Technische Hochschule Ingolstadt
Fakultät Informatik

Programmierung 2

Java-Einführung

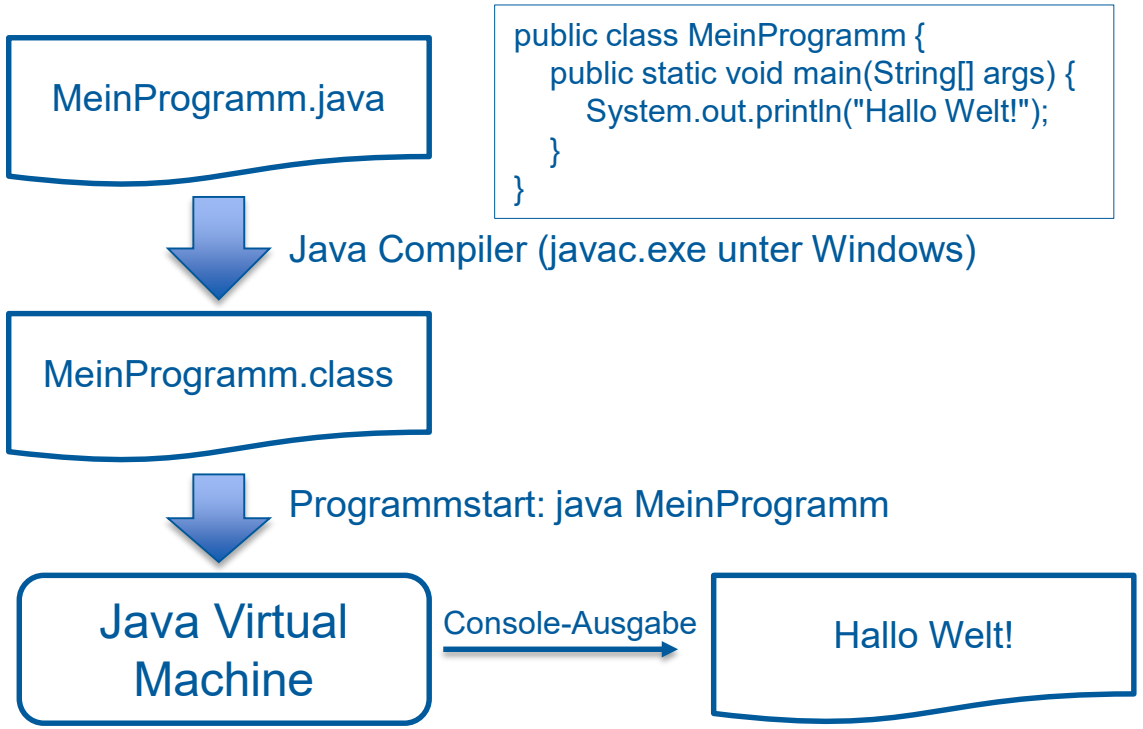
Prof. Dr. H.-M. Windisch

Themen

- Java und die Java Virtual Machine
- Hauptprogramm
- (Implizite) Typisierung
- Verschiedenes: Syntax von Anweisungen, Operatoren, Kommentare
- Funktionen
- Blöcke
- Kontrollfluss: if, while, for, do-while
- Ausgabe auf Console: wird mit den Beispielen erläutert
- Aufgabe: Umbau eines gegebenen Python Programms in ein Java-Programm

Java wurde 1995 von Sun als neue, objektorientierte Programmiersprache entwickelt. Die Sprache wird noch immer weiterentwickelt; der gegenwärtige Stand heißt Java 13.

Java-Compiler erzeugen keinen echten Maschinencode, sondern sogenannten virtuellen Bytecode, der zur Laufzeit von der Java Virtual Machine (JVM) interpretiert wird:



Vorteile des Bytecode-Konzeptes

- Bytecode ist sehr kompakt
- Bytecode ist portabel, nur die JVM muss an die jeweilige Plattform angepasst werden
- auch die Bibliotheken sind portabel, da sie ebenfalls in Bytecode vorliegen
- Java-Plattformen: PC, Handy, Smart Phone, Set-top Box, DVD-Spieler usw.

