Seminar Software Design Pattern

Factory Method, Abstract Factory, Prototype

Ralf Rublack

Betriebliche Informationssysteme Institut für Informatik
Universität Leipzig

13.05.2009

Gliederung

Gang of Four[Gamma et al., 1995]

- Lösung-Schablonen für Entwicklungsprobleme
- Design Patterns - Elements of Reusable Object-Oriented Software
- James Coplien, Richard Helm, Ralph Johnson, John Vlissides

Patternklassen der GoF

- Erzeugende Muster (Creational Patterns)
- Strukturelle Muster (Structural Patterns)
- Verhaltensmuster (Behavioral Patterns)

weitere Patternklassen

- Architekturmuster
- Analysemuster
- Idiome
- Kommunikationsmuster
- Organisationsmuster
- Antimuster

nur genannt

- Singelton nur einmal erzeugtes Objekt
- Fewton Anzahl der erzeugten Objekte kontrolliert
- Builder Erzeugung komplexer Objekte

im weiteren Vortrag

- Factory Methode
- Abstract Factory
- Prototype

Beispiel

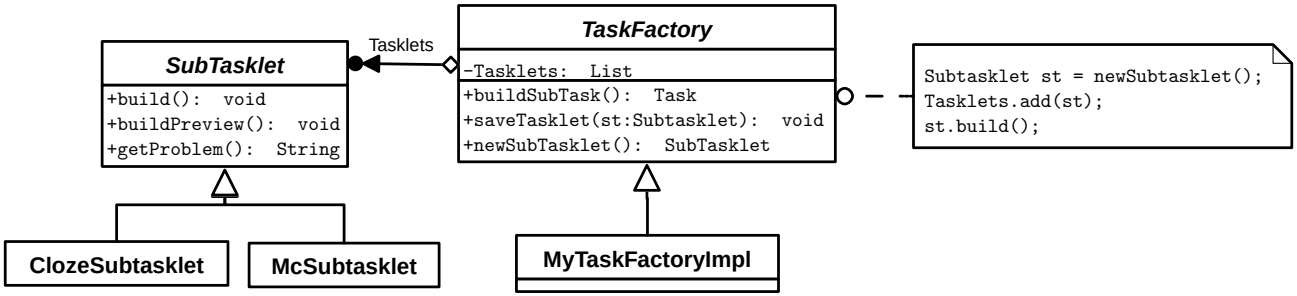
- Entwicklung eines Framework für E-Assessment
- Einsatz mit unterschiedlichen JAVA Frameworks
- unterschiedliche Aufgabentypen
- unterschiedliche Repräsentationen
- unterschiedliche Persistenzschichten

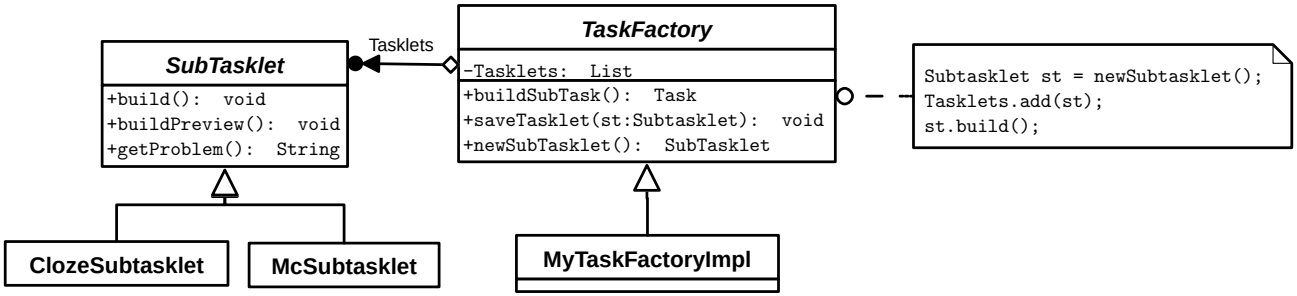
Factory Method

Factory Method

Kontext

- Implementierung von Frameworks/Klassenbibliotheken
- Interfaces zur Erzeugung von Objekten
- Definieren Beziehungen zwischen Objekten durch abstrakte Klassen


SubTasklet
+build(): void
+buildPreview(): void
+getProblem(): String
Tasklets
TaskFactory
-Tasklets: List
+buildSubTask(): Task
+saveTasklet(st:Subtasklet): void
+newSubTasklet(): SubTasklet
Subtasklet st = newSubtasklet();
Tasklets.add(st);
st.build();
ClozeSubtasklet
McSubtasklet
MyTaskFactoryImpl



