1. Einführung in die objekt-orientierte Programmierung

ARINKO

Objektorientierung ist:

- Logische Folge der fortschreitenden Abstraktion der programmierten Maschine durch Programmiersprachen.
- Vermeintliches Allheilmittel für Probleme der Softwarekrise.*

Softwarekrise (lt. Wikipedia):

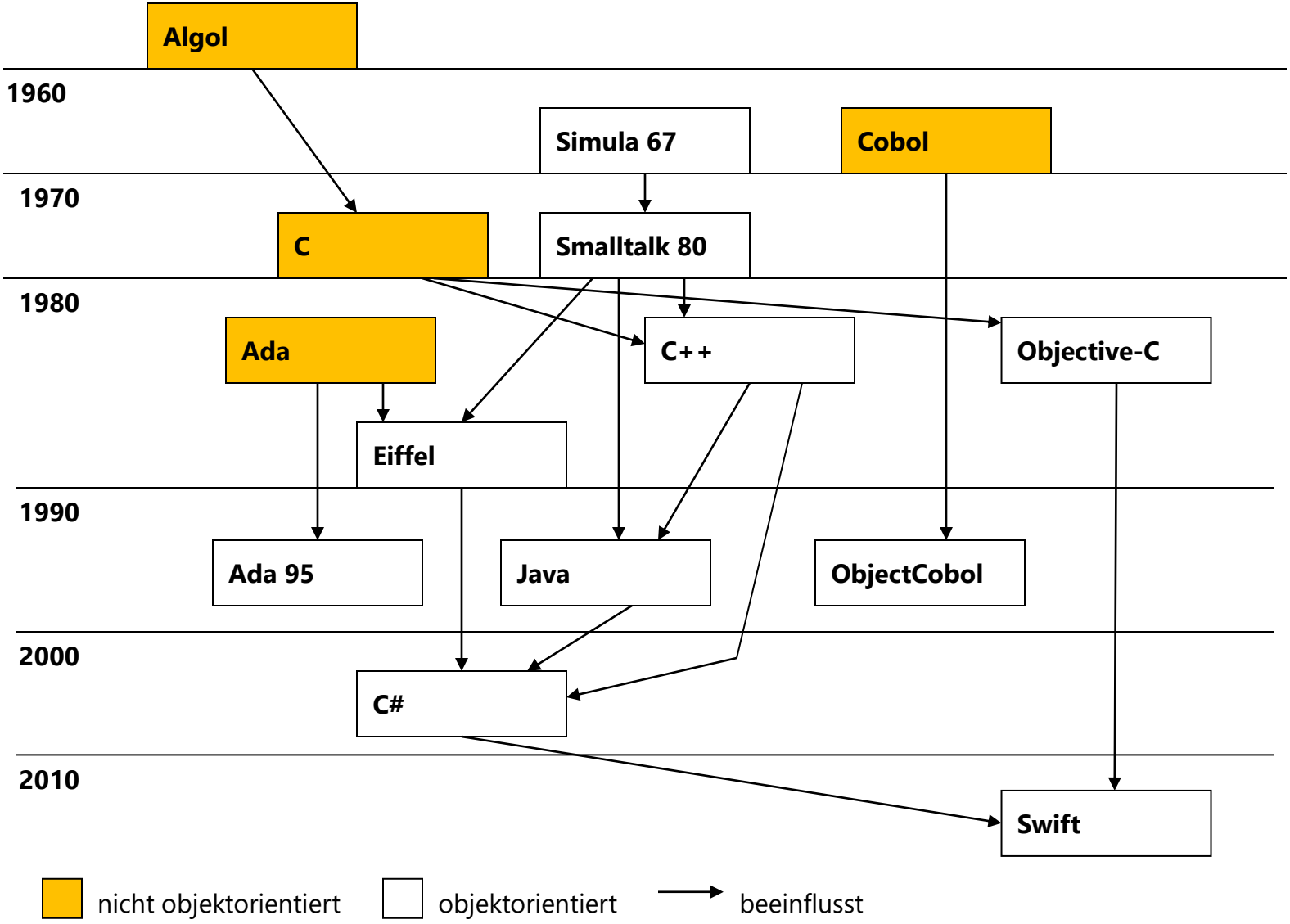
- "The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."
  (Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM)

The causes of the software crisis were linked to the overall complexity of hardware and the software development process. The crisis manifested itself in several ways:

- Projects running over-budget
- Projects running over-time
- Software was very inefficient
- Software was of low quality
- Software often did not meet requirements
- Projects were unmanageable and code difficult to maintain
- Software was never delivered

Fortschreitende Abstraktion bei Programmiersprachen

- Hardware → Assembler
  - Einzeldarstellung von Bits → Oktal, Dezimal, Hexadezimalcodes
  - binäre Maschinenbefehle → menschenlesbare Mnemonics
  - Speicheradressen → Bezeichner
- Assembler → Hochsprache
  - Maschinenoperation → Ausdruck
  - Daten & Berechnungsvorschriften → Parameter & Funktionen
  - Befehlssequenzen → Prozeduren
- Spezielle Operationen → Funktions- und Prozedurparameter
  - Funktionszeiger
- Objektorientierung



Quelle: http://www.wikipedia.de/ (stark vereinfacht)

