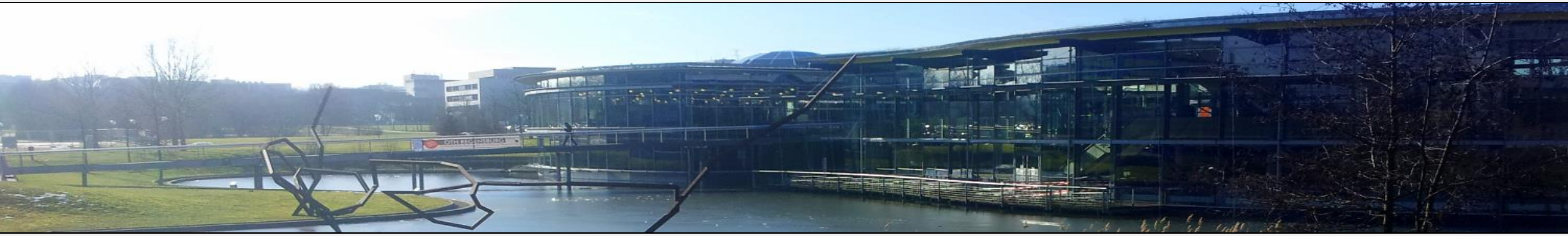


Factory Pattern

und deren Auswirkungen auf die Softwarearchitektur in der Praxis

Lernziele

1. Wozu dienen Entwurfsmuster?
2. Wie nutzt man die *Fabrikmethode*?
3. Wie nutzt man die *Abstrakte Fabrik*?
4. Welche Vor- und Nachteile bieten die vorgestellten Muster?

Was sind Entwurfsmuster?

- Lösungsschablonen für regelmäßig auftretende Entwurfsprobleme

Erzeugungsmuster

- Fabrikmethode
- abstrakte Fabrik
- Singleton
- …

Strukturmuster

Verhaltensmuster

Was versprechen wir uns von ihnen?

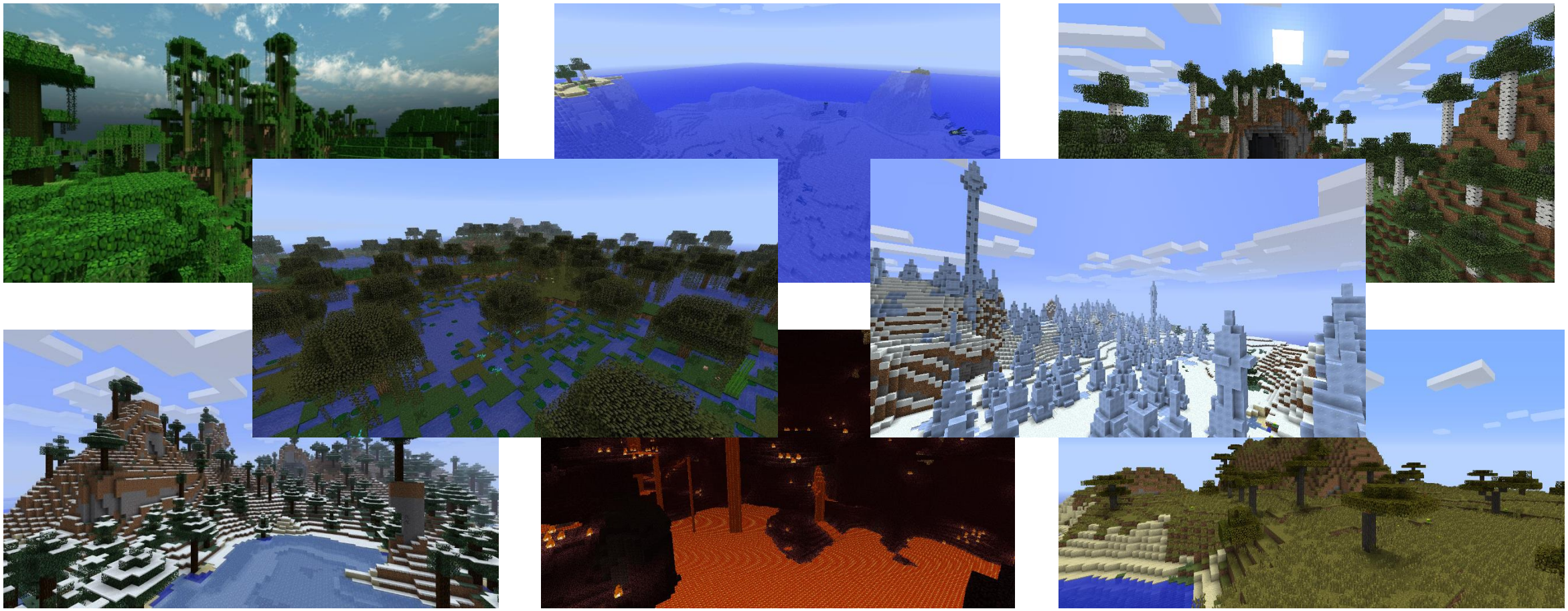
- Häufig auftretende Entwurfs-Probleme standardisiert lösen
- Entwickler-Vokabular standardisieren
- Code einfacher wartbar machen
- Kosten reduzieren