Problem

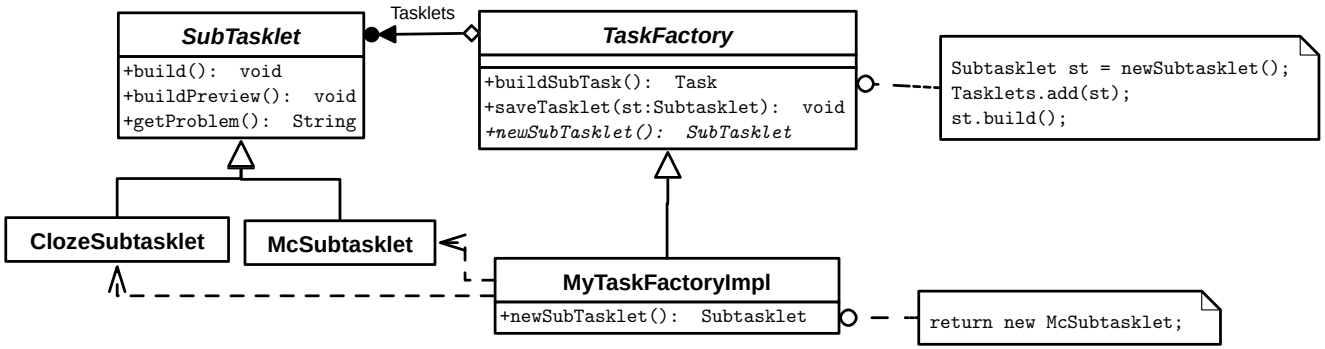
- Instanzerzeugung Implementierungsäbhängig
- TaskFactory kennt die Subklassen von SubTasklet nicht
- TaskFactory weiß nur wann nicht welches SubTasklet
- Framework kennt nur die abstrakte Klasse
- nicht instanzierbar

Lösungsansatz

- newSubtasklet abstrakte Methode
- mögliche Defaultimplementierung
- Implementierung in Anwendung

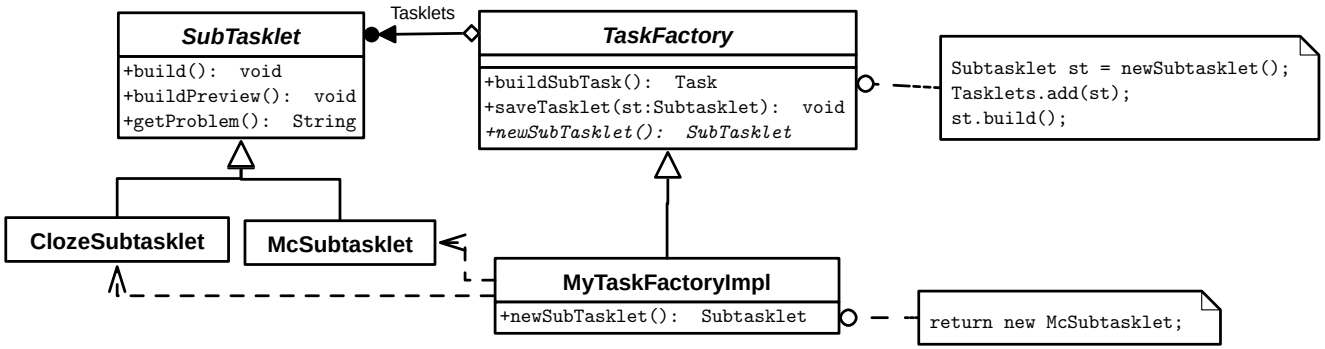
Lösungsansatz

- newSubtasklet abstrakte Methode
- mögliche Defaultimplementierung
- Implementierung in Anwendung



Charakteristik

- Implementation von newsubTasklet() in MyTaskFactoryImpl
- Kenntnis der Subklasse aus Framework entfernt
- newSubTasklet() wird Factory Methode genannt
- sie erstellt das Objekt



Elemente des Patterns

- Produkt (Interface)
- KonkretesProdukt (Implementation zu Produkt)
- Erzeuger (Applikationsklasse)
- KonkreterErzeuger (spezielle Applikationsklasse)

