Objektorientierung II

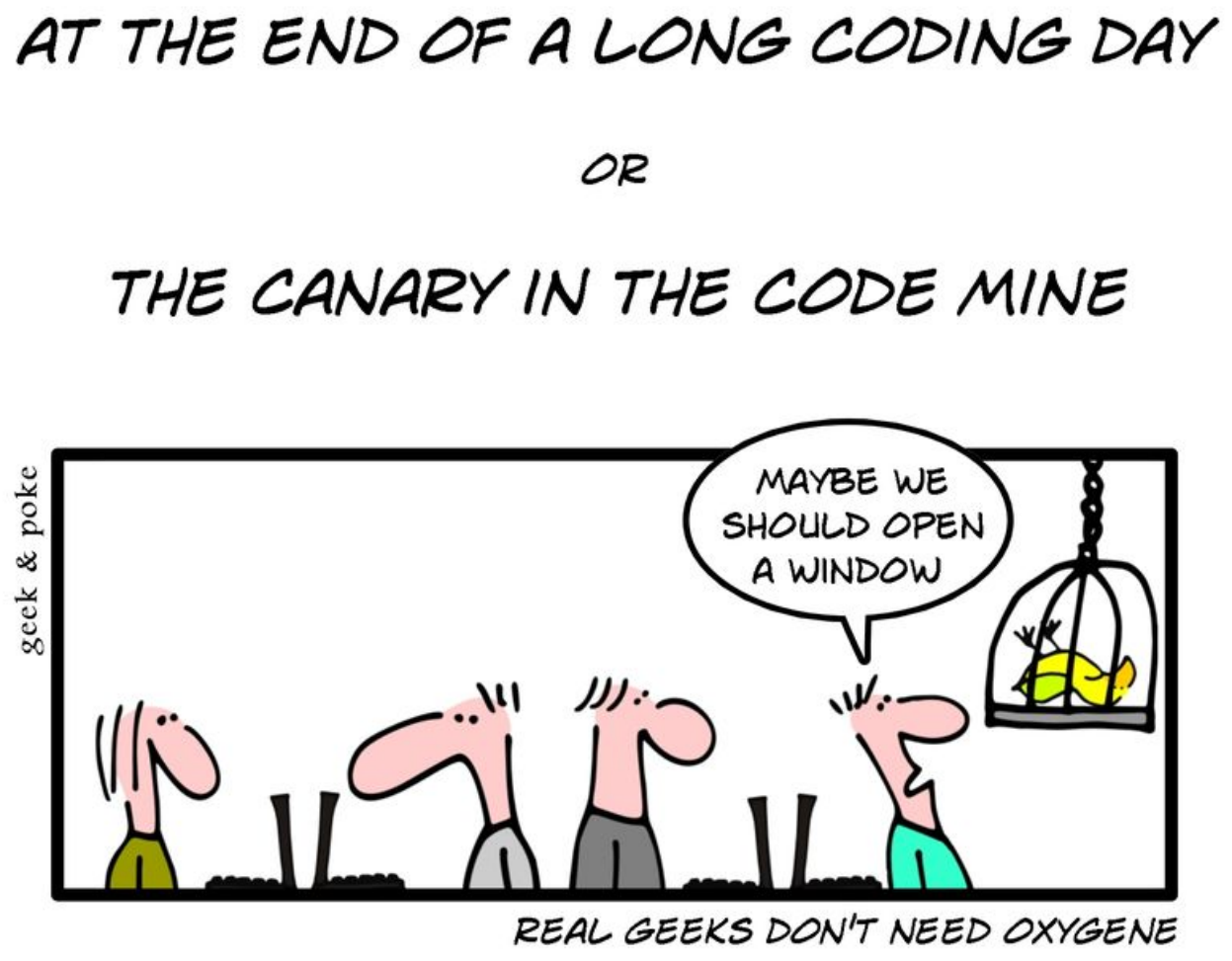
Javakurs 2010 - LE6

Sebastian Koch, Freitagsrunde

TU Berlin, 25. März 2010

Inhaltsübersicht

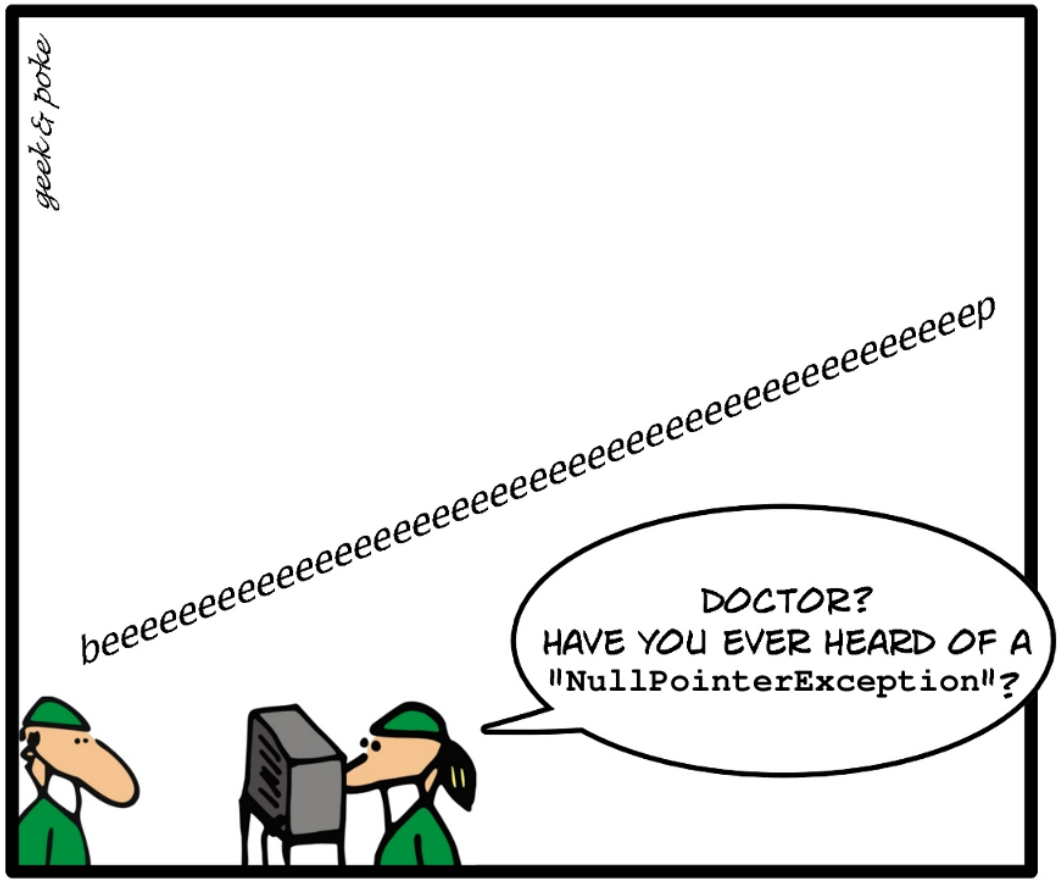
Ein Wort zur Übung . . .



[1]

[1] geekandpoke.typepad.com

Arbeit mit Objekten



RECENTLY IN THE OPERATING ROOM [1]

[1] geekandpoke.typepad.com

Vier Säulen der Objektorientierung

Motivation Objektorientierung II

®

[1]

≈ 2,5 Mio SLOC (source lines of code)

[1] mozilla.com/en-US/about/logo

Was sind Objekte?

Objekte ...