Objekt

- „Ein Objekt besitzt einen Zustand, reagiert mit einem definierten Verhalten auf seine Umgebung und besitzt eine Objektidentität, die es von allen anderen Objekten unterscheidet. Jedes Objekt ist Exemplar einer Klasse."
  (Heide Balzert; Lehrbuch der Objektmodellierung; Spektrum Akad. Verl. 1999)

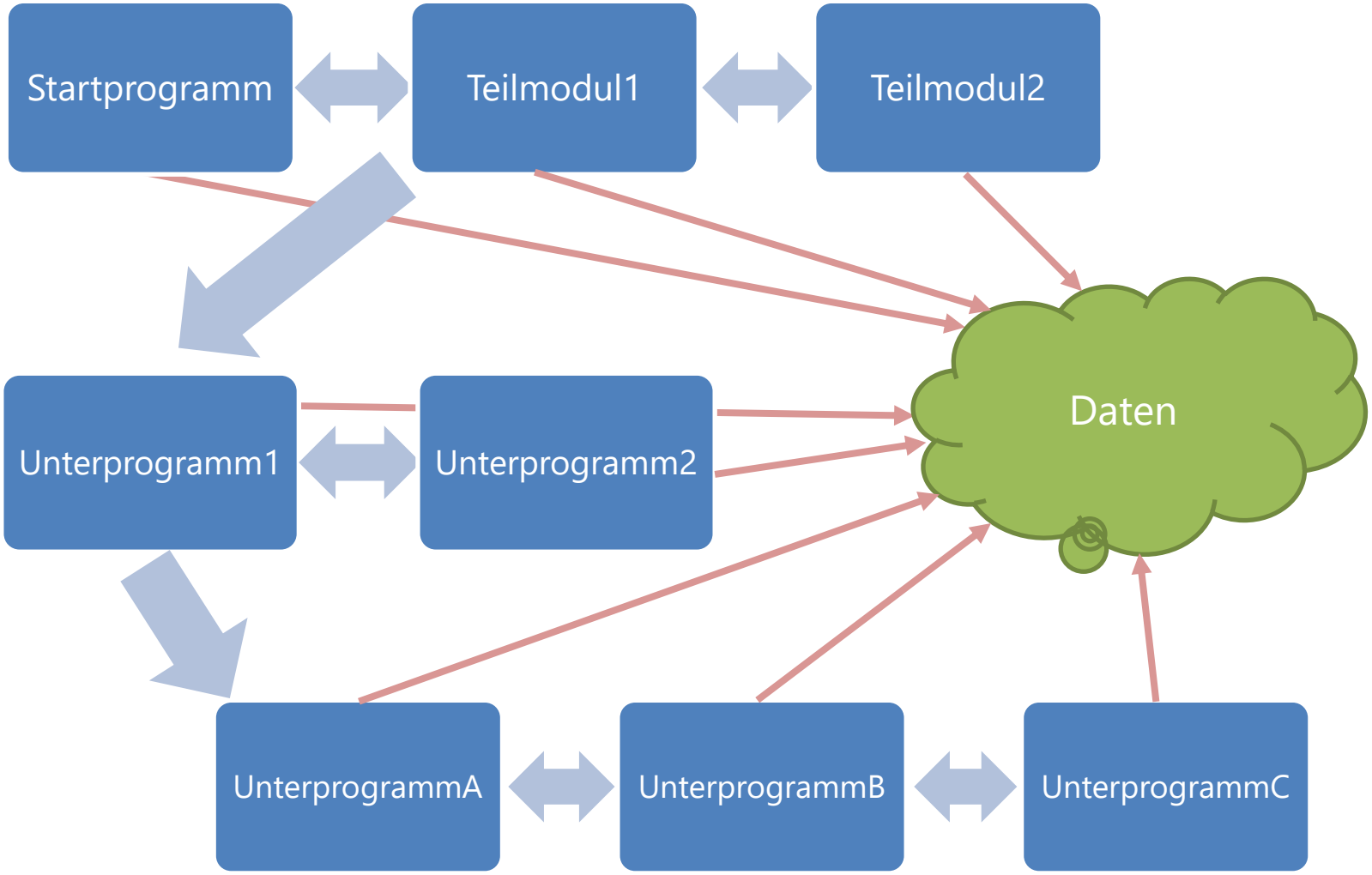
Klasse

- „Beschreibung von einem oder mehreren Objekten, die gemeinsame Attribute und Services (Methoden) aufweisen. Dazu gehört auch eine Definition, wie neue Objekte dieser Klasse erzeugt werden können."
  (Peter Coad, Edward Yourdon; Object-Oriented Analysis; 2nd Ed., Object International 1991)

