

LU05a - Vererbung

Interfaces

Eine Klasse kann eine oder gar mehrere Interfaces implementieren. Nichtabstrakte Klassen müssen alle Methoden ihrer implementierten Interfaces implementieren.

UML	Java	Typescript
<pre> classDiagram class Animal { ~eat(): void ~sleep(): void } class Bird { ~layEggs(): void ~fly(): void } class Penguin { +eat(): void +sleep(): void +layEggs(): void +fly(): void } Animal < -- Bird Bird < .. Penguin </pre>	<pre> public interface Animal {...} public interface Bird extends Animal {...} public class Penguin implements Bird {...} </pre>	<pre> interface Animal {...} interface Bird extends Animal {...} class Penguin implements Bird {...} </pre>

Abstrakte Klassen

Eine abstrakte Klasse ist eine Zwischenform zwischen einem Interface und einer nichtabstrakten Klasse. Sie kann Methoden bereits implementieren oder ohne Implementation definieren. Von einer abstrakten Klasse kann keine direkte Instanz erstellt werden, nur von einer erbenden, nichtabstrakten Klasse.

UML	Java
<pre>classDiagram class Animal { <<interface>> +eat(): void } class Mammal { <<abstract class>> +eat(): void +makeSound(): void +breathe(): void } class Dog { +makeSound(): void } Animal < .. Mammal Mammal < -- Dog</pre>	<pre>interface Animal { void eat(); } abstract class Mammal implements Animal { @Override public void eat() { System.out.println("This mammal is eating."); } abstract void makeSound(); public void breathe() { System.out.println("Breathing air..."); } } class Dog extends Mammal { @Override void makeSound() { System.out.println("Woof!"); } }</pre>

Abstrakte Klassen sind daher nützlich, wenn man gewisse Basisfunktionen in einer gemeinsamen Parentklasse implementieren will, aber wiederum andere Methoden erst in der jeweiligen Implementierungsklasse.

From:
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:
<https://wiki.bzz.ch/de/modul/ffit/3-jahr/java/learningunits/lu05/a>

Last update: **2025/09/16 06:52**