- sind Instanzen von Klassen
- sind Einheiten von Attributen und Methoden
- haben eine Identität, einen Zustand und ein Verhalten
- führen logisch zusammenhängenden Code zusammen
- ermöglichen auch komplexen Code einfach zu verstehen
- erlauben das einfache Wiederverwenden von Code

Ein Beispielobjekt?

```
1 class Human{
2     public String name;
3     public int age;
4
5     public Human(String name, int age){
6         this.name = name;
7         this.age  = age;
8     }
9 }
```

Kapselung

Was passiert bei folgendem Programmfragment?

```
1    Human lenny = new Human("L. L. Hofstadter", 27);
2    lenny.age = -12;
3
4    Human shelly = new Human("S. L. Cooper", 1337);
```

Wie kann man so etwas verhindern?

Eine gekapselte Klasse

```
1  class Human{
2      private String name; //Zugriff begrenzt
3      private int age;
4
5      public Human(String name, int age){
6          this.name = name;
7          this.setAge(age);}
8
9      public void setName(String name){
10         this.name = name;}
11
12     public void setAge(int age){
13         if (age >= 0 and ...){ //illegale Werte
14             this.age = age;}}  //verhindern
15
16     public void getName(){...}
17     public void getAge(){...}
18 }
```

Modifizierer - Arten und Verwendung

- Zugriff: public, protected, package(default), private
- static, final, abstract, ...
- Werden vor Klassen, Variablen, Methoden geschrieben und verändern sie

Zugriffskontrolle

```
1  public class Password{
2      private String password = "1234";
3
4      public void assign(String oldPW, String newPW){
5          if (password.equals(oldPW) && newPW != null){
6              this.password = newPW;
7              System.out.println("Passwort gesetzt.");
8          }
9          else
10             System.out.println("Passwort nicht gesetzt.");
11     }
12
13     public boolean check(String passwordToCheck){
14         return password.equals(passwordToCheck);
15     }
16 }
```

Was bedeutet static?

- Bindung einer Variablen oder Methode an eine Klasse
- Aufruf erfolgt dann über den Klassennamen (z. B. `Math.PI`)
- Sind nicht von einem Zustand abhängig
- Anwendungen: Konstanten, Zustandsunabhängige Methoden, Datenaustausch

Zustandsunabhängige Methoden

- `Math.max()` - Maximum zweier Zahlen
- `Math.sin()` - Berechnung des Sinus
- `Integer.parseInt()` - Konvertierung String nach Int
- ...

Datenaustausch

```
1  public class Human{
2      private String name;
3      private int age;
4      private int id;
5
6      private static int idcounter = 0;
7
8      public Human(String name, int age){
9          this.name = name;
10         this.age  = age;
11         this.id = Human.generateID();
12     }
13
14     private static int generateID(){
15         return idcounter++;
16     }
17
18     {...}
19 }
```

Standardmethoden

"Damit es gerecht zugeht, erhalten Sie alle die gleiche Prüfungsaufgabe: Klettern Sie auf diesen Baum!"

Die zwei Wichtigsten

- `public boolean equals(Object o)` - zum Vergleichen
- `public String toString()` - zum Ausgeben

Referenzvergleich

```
1   Human lenny = new Human("L. L. Hofstadter", 27);
2   Human shelly = new Human("S. L. Cooper", 28);
3   Human shelly2 = new Human("S. L. Cooper", 28);
4
5   System.out.println(lenny == shelly);
6   System.out.println(shelly == shelly2);
```

Was wird ausgegeben?

`false, false`

Vergleich per equals (Implementierung)

```
1  public class Human{
2      {...}
3      public boolean equals(Human otherHuman){
4          if (otherHuman == null){ //Sonst NullPointerException
5              return false;
6          }
7
8          return (this.name.equals(otherHuman.getName())
9                  && this.age == otherHuman.getAge());
10     }
11 }
```

Zustandsvergleich per equals (Beispiel)

```
1   Human lenny = new Human("L. L. Hofstadter", 27);
2   Human shelly = new Human("S. L. Cooper", 28);
3   Human shelly2 = new Human("S. L. Cooper", 28);
4
5   System.out.println(lenny.equals(shelly));
6   System.out.println(shelly.equals(shelly2));
```

Was wird ausgegeben?

