Einführung in die Programmierung

Georg Ringwelski

Einf. Programmierung

Ziele der Lehrveranstaltung

- Programmieren können in einer imperativen Sprache (z.B. Java)
- Syntax und Semantik der OOP in Java
- Einige APIs von Java benutzen können

Nach 2. Sem: Niveau OCA Java SE Programmer

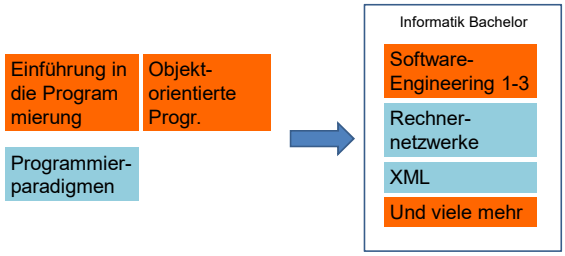
Einf. Programmierung

Erwartete (nicht erforderliche) Vorkenntnisse

- Umgang mit Computer, Betriebssystem, Dateien etc
- Umgang mit Internet
- Englische Texte verstehend lesen
- Selbstständiges Arbeiten bis zur Erreichung eines Zieles

Einf. Programmierung

Ausbildung, Objekt-Orientierte SWE bei IIb, IWb



Was ist Ihr Ziel?

Brief an mich selbst (5 Minuten):

Dieses Computerprogramm möchte ich mal selbst entwickeln?

Beförderungsdauer: ca. 9 Monate…

Organisatorisches

- Vorlesung: Fr 8:00 Uhr, GII 010
- Übung: Fr, 10:00 im GII 303 und Mi, 8.00 Uhr im GII 207
  - Präsentation der Lösungen, Lösen von Problemen mit den Aufgaben
- Prüfungsvorleistung VT für IIb und IWb: Jede Woche muss eine Aufgabe präsentiert werden, davon sind 80% (=11/13) für VT nötig, Beginn: KW 41 (erster Abgabetermin: 12.10.)
- Prüfungsleistung: Klausur (120 Minuten)
- Website: hszg.de/f-ei/gr/ …Lehre…
- Sprechstunde: auf Anfrage
- Fragen?

Literatur

- The Java Tutorial
  http://docs.oracle.com/javase/tutorial/
- T.Kröckertskothen: „Java, 1. Band“ RRZN, Uni Hannover
  - gebundene Version im Sekretariat für 5,20€
- Bücher , Amazon.de → 26.463 Treffer zu „Java“
- Diverse Java-Bücher in Bibliothek

Einführung in die Programmierung

1. Die Programmiersprache Java
2. Imperative Programmierung
3. Einführung Objektorientierte Programmierung

1. Die Programmiersprache Java

1. Was ist eine Programmiersprache?
2. Einsatzbereiche von Java
3. Java benutzen

Partnerinterview (5 Minuten):

- Welche Programmiersprachen kennst Du?
- Hast Du schon mal programmiert?
- Was hast Du schon programmiert?

Beantworten Sie gemeinsam folgende Frage:
„Was ist eine Programmiersprache?“

1.1 Was ist eine Programmiersprache?

1.1 Was ist eine Programmiersprache?

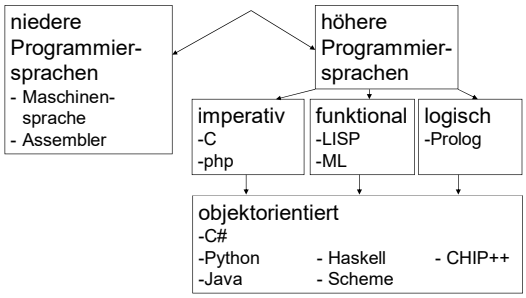
Eine Sprache zur Formulierung von Rechenvorschriften

(Algorithmen und Datenstrukturen)

featuring

1.1 Was ist eine Programmiersprache?

Abstraktion in Programmiersprachen



1. Die Programmiersprache Java

1. Was ist eine Programmiersprache?
2. Einsatzbereiche von Java
3. Java benutzen

1.2 Einsatzbereiche von Java

1. Java ist seit den 1990 Jahre in Entwicklung
2. läuft heute auf
   - ca. 1.1 Mrd Desktop Computern
   - ca. 2.3 Mrd Smartphones
   - Allen BlueRay Spielern
   - Uvm.
3. Native Android Apps werden in Java implementiert

