Aufzählungen (Enumerations)

Carsten Gips (HSBI)

Motivation

```java
public class Studi {
    public static final int IFM = 0;
    public static final int ELM = 1;
    public static final int ARC = 2;

    public Studi(String name, int credits, int studiengang) {
        // Wert für studiengang muss zwischen 0 und 2 liegen
        // Erwünscht: Konstanten nutzen
    }

    public static void main(String[] args) {
        Studi rainer = new Studi("Rainer", 10, Studi.IFM);
        Studi holger = new Studi("Holger", 3, 4);   // Laufzeit-Problem!
    }
}
```

Probleme: Typsicherheit, Kontext

Verbesserung: Einfache Aufzählung

```java
public enum Fach {
    IFM, ELM, ARC
}


public class Studi {
    public Studi(String name, int credits, Fach studiengang) {
        // Typsicherheit für studiengang :-)
    }

    public static void main(String[] args) {
        Studi rainer = new Studi("Rainer", 10, Fach.IFM);
        Studi holger = new Studi("Holger", 3, 4);    // Syntax-Fehler!
    }
}
```

Einfache Aufzählungen: Eigenschaften

```
public enum Fach {
    IFM, ELM, ARC
}
```

1. Enum-Konstanten (`IFM`, ...) sind implizit `static` und `final`
2. Enumerations (`Fach`) nicht instantiierbar
3. Enumerations stellen einen neuen Typ dar: hier der Typ `Fach`
4. Methoden: `name()`, `ordinal()`, `values()`, `toString()`

Erinnerung: Bedeutung von static und final

Einfache Aufzählungen: Eigenschaften (cnt.)

```java
// Referenzen auf Enum-Objekte können null sein
Fach f = null;
f = Fach.IFM;

// Vergleich mit == möglich
// equals() unnötig, da Vergleich mit Referenz auf statische Variable
if (f == Fach.IFM) {
    System.out.println("Richtiges Fach :-)");
}

// switch/case
switch (f) {
    case IFM:   // Achtung: *NICHT* Fach.IFM
        System.out.println("Richtiges Fach :-)");
        break;
    default:
        throw new IllegalArgumentException("FALSCHES FACH: " + f);
}
```

Enum: Genauer betrachtet

```
public enum Fach {  IFM, ELM, ARC  }
```

Compiler sieht (*in etwa*):

```
public class Fach extends Enum {
    public static final Fach IFM = new Fach("IFM", 0);
    public static final Fach ELM = new Fach("ELM", 1);
    public static final Fach ARC = new Fach("ARC", 2);

    private Fach( String s, int i ) { super( s, i ); }
}
```

=> **Singleton-Pattern** für Konstanten

Enum-Klassen: Eigenschaften

```
public enum Fach {
    IFM,
    ELM("Elektrotechnik Praxisintegriert", 1, 30),
    ARC("Architektur", 4, 40),
    PHY("Physik", 3, 10);

    private final String description;
    private final int number;
    private final int capacity;

    Fach() { this("Informatik Bachelor", 0, 60); }
    Fach(String descr, int number, int capacity) {
        this.description = descr;  this.number = number;  this.capacity = capacity;
    }
    public String getDescription() {
        return "Konstante: " + name() + " (Beschreibung: " + description
                + ", Kapazitaet: " + capacity + ", Nummer: " + number
                + ", Ordinal: " + ordinal() + ")";
    }
}
```

Wrap-Up

- Aufzählungen mit Hilfe von `enum`
- Komplexe Enumerations analog zu Klassendefinition: Konstruktoren, Felder und Methoden (keine Instanzen von Enum-Klassen erzeugbar)
- Enum-Konstanten sind implizit `final` und `static`
- Compiler stellt Methoden `name()`, `ordinal()` und `values()` zur Verfügung

LICENSE