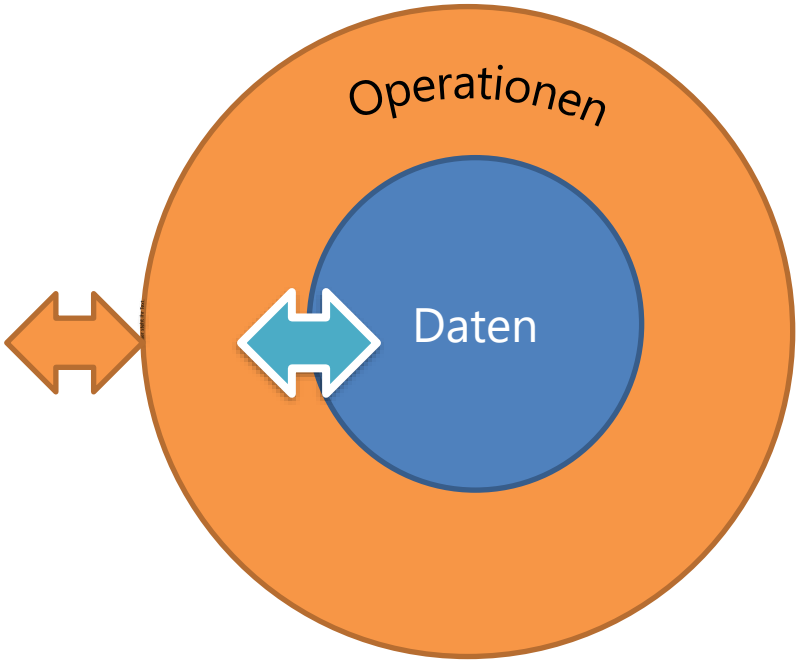
Vererbung

- Vererbung ist „die ausdrückliche Darstellung von Attribut- und Servicegemeinschaften (mehr als nur das Auslagern gemeinsamer Attribute und Services in einen Supertyp)."
  (Peter Coad, Edward Yourdon; Object-Oriented Analysis; 2nd Ed., Object International 1991)

	Objekte	Klassen	Vererbung	Polymorphie
Objektbasiert Ada, JavaScript	X			
Klassenbasiert CLU	X	X		
Objektorientiert Smalltalk, C++, Java, Ada95	X	X	X	X

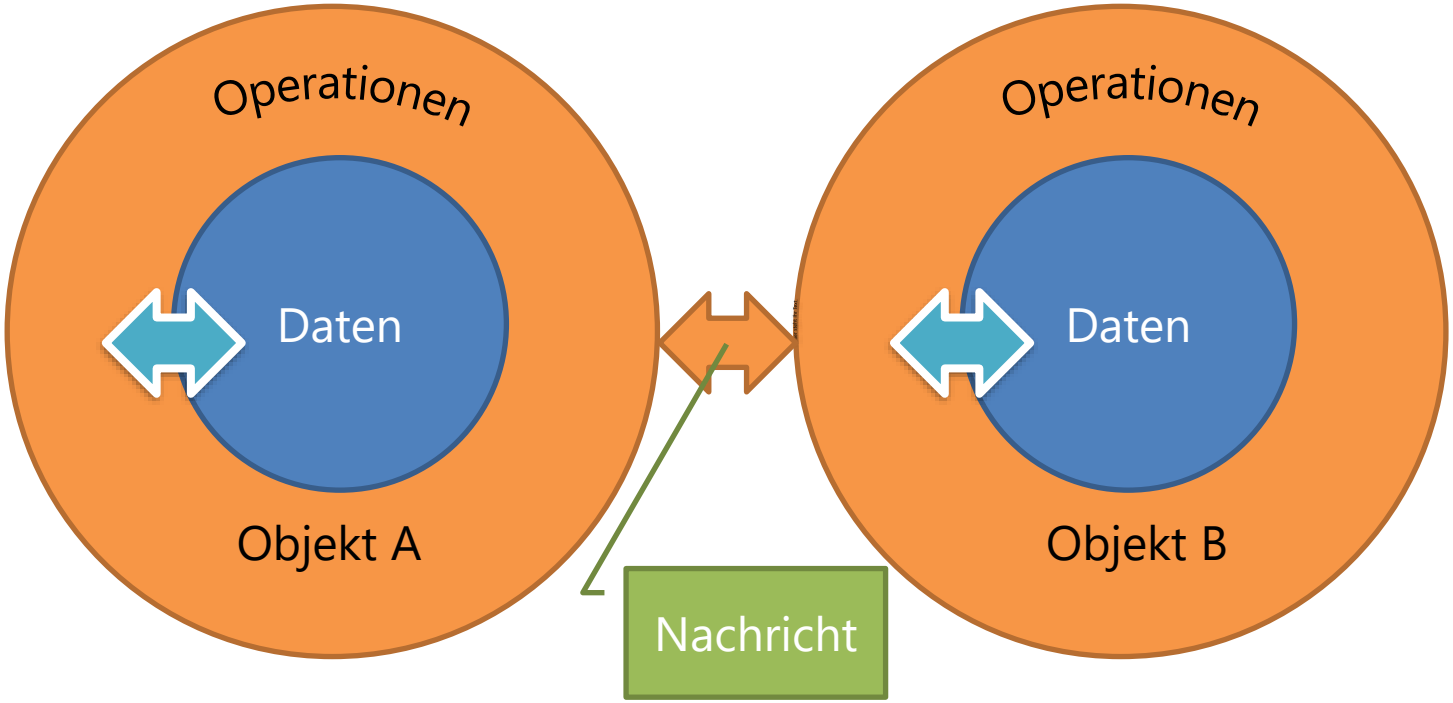

Startprogramm
Teilmodul1
Teilmodul2
Unterprogramm1
Unterprogramm2
Daten
UnterprogrammA
UnterprogrammB
UnterprogrammC

- Im Zentrum stehen Elemente, die eine Einheit darstellen aus Daten und Operationen.
- Der Zugriff auf die Daten erfolgt ausschließlich über die dafür vorgesehenen Operationen.
- Vorgehensweise:
  - Identifiziere die Elemente.
  - Identifiziere die Beziehungen zwischen den Elementen.



