Algorithmen und Datenstrukturen

Vorlesung #01 – Einführung in Java Teil 1

Benjamin Blankertz

Lehrstuhl für Neurotechnologie, TU Berlin

benjamin.blankertz@tu-berlin.de

18 · Apr · 2023

Themen der heutigen Vorlesung

- Informationen zu Java
- Überlick der Unterschiede C – Java
- Primitive Datentypen
- Arrays
- Objektorientierte Programmierung
- Klassenkonzept: Attribute und Methoden

Java

- Java wurde als Programmiersprache für das Internet entwickelt.
- Durchbruch durch Implementation im *Netscape Navigator 2.0* 1995.

Durch die Orientierung Richtung Internet ergeben sich folgende Anforderungen:

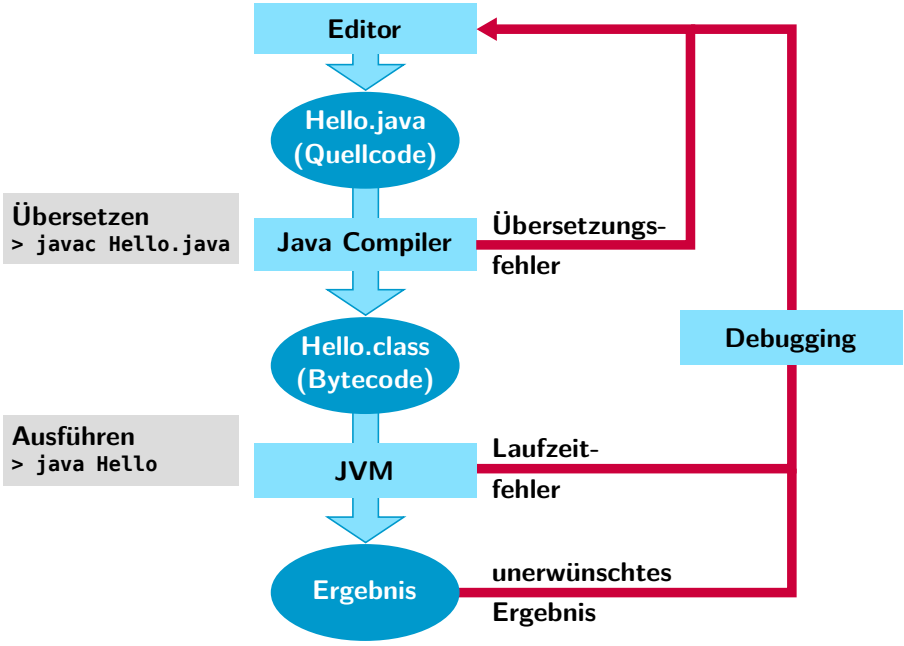
- Programmcode, der über das Internet empfangen wird, sollte keinen Schaden anrichten
- Dies schließt eine Adaption von C oder C++ aus (ungültige Zeiger können unkontrolliert Speicher beschreiben)

Vorzüge von Java

- Objektorientierung
- Das Konzept der *Java Virtual Machine* (JVM)
  - Java Byte Code
  - Unabhängig von Hardware und Betriebssystem
- Referenzen an Stelle von Zeigern (mehr Sicherheit)
- Gute Speicherverwaltung - automatische Freigabe
- *Open Source*

Zur Installation (Java IDE Intellij IDEA): Hinweise auf ISIS und in den Tutorien.

Ablauf des Programmierens in Java (ohne IDE)



Beispiele für Unterschiede C – Java

	C	Java
Ausführung	Compiler erzeugt Maschinencode	Byte Code, muss interpretiert werden
Array Deklaration	`int *x = malloc(N*sizeof(*x));`	`int[] x = new int[N];`
Array Größe	unbekannt für das Array	`x.length`
Datenstruktur definieren	`struct`	`class` - Klassen mit Methoden

Bibilotheksfunktionen nutzen	`#include "math.h"`
	`x = sqrt(3.14);`
	Funktionen sind global
		`x = Math.sqrt(3.14);`
		Funktionen haben *namespaces*

Speicher freigeben	`free`	automatische Speicherbereinigung
Variablen automatisch initialisiert	nicht garantiert	Instanzvariablen und Array Elemente initialisiert mit `0`, `null`, bzw. `false`
Variablennamen Konvention	`mean_square_error`	`meanSquareError`

Siehe die vollständigere Liste im Skript.

