Vererbung Teil 3 - Das Interface

Wir erinnern uns an die Grundlagen

Begriffe

- Basisklasse (auch Superklasse): Die Klasse von der geerbt wird.
- Abgeleitete Klasse (auch Subklasse): Die Klasse, die erbt.
- Einfachvererbung: Die *abgeleitete Klasse* erbt von **einer** Basisklasse
- Mehrfachvererbung: Die *abgeleitete Klasse* erbt von **mehreren** Basisklassen

Was passiert bei der Vererbung

- Die erbende Klasse übernimmt automatisch alle *Attribute* und *Methoden* der Basisklasse, die nicht als **private** gekennzeichnet sind.

Die normale Klasse als Basisklasse

- Eine normale Klasse *kann* Vorlage sein, muss aber nicht
- Man kann von einer normalen Basisklasse Klasse trotzdem Instanzen erzeugen
- Eine normale Klasse darf *nur* normale (keine abstrakten) Methoden enthalten!
- Erbt man von einer normalen Klasse, *kann* eine vorhandene Methode der Basisklasse überschrieben werden, muss aber nicht

Die abstrakte Klasse als Basisklasse

- Eine abstrakte Klasse dient *nur* als Vorlage für andere Klassen!
- Man kann von einer abstrakten Klasse keine Instanzen erzeugen
- Eine abstrakte Klasse *kann* abstrakte Methoden enthalten, muss aber nicht
- Eine abstrakte Klasse *kann* aber auch normale Methoden enthalten
- Eine abstrakte Methode *muss* in der *abgeleiteten Klasse* implementiert werden, es sei denn die abgeleitete Klasse ist auch abstrakt!
- Eine abstrakte Klasse kann nicht gleichzeitig auch eine *final* Klasse sein

Das Interface als Basisklasse

Was ein Interface ist

Ein Interface als Klassentyp ist einer abstrakten Klasse sehr ähnlich. Es stellt eine gemeinsame Schnittstelle über alle erbenden Klassen dar.

Was anders als bei normalen und abstrakten Klassen ist

- Ein Interface kann, wie eine abstrakte Klasse auch, nicht instanziiert werden.
- Ein Interface enthält keine Konstruktoren
- Ein Interface kann folgende Methodentypen enthalten:
  - Abstrakte Methoden
  - Default-Methoden (seit Java 8 - damit man Interfaces mit Methoden erweitern kann, ohne alle erbenden Klassen anzupassen und bestimmen kann, welche Methoden überschrieben werden müssen, und welche von erbenden Klassen implementiert werden müssen)
  - Statische Methoden
- Alle Methoden eines Interfaces sind standardmäßig public
- Eine Interface-Methode kann entweder eine *default* oder eine *static* Methode sein. (Beides gleichzeutig geht nicht!)
- Ein Interface kann statische oder nicht statische Konstanten enthalten.
- Ein Interface kann keine normalen Attribute enthalten.
- Ein Interface kann von einem oder mehreren anderen Interfaces erben (*extends*)
- Eine, ein Interface implementierende, nicht abstrakte, Klasse muss alle Methoden des Interfaces implementieren, die nicht als *default* oder *static* gekennzeichnet sind
- Eine Klasse kann mehrere Interfaces implementieren

Generischer Code

```
[modifier] interface InterfaceName [extends interfaceName1[,interfaceName2...]]
{
    // Static or non static constant
    [public] [static] [final] type identifier1 = value;

    // Static or non static abstract method
    [public] [static] type identifier2([parameter]);

    // Non static default method
    [public] default type identifier3([parameter]) {
        // Code here
    }

    // Static non default method
    [public] static type identifier4([parameter]) {
        // Code here
    }
}
```

Ein Beispiel: Ein einfaches Interface mit erbender Klasse und Demoprogramm

```java
/**
 * Very simple sample Interface.
 * This is the base for the inheriting class.
 */
public interface ISimpleInterface {
    // A non static constant
    public final String Greeting = "Hello!";

    // A static constant
    public static final String StaticGreeting = "Static Hello!";

    // An abstract method which needs to be implemented when inheriting this
    // interface (non static)
    public void SayHello();
}
```

```java
/**
 * Simple implementation for the base interface.
 */
public class SimpleClass implements ISimpleInterface {
    // Ask yourself: Why are "Greeting" and "StaticGreeting" missing here?

    /**
     * Default (non parameterized) constructor for SimpleClass
     */
    public SimpleClass() {
        System.out.println("SimpleClass created! (Default constructor)");
    }

    /**
     * Displays a greeting text in the console.
     */
    public void SayHello() {
        System.out.println(this.Greeting);
    }
}
```

```java
/**
 * Entry point class for the simple interface demo.
 */
public class SimpleClassMain {
    public static void main(String[] args) {
        // Not working --> interfaces cannot be instanciated
        ////ISimpleInterface simpleInterface = new ISimpleInterface();

        // Working
        // Instanciate new SimpleClass
        SimpleClass simpleClass = new SimpleClass();
        simpleClass.SayHello();
        System.out.println("Non static greeting constant: "
        + simpleClass.Greeting);

        System.out.println("---------------------------");

        // Working too
        // Ask yourself: Do I know WHY this works?
        ISimpleInterface simpleClass2 = new SimpleClass();
        simpleClass2.SayHello();
        System.out.println("Non static greeting constant: "
        + simpleClass2.Greeting);

        System.out.println("---------------------------");

        // Using the static constant of ISimpleInterface
        // Ask yourself: Do I know WHY this works?
        System.out.println("Static greeting constant: "
        + ISimpleInterface.StaticGreeting);
    }
}
```

Weitere Beispiele

Zusätzlich zu dem gegebenen einfachen Beispiel finden Sie zwei weitere Beispiele für Interfaces in der Nähe dieser Lernhilfe. Darin habe ich noch ein mittleres und ein etwas komplexeres Beispiel (beide fiktiv) für Sie aufgebaut. Schauen Sie sich die beiden Beispiele ruhig einmal an, und versuchen Sie die darin enthaltenen Fragen für sich selbst zu beantworten. Fragen zu allen drei Beispielen können wir gern im Tutorium besprechen.