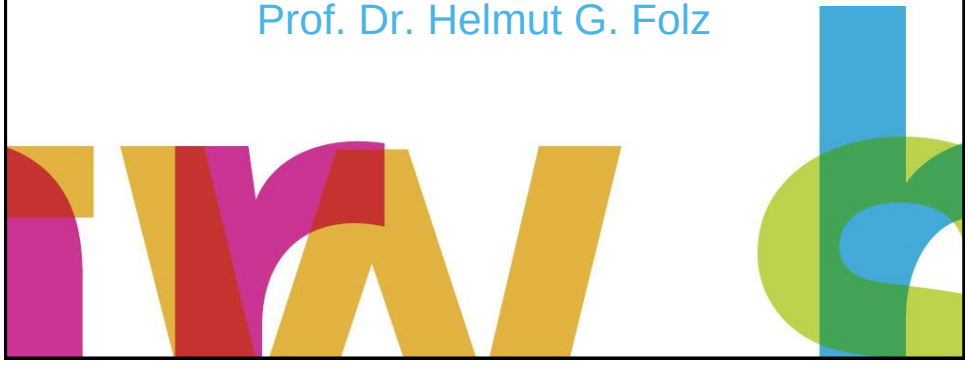
ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Aufzählungstypen

Prof. Dr. Helmut G. Folz



Einführung

- **Aufzählungstypen** sind Datentypen, die als Wertebereich eine endliche geordnete Menge von Konstanten zulassen.
- Aufzählungstypen sind in Java ab der Version 5 verfügbar.
- Einfachstmögliche Definition:

```
enum Typname { Wert1, Wert2, ... }
```

Eigenschaften

```
enum Typname { Wert1, Wert2, ... }
```

- Durch diese Deklaration wird ein Datentyp **`Typname`** vereinbart, der als möglichen Inhalt die Werte `Typname.Wert1`, `Typname.Wert2`, usw. annehmen kann.
- Unter Verwendung des Typnamens können Variablen oder Parameter deklariert werden und die Werte können diesen zugewiesen oder in Ausdrücken zur Abfrage verwendet werden.
- Aufzählungen sind als Klassen und ihre Werte als Objekte realisiert.

Eigenschaften

Enums

- besitzen eine **`toString`**–Methode, die den Namen des Wertes im Klartext ausgibt.
- können mit **`equals`** auf Gleichheit geprüft werden.
- Besitzen eine Methode **`ordinal`**, die den Index des Wertes innerhalb des Aufzählungstyps zurückgibt.
- können in **`switch`**–Anweisungen verwendet werden.
- Der Enum-Typ besitzt eine Klassen-Methode **`values`**, die eine Referenz auf ein Array mit allen möglichen Werten zurückliefert.

Beispiel: Farbentest

```
public class FarbenTest {
    enum Farbe {ROT, GRUEN, BLAU, GELB}

    public static void farbVergleich(Farbe f1, Farbe f2) {
        System.out.print(f1);
        System.out.print(f1.equals(f2) ? " = " : " != ");
        System.out.println(f2);
    }

    public static String toRGB(Farbe f) {
        String ret = "?";
        switch (f) {
           case ROT:   ret = "(255,0,0)"; break;
           case GRUEN: ret = "(0,255,0)"; break;
           case BLAU:  ret = "(0,0,255)"; break;
           case GELB:  ret = "(255,255,0)"; break;
        }
        return ret;
    }
```


Beispiel: Farbentest

```
public static void main(String[] args) {
       //Aufzaehlungsvariablen
       Farbe f1 = Farbe.ROT;
       Farbe f2 = Farbe.BLAU;
       Farbe f3 = Farbe.ROT;

       // toString() liefert den Namen
       System.out.println(f1);
       System.out.println(f2);
       System.out.println(f3);

       // equals funktioniert auch
       System.out.println();
       farbVergleich(f1, f2);
       farbVergleich(f1, f3);
       farbVergleich(f2, f3);
       farbVergleich(f1, f1);

       // Die Methode values()
       System.out.println();
       for (Farbe f : Farbe.values()) {
           System.out.println(f + "=" + toRGB(f));
    }
  }
```

Erweiterungen

- Aufzählungstypen werden in (lokale) Klassen übersetzt und ihre Werte sind Objekte dieser Klasse.
- Basisklasse aller Aufzählungstypen ist die Klasse `java.lang.Enum`
- Es besteht die Möglichkeit, den Aufzählungstyp um Attribute und Methoden zu erweitern.

Enum Farbe

```
public enum Farbe {
    ROT(255, 0 , 0),
    GRUEN(0, 255, 0),
    BLAU(0, 0, 255),
    GELB(255, 255, 0);

    private final int r;
    private final int g;
    private final int b;

    private Farbe(int r, int g, int b) {
        this.r = r;
        this.g = g;
        this.b = b;
    }

    public String toRGB() {
        return "(" + r + "," + g + "," + b + ")";
    }
}
```

Enum Farbe

```
public class FarbenTest2 {
    public static void main(String[] args) {
        // Alle Farben ausgeben
        for (Farbe f : Farbe.values()) {
            System.out.println(f + ":" + f.toRGB());
        }
        System.out.println();

        // Alle Werte von Farbe in ein Array schreiben
        // und ausgeben
        Farbe[] ftab = Farbe.values();
        for (int i = 0; i < ftab.length; i++)
            System.out.println(ftab[i] + ":"
                               + ftab[i].toRGB());
    }
}
```


Anwendung von EnumSet

- Die Klasse **`java.util.EnumSet`** ist eine Klasse zur Mengen-Verwaltung von Werten von Aufzählungstypen.
- Sie kann angewendet werden, um Teilmengen der Wertemenge eines Aufzählungstyps zu definieren, über die iteriert werden soll.