- Implementierungen austauschbar machen
- Verhalten änderbar gestalten (späte/dynamische Konfigurierbarkeit)

Beispiel Minecraft

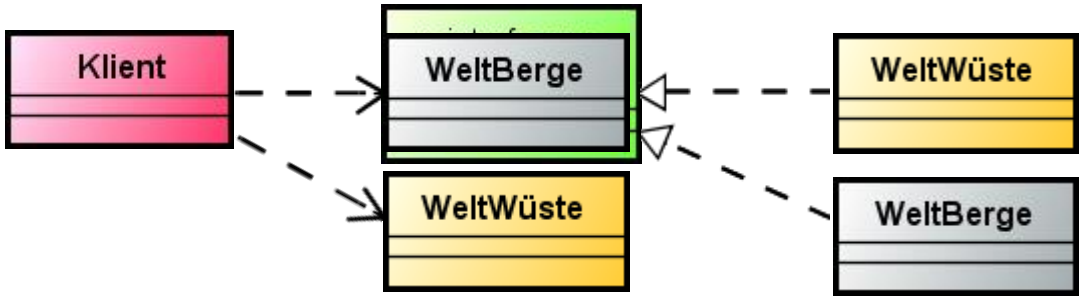
- > 50 Mio verkaufte Spiele, >12.000.000 Facebook Fans
- Für PC/Mac, iOS, Android, Xbox,…
- Game-based Learning in Schulen
- In Forschungseinrichtungen (Kooperation von Google und Caltech)

Welten in Minecraft



Wie kann man das umsetzen?

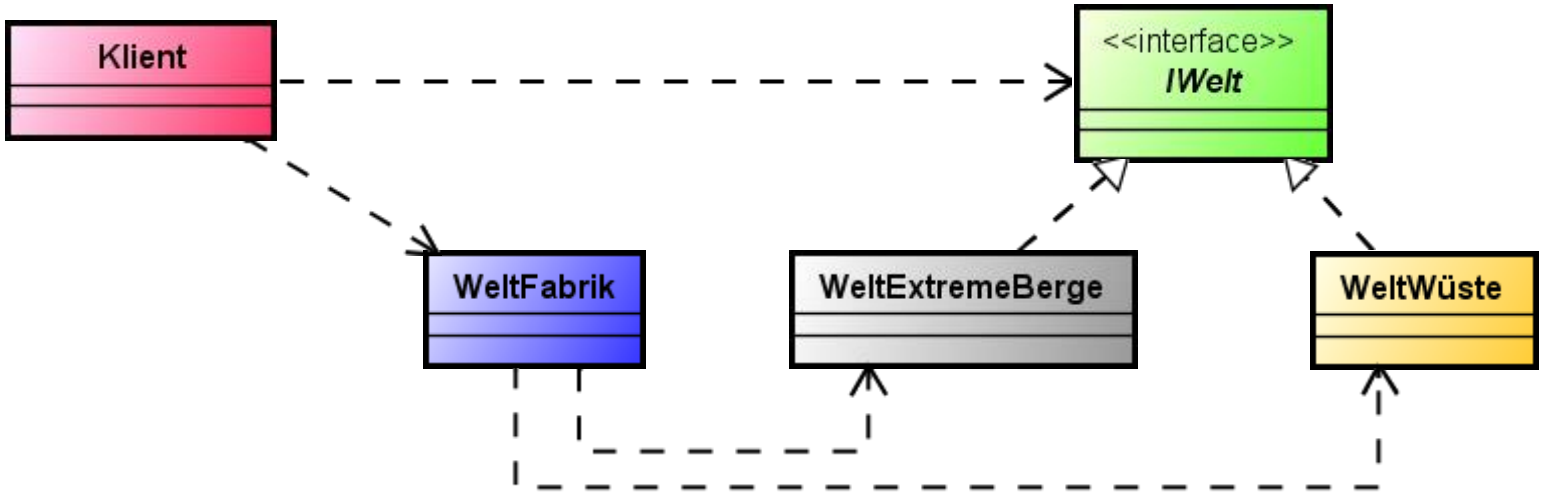
- Risiko bei Erweiterungen: fehlende Abstraktion, direkte Instanziierung und Nutzung
  ⇒ Code an vielen Stellen ändern
- Klient soll möglichst wenig über Implementierung wissen
- Typdeklarationen mit Interfaces statt Klassen (wenn sinnvoll)
  ⇒ Losere Kopplung