1.2 Erfolgsfaktoren

- **objektorientiert:** derzeit das angewandte Programmierparadigma
- **verteilt:** viel Unterstützung für verteilte Systeme (Internetanwendungen, Apps…)
- **relativ sicher:** im Internet einsetzbar
- **portabel:** Programme laufen auf jeder Hardware mit JVM (Java Virtual Machine)

1. Die Programmiersprache Java

1. Was ist eine Programmiersprache?
2. Eigenschaften von Java
3. Java benutzen

1.3 Java benutzen

Installation

- Java SE (standard edition) von oracle (ehem. sun)
- frei verfügbar auf http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - JDK 10 : Die Entwicklungsumgebung für Programme (enthält Laufzeitumgebung JRE), plattformspezifisch
  - Dokumentation: API-docs als HTML
- Profis entwickeln mit IDE (eclipse, NetBeans, AndroidStudio, …)
- Anfänger mit Kommandozeile und Editor (zB JEdit)
  - JEdit ist ebenfalls frei verfügbar: http://jedit.org

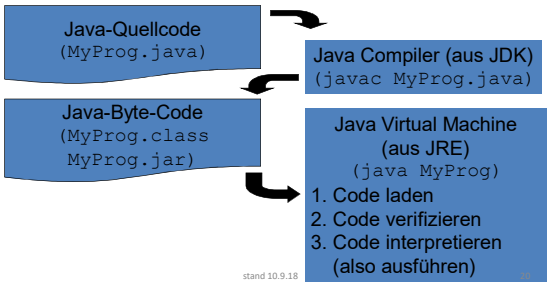
1.3 Java benutzen

Java JDK benutzen

1. Ggf. herunterladen
2. Ggf. installieren
3. **…java\jdk???\bin – Verzeichnis in „Umgebungsvariable“ PATH aufnehmen**
4. Kompilieren und ausführen in shell bzw. Eingabeaufforderung (cmd)

1.3 Java benutzen

Write once – run everywhere



1.3 Java benutzen

Alternativen

- Anstatt JEdit+shell: Integrated Development Environments (IDE)
  - Eclipse
  - NetBeans
  - IntelliJ
  - AndroidStudio
  - etc.
- Anstatt oracle JDK: OpenJDK, IBM Java und anderen

Einführung in die Programmierung

1. Die Programmiersprache Java
2. Imperative Programmierung
3. Einführung Objektorientierte Programmierung

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

2. 1 Hello World

Hello World in Java

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

     /* jetzt kommt die Ausgabe*/
     System.out.println("Hello World");

  }
}
```

Übungsaufgabe 1 (bis 10./12.10.)

a) Schreiben Sie ein Java-Programm, übersetzen es und führen es in Ihrer Arbeitsumgebung aus → HelloWorld.java

b) Schreiben Sie ein Java-Programm, das ein an der Kommandozeile eingegebenes Wort wieder ausgibt. Nennen Sie das Programm „Echo“.
   - Zum Beispiel soll auf eine Eingabe „java Echo Fritz“ die Ausgabe „Fritz“ folgen.
   - **Hinweis**: Auf das erste Argument der Kommandozeile hat man in Java mit der Variable „args[0]“ vom Typ String Zugriff. Auf das i-te Argument mit „args[i]“.

2. 1 Hello World

Formatierung von Java Programmen

- Trennzeichen sind: Leerzeichen, Tabulatoren, Zeilenumbrüche, Kommentare, Sequenzen von Trennzeichen

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

     /* jetzt kommt die Ausgabe*/
     System.out.println("Hello World");

  }
}
```


2. 1 Hello World

Formatierung von Java Programmen

- Anweisungen: Werden immer mit „ ; “ abgeschlossen

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

     /* jetzt kommt die Ausgabe*/
     System.out.println("Hello World");

  }
}
```


2. 1 Hello World

Formatierung von Java Programmen

- Blöcke beginnen mit „ { „ und enden mit „ } „ und fassen Anweisungen zusammen

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

     /* jetzt kommt die Ausgabe*/
     System.out.println("Hello World");

  }
}
```


2. 1 Hello World

Formatierung von Java Programmen

- Groß/Kleinschreibung ist relevant !!!
- Zeichensatz: vermeiden Sie Umlaute
- Kommentare:
  - beginnen mit /* und enden mit */
  - oder Rest der Zeile: //
- Konventionen:
  - Einrücken
  - camelCase