Anwendung von EnumSet (vereinfacht)

- EnumSet ist eine sogenannte **Collection-Klasse**, d. h. es ist möglich mit Hilfe der foreach-Schleife über die Inhalte eines EnumSet-Objektes zu iterieren.
- Der Parameter <E> ist ein **Typparameter**, d. h. E ist der Platzhalter für den Typ der Objekte, die die Menge aufnehmen kann
- `public static EnumSet<E> range(E from, E to);`
  erstellt die Teilmenge der Werte des Aufzählungstyps, die zwischen from und to liegen.
- `public static EnumSet<E> of(E e1, E e2, ... );`
  erstellt die Teilmenge der Werte des Aufzählungstyps, die aus den übergebenen Werten besteht.
- `public boolean contains(Object o);`
  überprüft, ob ein bestimmtes Objekt in der Menge vorhanden ist.

Beispiel: Anwendung EnumSet

```
public enum Wochentag {
    Sonntag, Montag, Dienstag,
    Mittwoch, Donnerstag, Freitag, Samstag
}
```

Beispiel: Anwendung EnumSet

```
import java.util.EnumSet;
import static bspEnum.Wochentag.*;

public class WochentagTest {
    public void start() {
        // Ein Bereich innerhalb des Aufzählungstyps
        EnumSet<Wochentag> werktage =
                            EnumSet.range(Montag, Freitag);

        // Eine Teilmenge innerhalb des Aufzählungstyps
        EnumSet<Wochentag> wochenende =
                            EnumSet.of(Samstag, Sonntag);
```


Beispiel: Anwendung EnumSet

```
        // Alle Werte des Aufzählungstyps durchlaufen
        for (Wochentag w : Wochentag.values()) // alle Werte des Typs
        {
            if (werktage.contains(w))
                System.out.println(w + " ist ein Werktag");
            else if (wochenende.contains(w))
                System.out.println(w + " ist am Wochenende");
        }

        // Nur Werktage durchlaufen
        System.out.print("\nWerktage: ");
        for (Wochentag w : werktage) {
            System.out.print(w + " ");
        }
        System.out.println();

    }

     public static void main(String[] args) {
        new WochentagTest().start();
    }
}
```

Anwendung in einem Dialogprogramm

- Bisherige Lösung für das Menü:

```
public class BankDialog {
    // Attribute
    private Bank bank1;
    private Scanner input = new Scanner(System.in);

    // Klassenkonstanten
    private static final int ANLEGEN     = 1;
    private static final int EINZAHLEN   = 2;
    private static final int ABHEBEN     = 3;
    private static final int LOESCHEN    = 4;
    private static final int UEBERWEISEN = 5;
    private static final int ENDE        = 0;
```


Anwendung in einem Dialogprogramm

- Lösung mit Aufzählungstyp:

```
public class BankDialog {
    // Attribute
    private Bank bank1;
    private Scanner input = new Scanner(System.in);

    // Aufzählungstyp für das Menü
    enum Funktion {ENDE, ANLEGEN, EINZAHLEN, ABHEBEN, LOESCHEN,
                   UEBERWEISEN  };
    private String menue;
```

Anwendung in einem Dialogprogramm

- Bisherige Menüausgabe

```
private int einlesenFunktion() {
    System.out.print(ANLEGEN     + ": anlegen; " +
                     EINZAHLEN   + ": einzahlen; " +
                     ABHEBEN     + ": abheben; " +
                     LOESCHEN    + ": löschen; " +
                     UEBERWEISEN + ": überweisen; " +
                     ENDE        + ": beenden -> ");

    return input.nextInt();
}
```


Anwendung in einem Dialogprogramm

- Menüausgabe mit Aufzählungstyp

```
public BankDialog() {
    StringBuffer sb = new StringBuffer();
    for (Funktion f : Funktion.values()) {
        sb.append(f.ordinal()).append(": ")
          .append(f).append("; ");
    }
    sb.append(" -> ");
    menue = sb.toString();
}

private Funktion einlesenFunktion() {
    System.out.print(menue);
    int eingabe = input.nextInt();
    if (eingabe >= 0 && eingabe < Funktion.values().length)
        return Funktion.values()[eingabe];
    else
        throw new IllegalArgumentException("Falsche Funktion: "
                                           + eingabe);
}
```

Anwendung in einem Dialogprogramm

- Ausführen der Funktion

```
private void ausfuehrenFunktion(Funktion funktion) {
    int kontonr, nachKontonr;
    double betrag;

    switch (funktion) {
        case ANLEGEN:
            bank1.legeKontoAn(einlesenKontonr(),
                              einlesenInhaber());
           break;

        case ABHEBEN:
            ...
```