- In der Objektorientierung heißen die Elemente „Objekte“.
- Objekte repräsentieren Dinge einer fachlichen „Domain“ (dt. Anwendungsbereich), die Informationen in Form von Daten halten und deren Interaktion/Verhalten von Interesse ist.
- Beispiele von Objekten:
  - „greifbare“ (konkrete) Gegenstände wie Vertrag, Smartphone, Auto
  - „nicht greifbare“ (abstrakte) Dinge wie Konzern, Wahl, SMS
- Man erkennt, dass Objekte hauptsächlich durch Hauptworte charakterisiert werden.
- Objekte:
  - kapseln Daten und Operationen
  - haben einen Zustand und können diesen verändern
  - haben „Verhalten“
  - senden und empfangen Nachrichten
  - haben eine Identität

- Interaktion zwischen Objekten erfolgen mit Nachrichten.
- Ein Objekt A sendet eine Nachricht an Objekt B.
- Objekt B verarbeitet die Nachricht, indem es die passende Operation ausführt und ggf. eine Antwort zurückliefert.
- Objekt A wartet so lange, bis es die Antwort von Objekt B erhalten hat bzw. bis die Operation von Objekt B fertig ist.



Attribute (engl. „attributes“ oder gerne auch „fields“):

- Informationen, Daten eines Objekts
- können unveränderlich (konstant) oder variabel sein

Methoden (engl. „methods“):

- Operationen eines Objekts
- können den Objektzustand verändern oder in Abhängigkeit von diesem agieren
- können Nachrichten/Daten (= Parameter) von ihrem Aufrufer empfangen
- können Antworten (= Rückgabewerte) an diesen zurückliefern

- Objekte können in Programmiersprachen unterschiedlich kreiert werden.
  - Prototyping: man schafft sozusagen Fakten, indem man durch Zugriff auf Objekte definiert, was sie können. Neue Objekte werden als Kopien dieser Prototypen erzeugt (ein Ansatz, den JavaScript verfolgt)
  - Klassenbasiert: Objekte werden nicht einzeln entworfen und implementiert, sondern durch Verallgemeinerung in Klassen beschrieben (Ansatz von C++, C#, Java, Python…)

Klassenbasierter Ansatz:

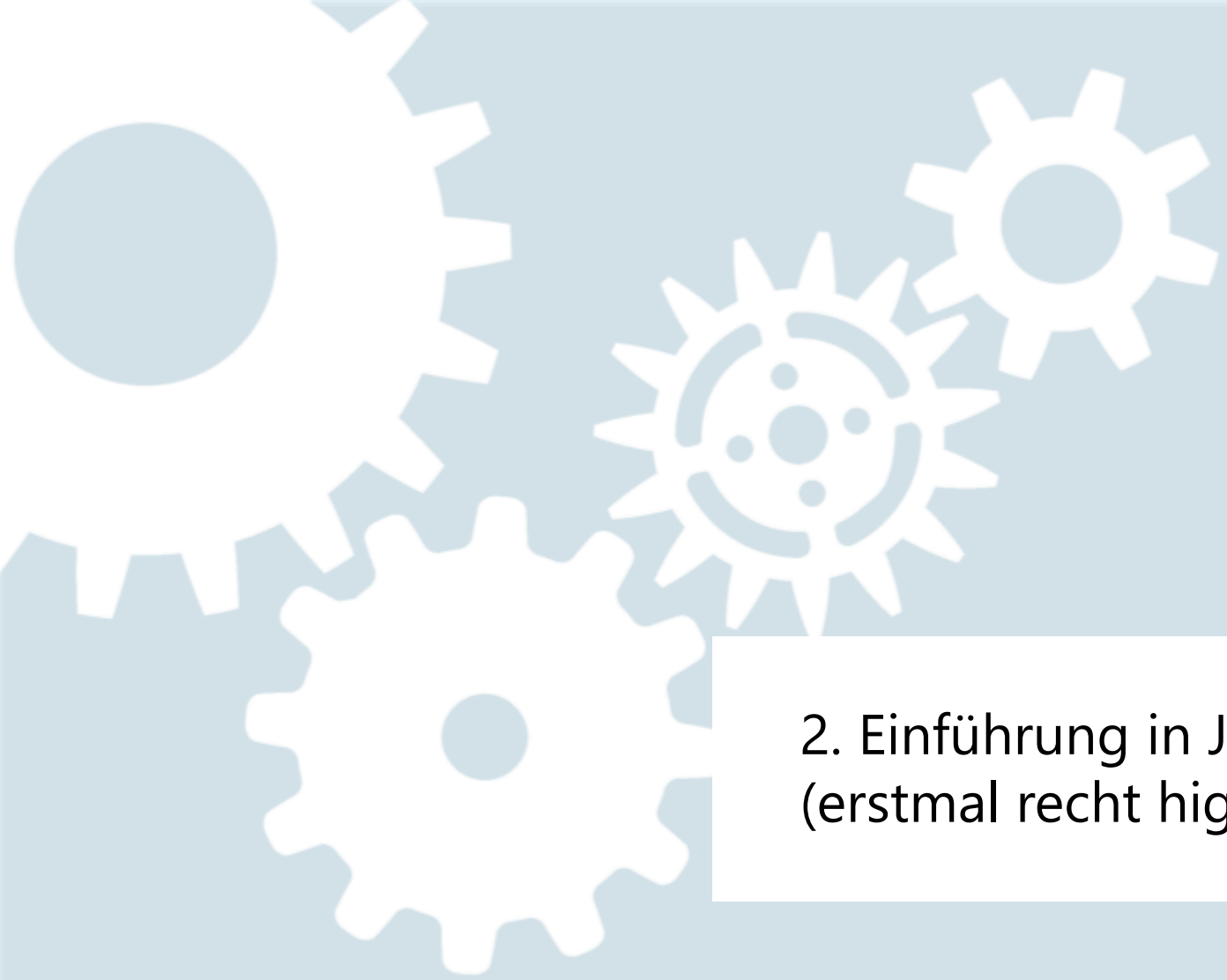
- Gleichartige Objekte werden identifiziert, deren gemeinsame Attribute und Methoden werden exemplarisch in einer „Klasse“ beschrieben
- Eine Klasse enthält also die Definitionen von Attributen und Methoden für zu erstellende Objekte.
- Objekte werden dann aus den passenden Klassen erzeugt. Eine Klasse ist somit eine Art „Bauvorschrift“ für Objekte.
- Die Java-Programmentwicklung besteht nahezu ausschließlich aus der Entwicklung von Klassen!