Aber: Was ist jetzt das Problem?

- Wie bekomme ich jetzt eine Wüstenwelt?
- Eben war das noch möglich, jetzt nicht mehr!

Fabrikmethode für Welten



Wie erzeuge ich jetzt eine neue Welt?

```
public class Klient:
…
WeltFabrik weltFabrik = new WeltFabrik();
IWelt gebirge  = weltFabrik.erzeugeWelt(Welttyp.EXTREME_BERGE);
IWelt wüste    = weltFabrik.erzeugeWelt(Welttyp.WÜSTE);

gebirge.besiedleWelt();
```

Beispiel: Flora in Minecraft

Eine Fabrik soll die Flora einer Landschaft erstellen:

- `ExtremeBerge` erzeugt Bäume, Berge
- `Wüste` erzeugt „abstrakte Bäume und Berge”

Realisierung mit Fabrikmethode

- Intention: - Objekte verschiedenen Typs auf Bestellung erzeugen
- Umsetzung: - Eigene Erstellerklasse als Fabrik mit Fabrikmethode
  - Erzeugung verschiedener Objekte über Fabrikmethode
- Klient: - Kein Aufruf von Konstruktor (new)
  - Sondern von Fabrikmethode (z.B. mit Welt-Typ)
- Ziele:
  - Entkopplung der Herstellung von der Verwendung eines Objekts
  - Leichte Austauschbarkeit der Implementierung oder des Produktionsprozesses