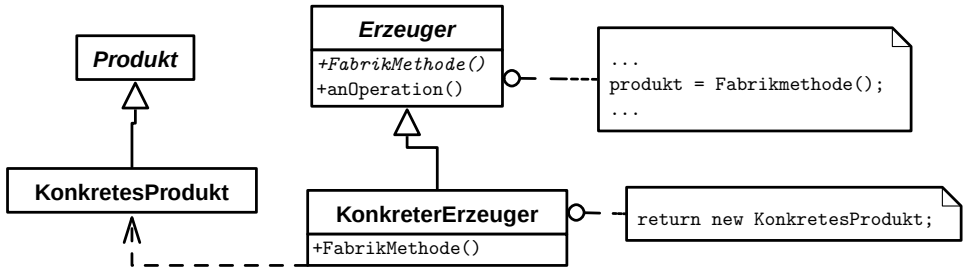


Abbildung: GoF Book S. 122

Anwendung und Konsequenz

- Erzeugerklasse kennt Objekttyp
- Subklassen sollen Typ bestimmen
- Client muss Subklasse von Creator sein

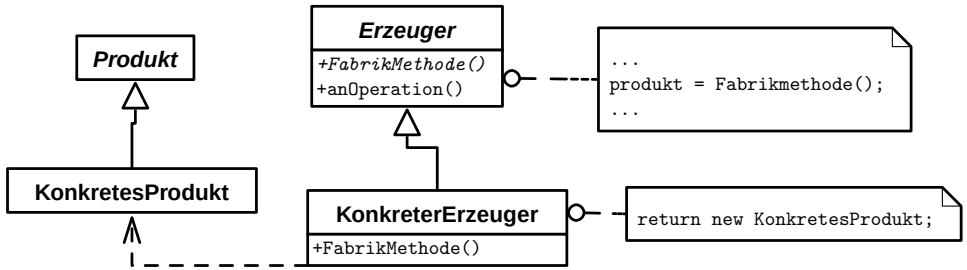


Abbildung: GoF Book S. 122

Vorteile

- keine Bindung
- indirekte Objekterzeugung
- Defaultimplementierung
- parallele Objektstrukturen verknüpft

Vorteile

- keine Bindung
- indirekte Objekterzeugung
- Defaultimplementierung
- parallele Objektstrukturen verknüpft

