Tagesprogramm

Aspekte und Annotationen

Software-Entwurfsmuster

Factory-Method

Prototype

Aspektorientierte Programmierung

Paradigma der Modularisierung — **Separation-of-Concerns**

AspectJ

Core-Concerns (Kernfunktionalitäten) in Klassen

Cross-Cutting-Concerns (Querschnittsfunktionalitäten) in Aspekten

Aspekt beschreibt, wie Programmcode zu ändern ist
(z.B. Zusatzcode vor oder nach Aufruf bestimmter Methoden ausführen)

Aspect-Weaver wendet Aspekte auf Klassen an

Annotationen

Annotation ist optionaler Parameter, der

- an (fast) beliebige Sprachkonzepte anheftbar ist,
- im Java-Code statisch gesetzt wird,
- von gesamter Werkzeugkette bis zur Laufzeit auslesbar ist.

```
@Override
public String toString() { ... }

@BugFix(who="Kaspar", date="2014-12-03", level=3)
public class Buggy { }
```

Definition von Annotationen

Syntax von Interfaces adaptiert

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
public @interface BugFix {
    String who() default "me"; // author of bug fix
    String date();             // when was bug fixed
    int level() default 1;     // importance level 1-5
}
```

Annotationen zur Laufzeit

Falls `@Retention(RUNTIME)` wird echtes Interface erzeugt:

```
public interface BugFix
       extends java.lang.annotation.Annotation {
    String who();
    String date();
    int level();
}
```

Zugriff durch Reflection (über `Class`)

Zweck von Entwurfsmustern

Benennen wiederkehrender Probleme und Lösungen

Austausch von Erfahrungen

Wiederverwendung von Erfahrung wo Wiederverwendung von Code versagt (sehr abstrakt, daher häufig wiederverwendbar)

Factory-Method (Virtual-Constructor)

Zweck: Definition einer Schnittstelle für Objekterzeugung

Anwendungsgebiete:

Klasse neuer Objekte bei Objekterzeugung unbekannt

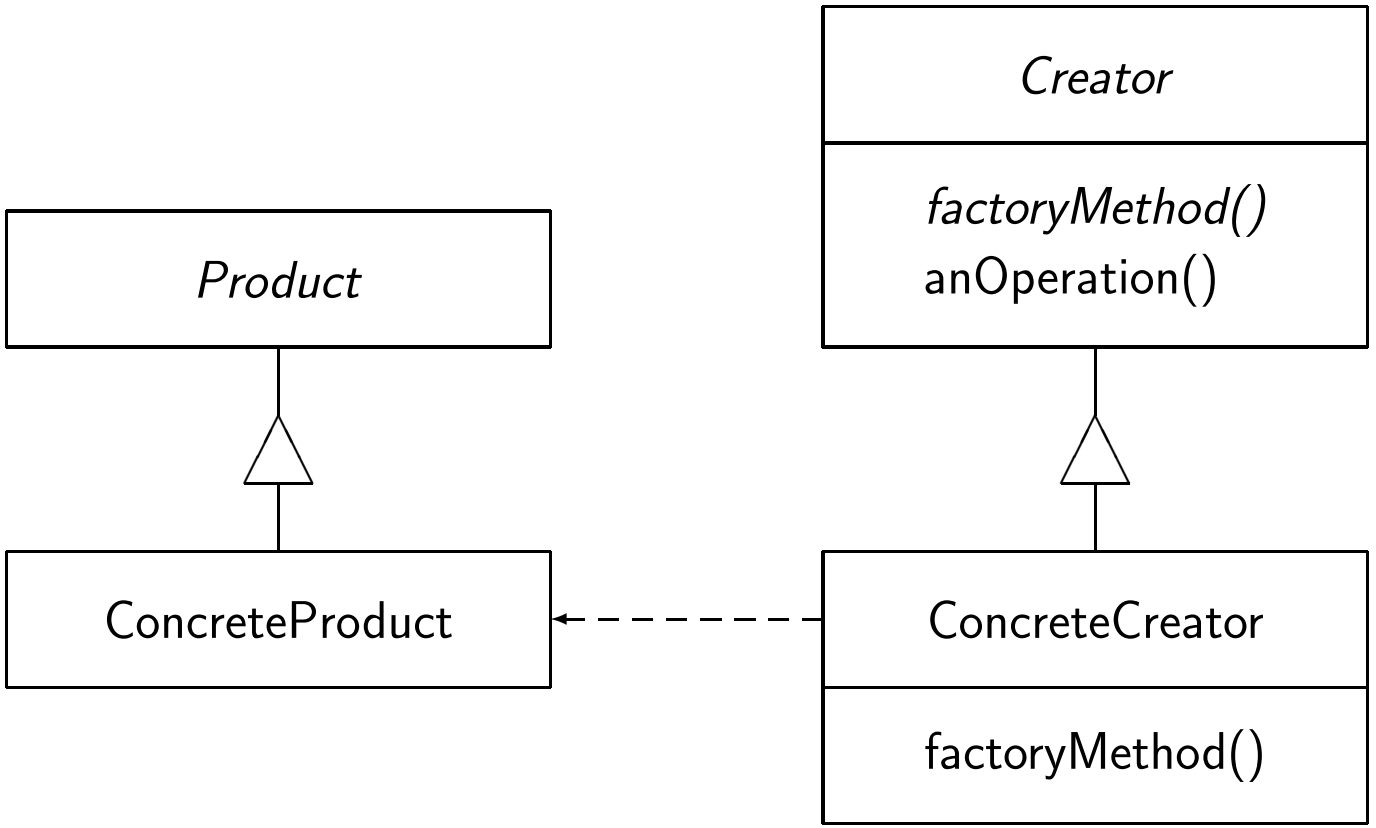
Unterklassen sollen Klasse neuer Objekte bestimmen

Klassen delegieren Verantwortlichkeiten an Unterklassen
(Wissen um Unterklasse soll lokal bleiben)

Factory-Method: Beispiel 1

