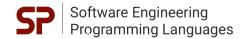


04-Objekte-5-Enumerations/Records

Objektorientierte Programmierung | Matthias Tichy





Lernziele

- Enumerations
- Record

Aufzählungstypen

Enumeration-Klassen erlauben die Definition von nominalskalierten Variablentypen.

```
public enum StandardOpenOption implements OpenOption {
    READ,
    WRITE,
    APPEND,
    TRUNCATE_EXISTING,
    CREATE,
    CREATE_NEW,
    DELETE_ON_CLOSE,
    ...
}
```

https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/nio/file/StandardOpenOption.html

Keine Alternative

- Klassische Umsetzung mit int-Konstanten (z.B. in C)
- Warum ist das keine (gute) Alternative?

```
public class StandardOpenOptions {
  public static final int READ = 1;
  public static final int WRITE = 1 << 1;
  public static final int APPEND = 1 << 2;
  ...
}</pre>
```

Enumeration können einen Body haben – mit Konstruktoren, Attributen und Methoden.

```
public enum GeldStuecke {
    Cent(1), Cent_2(2), Cent_5(5), Cent_10(10), Cent_20(20), Euro_1(100), Euro_2(200);

    private int cent_value;
    private GeldStuecke(int value) {
        this.cent_value = value;
    }
    public int getCent_Value() {
        return cent_value;
    }

James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, Daniel Smith, Gavin Bierman:
```

James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, Daniel Smith, Gavin Bierman The Java® Language Specification - Java SE 20 Edition - 2023-03-03 - **§8.9 Enum Classes** https://docs.oracle.com/javase/specs/jls/se20/html/jls-8.html#jls-8.9

Switch / Case mit Enumerations

```
public static void main(String[] args) {
                                                       Nutzung eines Aufzählungswertes
 GeldStuecke stueck = GeldStuecke. Cent 10;
                                                    switch als Ausdruck → Zuweisung
  String name = switch (stueck) {
  case Cent -> "1er";
                                                    von Ergebnissen
  case Cent 2 -> "2er";
  case Cent 5 -> "5er";
  case Cent_10 -> "10er";
  case Cent_20 -> "20er";
                                              Prüfung des Compilers auf Vollständigkeit
  case Euro 1 -> "Euro";
  case Euro 2 -> "2Euro";
System.out.format("Wert von %s mit Namen %s ist %d cent",
                   stueck, name, stueck.getCent_Value());
                                                                     Zugriff auf Methode
```

Records

Murmelgruppe 3

Java Playground auf https://www.online-java.com/

```
public static void main(String args[]) {
    var strings = Arrays.asList(new String[] { "one", "two", "three", "three", "four" });
    var result = // TODO

    System.out.println("Das Ergebnis ist " + result);
    // Das Ergebnis ist "three",5 und "four", 4
}
```

Record

Leichtgewichtiges Konstrukt für Datenaggregation

```
public static void main(String args[]) {
                                                                         record mit 2 Attributen:
                                                                          s: String
        record StringLengthTuple(String s, int length) {
                                                                          length: int
        var strings = Arrays.asList(new String[] { "one", "two", "three", "three", "four" });
        var result = new LinkedList<StringLengthTuple>();
        var distinctStrings = new HashSet<>();
        for (var s : strings) {
                if (s.length() >= 4 && !distinctStrings.contains(s)) {
                         result.add(new StringLengthTuple(s, s.length()));
                         distinctStrings.add(s);
        System.out.println("Das Ergebnis ist " + result);
```

Record

Leichtgewichtiges Konstrukt für Datenaggregation

"A record declaration specifies a new record class, a restricted kind of class that defines a simple aggregate of values."

```
record StringLengthTuple(String s, int length) {}

Record Component Record Component
```

- Keine Vererbung (→ extends Record)
- Automatisch:
 - Generierung von Methoden wie equals()
 - privates finales Attribut für jeden Record Component (immutable)
 - public getter-Methode für jeden Record Component

James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, Daniel Smith, Gavin Bierman: The Java® Language Specification - Java SE 20 Edition - 2023-03-03 - **§8.10 Record Classes** https://docs.oracle.com/javase/specs/jls/se20/html/jls-8.html#jls-8.10

Record

Leichtgewichtiges Konstrukt für Datenaggregation

- Statische Attribute und Methoden möglich
- Definition weiterer Methoden möglich
- Definition weiterer Konstruktoren möglich

```
record StringLengthTuple(String s, int length) {
    public StringLengthTuple {
        Objects.requireNonNull(s);
}

public StringLengthTuple(String s) {
        this(s, s.length());
}

Aufruf eines anderen Konstruktors
Weiterer Konstruktor
```

James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, Daniel Smith, Gavin Bierman: The Java® Language Specification - Java SE 20 Edition - 2023-03-03 - §8.10 Record Classes https://docs.oracle.com/javase/specs/jls/se20/html/jls-8.html#jls-8.10

Lernziele

- Enumerations
- Record