Nachteile

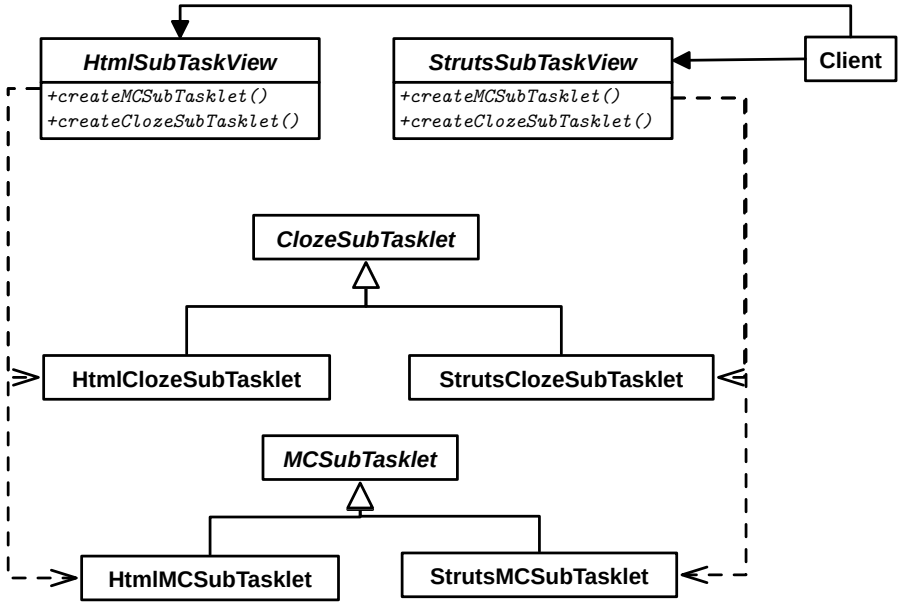
- Client Subklasse von Creator

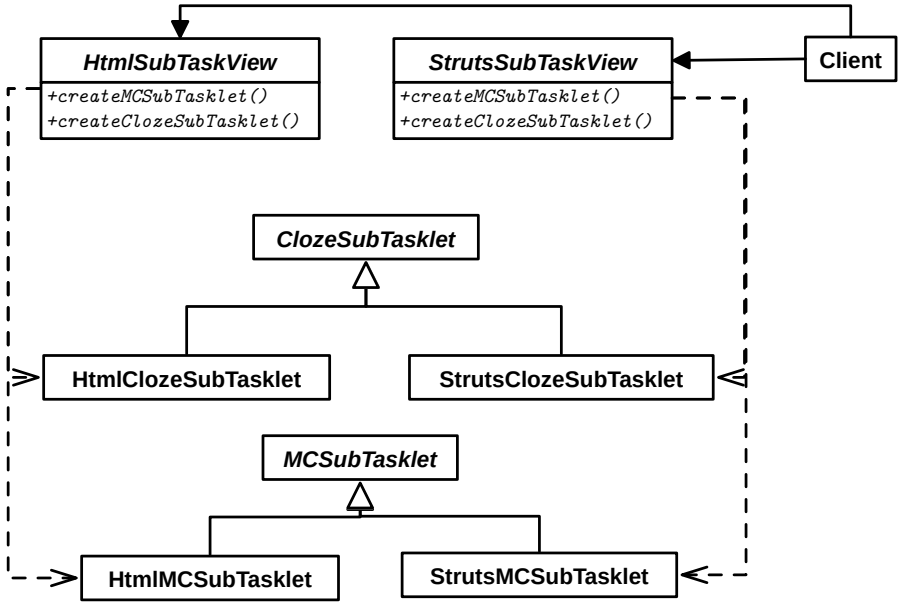
Abstract Method

Abstract Method

Kontext

- Framework stellt Struktur
- Framework soll implementierungsunabhängig sein
- Implementierung konfigurierbar austauschbar


HtmlSubTaskView
+createMCSubTasklet()
+createClozeSubTasklet()
StrutsSubTaskView
+createMCSubTasklet()
+createClozeSubTasklet()
Client
ClozeSubTasklet
HtmlClozeSubTasklet
StrutsClozeSubTasklet
MCSubTasklet
HtmlMCSubTasklet
StrutsMCSubTasklet



Problem

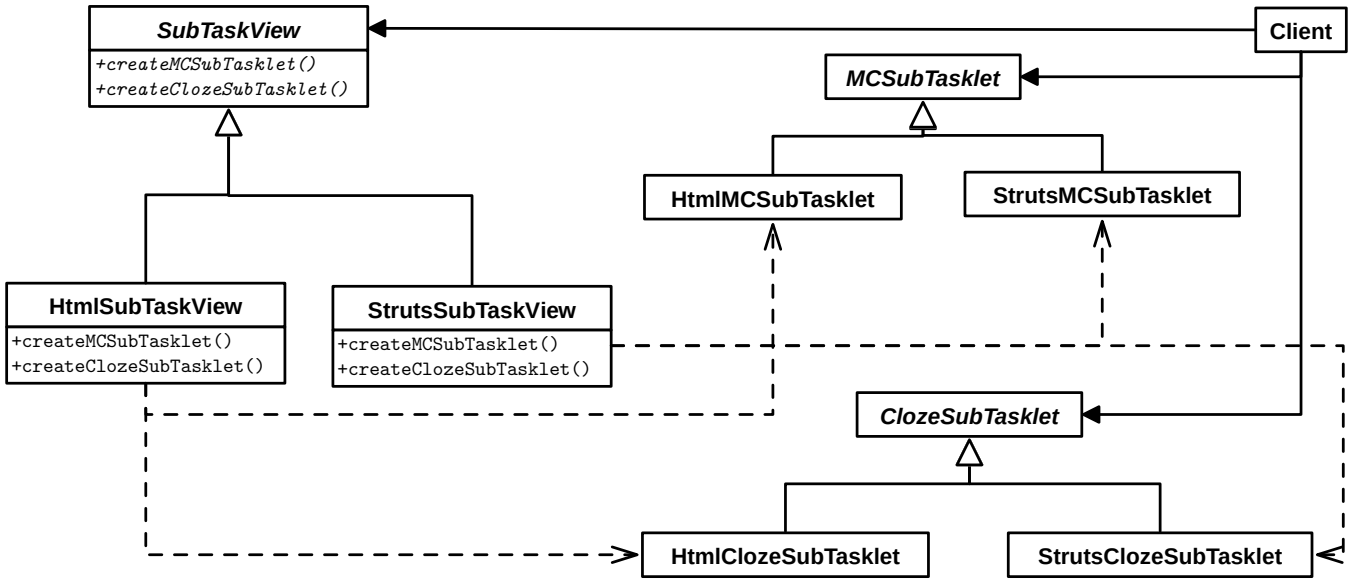
- ein Framework für Erstellung von Tests
- unterschiedliche Präsentationsframeworks
- Client muss die Implementierung kennen
- Präsentationsschichten sollten austauschbar sein

Lösungsansatz

- Client benutzt nur Interface
- Interface für SubTaskView mit Methoden für Objekterzeugung
- Implementierungen von SubTaskView
- Interfaces für Objekte

