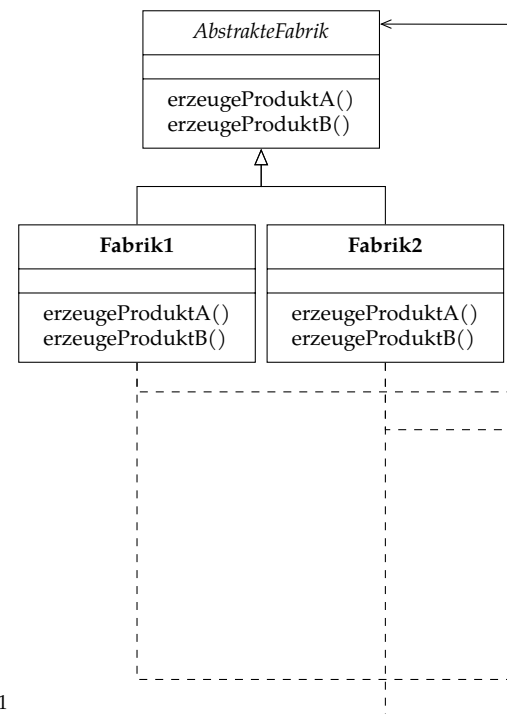


Abstrakte Fabrik (Abstract factory)

Weiterführende Literatur:

- Wikipedia-Artikel „Abstrakte Fabrik“
- <https://www.philippbauer.de/study/se/design-pattern/abstract-factory.php>
- Gamma u. a., *Design Patterns* CD, PDF Seite 77-84
- Schatten, *Best Practice Software-Engineering*, Kapitel 8.3.2, Seite 250-252
- Eilebrecht und Starke, *Patterns kompakt*, Kapitel 1.3, Seite 25-29
- Siebler, *Design Patterns mit Java*, Kapitel 11, Seite 127-145

Es wird eine Schnittstelle bereitgestellt, um Familien verbundener oder abhän-



giger Objekte zu erstellen, ohne die konkreten Klassen zu spezifizieren.¹

```
3 interface AbstraktesProduktA {
4 }
5
6 interface AbstraktesProduktB {
7 }
8
9 class ProduktA1 implements AbstraktesProduktA {
10     public ProduktA1() {
11         System.out.println("ProduktA1 wurde erzeugt");
12     }
13 }
14
15 class ProduktA2 implements AbstraktesProduktA {
16     public ProduktA2() {
```

¹Eilebrecht und Starke, *Patterns kompakt*, Seite 25.

```

17     System.out.println("ProduktA2 wurde erzeugt");
18 }
19 }
20
21 class ProduktB1 implements AbstraktesProduktB {
22     public ProduktB1() {
23         System.out.println("ProduktB1 wurde erzeugt");
24     }
25 }
26
27 class ProduktB2 implements AbstraktesProduktB {
28     public ProduktB2() {
29         System.out.println("ProduktB2 wurde erzeugt");
30     }
31 }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/Produkte.java](https://github.com/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/Produkte.java)

```

3 interface AbstrakteFabrik {
4     public AbstraktesProduktA erzeugeProduktA();
5
6     public AbstraktesProduktB erzeugeProduktB();
7 }
8
9 class Fabrik1 implements AbstrakteFabrik {
10     @Override
11     public AbstraktesProduktA erzeugeProduktA() {
12         return new ProduktA1();
13     }
14
15     @Override
16     public AbstraktesProduktB erzeugeProduktB() {
17         return new ProduktB1();
18     }
19 }
20
21
22 class Fabrik2 implements AbstrakteFabrik {
23     @Override
24     public AbstraktesProduktA erzeugeProduktA() {
25         return new ProduktA2();
26     }
27
28     @Override
29     public AbstraktesProduktB erzeugeProduktB() {
30         return new ProduktB2();
31     }
32 }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/AbstrakteFabrik.java](https://github.com/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/AbstrakteFabrik.java)

```

3 public class Klient {
4     public static void main(String[] args) {
5         // zentraler Austauschpunkt -> Implementierungsaustausch
6         AbstrakteFabrik fabrik1 = new Fabrik1();
7         fabrik1.erzeugeProduktA();
8         fabrik1.erzeugeProduktB();
9         // KonkretesProduktA1 wurde erzeugt
10        // KonkretesProduktB1 wurde erzeugt
11
12        AbstrakteFabrik fabrik2 = new Fabrik2();
13        fabrik2.erzeugeProduktA();
14        fabrik2.erzeugeProduktB();

```

```
15      // KonkretesProduktA2 wurde erzeugt
16      // KonkretesProduktB2 wurde erzeugt
17  }
18 }
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/Klient.java](https://github.com/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/Klient.java)

Literatur

- [1] Karl Eilebrecht und Gernot Starke. *Patterns kompakt. Entwurfsmuster für effektive Softwareentwicklung*. 2019.
- [2] Erich Gamma u. a. *Design Patterns CD. Elements of Resuable Object-Oriented Software*. 1995.
- [3] Alexander Schatten. *Best Practice Software-Engineering. Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. 2010.
- [4] Florian Siebler. *Design Patterns mit Java*. 1. Aufl. Hanser, 2014. ISBN: 978-3-446-44111-8.
- [5] Wikipedia-Artikel „Abstrakte Fabrik“. https://de.wikipedia.org/wiki/Abstrakte_Fabrik.