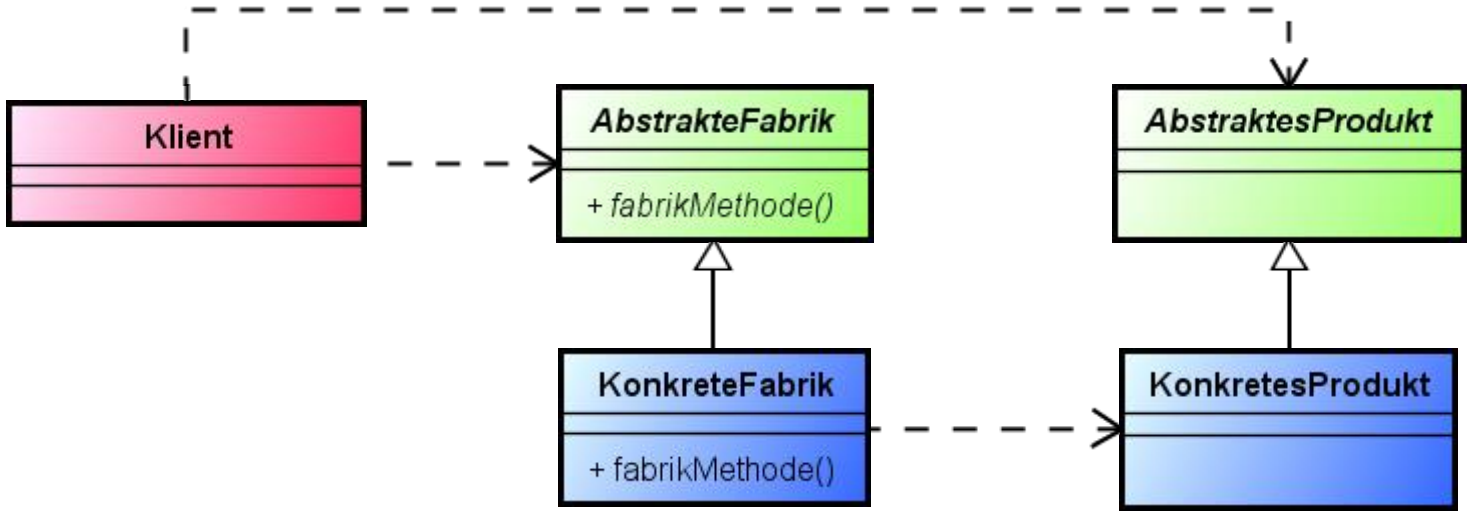
Fabrikmethode

- Klient entscheidet:
  - Aufruf einer abstrakten Fabrikmethode

wann wird ein Objekt erzeugt

- Subklasse der Fabrik entscheidet über:
  - Implementierung der abstrakten Fabrikmethode

wie wird Objekt erzeugt



Vor- und Nachteile (Fabrikmethode)

Vorteile:

+ **Lose Kopplung** zwischen Klient und Produkten
+ **Kapselung** der Objekterstellung in Fabrik
+ **Trennung** von Produkterstellung und Produktnutzung
+ **Reduzierte Komplexität** aus Klientensicht
+ **Lesbarkeit** und Wiederverwendbarkeit des Programmcodes

Nachteile:

– **Mehr Komplexität im Design** durch Subklassenbildung (eine konkrete Fabrik-Subklasse für jedes Produkt)
– **Mehraufwand** durch zusätzlichen Programmcode

Weitere Beispiele

- **Kennen Sie weitere Beispiele?**

- Zutatenfabrik für Gerichte (z.B. Pizza)
  - `erstelleTeig(), erstelleSoße(), …`

- Zugriff auf Datenbanken (SQL Server, Oracle, etc.)
  - `erstelleDBVerbindung(), erstelleDBAnweisung()`

Abstrakte Fabrik: Welten in Minecraft

Welt Extreme Berge:

- Hohe Berge
- Wenige Eichen
- Wasserfälle, etc.

Produktfamilie 1

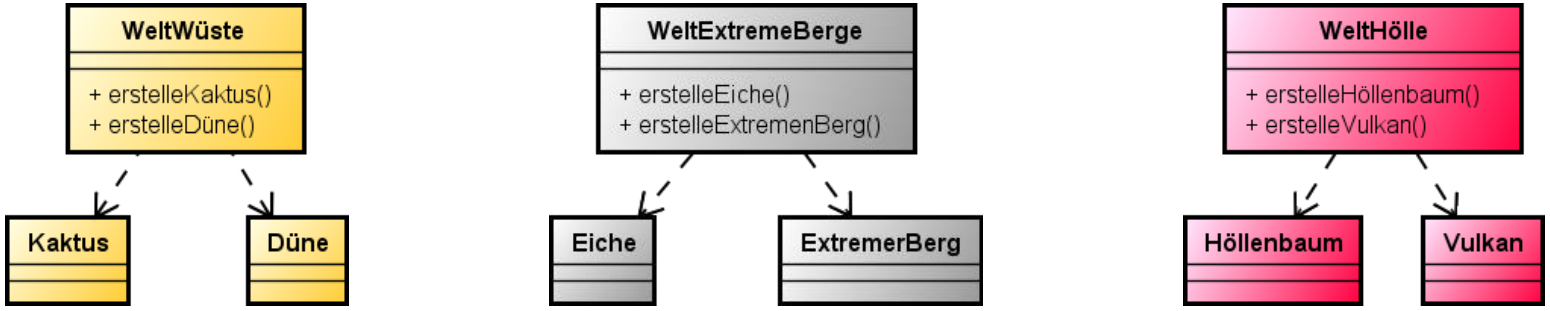
Welt Wüste:

- Flache Dünen-Landschaft
- Kakteen
- Wenig Wasser, etc.

