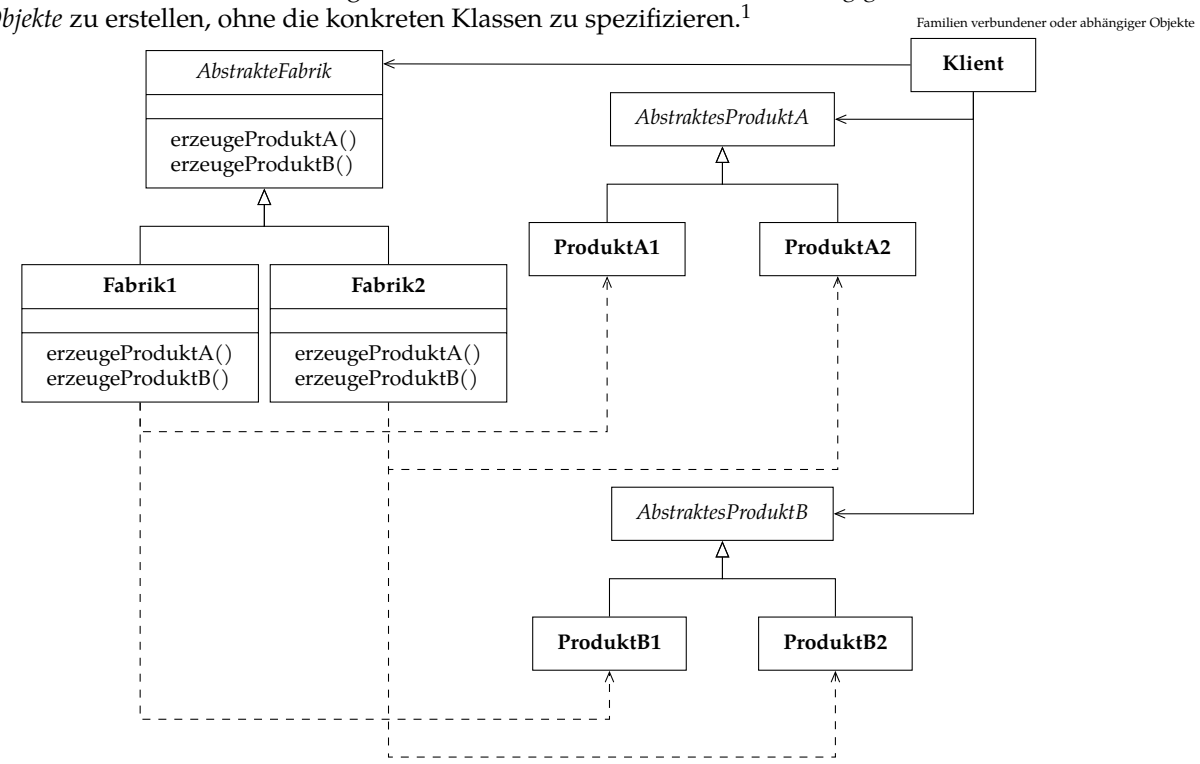
Abstrakte Fabrik (Abstract factory)

Weiterführende Literatur:

- *Wikipedia-Artikel „Abstrakte Fabrik“*
- `https://www.philipphauer.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

Zweck

Es wird eine Schnittstelle bereitgestellt, um *Familien verbundener oder abhängiger Objekte* zu erstellen, ohne die konkreten Klassen zu spezifizieren.[1]



Allgemeines Code-Beispiel

```
3  interface AbstraktesProduktA {
4  }
5
6  interface AbstraktesProduktB {
7  }
8
```

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

```java
class ProduktA1 implements AbstraktesProduktA {
  public ProduktA1() {
    System.out.println("ProduktA1 wurde erzeugt");
  }
}

class ProduktA2 implements AbstraktesProduktA {
  public ProduktA2() {
    System.out.println("ProduktA2 wurde erzeugt");
  }
}

class ProduktB1 implements AbstraktesProduktB {
  public ProduktB1() {
    System.out.println("ProduktB1 wurde erzeugt");
  }
}

class ProduktB2 implements AbstraktesProduktB {
  public ProduktB2() {
    System.out.println("ProduktB2 wurde erzeugt");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/Produkte.java`

```java
interface AbstrakteFabrik {
  public AbstraktesProduktA erzeugeProduktA();

  public AbstraktesProduktB erzeugeProduktB();
}

class Fabrik1 implements AbstrakteFabrik {
  @Override
  public AbstraktesProduktA erzeugeProduktA() {
    return new ProduktA1();
  }

  @Override
  public AbstraktesProduktB erzeugeProduktB() {
    return new ProduktB1();
  }

}

class Fabrik2 implements AbstrakteFabrik {
  @Override
  public AbstraktesProduktA erzeugeProduktA() {
    return new ProduktA2();
  }

  @Override
  public AbstraktesProduktB erzeugeProduktB() {
    return new ProduktB2();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/abstrakte_fabrik/allgemein/AbstrakteFabrik.java`

```java
public class Klient {
  public static void main(String[] args) {
    // zentraler Austauschpunkt -> Implementierungsaustausch
    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`

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`.