```
abstract class Document { ... }
class Text extends Document { ... }
...   // classes Picture, Video, ...
abstract class DocCreator {
    abstract Document create();
}
class TextCreator extends DocCreator {
    Document create() { return new Text(); }
}
...   // classes PictureCreator, VideoCreator, ...
class NewDocManager {
    private DocCreator c = ...;
    public void set(DocCreator c) { this.c = c; }
    public Document newDoc() { return c.create(); }
    }
```

Factory-Method: Struktur



Factory-Method: Eigenschaften

Anknüpfungspunkte (Hooks) für Unterklassen

→ flexibel und Entwicklung von Unterklassen vereinfacht

verknüpfen parallele Klassenhierarchien (Creator- und Product-Hierarchie)

Beispiel:
`generiereFutter` vom Typ `Futter` in `Tier` (abstrakt)
erzeugt in `Rind` neue Instanz von `Gras`
und in `Tiger` neue Instanz von `Fleisch`

oft große Anzahl an Unterklassen nötig

Prototype

Zweck: Prototyp-Objekt spezifiziert Art eines neuen Objekts,
Objekterzeugung durch Kopieren des Prototyps

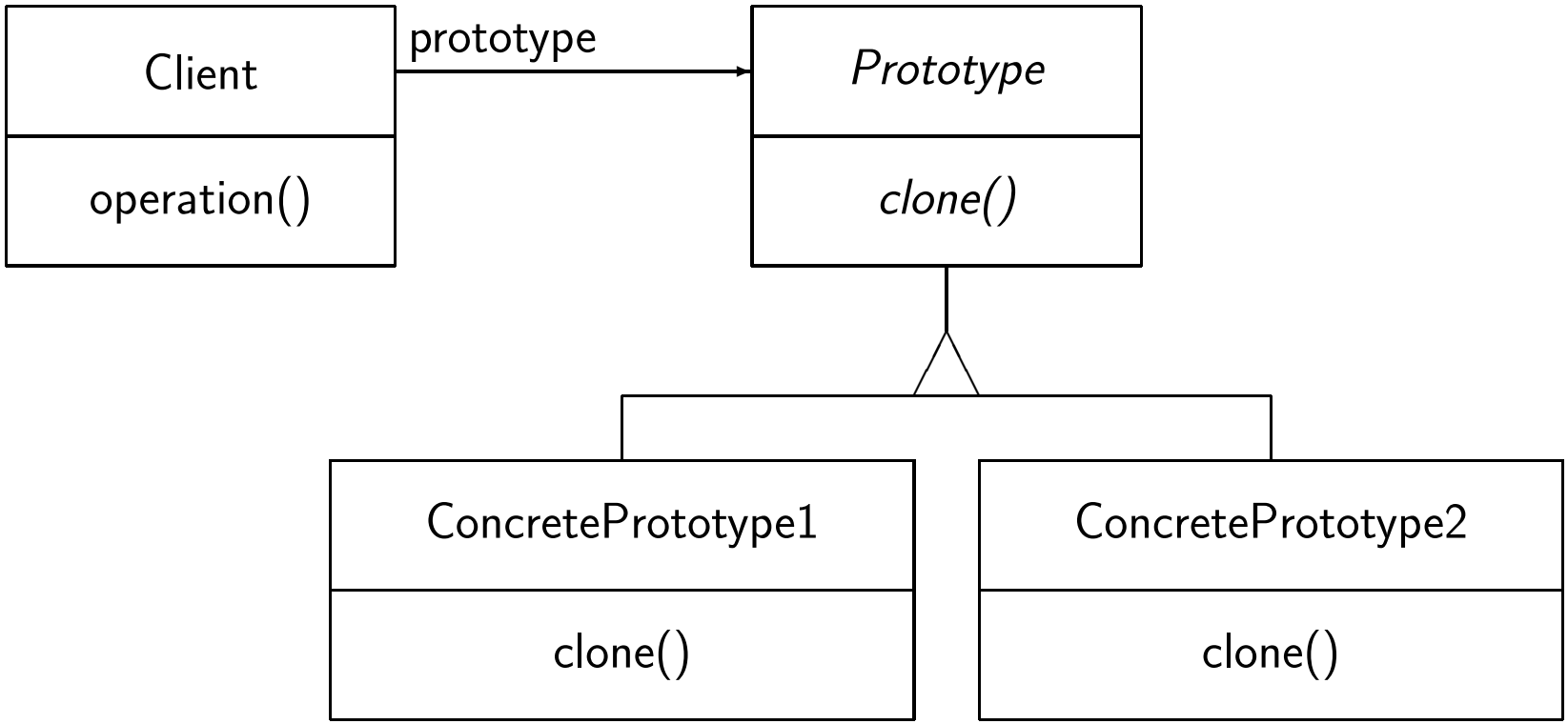
Anwendungsgebiete:

