {
        // kann dieses Dokument nicht bearbeiten
        return;
    }

    Dokument* dok = DoErzeugeDokument();

    if (dok) {
        FuegeDokumentHinzu(dok);
        DoVorDokumentOeffnen(dok)
        dok->Oeffne();
        dok->DoLiesEin()
    }
}
```



Bekannte Verwendungen

- Schablonenmethoden werden in fast jeder abstrakten Klasse verwendet



Verwandte Muster

- Fabrikmethoden werden oft von Schablonenmethoden aufgerufen
- Strategien
  - verwenden Delegation, um den gesamten Algorithmus zu ändern

Vielen Dank für Ihre Aufmerksamkeit

Fragen???