Produktfamilie 2

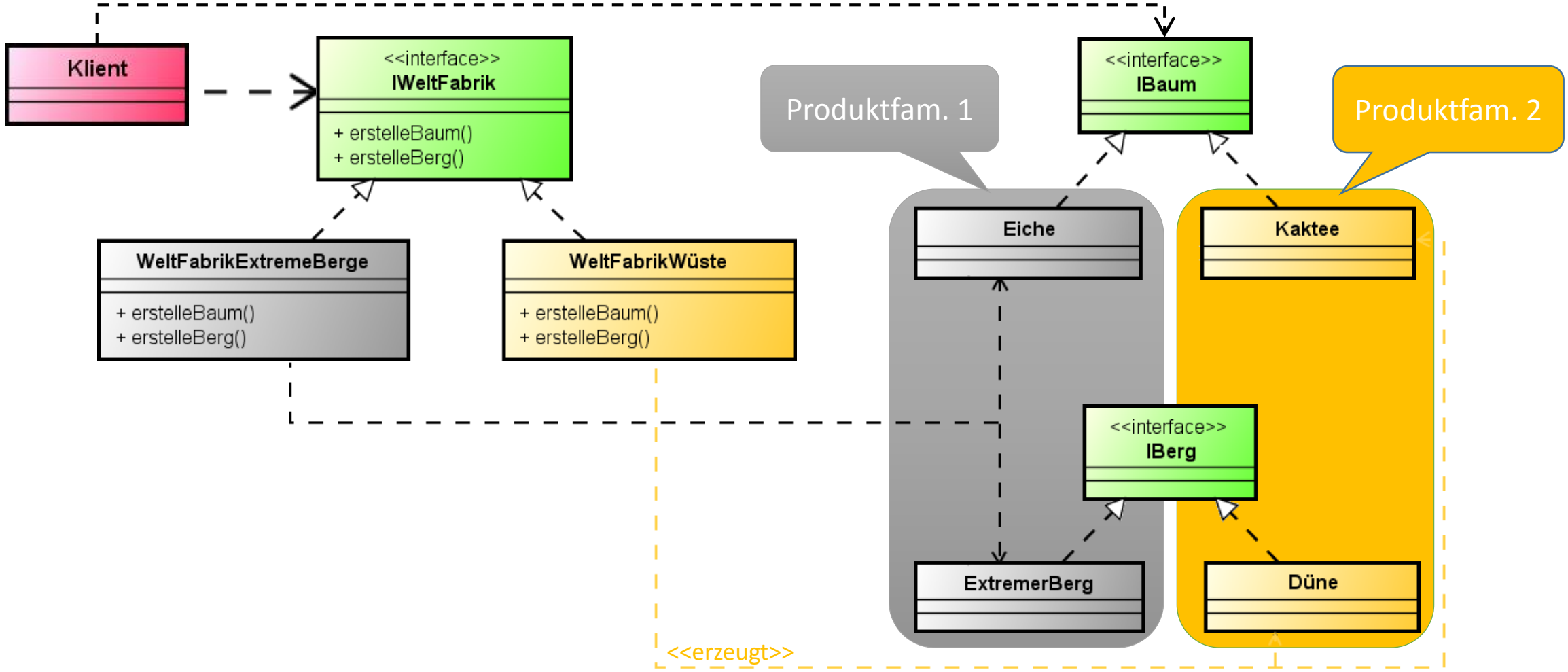
Muster: Abstrakte Fabrik

- Angenommen wir möchten eine WüstenWelt erstellen

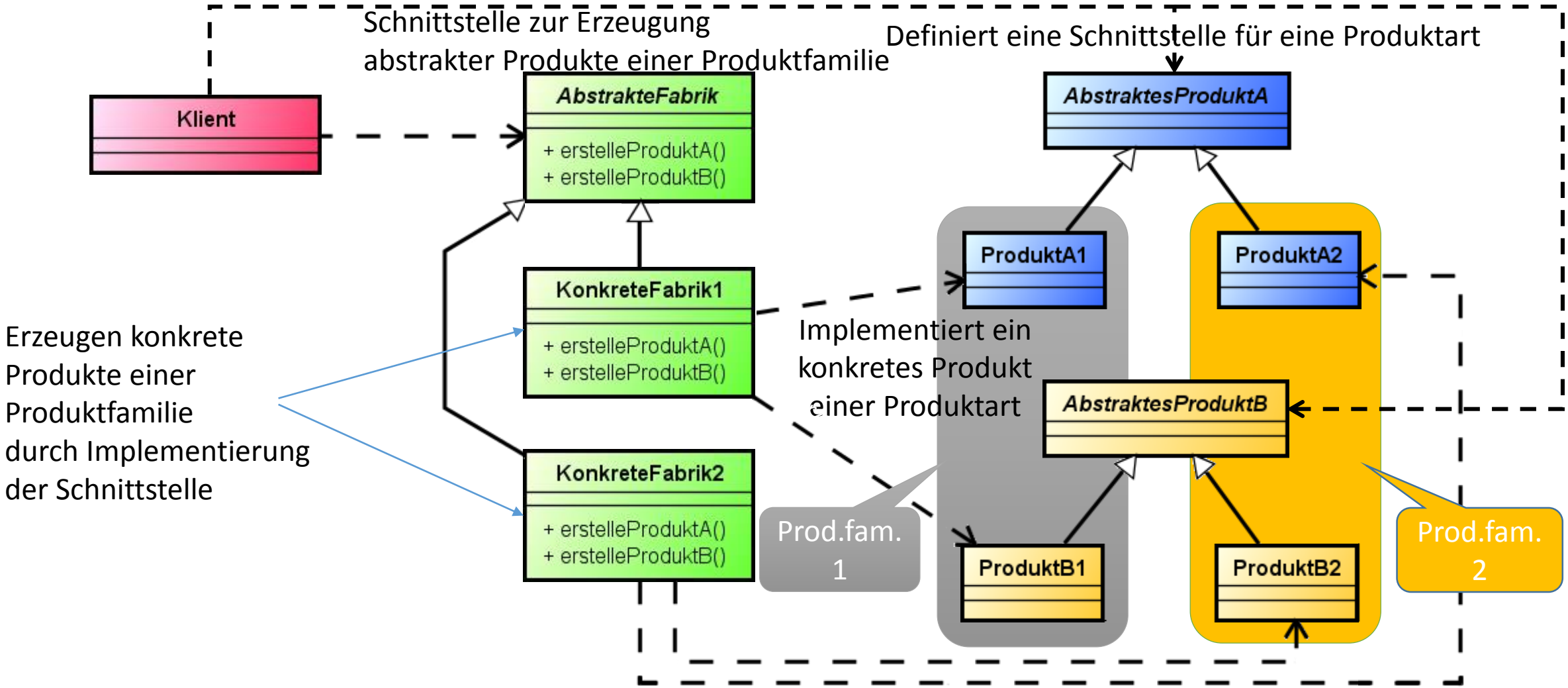


- Was passiert, wenn wir nun weitere Welten erstellen wollen?
- Problem:
  - Fehlende Abstraktion,
  - Enge Kopplung von Klient und Welten
  - Inkonsistenzen (Klient kann Elemente der Welten beliebig vermischen)
- Ziel: Entkopplung von Klient und Welten

Abstrakte Fabrik für Welten