- Objekte entstehen aus einer Klasse durch „Instanziierung“
- Objekte können daher auch alternativ als „Instanz einer Klasse“ bezeichnet werden (im Deutschen ist der Begriff Instanz leider anderweitig [juristisch] belegt, was zu Missverständnissen führen kann)
- Objekte können nur erzeugt werden, wenn es eine passende Klasse gibt.
- Umkehrschluss: jedes Objekt ist Instanz einer Klasse.

- Klassen entstehen zur Entwicklungszeit (Programmierung)
  „Development-Time“
- Objekte entstehen zur Laufzeit
  „Runtime“

- Alle Objekte einer Klasse kennen dieselben Methoden
- Jedes Objekt einer Klasse unterscheidet sich in seinem Zustand von anderen Objekten seiner Klasse (unterschiedliche Attributsbelegung)

- Jede Klasse definiert einen eigenen „Typ“.
- Objekte einer Klasse haben alle denselben Typ, den der Klasse.
- Der Java-Compiler und das Laufzeitsystem überprüfen Typzugehörigkeiten und verweigern bei Unstimmigkeiten die Compilierung bzw. brechen mit einem Laufzeitfehler ab.
- Java ist (bis Java 7) eine streng-typisierte Programmiersprache.
- Ab Java 8 wird in den sog. Lambda-Ausdrücken eine relativ große Freiheit gewährt.



2. Einführung in Java (erstmal recht high-level)

ARINKO

1.0 1.0.2	1995	Initiale Version. Besonderes Highlight: Applets.
1.1	1997	Ergänzung um innere Klassen
1.2	1998	Aufnahme von Swing und dem Collection-Framework. Benennung der Plattform in „Java 2“
1.3	2000	Erweiterungen der Plattform
1.4	2002	Erweiterungen der Plattform, erstmals neues Keyword: „assert“
1.5 → 5.0	2004	Zahlreiche Veränderungen (Generics, Annotations, Enhanced-for, Varargs, Enums…). Umbenennung in „Java SE“
6	2006	Erweiterungen der Plattform, Aufnahme von WebServices
7	2011	Erweiterungen der Plattform. Java nun vollständig bei Oracle.
8	2014	Erweiterung hinsichtlich funktionaler Programmierung (Lambdas und Streams). Neue Date-/Time-API.
9	2017	Erweiterungen der Plattform. Modul-Konzept.
10+	2018ff.	Ab jetzt halbjährlich erscheinende Minor-Releases. Java 11, 17 und 21 haben einen sog. Long-Term-Support („LTS“)