Übung im Plenum

Machen Sie daraus ein schönes Java Programm

```
//ein Programm, das einen Teiler berechnet
   class Teiler{public static void
   main(String[]a){int z1 =
   Integer.parseInt(a[0]);int z2 =
   Integer.parseInt(a[1]);
   System.out.println("res: "+GGT(z1,z2)); }
   int GGT(int a, int b){if(b==0) return
   a;return ggT(b,a%b);}}
```

Wiederholung

1. Welche Schritte sind notwendig, um vom Java Code zur Programmausführung zu kommen?
2. Wie funktioniert die Ein- und Ausgabe in Konsolenprogrammen mit Java?
3. Welche Schrift- und Formatierungskonventionen gibt es in Java?

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

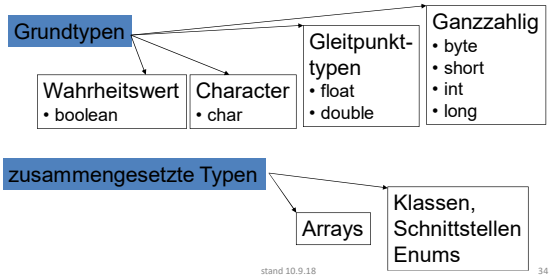
2.2 Typen, Variablen, Grundoperationen

Typen in der Programmierung

- Typen sind die „Einheiten“ von Daten im Computer
- Typen legen Speicherbereiche für Daten fest (z.B. wie viele Bytes)
- Typisierung erleichtert das Programmieren (Syntaktisch überprüfbare Semantik)
- In typisierten Sprachen hat jeder Wert einen Typ

2.2 Typen, Variablen, Grundoperationen

Typen in Java



2.2 Typen, Variablen, Grundoperationen

Beispiele für Typen

```
class Hello{
    public static void main( String[] a){
      long cnt = 0;
      char x = ´a´;
      boolean b = true;
      …
    }
}
```


2.2 Typen, Variablen, Grundoperationen

Variablen

- Variablen sind Bezeichner für Speicherzellen
- Sie sind Platzhalter für Werte
- Variablen haben
  - einen Namen
  - einen Typ
  - einen Wert (immer)
  - Einen definierten, endlichen(!) Wertebereich
  - (Sichtbarkeit, Änderbarkeit, Lebensdauer und andere Eigenschaften)

2.2 Typen, Variablen, Grundoperationen

Variablen in Java

- Jede Variable muss in Java **deklariert** werden (Variablendeklaration)
  - Ihr Name wird einem Speicherbereich zugeordnet
  - Der Speicherbereich wird durch den Garbage Collector verwaltet
- Deklarierte Variablen sollten **definiert** werden (Variablendefinition = Zuweisung)
  - Ihr wird ein Wert zugeordnet für den sie steht
  - Dieser Wert wird bei der Zuweisung berechnet und in der bezeichneten Speicherzelle abgespeichert.

2.2 Typen, Variablen, Grundoperationen

Beispiele für Variablendeklaration und -definition

```
class Hello{
    public static void main( String[] a){
      long cnt =0; int i =0;
      String s = a[0];
      …
    }
}
```


2.2 Typen, Variablen, Grundoperationen

Zuweisung von Variablen

- Variablen können ihren Wert während der Laufzeit verändern durch:

  <Variable> = <Ausdruck>;

- Links steht eine Variable
- Rechts ein auswertbarer Ausdruck
- Der Ausdruck rechts hat den gleichen oder einen allgemeineren Typ

2.2 Typen, Variablen, Grundoperationen

Beispiele Zuweisung von Variablen

```
int x;
x = 5;
-----------------------------------
int y =0; int z =3;
z = 3*(y+5);
y = y+4;
```


2.2 Typen, Variablen, Grundoperationen

Einzelübung

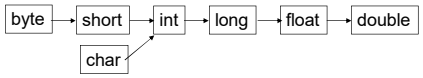
Schreiben Sie in Java ein „Teilprogramm“, das die Werte zweier int-Variablen austauscht.

```
int x = 2;
int y = 3;
…
// jetzt gilt: y=2 und x=3
```


2.2 Typen, Variablen, Grundoperationen

Typkonvertierungen in Java Grundtypen

- „kann-dargestellt-werden-als-Hierarchie“ auf den Grundtypen:



- automatische Konvertierung von Java in Pfeilrichtung bei Zuweisungen
- (sonst: explizites type-casting)

2.2 Typen, Variablen, Grundoperationen

Übung: welche Zuweisungen sind zulässig?