`false, true`

Ausgabe von Objekten

```
1    Human lenny = new Human("L. L. Hofstadter", 27);
2    Human shelly = new Human("S. L. Cooper", 28);
3
4    System.out.println(lenny);
5    System.out.println(shelly);
```

Was wird ausgegeben?

`Human@9304b1, Human@190d11`

Ausgabe mit toString (Implementierung)

```
1 class Human{
2     {...}
3     public String toString(){
4         return "Name: "+this.name +", \t Alter: "
5         +this.age +", \t ID: "+this.id;
6     }
7 }
```

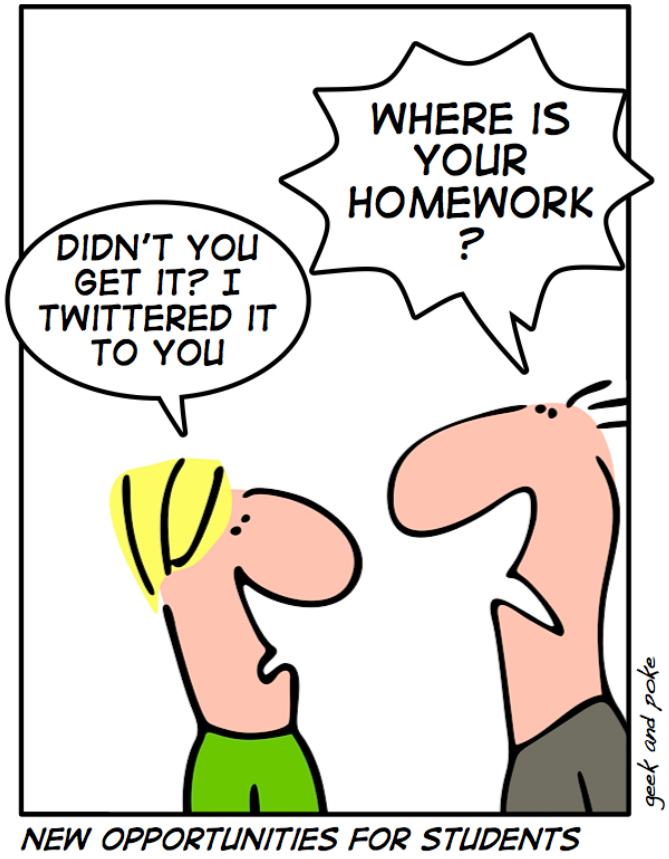
Ausgabe mit toString (Beispiel)

```
1    Human lenny = new Human("L. L. Hofstadter", 27);
2    Human shelly = new Human("S. L. Cooper", 28);
3
4    System.out.println(lenny);
5    System.out.println(shelly);
```

Was wird ausgegeben?

```
Name: L. L. Hofstadter,  Alter: 27,  ID: 0
Name: S. L. Cooper,  Alter: 28,  ID: 1
```

Fragenzeit



[1]

[1] geekandpoke.typepad.com

Was ist Vererbung?

Erklärung am Beispiel Shape

Vererbung - Ein Überblick

- Eine Klasse kann mittels Vererbung erweitert werden
- Die erbende Klasse erbt alle Attribute und Methoden
- Methoden können neu definiert werden und vorhandene überschreiben
- Mit `super.`*`funktionsname`*`()` kann eine Methode aus der Elternklasse aufgerufen werden
- Alle Klassen in Java erben von der Klasse `Object`
- "The object-oriented version of 'Spaghetti code' is, of course, 'Lasagna code'. (Too many layers)."

