Java Enum

Eine Menge bekannter und typsicherer Konstanten

Enum in unterschiedlichen Sprachen

C und C++

Aufzählung von benannten Integer-Konstanten

```
enum Color { red, green, blue }; // Verwendung: color = red;
```

C# und C++ (ab C++11)

Wie C, aber mit Namensraum

```
enum Color { Red, Green, Blue }; // Verwendung: color = Color.Red;
```

Java vor Version 5.0

Keine Unterstützung in der Sprache

Muster: Integer-Konstanten in Klassen

```
public class Color {
  public static final int COLOR_RED   = 0;
  public static final int COLOR_GREEN = 1;
  public static final int COLOR_BLUE  = 2;
}
```

Java ab Version 5.0

Sprachunterstützung (zB: Schlüsselwort enum, Klassenbibliothek EnumSet)

Spezielle Klasse (nicht vererbbar, Konstanten sind definierte Instanzen)

```
public enum Color { RED, GREEN, BLUE }
```

Java-Enums können mehr!

Typsicher

```
Color color = Coin.ONE_CENT; // Fehler: incompatible types
```

Vergleichbar (Comparable)

```
if (coin.compareTo(Coin.FIFTY_CENT) < 0) { … }
```

Serialisierbar (Serializable)

Übertragen über Netzwerk, für entfernte Methodenaufrufe

Konstante zu Integer (ordinal()**)**

```
Color color = Color.Red;
System.out.println(color.ordinal());        // 0, 1 oder 2 für RED, GREEN bzw. BLUE
```

Konstante zu String (toString()**)**

```
String colorName = color.toString();        // "RED", "GREEN" oder "BLUE"
```

String zu Konstante (<Enum>.valueOf(String)**)**

```
Color color = Color.valueOf("RED");
```

Switch-Unterstützung

```
switch (color) {
  case RED: …
  …
}
```

```
public enum Color { RED, GREEN, BLUE }
public enum Coin  {
  ONE_CENT, TWO_CENT, FIVE_CENT,
  TEN_CENT, TWENTY_CENT, FIFTY_CENT,
  ONE_EURO, TWO_EURO
}
```

Java-Enums können mehr!

Aufzählbar – values()

```
for (Color color : Color.values()) {
    System.out.println(color);
}
```

Eigene Konstruktoren, Methoden und Felder

```
public enum Coin {
  ONE_CENT(1), TWO_CENT(2), FIVE_CENT(5), TEN_CENT(10),
  TWENTY_CENT(20), FIFTY_CENT(50),
  ONE_EURO(100), TWO_EURO(200);

  private final int valueInCent;

  Coin(int valueInCent) { this.valueInCent = valueInCent; }

  public int getValueInCent() { return valueInCent; }

  public Coin getNextBiggerCoin() { return getNextFittingCoin(getValueInCent() + 1); }

  public static Coin getNextFittingCoin(int cent) {
    for (Coin coin : values()) {
      if (coin.getValueInCent() > cent) { return coin; }
    }
    return null;
  }
}
```

ⓘ Achtung, jeder Code ist technisch möglich, aber Enums müssen konstant sein, damit sie definitionsgemäß funktionieren!

Java-Enums können mehr!

Eigene Methoden pro Konstante

```
public enum Coin {
  ONE_CENT(1)      { public Coin getNextSmallerCoin() { return null;        } },
  TWO_CENT(2)      { public Coin getNextSmallerCoin() { return ONE_CENT;    } },
  FIVE_CENT(5)     { public Coin getNextSmallerCoin() { return TWO_CENT;    } },
  TEN_CENT(10)     { public Coin getNextSmallerCoin() { return FIVE_CENT;   } },
  TWENTY_CENT(20)  { public Coin getNextSmallerCoin() { return TEN_CENT;    } },
  FIFTY_CENT(50)   { public Coin getNextSmallerCoin() { return TWENTY_CENT; } },
  ONE_EURO(100)    { public Coin getNextSmallerCoin() { return FIFTY_CENT;  } },
  TWO_EURO(200)    { public Coin getNextSmallerCoin() { return ONE_EURO;    } };

  private final int valueInCent;

  Coin(int valueInCent) { this.valueInCent = valueInCent; }

  public int getValueInCent() { return valueInCent; }

  public abstract Coin getNextSmallerCoin();

  public Coin getNextBiggerCoin() { return getNextFittingCoin(getValueInCent() + 1); }

  public static Coin getNextFittingCoin(int cent) {
    for (Coin coin : values()) {
      if (coin.getValueInCent() > cent) { return coin; }
    }
    return null;
  }
}
```

ⓘ Achtung, jeder Code ist technisch möglich, aber Enums müssen konstant sein, damit sie definitionsgemäß funktionieren!

Literatur

Joshua Bloch, 2008, Effective Java, Second Edition (Chapter 6, Enums and Annotations)

Gosling, et al., 2015, The Java Language Specification, Java SE 8 Edition (8.9 Enum Types)

Christian Ullenboom, 2011, Java ist auch eine Insel (5.4.3 Aufzählungen mit enum)

Joshua Bloch, 2001, Effective Java, Programming Language Guide (Item 21, typsichere Enums für Java vor Version 5.0)