Primitive Datentypen

In Java gibt es die folgenden **primitiven Datentypen** (*primitive data types*):

Datentyp	Inhalt	Wertebereich
`boolean`	1 Bit Wahrheitswert (Größe undefiniert)	`true`, `false`
`char`	16 Bit Unicode	Unicode Characters
`byte`	vorzeichenbehaftete ganze Zahl in 8 Bit	$-2^7 \cdots 2^7-1 = 127$
`short`	vorzeichenbehaftete ganze Zahl in 16 Bit	$-2^{15} \cdots 2^{15}-1 = 32767$
`int`	vorzeichenbehaftete ganze Zahl in 32 Bit	$-2^{31} \cdots 2^{31}-1 = 2147483647$
`long`	vorzeichenbehaftete ganze Zahl in 64 Bit	$-2^{63} \cdots 2^{63}-1 = 9223372036854775807$
`float`	Gleitkommazahl in 32 Bit	$\pm\ 2^{127} \approx 10^{38}$, 7 signifikante Stellen
`double`	Gleitkommazahl in 64 Bit	$\pm\ 2^{1023} \approx 10^{308}$, 15 signifikante Stellen

Am häufigsten werden `boolean`, `int` und `double` verwendet, sowie der abstrakte Datentyp `String`.

Vorsicht bei der Wahl der Datentypen

- Datentypen sollten mit Bedacht gewählt werden, siehe
- YouTube *number of views* als `int` (Bereich bis 2.147.483.647)
- bis Dez. 2014!

YouTube ▸ Öffentlich 01.12.2014

We never thought a video would be watched in numbers greater than a 32-bit integer (=2,147,483,647 views), but that was before we met PSY. "Gangnam Style" has been viewed so many times we had to upgrade to a 64-bit integer (9,223,372,036,854,775,808)!

Hover over the counter in PSY's video to see a little math magic and stay tuned for bigger and bigger numbers on YouTube.

Übersetzen

[https://web.archive.org/web/20190208205643/https://plus.google.com/+youtube/posts/BUXfdWqu86Q]

Arrays

Drei Schritte, um in Java ein Feld (*array*) oder Array anzulegen:

- Deklaration
- Speicher Reservierung
- Initialisierung

```
double[] x;                  // Variable als Array deklarieren
x = new double[K];           // und erzeugen (Speicher reservieren)
for (int k = 0; k < K; k++)
  x[k] = 0.0;                // initialisieren
```

Alle drei Schritte können in einer Zeile erledigt werden (automatische Initialisierung von Java):

```
double[] x = new double[K];
```

Deklaration mit Initialisierung durch eine Werteliste.

```
int[] fibo = { 0, 1, 1, 2, 3, 5, 8 };
```

Mehrdimensionale Arrays

- In Java gibt es keine 'echten' mehrdimensionalen Array.
- Man bildet Arrays von Arrays (von Arrays ...)
- Dies hat viele wichtige Konsequenzen, z. B. beim Kopieren und Vergleichen.

```
// Deklaration 2-dim Array:
int[][] x;
// Erzeugen eines 2-dim Arrays
x = new int[4][3];
for (int i = 0; i < 4; i++)
  for (int j = 0; j < 3; j++)
    x[i][j] = i - j;
```

Die Arrays müssen nicht rechteckig sein:

```
int[][] x;
x = new int[5][];
for (int i = 0; i < x.length; i++) {
  x[i] = new int[i+1];
  for (int j = 0; j < x[i].length; j++)
    x[i][j] = i - j;
```