Abstrakte Fabrik (Produktfamilien)



Vor- und Nachteile (Abstrakte Fabrik)

Vorteile:

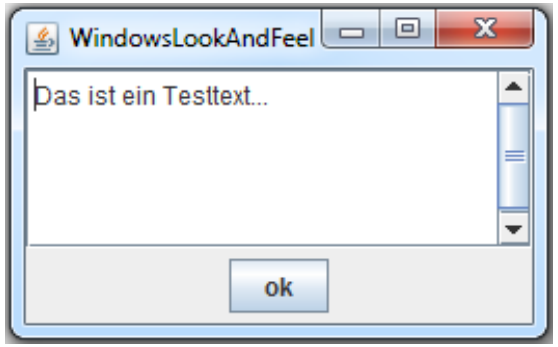
+ Kapselung der Objekterstellung in Fabrik (→ Entkopplung)
+ Klient kennt konkr. Implementierungen nicht (→ Entkopplung)
+ Wechsel zwischen Produktfamilien (→ z.B. Look and Feel)
+ Leichte Erweiterung um neue Produktfamilien (→ z.B. Dschungel)
+ Konsistente Benutzung von Produktfamilien (→ z.B. kein Kaktus im extremen Gebirge, keine Motif-Scrollbar im Windows 8 Fenster)

