Einführung in die Programmierung

by André Karge

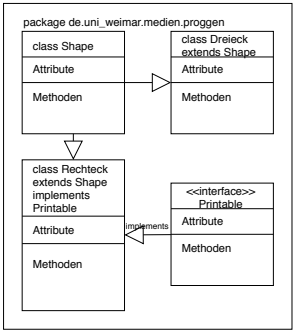
Übung - Wiederholung part 1

Heute

- Klassenstruktur
- Clean Code & Coding Conventions
- Enum

Klassenstruktur

Klassenstruktur




```java
package de.uni_weimar.medien.proggen;
public class Shape {
  // ---- alle Attribute ----
  protected String name;
  // ---- alle Methoden ----
  public Shape(String name) {...} // = constr.
  public double calcArea() {...}
  // ---- getter & setter ----
  public void setName(String name) {...}
  public String getName() {...}
}
public class Dreieck extends Shape {
  // ---- alle Attribute ----
  private double a;
  private double b;
  private double c;
  // ---- alle Methoden ----
  public Dreieck(String name, // = constr.
                 double a,
                 double b,
                 double c) {
    super(name);
    ...
  }
  @Override public double calcArea() {...}
  // ---- getter & setter ----
  public void set(double a, double b, double c) {...}
  public double getA() {...}
  public double getB() {...}
  public double getC() {...}
}
```

```java
public interface Printable {
  // ---- alle Methoden ----
  void printToConsole();
}
public class Rechteck extends Shape
    implements Printable {
  // ---- alle Attribute ----
  private double x;
  private double y;
  // ---- alle Methoden ----
  public Rechteck(String name,  // = constr.
                  double x,
                  double y) {
    super(name);
    ...
  }
  @Override public double calcArea() {...}
  public void printToConsole() {...}
  // ---- setter & getter ----
  public void set(double x, double y) {...}
  public double getX() {...}
  public double getY() {...}
}
```

Clean Code & Coding Conventions

Clean Code & Coding Conventions

- Klassen und Interfaces beginnen immer mit Großbuchstaben (class Shape, class Star, interface Printable)
- Attribute beginnen immer mit Kleinbuchstaben (private int alter, private String textFeld)
- Methoden beginnen auch immer mit Kleinbuchstaben (public static int calcSum())
- Bezeichner sollten immer Aussagekräftig sein
- Bezeichner können im *CamelCase* geschrieben werden

Clean Code & Coding Conventions

Kommentare

- Man macht an jede Stelle im Programmcode Kommentare, wo es nicht direkt ersichtlich ist, was die Prozedur macht

```
public int calcSum(int a, int b) {
  // return the sum of a and b ----> Vollkommen unnötiger Kommentar
  return a + b;
}
```

- Viel sinnvoller:

```
String text = "TheQuickBrownFoxJumpsOverTheLazyDog.";
char chars[] = text.toCharArray();
String reversedString = "";
// invertiere String text
for(int i = chars.length-1; i >=0;i--){
  reversedString = reversedString + chars[i];
}
```

Clean Code & Coding Conventions

Docstrings

- Werden immer vor Methoden geschrieben und Beschreiben die Methode, Parameter und Rückgabewerte

```
/**
* Berechnet die Quersumme der übergebenen Ganzzahl.
* Gibt -1 Zurück, wenn die Zahl negativ ist.
* @param eingabe Zahl von der die Quersumme berechnet werden soll
* @return Quersumme der Zahl oder -1
*/
public int calcQuerSumme(int eingabe) {...}
```

Clean Code & Coding Conventions

Faustregel

Haltet euch an die Coding Conventions, die in der Vorlesung besprochen wurden!
Das heißt:

- Nutzt Kommentare
- Schreibt Docstrings
- Nutzt Aussagekräftige Bezeichner
- Achtet auf korrekte Kapselung
- Achtet auf korrekte Einrückung
- Klassen, Interfaces und Komplexe Datentypen $\rightarrow$ beginnt mit Großbuchstabe
- der Rest $\rightarrow$ beginnt mit Kleinbuchstabe

Enum

Enum

Wiederholung Enum

- = Aufzählungstypen
- Werden beim Kompilieren zu Klassen mit Konstanten übersetzt
- Können auch noch Attribute haben

```
public enum Getraenk {
  BIER(5.3), WEIN(12.0), TRAUBENMOST(3.0), WASSER(0);
  private final double alkoholGehalt;
  public Getraenk(double alkoholGehalt) {
    this.alkoholGehalt = alkoholGehalt;
  }
  public double getAlkoholGehalt() {
    return this.alkoholGehalt;
  }
  public double calcAbbauZeit(double getraenkMaenge,
                              double koerpergewicht) {
    double maengeAlkoholKg = getraenkMaenge * this.alkoholGehalt / 100;
    double abbauZeit = 9090.9 * maengeAlkoholKg / koerpergewicht; // Formel: 0,11g / (kg * h)
    return abbauZeit;
  }
}
```

Enum

Wiederholung Enum

```
public static void main(String[] args) {
  System.out.println( "Alkoholgehalt von Bier: " + Alkohol.BIER.getAlkoholGehalt() + " %" );
  System.out.println("Abbauzeit Bier: " + Alkohol.BIER.calcAbbauZeit( 0.5, 80 ) );
  System.out.println("Abbauzeit Wein: " Alkohol.WEIN.calcAbbauZeit( 0.2, 70 ) );
}
```

Übungsaufgaben

Übungsaufgaben

Klassenstruktur

1. Schreiben Sie eine Klassenstruktur für Lebewesen
2. Lebewesen haben alle eine artenbezeichnung (String) und ein *alter* (int). Dazu haben Lebewesen eine Funktion altern, die das alter um eins erhöht
3. Leiten Sie die Klassen Tiere, Pflanzen von Lebewesen ab
4. Tiere können auf einem Kontinent leben, erstellen Sie ein Enum für diese Kontinente. Kontinente haben die Attribute hemisphereSN und hemisphereOW, für den Sektor des Kontinents auf der Welt (bwps.: Australien = Sued+Ost) (kann auch ein Enum sein)
5. Pflanzen haben ein Attribut wirkungsgrad und implementieren das Interface Photosynthese mit der Methode doPhotosysnthese
6. Achten Sie auf korrekte Kapselung (public, private, protected) ihrer Attribute und Methoden und schreiben Sie Getter & Setter

Bitte Schickt mir eure Fragen bis spätestens nächste Woche **Mittwoch** (05.02.2020)

Fragen?