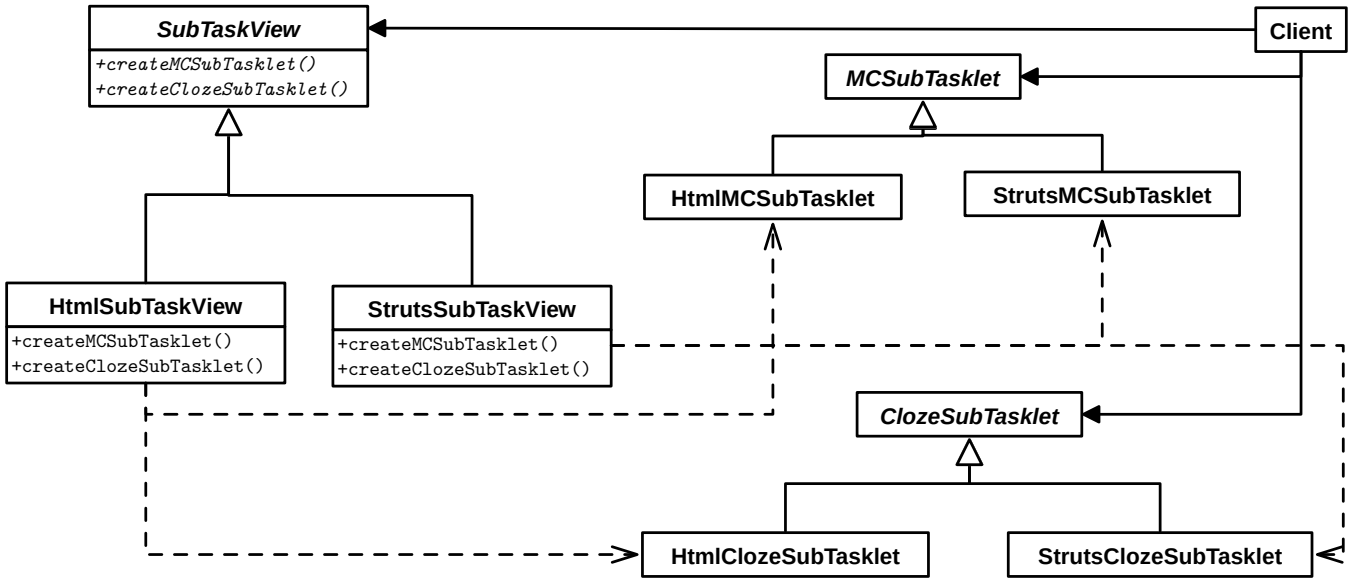
Lösungsansatz

- Client benutzt nur Interface
- Interface für SubTaskView mit Methoden für Objekterzeugung
- Implementierungen von SubTaskView
- Interfaces für Objekte



Charakteristik

- Client benutzt nur Interfaces
- Implementierungen erzeugen Objekte



Elemente des Patterns

- AbstrakteFactory
- KonkreteFactory
- AbstraktesProdukt
- KonkretesProdukt
- Client

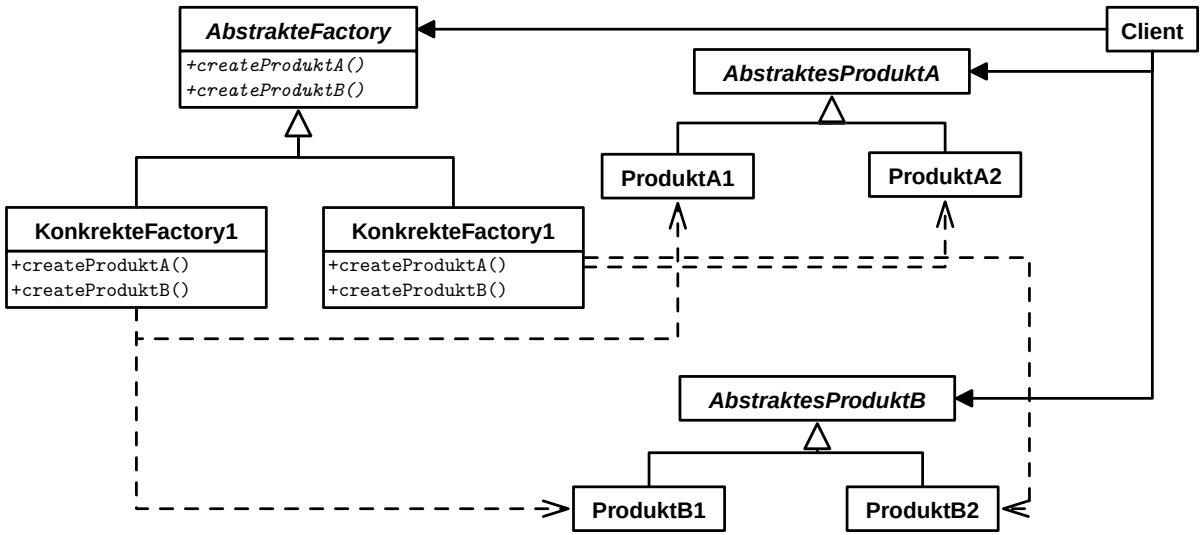


Abbildung: GoF Book S. 101

Anwendung und Konsequenz

- KonkreteFactory meist Singelton
- ermöglicht Produktfamilien
- leichter Austausch von Implementierungen
- Versteckt Implementierung vor Client