Java 2

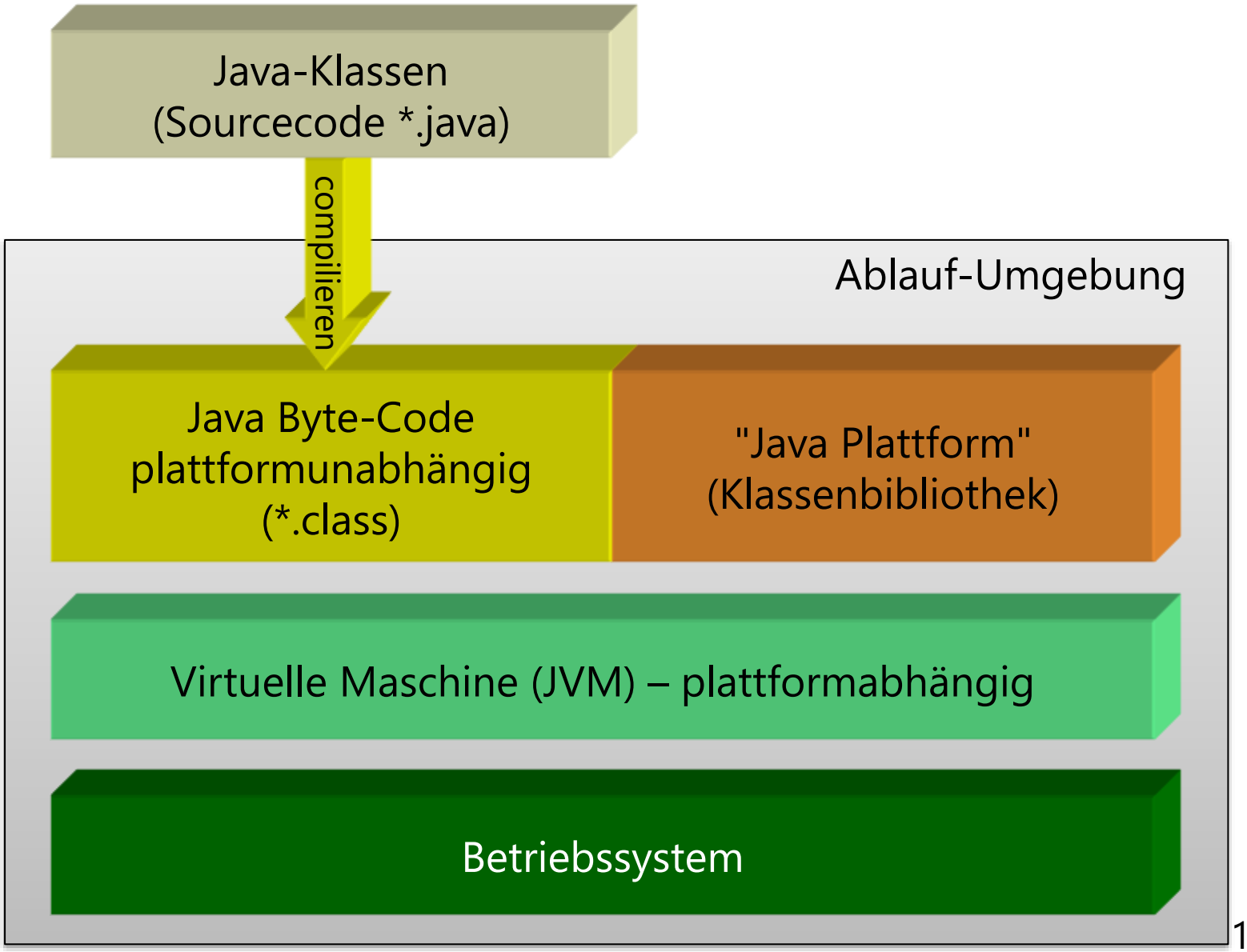
Java SE

- »Java: A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language.«
  Sun Microsystems
- Das Wesentliche an dieser Aussage ist, dass der Erfolg von Java damit zusammenhängt, weil alle diese Punkte – jeweils zumindest zufriedenstellend – erfüllt sind.
- Java bietet einen guten Mix aus den Programmiersprachen der ausgehenden 90er-Jahre, insbesondere C, C++, Smalltalk.

Java umfasst

- die Programmiersprache „Java“ (inkl. Tools)
- die virtuelle Maschine für den Ablauf von Java-Programmen („java virtual machine“, kurz „JVM“)
- die Java „Plattform“

- Java ist eine objektorientierte Programmiersprache.
- Sie verbindet Konzepte gängiger Programmiersprachen der 1990er-Jahre, v.a. Smalltalk und C++
- …hat aber bewusst nicht die Komplexität von C++
- …und verfügt natürlich auch über eigene Konzepte, die wiederum von anderen Programmiersprachen aufgenommen wurden.
- Beispiele von Konzepten aus Smalltalk:
  - Objektzugriff über Referenzen
  - Compilation in maschinenunabhängigen Code
  - Einfachvererbung
  - Garbage-Collection
- Beispiele von Konzepten aus C++:
  - Syntax der Anweisungen und Schlüsselwörter
  - Syntax der Klassenkonstruktion
  - Syntaktische Anleihen an den Umgang mit Attributen, Methoden und Objekten


Java-Klassen
(Sourcecode *.java)
compilieren
Ablauf-Umgebung
Java Byte-Code
plattformunabhängig
(*.class)
"Java Plattform"
(Klassenbibliothek)
Virtuelle Maschine (JVM) – plattformabhängig
Betriebssystem

- Die Java Plattform besteht aus zahlreichen vorgefertigten und mitgelieferten Klassen.
- Diese Klassen decken die unterschiedlichsten Bereiche ab und zeigen sehr gut, dass Java eine „multi purpose“ Sprache ist.
- Die Klassen sind in sog. Paketen (eine Art low-level Modul-Idee) organisiert.
- Beispiele für Funktionalitäten in den Paketen:

Applets	Remote Kommunikation	Sound
GUIs mit AWT	Web-Services	XML
GUIs mit Swing	Security	Mathematikfunktionen
Bildverarbeitung	Datenbankzugriffe	Internationalisierung
Printing	Threading	nativen Code anbinden
Komponentenkonzept	Logging	Meta-Programmierung
File-IO	Preferences	Streaming und Lambdas
Networking	ZIP-Dateien	Zeit und Datum

JRE

- steht für „Java Runtime Environment“
- Dieses enthält alles, was man braucht, um Java-Programme ablaufen zu lassen.
- D.h. im wesentlichen
  - die (komplette) Java-Plattform
  - die virtuelle Maschine

JDK

- steht für „Java Development Kit“
- Dieses wird u.a. von Oracle kostenlos zur Verfügung gestellt und umfasst u.a.
  - ein JRE
  - Den Compiler
  - Zahlreiche Utilities (JavaDoc, JAR-Builder, Monitoring-Tool etc.)
  - Optional erhältlich: Dokumentation in HTML