- int x = 5; short y = x;
- byte x = 5; long y = x
- char x = ‘ a‘; int y = x;

2.2 Typen, Variablen, Grundoperationen

Explizite Typumwandlung (casting)

- Typumwandlungen sind auch gegen die „kann-dargestellt-werden-als“-Hierarchie möglich
- Z.B. long x = 5; int y = (int)x;
- Syntax:
  <Variable> = (<Typ>) <auswertbarer Term>
- **Aber Vorsicht: Das kann zu Rechen- oder Laufzeitfehlern führen, wenn eine Umwandlung unmöglich ist**

2.2 Typen, Variablen, Grundoperationen

Übung, Demo am Rechner:

Sind diese Umwandlungen zulässig?
Sind sie überhaupt nötig?
Führen sie zu Laufzeitfehlern?
Welchen Wert hat jeweils y?

Wir probieren das aus…

- int x = 5; long y = (long)x;
- double x = 5.5; short y = (short)x;
- int x = 9999; byte y = (byte)x;
- char x = ‘ a‘; int y = x;
- int x = 5; String y = (String)x

2.2 Typen, Variablen, Grundoperationen

Initialisierung von Variablen in Java

- Initialisierung ist die „erste“ Definition einer Variable im Programm
- Die initialen Werte können auch aus Termen berechnet werden
- Variablen müssen in Java initialisiert werden (definite assignment: ggf. macht das der Compiler für Sie: byte, char, short, int, long: 0, float, double: 0.0)

2.2 Typen, Variablen, Grundoperationen

Konstante Werte in Java

- byte, short, int:
  - einfache Zahlen, zB 5
  - Binärzahlen, zB 0b101
  - Hexadezimal, zB 0xa2
- long: wie oben aber mit „L“ (zB 5L)
- double: mit Punkt, zB 0.31 (oder explizit: 1d)
- float wie double aber mit „f“, zB 0.31f
- char: mit einfachen Anführungszeichn, zB ‘a‘
- boolean: „true“ oder „false“
- Strings: doppelte Anführungszeichen zB “xx“

2.2 Typen, Variablen, Grundoperationen

Exkurs: Ausgabe von Variablen in Java

- Die Methode `System.out.print(X)` gibt den Wert der String-Variablen `X` in der Konsole aus
- Die Methode `System.out.println(X)` gibt zusätzlich eine neue Zeile aus
- Mit dem Operator „+“ können Ausgaben aneinandergehängt werden: System.out.print(“Hallo“+X)
- (dabei findet eine Umwandlung von X in String statt)

2.2 Typen, Variablen, Grundoperationen

Terme werden mit Grundoperationen gebildet

- Ein Ausdruck (=Term, expression) besteht aus Operanden und Operatoren
- Operanden sind Variablen, Konstanten oder Ausdrücke
- Operatoren haben Eingabetyp(en) und Ausgabetyp
- In gültigen Ausdrücken stimmen die Typen

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in Java

Verknüpfungen für alle Zahlen (byte bis double)

+ : Addition

- : Subtraktion

* : Multiplikation

/ : Division (bei ganzzahligen Typen: ganzzahlig)

% : Modulo, Rest bei ganzzahliger Division

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in Java

Typ des Ergebnisses der Verknüpfungen:

- immer einer der Typen der Eingabe
- der größere in der „kann-dargestellt-werden-als-Hierarchie“
- Das gilt auch bei der Division
  - zB 5/2 → 2 bzw. 5.0/2 → 2.5

2.2 Typen, Variablen, Grundoperationen

Übung im Plenum

Was ist Typ und Wert der Auswertung folgender Terme?