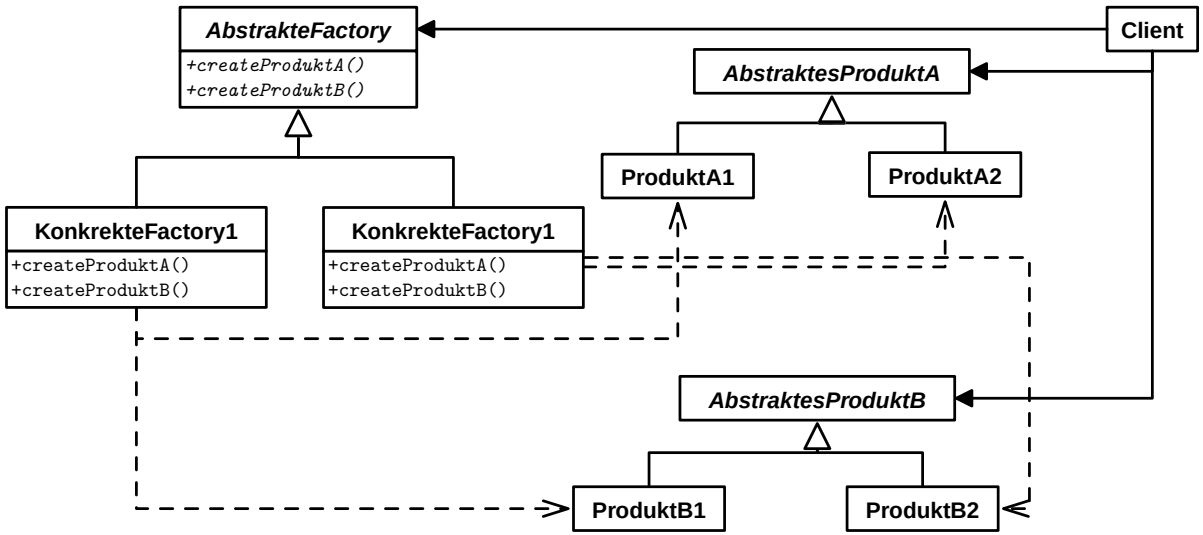


Abbildung: GoF Book S. 101

Vorteile

- isoliert Implementierung
- Kontrolle der Objektgenerierung
- einfacher Austausch
- Konsistenz

Vorteile

- isoliert Implementierung
- Kontrolle der Objektgenerierung
- einfacher Austausch
- Konsistenz

Nachteile

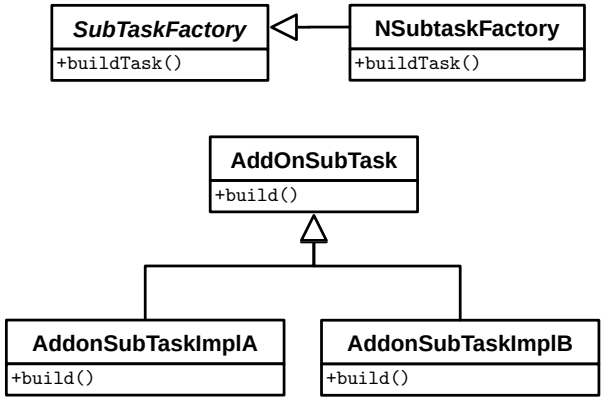
- neue Klassen kosten viel Aufwand

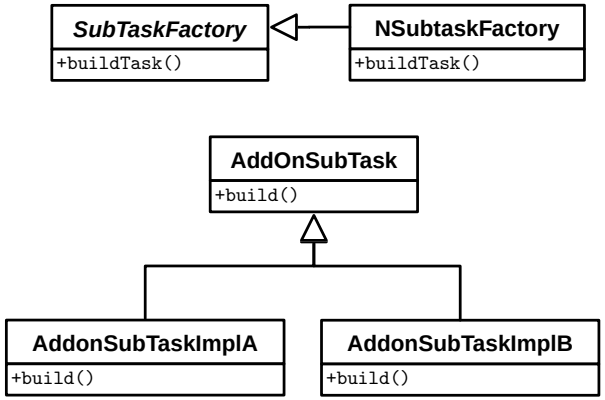
Prototype

Prototype

Kontext

- Objekte im laufenden System spezifizieren
- Funktionalität nutzen
- spezifische Erweiterung[Pree, 1995]
- geringe Objektdifferenz
- keine komplexe Objekthierachie