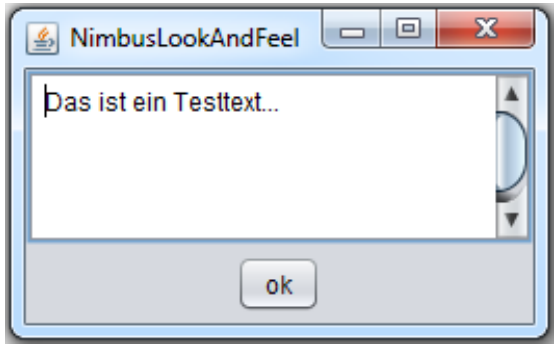
Nachteile:

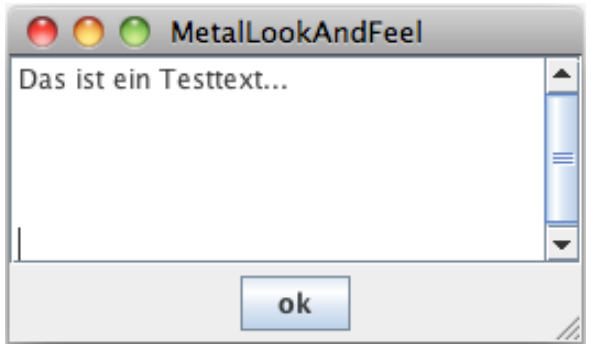
– Hohe Komplexität (s. Klassendiagramm)
– Neue Produkte erfordern Änderungen in gesamter Fabrik-Hierarchie

Weitere Beispiele

- Java Look and Feel (Windows, MacOSX, etc.)
  - Abstrakt: LookAndFeel, Button, EditBox
  - Konkret: WindowsLookAndFeel, WindowsButton, WindowsEditBox







- Spielesammlung (Schach, Mühle, etc.)
  - Abstrakt: Spielfabrik, Spielbrett, Spielfigur
  - Konkret: Schachfabrik mit Schachbrett und Schachfiguren

Lessons learned!

Auswirkungen auf die Software-Architektur in der Praxis

- Fabrikmethode:
  - Ermöglicht losere Kopplung zwischen Nutzer und Produkten
- Abstrakte Fabrik:
  - Ermöglicht Realisierung von Produktfamilien und leichte Ersetzbarkeit
- Beide Entwurfsmuster:
  - Erhöhen Komplexität der Klassenhierarchie (insb. bei abstrakter Fabrik)
  - Sie sind immer mit Bedacht einzusetzen (keine „Entwurfsmusteritis“)
  - Verständnis ist wichtiger als starrer Mustereinsatz

Fabrik-Muster in der Praxis

- Fabrikmethode:
  - Viele Anwendungsfälle, wird häufig genutzt (Komplexität aus Klient auslagern)
  - Erleichtert Testbarkeit

- Abstrakte Fabrik:
  - Anwendungsfall selten, daher wenig Nutzung
  - Nutzung z.B. in Klassenbibliotheken (JDK Look and Feels)

Fabrikmethode oder abstrakte Fabrik?

- Menge unterschiedlicher Anzeigertypen für die dynamische Fahrgastinformation im Personennahverkehr

- GUIs für Geräteklassen (Smartphone, Tablet, PC)

Wer noch tiefer einsteigen möchte…

- „Entwurfsmuster“, E. Gamma et al.

- „Entwurfsmuster von Kopf bis Fuß“, E. Freeman et al.

- „Der Weg zum Java-Profi“, M. Inden

Bezug zu anderen Mustern

- Fabrikmethode ↔ Erzeugungsmethode
  - Fabrikmethode: Kapselung des gesamten Herstellungsprozesses
- Fabrikmethode oft im Zusammenhang mit Singleton-Muster
- Fabrikmethoden werden in abstrakten Fabriken genutzt