- 3/2
- 3/2L
- 3d/2
- 3d/(float)2

2.2 Typen, Variablen, Grundoperationen

Übung in Kleingruppen

a) Schreiben Sie ein Java Programm, das die Höhe und Breite eines Dreiecks einliest und dessen Flächeninhalt ausgibt. (=1/2*h*b)

b) Schreiben Sie ein Programm, das den Radius eines Kreises einliest und dessen Umfang (=2πr) und Fläche (=πr^2)ausgibt

Präsentieren Sie Ihre Lösung

2.2 Typen, Variablen, Grundoperationen

Übungsaufgabe 2

a) Schreiben Sie ein Java Programm, das an der Kommandozeile zwei Zahlen einliest. Diese beiden Zahlen werden dann in unterschiedlichen Variablen vom Typ byte, long, und double **kopiert**. Führen Sie dann die Division der Zahlen unterschiedlichen Typs aus und geben das Ergebnis aus.

Probieren Sie ihre Programm mit unterschiedlichen Werten aus: -9999999, -1, 0, 1, 999999. Beobachten Sie und versuchen Sie sich Überraschungen zu erklären.

b) Schreiben Sie ein Programm, dass für 1111/333 drei unterschiedliche Ergebnisse liefert

Wiederholung

- Welche Grundtypen gibt es in Java und welche Werte sind damit darstellbar?
- Wie werden Variablen deklariert und wie definiert?
- Auf welche zwei Arten können die Typen von Werten umgewandelt werden?
- Welcher Typ ergibt sich für Terme über Ausdrücken unterschiedlichen Typs?
- Welche Operatoren für Zahlen gibt es in Java?

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in Java

Unäre Operatoren für Zahlen

+ / - : Vorzeichenoperatoren (prefix)

++ / -- : bei prefix Prä-Inkrement/Dekrement

++ / -- : bei postfix: Post-Inkrement/Dekrement

2.2 Typen, Variablen, Grundoperationen

Übung: welchen Wert haben x und y?

a) int x = 3; int y = ++x + 3;
b) int x = 3; int y = x / 2;
c) int x =3; int y = -x ++;
d) int x = 3; int y = 10 % x--;
e) int x=3; int y = 3+ ++x

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in der Programmierung

- Operatoren sind meist überladen, also für mehrere Typen definiert
- Es gelten Rechenegeln aus der Mathematik
  - Punkt-vor-Strich
  - Assoziativität
  - Kommutativität
  - Distributivgesetz
  - Wenn keine Klammern da sind: von links nach rechts
- Unäre Operatoren binden (i.d.R.) stärker vgl. Präzedenzregeln im Java Tutorail

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in Java

Logische Operatoren (Eingabe-typen: boolean, Ausgabetyp: boolean)

! : nicht (unär, prefix)

&& : und (binär, infix)

|| : oder (binär, infix)

2.2 Typen, Variablen, Grundoperationen

Übung: Welchen Wert hat c?

- boolean a,b,c; a = true; b = true; c = a&&b;
- boolean a,b,c; a = false; b = true; c = a&&b;
- boolean a,b,c; a = false; b = true; c = a||b;
- boolean a,b,c; a = false; b = true; c = !a&&b;

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in Java

Bitoperatoren(Eingabe-typen: int)

~: alle bits „umdrehen“ (unär)

& | ^ : bitweise und/oder/xor (binär, infix)

<<, >> : bitshift links/rechts (unär)

<<<, >>>: bitshift mit 0 „nachfüllen) (unär)

2.2 Typen, Variablen, Grundoperationen

Grundoperationen in Java

Relationale Operatoren für Zahlen (Eingabe-typen: Zahlen, Ausgabetyp: boolean)

> bzw < : größer bzw kleiner

>= bzw <= : größer bzw kleiner oder gleich

== : gleich

!= : ungleich

(Das sind wirklich Operatoren und liefern ein Ergebnis!)

2.2 Typen, Variablen, Grundoperationen

Übung: Welchen Wert hat b?

- boolean b ; int x = 3; int y = 4; b = x<y;
- boolean b ; int x = 3; int y = 4; b = x == y;
- boolean b ; int x = 3; int y = 3; b = x<=y;

Exkurs

Vergleiche in Java

- einfache Datentypen (int …) kann man mit ==, <= etc vergleichen
- Zusammengesetzte idR Typen nicht!
- Der Vergleich von Objekten mit `o1 == o2` ist nur wahr, wenn es *das selbe* Objekt ist
- Der Vergleich `o1.equals(o2)` ist wahr, wenn es *das gleiche Objekt* ist
- **String** ist ein zusammengesetzter Typ, (nämlich eine Klasse), Strings sind Objekte dieser Klasse, ABER seit 1.6 für Konstanten ebenfalls mit == vergleichbar
- **Tipp: verwenden Sie lieber immer equals wenn Sie „das gleiche“ meinen**