Zwei Möglichkeiten der Deklaration mit einer Werteliste:

```
int[][] square = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
int[][] triangle = {{1}, {2, 3}, {4, 5, 6}};
```

Iterieren über Arrays

Um alle Elemente eines Arrays anzusprechen, kann man über die Indizes iterieren:

```
double[] folge = {12, -4, 5.6, 17};
double sum = 0.0;
for (int k = 0; k < folge.length; k++)
  sum += folge[k];
```

Oder man iteriert direkt über das Array: Diese Mechanimus wird in der nächsten Vorlesung besprochen (`Iterable`).

```
double[] folge = {12, -4, 5.6, 17};
double sum = 0.0;
for (double zahl : folge)
    sum += zahl;
```

Direkte Iteration über mehrdimensionale Arrays:

```
int[][] triangle = {{1}, {2, 3}, {4, 5, 6}};
int sum = 0;
for (int[] row : triangle)
    for (int element : row)
        sum += element;
```

Objektorientierte Programmierung (OOP)

Die Programmiersprache Java umfasst folgende Aspekte:

- imperative Prinzipen (Variablen, Operatoren, Fallunterscheidung, Schleifen, einfache statische Methoden, wie in C)
- allgemeine Objektorientierung (Objekte, Klassen, Vererbung, Schnittstellen)
- spezielle Erweiterungen von Java (Ausnahmen, Generics, Closures)

Das Konzept von Klassen und Objekten

Realität

reales Objekt

- hat Eigenschaften (Attribute)
- man kann etwas damit machen (Operationen)

⇓ **Modellierung / Abstraktion** (Programmierung)

Code

Klasse (fasst Objekte eines Typs zusammen)

- Definiert, welche Eigenschaften das Objekt besitzen kann (Attribute)
- Implementation der Operationen: Methoden

⇓ **Instanziierung** (beim Programmablauf)

Speicher

Objekt (Exemplar, Instanz der Klasse)

- Besitzt konkrete Eigenschaften (Attribute haben Werte)
- Anwendung einer Operation: Aufruf der Instanzmethode für das konkrete Objekt.

Klassen in Java

- Eine **Klasse** (*class*) definiert zusammengesetzte Datenstrukturen (wie **`struct`** in C) und die darauf zulässigen Operationen (objektbezogene **Methoden**).
- Jede Klasse besitzt mindestens einen **Konstruktor**, eine Methode mit dem Klassennamen, bei der kein Ausgabeargument angegeben wird, auch nicht `void`.
- Der Konstruktor wird aufgerufen, wenn man mit **`new`** ein Exemplar oder Instanz (*instance*) der Klasse erzeugt. Dies ist dann ein **Objekt**.
- Es kann mehrere Konstruktoren geben, wenn sie unterschiedliche Signaturen (Sequenz der Argumenttypen) haben.
- Dabei rufen häufig komplexere Konstruktoren die einfacheren Konstruktoren auf: **Verkettung von Konstruktoren**.
- Das Schlüsselwort **`this`** ist eine Referenz auf das Objekt, für das es aufgerufen wird. Mit `this(...)` kann ein anderer Konstruktor ausgerufen werden. Das muss aber immer die erste Codezeile im Konstruktor sein.
- Die Daten sind nach außen gekapselt: Der Zugriff ist nur über die festgelegten Methoden möglich, siehe Abstrakte Datentypen in der folgenden Vorlesung.

Kategorien von Variablen in Klassen

In Klassen gibt es drei Kategorien von Variablen:

- **Klassenvariablen** sind Datenfelder, die als `static` definiert werden. Sie existieren nur einziges mal pro Klasse und alle Objekte der Klasse können auf sie zugreifen.
- **Instanzvariablen** existieren unabhängig voneinander für jede Instanz der Klasse. Sie repräsentieren Eigenschaften bzw. den momentanen Zustand des konkreten Objektes.
- Lokale Variablen werden *innerhalb* von Methoden definiert und existieren nur für die Dauer des Methodenaufrufs.