Abstrakte Klassen

- Eine abstrakte Klasse kann abstrakte Methoden und implementierte Methoden enthalten
- Von einer abstrakten Klasse kann keine Instanz erstellt werden
- Eine Subklasse muss alle abstrakten Methoden implementieren oder selbst abstrakt sein

Anwendung auf unser Beispiel Shape

Interfaces

- Ermöglichen die Mehrfachvererbung
- Von einem Interface kann keine Instanz erstellt werden
- Es können nur Methodenköpfe und Variablen vorgegeben werden
- Eine Subklasse muss alle durch das Interface vorgeschriebenen Methoden implementieren

Mehr dazu in der Übung ...

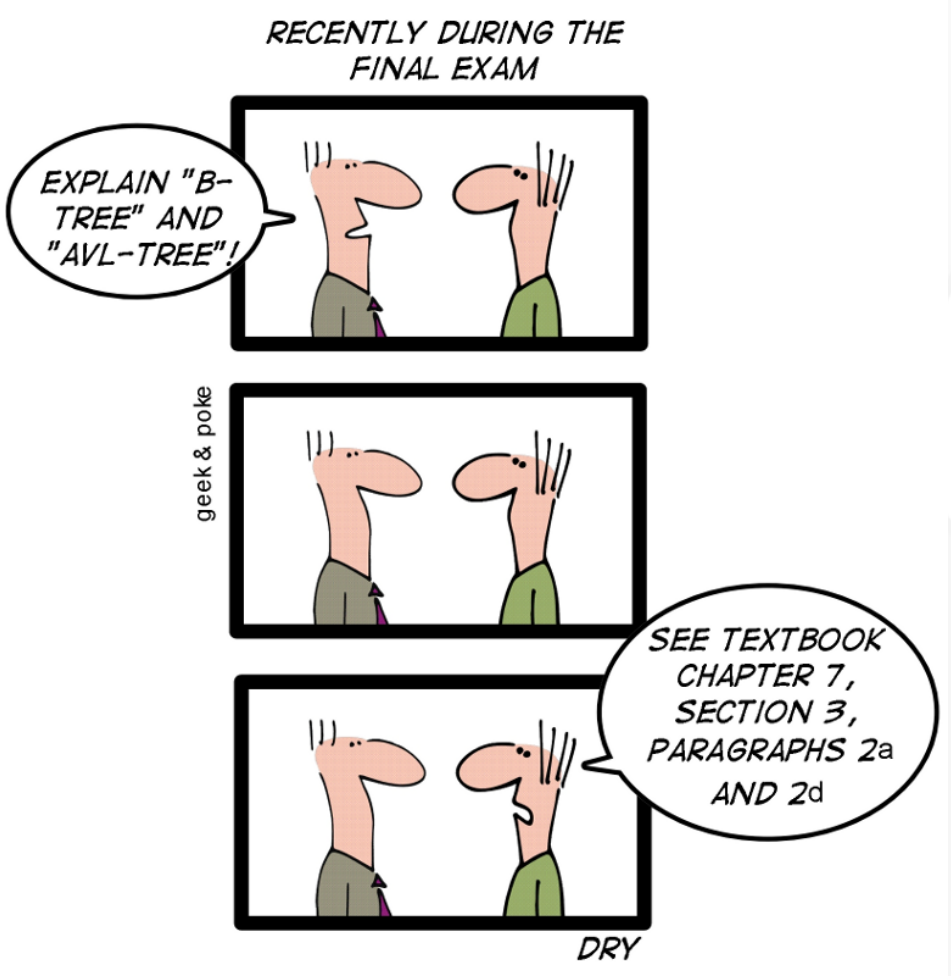
Modifizierer und Standardmethoden

- Modifizierer `public`, `protected`, `package`, `private` zur Zugriffsänderung
- Modifizierer `static`, `final`, `abstract` wofür?
- Anwendung vor der Klasse, Variable, Methode
- Standardmethoden: `equals` zum Vergleich und `toString` zur Ausgabe
- Werden von `Object` geerbt und überschrieben