SubTaskFactory
+buildTask()
NSubtaskFactory
+buildTask()
AddOnSubTask
+build()
AddonSubTaskImplA
+build()
AddonSubTaskImplB
+build()



Problem

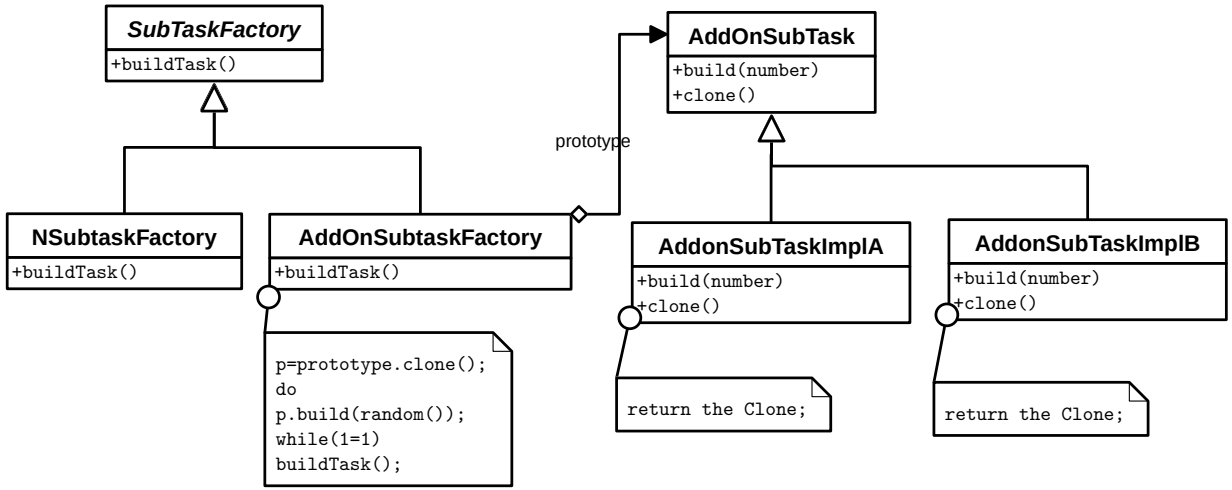
- Erweiterung um neue Objekte
- dynamisches Laden von Objekten
- Objektgenerierung parametrisieren
- Integration in bestehende Struktur

Lösungsansatz

- Einführung Factorysubklasse
- AddOnsubtaskFactory benutzt Prototyp
- Objektgenerierung mit Parameter
- AddOnSubTask Klonmethode
- Implementierung in Subklassen

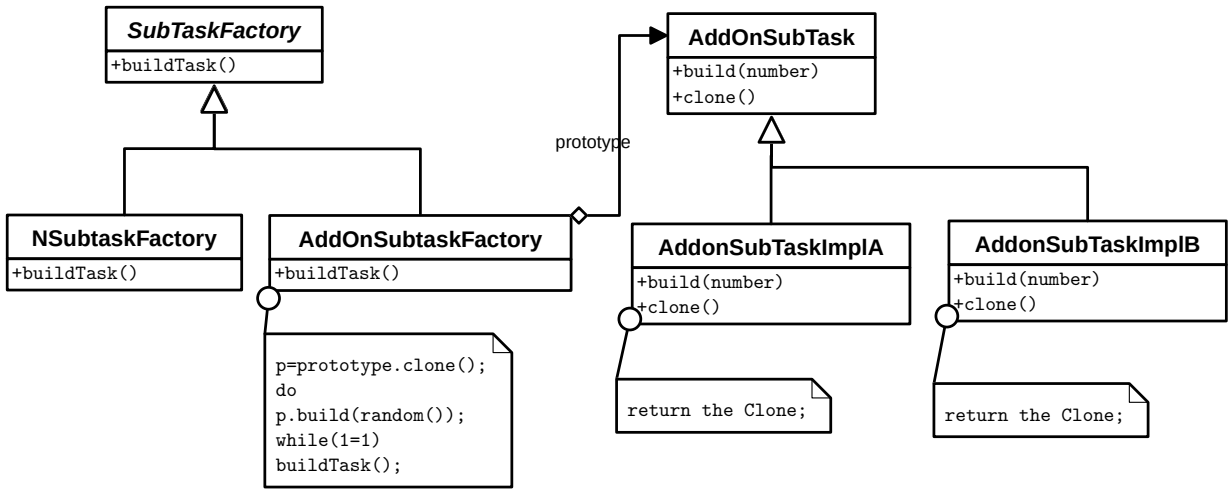
Lösungsansatz

- Einführung Factorysubklasse
- AddOnsubtaskFactory benutzt Prototyp
- Objektgenerierung mit Parameter
- AddOnSubTask Klonmethode
- Implementierung in Subklassen