Nachteile des Bytecode-Konzeptes

- Interpretation von Bytecode ist langsamer als Ausführung von Maschinencode
- aber: neue Compilertechniken (z.B. Hot-Spot-Compiler) beschleunigen die Ausführung, heute keine Einschränkung mehr für die meisten Anwendungen!

Hauptprogramm

Ein Java-Programm besteht typischerweise aus mehreren Klassendefinitionen (siehe Kapitel 1).
Zum Start eines Java-Programms wird eine der Klassen genutzt, die eine main-Funktion besitzt (es kann beliebig viele Klassen mit einer main-Funktion geben).

Das einfachste Java-Programm ist nachfolgend abgebildet – es gibt „Hello World!“ auf die Console aus.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Anmerkungen:

- Es wird eine Klasse namens HelloWorld mit einer main-Funktion definiert.
- Beim Start des Programms wird automatisch die main-Funktion ausgeführt, d.h. es wird die println-Funktion aufgerufen, welche die Zeichenkette „Hello World!“ auf die Console ausgibt.
- Die Hintergründe für den Funktionsergebnistyp „public static void“ bzw. den „String[] args“-Parameter der Funktion werden später erläutert.
- Zur Ausführung muss die Klasse in einer Datei namens „HelloWorld.java“ definiert sein!

Java-Einführung

(Implizite) Typisierung

Python kennt keine expliziten Datentypen mit Ausnahme selbstdefinierter Klassen

Java schon!

Beispiel:

Python	Java
x = 1	int x = 1; oder int x; x = 1;

Primitive Datentypen in Java

Wofür	Python Datentyp	Java Datentyp(en)	**Java-Beispiel**
Zeichen	str	char (8 Bit)	char c = 'a‘;
Ganze Zahlen	int	int (32 oder 64 Bit), short(16 Bit), long (64 Bit)	long langeZahl = 1234567890999999999L;
Wahrheitswert	bool	boolean (Werte true bzw. false)	boolean aktiv = true;
Gleitpunktzahlen	float	float (single precision), double (double precision)	float ueberweisung = 1250.50f;
Strings	str	Klasse String	String begruessung = "Hallo!";

- Anweisungen werden in Java immer mit einem ";" beendet. Eine Zeile kann beliebig viele Anweisungen enthalten (aber nicht gut lesbar!).
- Anders als Python kennt Java Operatoren für Inkrement (+1) und Dekrement (-1) von Werten

  Beispiel: Python: `x += 1` Java: `int x = 5; x++;`
- Sowohl Inkrement als auch Dekrement gibt es in Präfix und Postfix-Form (analog mit "--"):
  - Präfix: `y = ++x; // x sei 2` => inkrementiere x und liefere den Wert von x: y -> 3, x -> 3
  - Postfix: `y = x++; // x sei 2` => liefere den Wert von x und inkrementiere x: y -> 2, x -> 3

- Java kennt **arithmetische Operatoren** analog zu Python, die Operatoren ** bzw. // gibt es aber nicht!
- Java kennt **logische Operatoren**, diese heißen aber anders:

Java	Python
||	or
&&	and
!	not

Java-Einführung

Verschiedenes (2)

- Java kennt **Vergleichsoperatoren** analog zu Python: ==, !=, <, <=, >, >=
- Java kennt keine Mehrfachzuweisung:

  Beispiel: a, b = 1, 2 in Java: int a = 1; int b = 2;
- Kommentare
  - Einzeilig bis zum Zeilenende: Python: # Java: //
  - Mehrzeilig
    - Python: """ … """ oder ' ' ' … ' ' '
    - Java: /* … */

