Interfaces: Default-Methoden

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

Problem: Etablierte API (Interfaces) erweitern

```
interface Klausur {
    void anmelden(Studi s);
    void abmelden(Studi s);
}
```

Problem: Etablierte API (Interfaces) erweitern

```
interface Klausur {
    void anmelden(Studi s);
    void abmelden(Studi s);
}
```

=> Nachträglich noch `void schreiben(Studi s);` ergänzen?

Default-Methoden: Interfaces mit Implementierung

```
interface Klausur {
    void anmelden(Studi s);
    void abmelden(Studi s);

    default void schreiben(Studi s) {
        ...     // Default-Implementierung
    }

    default void wuppie() {
        throw new java.lang.UnsupportedOperationException();
    }
}
```

Hinweis: Mehrfachvererbung

Auflösung Mehrfachvererbung: 1. Klassen gewinnen

```java
interface A {
    default String hello() { return "A"; }
}
class C {
    public String hello() { return "C"; }
}
class E extends C implements A {}


/** Mehrfachvererbung: 1. Klassen gewinnen */
public class DefaultTest1 {
    public static void main(String... args) {
        String e = new E().hello();
    }
}
```

Demo: defaultmethods.rule1.DefaultTest1

Auflösung Mehrfachvererbung: 2. Sub-Interfaces gewinnen

```java
interface A {
    default String hello() { return "A"; }
}
interface B extends A {
    @Override default String hello() { return "B"; }
}
class D implements A, B {}


/** Mehrfachvererbung: 2. Sub-Interfaces gewinnen */
public class DefaultTest2 {
    public static void main(String... args) {
        String e = new D().hello();
    }
}
```

Demo: defaultmethods.rule2.DefaultTest2

Auflösung Mehrfachvererbung: 3. Methode explizit auswählen

```java
interface A {
    default String hello() { return "A"; }
}
interface B {
    default String hello() { return "B"; }
}
class D implements A, B {
    @Override public String hello() { return A.super.hello(); }
}


/** Mehrfachvererbung: 3. Methode explizit auswählen */
public class DefaultTest3 {
    public static void main(String... args) {
        String e = new D().hello();
    }
}
```

Demo: defaultmethods.rule3.DefaultTest3

Quiz: Was kommt hier raus?

```java
interface A {
    default String hello() { return "A"; }
}
interface B extends A {
    @Override default String hello() { return "B"; }
}
class C implements B {
    @Override public String hello() { return "C"; }
}
class D extends C implements A, B {}


/** Quiz Mehrfachvererbung */
public class DefaultTest {
    public static void main(String... args) {
        String e = new D().hello(); // ???
    }
}
```

Interfaces vs. Abstrakte Klassen

- **Abstrakte Klassen**: Schnittstelle und Verhalten und Zustand
- **Interfaces**:
  - vor Java 8 nur Schnittstelle
  - ab Java 8 Schnittstelle und Verhalten
- Design:
  - Interfaces sind beinahe wie abstrakte Klassen, nur ohne Zustand
  - Klassen können nur von **einer** (abstrakten) Klasse erben, aber **viele** Interfaces implementieren

Wrap-Up

Seit Java8: Interfaces mit Implementierung: **Default-Methoden**

- Methoden mit dem Schlüsselwort `default` können Implementierung im Interface haben
- Die Implementierung wird vererbt und kann bei Bedarf überschrieben werden
- Auflösung von Mehrfachvererbung:
  - Regel 1: Klassen gewinnen
  - Regel 2: Sub-Interfaces gewinnen
  - Regel 3: Methode explizit auswählen
- Unterschied zu abstrakten Klassen: **Kein Zustand**

LICENSE

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.