Klasse des neuen Objekts erst zur Laufzeit bekannt

Vermeidung von Creator- parallel zu Product-Hierarchie (Factory-Method)

jede Instanz hat einen von wenigen Zuständen
→ Kopieren einfacher als Konstruktoraufruf, übernimmt Objektzustand

Prototype: Struktur



Prototype: Eigenschaften

versteckt Product-Klassen (aus Factory-Method) vor Anwendern,
daher beeinflussen geänderte Product-Klassen Anwender nicht

Prototyp-Menge dynamisch änderbar (Klassenstruktur nicht)

Prototypen dynamisch änderbar (Klassen nicht)

$\rightarrow$ in hochdynamischen Systemen:
Verhalten durch Objektkomposition statt Klassendefinition festlegbar

vermeidet große Anzahl an Unterklassen

erlaubt dynamische Konfiguration von Programmen auch in Sprachen wie C++

Prototype: Implementierungshinweise

`clone` in Java für **flache** Kopien in `Object` vordefiniert
(verwendbar wenn `Cloneable` implementiert)

Erzeugen **tiefer** Kopien schwierig — zyklische Strukturen

Prototyp-Manager zur Verwaltung der Prototypen

`clone` hat keine (geeigneten) Parameter, oft Initialisierungsmethoden nötig

von dynamischen Sprachen direkt unterstützt