

Enumeration Types

Aufzählungstypen

bestehen aus einer festen (und normalerweise kleinen) Anzahl benannter Konstanten

Bsp.:

- Spielkartenfarben: Karo, Kreuz, Herz, Pik
- Wochentage: Montag, ..., Sonntag
- Noten: Sehr gut, ..., Ungenügend
- ...

vor Version 1.5 keine direkt Unterstützung in Java, Modellierung z.B. durch

- `final`-Konstanten vom Typ `int` o.ä., nicht typ-sicher!
- Klasse `Wochentag` mit sieben Unterklassen, sehr aufwändig

Aufzählungstypen in Java

Bsp.: Wochentage

```
public enum Wochentag {  
    MO, DI, MI, DO, FR, SA, SO;  
}
```

definiert Typ mit 7 Werten

Werte zu verwenden z.B. als Wochentag.DO

Aufzählungstypen als Klassen

Deklaration der Form `enum A { ... }` wird vom Compiler in normale Klasse übersetzt

Enum-Typen können auch Methoden haben

Konstanten können assoziierte Werte haben

Beispiel

```
public enum Wochentag {  
    MO(0), DI(1), MI(2), DO(3), FR(4), SA(5), SO(6);  
  
    private tag;  
    private static final MAX = 7;  
  
    Wochentag(int t) {  
        tag = t;  
    }  
  
    public index() {  
        return tag;  
    }  
  
    public static int abstand(Wochentag x, Wochentag y) {  
        return (y.index() - x.index() + MAX) % MAX;  
    }  
}
```

Aufzählungstypen im Typsystem

Enum-Typen erweitern immer `java.lang.Enum` implizit

`public enum A extends B { ... }` nicht zulässig

cleverer Mechanismus (in der Konstruktion von `Enum`, nicht im Compiler) sorgt dafür, dass folgendes bereits syntaktisch nicht korrekt ist

```
enum A { A1, A2 }  
enum B { B1, B2 }
```

```
...  
if (B1.compareTo(A2)) { ... }
```

Zu beachten

- Konstruktoren in Enum-Typen nicht `public` machen
- Enum-Typen können nicht mithilfe von `extends` etwas erweitern
- Werte eines Enum-Typs sind automatisch geordnet, wie üblich mit `compareTo` erfragen
- Parameter an Konstanten können sogar Methoden sein
- statische Methode `values()` liefert `Collection` der einzelnen Werte, kann z.B. mit `Iterator` durchlaufen werden