Charakteristik

- Prototypklassen haben Clone()
- meist mit Parameter



Elemente des Patterns

- Prototype (Klone Interface)
- ConcretePrototype (Klone Implementierung)
- Client

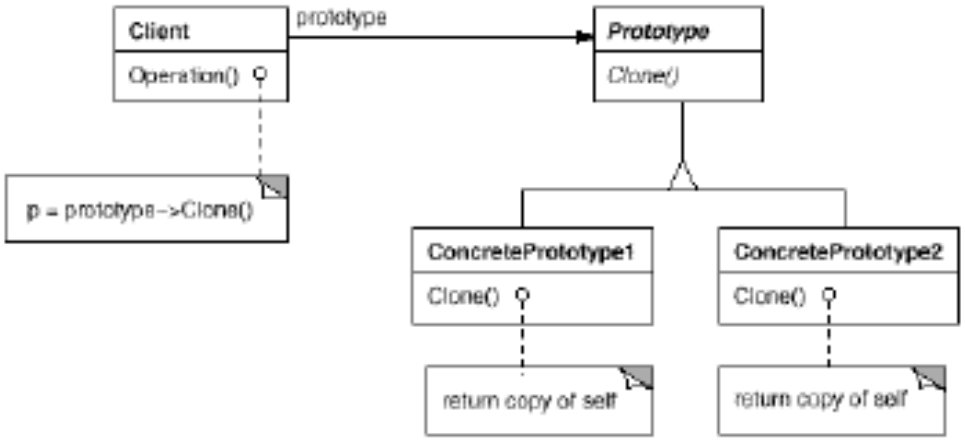


Abbildung: GoF Book S. 135

Anwendung und Konsequenz

- Verringerung der Klassenanzahl
- Versteckt Implementierung vor Client

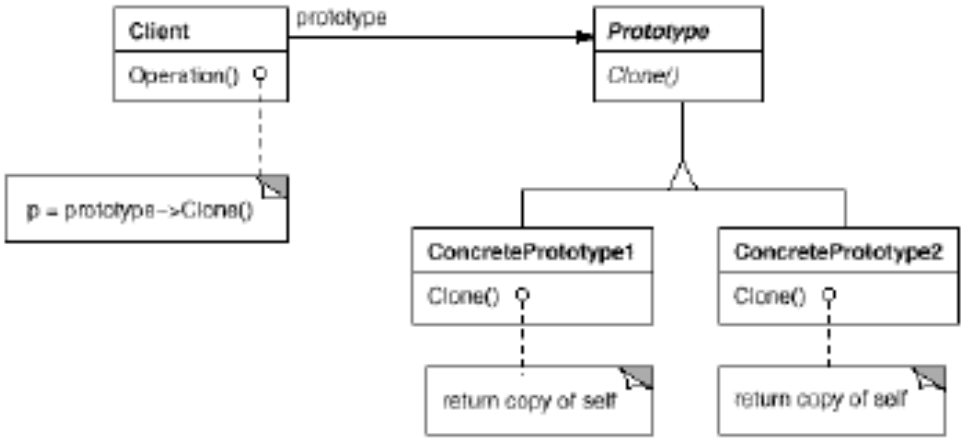


Abbildung: GoF Book S. 135

Vorteile

- isoliert Implementierung
- weniger Objektklassen
- dynamische Binden
- neue Objekte einfach
- Parameter für Objekterstellung
- komplexe Objekte
- weniger Subklassen

Vorteile

- isoliert Implementierung
- weniger Objektklassen
- dynamische Binden
- neue Objekte einfach
- Parameter für Objekterstellung
- komplexe Objekte
- weniger Subklassen

Nachteile

- Klone Methode
- aufwendig zu implementieren

Literatur


Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995).
Design Patterns: Elements od Reusable Object-Oriented Software.
Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, New York, NY.


Pree, W. (1995).
Design Patterns for Object-Oriented Software Development.
ACM Press Books. Addison-Wesley Publishing Company, Wokingham.

vielen Dank für die Aufmerksamkeit

Fragen ??