- Java gibt es in 2 bzw. 3 völlig unterschiedlich ausgebauten Editionen, für unterschiedliche Zielrichtungen.
- Java Standard Edition (Java SE)
  - Das normale Desktop-Java
  - Laufzeitumgebung ist eine virtuelle Maschine
- Java Enterprise Edition (Java EE, EE4J)
  - Spezifikation einer Umgebung für robuste Geschäftsanwendungen
  - Laufzeitumgebung ist ein Java-EE-Application-Server, der in Java implementiert ist und auf einer virtuellen Maschine abläuft
- Java Micro Edition (Java ME)
  - Idee einer Java SE für Kleinsysteme (Smartphones, Settop-Boxen, Router, embedded Devices etc.)
  - Inzwischen durch unterschiedlich paketierte Java SE Varianten („profiles“) verdrängt

3. Ein erstes Java-Programm (endlich)

ARINKO

Klassenkopf

```
public class Konto
{
    private String kontonummer;
    private double zinsen;
    private int kontostand;

    public Konto(String n, double z, int k)
    {
        kontonummer = n;
        zinsen = z;
        kontostand = k;
    }

    public String getKontonummer()
    {
        return kontonummer;
    }

    ...hier steht noch mehr... ausgelassen

    public void setKontostand(int stand)
    {
        kontostand = stand;
    }
}
```

Klassenrumpf

```
public class Konto
{
    private String kontonummer;
    private double zinsen;
    private int kontostand;

    public Konto(String n, double z, int k)
    {
        kontonummer = n;
        zinsen = z;
        kontostand = k;
    }

    public String getKontonummer()
    {
        return kontonummer;
    }

    ...hier steht noch mehr... ausgelassen

    public void setKontostand(int stand)
    {
        kontostand = stand;
    }
}
```

Attribute

- Attribute sind Variablen
- String ist eine Klasse der Java-Plattform
- int und double sind eingebaute Zahlentypen

Methoden

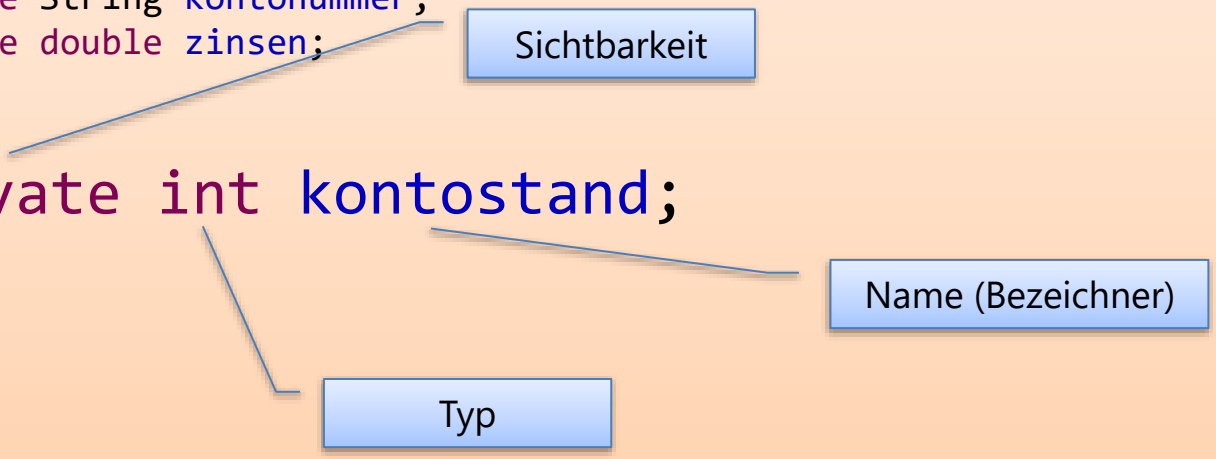
- Methoden können Resultate (hier vom Typ String) zurückliefern
- void-Methoden liefern nichts zurück
- Methoden können Parameter haben

```
public class Konto
{
    private String kontonummer;
    private double zinsen;

    private int kontostand;

    ...hier steht noch mehr... ausgelassen

}
```



```
public class Konto
{
    ...hier steht noch mehr... ausgelassen

    public String getKontonummer()
    {
        return kontonummer;
    }

    public void setKontostand(int stand)
    {
        kontostand = stand;
    }
}
```

Sichtbarkeit

Rückgabetyp

Name (Bezeichner)

kein Parameter

Rückgabe eines Wertes passend zum Rückgabetyp

```
public class Konto
{
    ...hier steht noch mehr... ausgelassen


    public String getKontonummer()
    {
        return kontonummer;
    }


    public void setKontostand(int stand)
    {
        kontostand = stand;
    }
}
```

keine Rückgabe

Name des Parameters

Parametertyp

Zuweisung des Parameters (stand) an ein Attribut (kontostand)

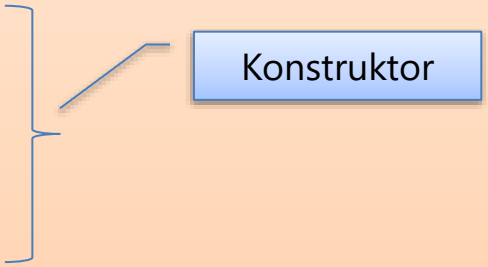
```
public class Konto
{
    private String kontonummer;
    private double zinsen;
    private int kontostand;

    public Konto(String n, double z, int k)
    {
        kontonummer = n;
        zinsen = z;
        kontostand = k;
    }

    public String getKontonummer()
    {
        return kontonummer;
    }

    ...hier steht noch mehr... ausgelassen

    public void setKontostand(int stand)
    {
        kontostand = stand;
    }
}
```



