ENUM

Aufzählungstyp

Bedeutung

- Oft werden Datentypen mit bestimmten Werten gebraucht
- Enum´s sind keine Zahlen oder Wahrheitswerte

- Beispiele
  - *Farben: rot, grün, gelb*
  - *Wochentage: Mo, Di, Mi, Do, Fr, Sa, So*
  - *Spielerposition: Torwart, Kreisläufer, Rückraum, Außen*
- Wenn wir die Codierung über Zahlen vornehmen, wäre das möglich aber irreführend
- Aufzählungstyp erlaubt Typdefinition mit begrenzter Wertmenge
- Synonyme: Enumeration

Syntax

- Syntax
  - *enum enumtype {enumelement(,enumelementN)*}*
  - *Aufzählungstyp benannt durch enumtype*
- Konventionen
  - *enumtype wie Klassenname*
  - *enumelement groß geschriebenes, englisches Substantiv*
- Beispiele
  - *enum Color {Red, Green, Blue, Yellow}*
  - *enum Day {Mon, Tue, Wed, Thu, Fri, Sat, Sun }*
- Eigenschaften
  - *Einzelne Werte innerhalb des Aufzählungstyp sind Unique*
  - *Gleichnamige Werte in verschiedenen Aufzählungstypen sind inkompatible*
  - *Zugriff: Day.Mon*

Verwendung

- Gegeben
  - *enum Color {Red, Green, Blue, Yellow}*
- Verwendung
  - *Gleichberechtigt zu anderen Typen in Java*
  - *Deklaration einer Variablen*
    - Color c;
  - *Zugriff auf das Literal eines Aufzählungstyp*
    - Color.Red
  - *Zuweisen eines Wertes*
    - c = Color.Red;
  - *Vergleich von Werten / Objekten eines Aufzählungstyps*
    - if (c == Color.Red) ….;
    - Vergleich mit == ausreichend, equals() nicht erforderlich

Vergleichbare Klassendefinition

- Eigenschaften
  - *Aufzählungstyp ist eine spezielle Art von Klasse*
  - *Also euch ein Referenztyp*
  - *Aufzählungswerte sind finale statische Datenelemente*
  - *Zur Laufzeit kein neuer Aufzählungswert erzeugbar!*
- Aufzählungstyp
  - *enum Color {Red, Green, Blue, Yellow}*
- Vergleichbare Klassendefinition
  - ```
    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();
    }
    ```
    ```

Vordefinierte Methoden

- boolean equals (Object x)
  - *Vergleicht aktuellen Aufzählungswert mit x*
  - *Liefert true, wenn beide gleich sind, sonst false;*
- int compareTo (E x)
  - *Vergleicht aktuellen Aufzählungswert mit x bzgl. der Reihenfolge der Definition*
- int ordinal ()
  - *Liefert den Index des aktuellen Aufzählungswertes bzgl. der Definitionsreihenfolge*
  - *Erster Wert hat den Index 0*
- Es gibt noch weitere vordefinierte Methoden

Eigene Methoden

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

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

Aufruf mit dem Aufzählungswert als Zielobjekt

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

Eigene Methoden

Definition von Datenelementen in Aufzählungstyp möglich

Beispiel: Datenelement zum Kennzeichnen von Wochenendtagen

- Test über Bedingung nicht mehr notwendig, nutzen von weekend
- Initialisieren des Datenelementes in einem Konstruktor
- Argumente für Konstruktor in Definition mit aufgezählt

```
Enum Day {
        Mon(false), Tue(false), Wed(false), Thu(false), Fri(false), Sat(true), Sun(true);
        private final boolean weekend;
        Day(boolean w) { weekend = w; }
        boolean isWeekend() {return weekend; }
}
```

Compiler verhindert Erzeugen neuer Aufzählungswerte (trotz Konstruktor)