Vererbung

- Ableitung neuer Klassen von Elternklassen
  - Erweiterung der Elternklasse durch die Kindklasse
  - Überschreiben von Methoden der Elternklasse
- Code kann wiederverwendet werden
- Als Zusatz können abstrakte Klassen und Interfaces benutzt werden

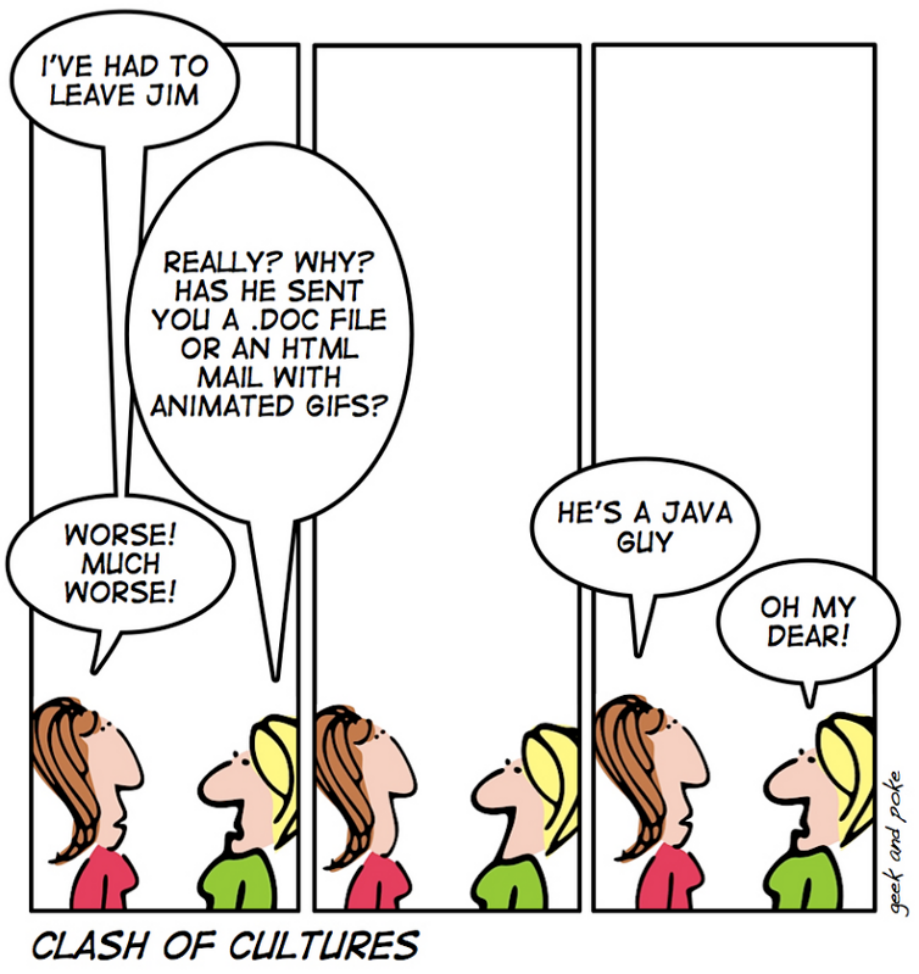
Klausurhinweis



[1]

[1] geekandpoke.typepad.com

Java?



[1]

[1] geekandpoke.typepad.com

Weitere Recherchemöglichkeiten

- Polymorphie
- Exceptions
- Generics
- Ein-/Ausgabe
- Collections, Iteratoren
- IDE: Eclipse, NetBeans

Referenzen

- http://java.sun.com/j2se/1.5.0/docs/api
- http://openbook.galileocomputing.de/javainsel8/
- http://www.javabuch.de
- http://geekandpoke.typepad.com
- http://www.mozilla.com/en-US/about/logo/