

Softwareentwicklung II (IB)

Enum-Klassen

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 28.01.2020 17:34

Inhaltsverzeichnis

Idee	1
Definition	2
Eigenschaften	2
Enumtypen	2
Enums als Referenztypen	3
Vordefinierte Methoden	3
Methoden	4
Objektvariablen	4
Objektvariablen und Konstruktoren	4

Idee

- `int`, `double` und `boolean` speichern Zahlen und Wahrheitswerte
- Oft abgegrenzte Sammlung von Werten gebraucht, die weder Zahlen noch Wahrheitswerte sind
- Beispiele: Farben, Wochentage, Geschlecht, Schachfiguren
- Codierung als Zahlen- oder Wahrheitswerte technisch möglich, aber logisch willkürlich und irreführend
- **Aufzählungstypen** (engl. enumeration types, „Enums“) erlauben Typdefinition für begrenzte (aufzählbare) Wertemengen

Definition

- Definitionsschema für **Enumtypen**:

```
enum enumtype {enumelement, enumelement, ...}
```

- enumtype benennt den Typ. Gleiche Namenskonventionen wie Klassennamen
- **Enumwerte** enumelement in den geschweiften Klammern aufzählen
- Per Konvention groß geschriebene, englische Substantive
- Beispiele:

```
enum Color {Red, Green, Blue, Yellow}
```

```
enum Day {Mon, Tue, Wed, Thu, Fri, Sat, Sun}
```

```
enum Sex {Female, Male}
```

```
enum ChessPiece {Pawn, Rook, Knight, Bishop, Queen, King}
```

Eigenschaften

- Enumwerte innerhalb eines Typs müssen eindeutig sein

```
enum Choice {Foo, Bar, Baz, Foo} // Fehler, Foo doppelt
```

- Enumwerte in verschiedenen Enumtypen unabhängig

```
enum Choice {Foo, Bar}
```

```
enum Selection {Baz, Foo} // ok
```

- Enumwerte in unterschiedlichen Enumtypen sind inkompatibel, auch bei gleichen Namen

Enumtypen

- Enumtypen gleichberechtigt mit anderen Typen
- Beispiel: Definition einer Variablen

```
Color c;
```

- **Enum-Literal** = Typnamen und Elementnamen nach dem Schema enumtype.enumelement

- Beispiel: Zuweisen eines Wertes:

```
c = Color.Red;
```

- Vergleich eines Wertes:

```
if (c == Color.Yellow) ...
```

Enums als Referenztypen

- Definition eines Enumtyps = spezielle Klassendefinition
- Enumtypen sind Referenztypen
- Enumwerte = Klassenvariablen
- Beispiel: Definition

```
enum Color {Red, Green, Blue, Yellow}
```

- in etwa äquivalent zu

```
class Color {  
    final static Color Red    = new Color();  
    final static Color Green  = new Color();  
    final static Color Blue   = new Color();  
    final static Color Yellow = new Color();  
}
```

- Syntax der Enum-Literale: Klassenvariable Red der Klasse Color: `Color.Red`
- Neue Enumobjekte können nicht erzeugt werden

Vordefinierte Methoden

static E valueOf(String s) Enumwert mit dem Namen `s` (`IllegalArgumentException`, wenn der Name nicht existiert.)

static E[] values() Array mit allen Enumelementen

String toString() Name dieses Enumwertes als String

boolean equals(Object x) Vergleicht diesen Enumwert mit `x`: `true` wenn beide gleich sind, `false` ansonsten

int hashCode() Kennnummer (Hashcode) dieses Enumwertes

int compareTo(E x) Vergleich dieses Enumwertes mit `x` gemäß Definitionsreihenfolge

int ordinal() Index dieses Enumwertes gemäß Definitionsreihenfolge (erster Wert = Index 0)

- Begriffe „Enumelement“, „Enumwert“, „Enumobjekt“ synonym:
 - Enumobjekte sind eindeutig,
 - zusätzliche Objekte können nicht erzeugt werden,
 - Vergleich mit `==` statt `equals` ausreichend

Methoden

- In Enums können Methoden definiert werden

```
enum Day {
    Mon, Tue, Wed, Thu, Fri, Sat, Sun;

    boolean isWeekend() {
        return this == Sat || this == Sun;
    }
}
```

- Aufruf mit Enumwert als Zielobjekt:

```
Day today = ...;
if (today.isWeekend())
    ...
```

Objektvariablen

- Neben Methoden auch Objektvariablen in Enumklassen
- Beispiel: boolean-Objektvariable für Tage des Wochenendes:

```
enum Day {
    ...
    private final boolean weekend;
}
```

isWeekend benutzt die Objektvariable (Test einer Bedingung unnötig):

```
enum Day {
    ...
    boolean isWeekend() {
        return weekend;
    }
}
```

Objektvariablen und Konstruktoren

- Initialisierung der Objektvariablen in einem Konstruktor:

```
enum Day {
    Day(boolean w) {
        weekend = w;
    }
}
```

- Argumente für Konstruktor in der Definition:

```
enum Day {  
    Mon(false), Tue(false), Wed(false), Thu(false),  
    Fri(false), Sat(true), Sun(true);  
}
```

- Trotz Konstruktor: Compiler verhindert Erzeugen neuer Enumwerte