- Ein Konstruktor ist eine spezielle Methode, die bei der Erzeugung eines Objektes (automatisch) aufgerufen wird
- Hier initialisiert der Code des Konstruktors die Attribute auf Basis der Eingabeparameter

- In allen C-ähnlichen Sprachen, so auch in Java, ist der Startpunkt eines Programmes eine Funktion/Routine/Methode mit festem, vorgegebenen Format.
- Alle Methoden in Java müssen einer Klasse zugeordnet sein. Hier im Beispiel die Klasse `KontoTest`.

```
public class KontoTest
{

    public static void main(String[] args)
    {

        Konto konto1 = new Konto("DE68-0815-4711", 0.01, 0);

        System.out.print("Auf dem Konto mit Kontonummer ");
        System.out.print(konto1.getKontonummer());
        System.out.print(" befinden sich ");
        System.out.print(konto1.getKontostand());
        System.out.println(" EUR.");
    }
}
```

KontoTest – Erzeugung eines Objektes, Referenz

```
public class KontoTest
{
    public static void main(String[] args)
    {
        Konto konto1 = new Konto("DE68-0815-4711", 0.01, 0);

        System.out.print("Auf dem Konto mit Kontonummer ");
        System.out.print(konto1.getKontonummer());
        System.out.print(" befinden sich ");
        System.out.print(konto1.getKontostand());
        System.out.println(" EUR.");
    }
}
```

lokale Variable

Erzeugung eines Objektes

Ausgabe

- Der Operator new liefert einen Zeiger/Pointer auf neu erstellte Objekte.
- Zeiger/Pointer sind in Java nicht direkt manipulierbar (beispielsweise „Pointer-Arithmetik“ in C)
- Um die Unterscheidung zu klassischen Pointern aufzuzeigen, werden sie in Java **Referenz** genannt.
- konto1 „zeigt“ nach der Zuweisung auf das erstellte Objekt.

KontoTest – Zugriffe auf ein Objekt

```
public class KontoTest
{
    public static void main(String[] args)
    {

        Konto konto1 = new Konto("DE68-0815-4711", 0.01, 0);

        System.out.print("Auf dem Konto
        System.out.print(konto1.getKontonummer());
        System.out.print(" befinden sich ");
        System.out.print(konto1.getKontostand());

        System.out.println(" EUR.");
    }
}
```

lokale Variable

Zugriff auf eine Methode des Objektes

Der Rückgabewert der Methode getKontostand() wird als Eingangsparameter der Methode print() übergeben

```
public class KontoTest
{
    public static void main(String[] args)
    {
        Konto konto1 = new Konto("DE68-0815-4711", 0.01, 0);
        System.out.print("Auf dem Konto mit Kontonummer ");
        System.out.print(konto1.getKontonummer());
        System.out.print(" befinden sich ");
        System.out.print(konto1.getKontostand());
        System.out.println(" EUR.");

        Konto konto2 = new Konto("DE68-XXXX-0042", 0.005, 5000);
        double zinsbetrag = konto2.getKontostand() * konto2.getZinsen();
        System.out.println("Nächster monatlicher Zinsbetrag wäre " + zinsbetrag);

        konto2.setKontostand(5200);
        System.out.print("Auf dem Konto mit Kontonummer ");
        System.out.print(konto2.getKontonummer());
        System.out.print(" befinden sich jetzt ");
        System.out.print(konto2.getKontostand());
        System.out.println(" EUR.");
    }
}
```

Erzeugung eines neuen Objektes

Änderung

Ausgabe-Verkettung

- Das Java Development Kit (JDK) kommt u.a. mit folgenden Tools:
  - `javac` – der Java Compiler
  - `java` – die virtuelle Maschine
  - `javap` – ein Dissassembler für Bytecode
  - `javadoc` – ein Generator für Entwicklerdokumentation
  - `jvisualvm` bzw. `jconsole` – eine Monitoring-Anwendung zur Beobachtung der virtuellen Maschine
- Wichtig im Umgang:
  - `javac` arbeitet mit Dateien (`.java`)
  - `java` arbeitet mit Klassen und sucht sich selbsttätig die passenden Bytecode-Dateien (`.class`)
  - WO `java` genau nach Bytecode sucht, bestimmt der sog. Class-Search-Path (kurz: Classpath)

Compilieren und Ausführen auf der Kommandozeile

```
C:\Java\DHBW>dir /b
Konto.java
KontoTest.java

C:\Java\DHBW>javac KontoTest.java

C:\Java\DHBW>dir /b
Konto.class
Konto.java
KontoTest.class
KontoTest.java

C:\Java\DHBW>java KontoTest
Auf dem Konto mit Kontonummer DE68-0815-4711 befinden sich 0 EUR.
Nächster monatlicher Zinsbetrag wäre 25.0
Auf dem Konto mit Kontonummer DE68-XXXX-0042 befinden sich jetzt 5200
EUR.

C:\Java\DHBW>
```


Praxis
Theorie