Funktionen heißen in Java **Methoden**

Unterschiede am Beispiel:

Python	Java
def addiere(x, y):	int addiere(int x, int y) {
summe = x + y	int summe = x + y;
return summe	return summe;
	}

Bemerkungen

- Methoden müssen explizit typisiert werden: Ergebnistyp und Parametertypen
- Die Einrückung dient der Lesbarkeit, ist aber syntaktisch nicht relevant!
- Der Methodenrumpf wird durch einen Block { … } realisiert

Blöcke

Blöcke gibt es in Python und in Java

Unterschied:

- eine Anweisung (z.B. bei if) muss in Python durch Einrückung als Block gekennzeichnet werden
- allgemein: Python-Blöcke sind durch Einrückung gekennzeichnet, in Java werden Block-Klammern genutzt!
- Blöcke können lokale Variablen enthalten, deren Lebenszeit ist an die Blockausführung gebunden
- in Java: bei nur einer Anweisung kein Block nötig, aber möglich (wird auch empfohlen!)

Beispiel

```
Python          Java
if a < 5:       if (a < 5) {
    x = 1           x = 1;
    y = 2           y = 2;
else:           } else {
    x = 0           x = 0;
    y = 0           y = 0;
                    }
```

Java-Einführung

Kontrollfluss: if, while, for, do-while

if-Anweisung am Beispiel

Python
```
if a < 5:
    x = 1
    y = 2
elif a > 10:
    x = 3;
else:
    x = 0
    y = 0
```

Java
```
if (a < 5) {
    x = 1;
    y = 2;
} else if (a > 10) {
    x = 3;
} else {
    x = 0;
    y = 0;
}
```

for-Schleife am Beispiel

Python
```
for i in range(1, 11, 1):
    print(i)
```

Java
```
for (int i = 1; i < 11; ++i)
    System.out.println(i);
```

while-Schleife am Beispiel

Python
```
x, summe = 1, 0
while x < 10:
    summe += x
    x += 1

print(x, summe)
```

Java
```
int x = 1, summe = 0;
while (x < 10) {
    summe += x;
    x++; // oder x += 1;
                    }
System.out.println(x + " " + summe);
```

do-while-Schleife am Beispiel

Python
```
eingabe = ,‘
while eingabe != 'ende‘
    eingabe = input('Eingabe: ')
    print(eingabe)
```

Java
```
Scanner input = new Scanner(System.in);
do {
    String eingabe = scanner.next();
    System.out.println(eingabe);
while (!eingabe.equals("ende"));
```

foreach-Schleife am Beispiel

Python
```
stringliste = [ 'Hallo', 'Welt!‘]
for s in stringliste:
    print(s)
```

Java
```
String[] stringliste = new String[] { "Hallo", "Welt!" }; // stringliste als String-Array
for (String s : stringliste)
    System.out.println(s);
```

Java-Einführung

Aufgabe: Umbau eines gegebenen Python Programms in ein Java-Programm

Gegeben sei folgendes Python-Script. Schreiben Sie ein äquivalentes Java-Programm, nutzen Sie den vorgegebenen Code-Rahmen!

```
def summe(a, b, c):
    ''' Liefert die Summe von a, b und c '''
    return a + b + c

def ggT(x, y):
    ''' Liefert den grössten gemeinsamen Teiler von x und y '''
    while x > 0:
        if x < y:
            x, y = y, x
        x -= y
    return y

wert1, wert2, wert3 = 10, 20, 30
print('Summe: ', summe(wert1, wert2, wert3))
print('ggT(', wert1, ', ', wert2, ') =', ggT(wert1, wert2))
```

```
package python2java;

public class PythonScript {
    // hier ihre Methoden (=Funktionen in Python)



    public static void main(String[] args) {
        // hier ihr Hauptprogramm (= oberste Scriptebene)



    }
}
```

beachte:
Methoden müssen analog zu main als „public static“ definiert werden (wird später erklärt)!