Kategorien von Methoden in Klassen

In Klassen gibt es zwei Kategorien von Methoden:

- **Klassenmethoden**
  - beziehen sich auf kein konkretes Objekt der Klasse
  - können nur auf Klassenvariablen und nicht auf Instanzvariablen zugreifen.
  - (können Objekt der eigenen Klasse als Argument nehmen)
  - Deklaration durch **`static`**

  Aufruf über Klassennamen, z.B. `Math.log(17)`
- **Instanzmethoden** (oder Objektmethoden)
  - beziehen sich auf eine konkrete Instanz bzw. Objekt der Klasse
  - existieren unabhängig voneinander für jede Instanz der Klasse.
  - können auf private Eigenschaften dieses Objektes zugreifen.

  Aufruf über den Variablenamen,
  z.B. `str.length()` für eine Variable `str` der Klasse `String`.

Objekte

- Objekte sind Instanzen von Klassen; sie sind charakterisiert durch
  - **Identität:** Speicherbereich des Objektes
  - **Zustand:** Wert des Datentyps (Instanzvariablen)
  - **Verhalten:** Definiert durch die Objektmethoden
- Erzeugung durch **`new`** $\Rightarrow$ Konstruktoraufruf
- Konstruktor: reserviert Speicher, initialisiert Werte und gibt eine Referenz auf das Objekt zurück
- Objekte sind im Gegensatz zu primitiven Datentypen Referenztypen.
- Wenn `new` mehrfach aufgerufen wird, werden mehrere Objekte derselben Klasse erzeugt, jedes mit einer eigenen Identität.

Rückschau

Nach dieser Vorlesung sollten Sie

- die Geschichte von Java kennen, insbesondere
  das Konzept der JVM mit Java Byte Code
- wissen, wie man kleine Java Programm schreibt, kompiliert und startet
- die primitiven Datentypen von Java kennen
  und Arrays benutzen können
- das Konzept der objektorientierten Programmierung verstanden haben
- und den Unterschied von Klassen- und Instanzvariablen kennen.

Literatur

- Sedgewick R & Wayne K, **Introduction to Programming in Java: An Interdisciplinary Approach.** 2. Auflage, Addison-Wesley Professional, 2017. Onlinefassung: `https://introcs.cs.princeton.edu/java`
- Ullenboom C, **Java ist auch eine Insel.** 13. Auflage, Rheinwerk Computing, 2018. Onlinefassung: `http://openbook.rheinwerk-verlag.de/javainsel`
- Vornberger O, **Algorithmen. Skript zur Vorlesung im WS 2016/2017.** PDF und HTML unter `http://www-lehre.inf.uos.de/~ainf`

Spezielle Aspekte:

- `https://introcs.cs.princeton.edu/java`, hier insbesondere die Seite `.../faq/c2java.html`
- `https://wiki.byte-welt.net/wiki/Warum_Instanzvariablen_private_deklarieren`
- `https://de.wikibooks.org/wiki/Java_Standard:_Klassen`
- `https://javapapers.com/core-java/java-history`

Danksagung

Außer der angegeben Literatur basiert die Zusammenstellung des Materials dieses Kurses auf den Vorlesungen meiner Vorgängerinnen und Vorgänger Oliver Brock, Anja Feldmann und Marc Alexa. Vielen Dank für die Weitergabe!

Die Folien wurden mit LaTeX erstellt unter Verwendung vieler Pakete, u.a. beamer, listings, lstbackground, pgffor und colortbl sowie eine Vielzahl von Tipps auf `tex.stackexchange.com` und anderen Internetseiten.

Bildreferenzen

- Seite 7, Google+ Meldung zu YouTube *number of views overflow*: Screenshot von `https://plus.google.com/+YouTube/posts/BUXfdWqu86Q`
- Alle anderen Abbildungen wurden für diese Vorlesung erstellt.

Index