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

2.3 Steueranweisungen

Anweisungen/Befehle in Programmiersparchen

- Anweisungen sind
  - einfache Anweisungen: `<Anweisung>;`
  - Blöcke: `{<Anweisung1>;…<AnweisungN>;}`
  - Konstrollstrukturen: Verzweigungen oder Schleifen (s.u.)
- In imperativen Programmen werden Anweisungen der Reihe nach ausgeführt
- Beispiele:
  - `int x;`
  - `x = x + 42;`
  - `System.out.println(x);`

2.3 Steueranweisungen

Kontrollstrukturen in Programmiersparchen

- Definieren unter welchen Bedingungen bestimmte Anweisungen ausgeführt werden
- Meist abhängig von Eingabe-Werten
- „Bedingungen“ sind Ausdrücke mit dem Ausgabetyp `boolean`

2.3 Steueranweisungen

Kontrollstrukturen in Java

Verzweigungen

```
if (<Bedingung>)
  <Anweisung>

if (<Bedingung>)
   <Anweisung>
else <Anweisung>
```


2.3 Steueranweisungen

Übung

Schreiben Sie ein Java-Programm das die größere von zwei eingegebenen Zahlen ausgibt

```
public static void main(String[] a){
   int x = Integer.parseInt(a[0]);
   int y = Integer.parseInt(a[1]);
   int z; // das groessere
   //TODO
   System.out.println(z+ „ist groesser“);
}
```


2.3 Steueranweisungen

Klausuraufgabe

(2 Punkte) Gegeben seien die Variablen **a,b,c** und **d** vom Typ **boolean.** Definieren Sie in Java: „ wenn **a** oder **b** aber nicht **c** gilt, dann soll „XX“ ausgegeben werden, ansonsten, falls **c** und **d** gilt soll „YY“ ausgegeben werden.

2.3 Steueranweisungen

Kontrollstrukturen in Java

Verzweigungen (forts.)

```
switch (<Ausdruck>) {
   case <konstante1> :<Anweisung1>
     …
   case <konstanteN> :<AnweisungN>
   default : <Anweisung>
}
```

`<Ausdruck>` ist vom Typ char, byte, short, int, String (seit v1.7)

2.3 Steueranweisungen

Kontrollstrukturen in Java

Verzweigungen (forts.)

```
switch (<Ausdruck>) {
   case <k1> : <A1> ; break;
     …
   case <kN> : <AN> ; break;
   default : <Anweisung>
}
```

2.3 Steueranweisungen

Kontrollstrukturen in Java

Schleifen

```
while( <Bedingung> )
   <Anweisung>
```

```
do
   <Anweisung>
while( <Bedingung> )
```


2.3 Steueranweisungen

Beispiel, while

```
int a=0;
while( a < 10 ){//irgendwann false
   System.out.println("Hallo"+a);
   a++;
}
```


Übungsaufgabe 3

b) Schreiben Sie ein Java Programm, das die Zeichenketten in der Kommandozeile einliest und zählt, wie oft das Wort „Hallo“ eingegeben wurde.

Hinweise:

- um in Java zwei Zeichenketten a und b zu vergleichen benutzt man den Ausdruck „a.equals(b)“, der wahr ist, wenn a und b gleich sind.
- Die Anzahl der Elemente eines Arrays, wie z.B. den Kommandozeilenparmetern „String[] args“ kann man mit dem Ausdruck „args.length“ abfragen

c) Schreiben Sie ein Programm, das Dreiecke ausgibt…

Wiederholung

- Welche Operatoren auf dem Datentyp boolean gibt es in Java?
- Wie werden in Java bedingte Anweisungen implementiert?

2.3 Steueranweisungen

Kontrollstrukturen in Java

Verzweigungen (forts.)

```
switch (<Ausdruck>) {
   case <konstante1> :<Anweisung1>
     …
   case <konstanteN> :<AnweisungN>
   default : <Anweisung>
}
```

`<Ausdruck>` ist vom Typ char, byte, short, int, String (seit v1.7)

2.3 Steueranweisungen

Kontrollstrukturen in Java

Verzweigungen (forts.)

```
switch (<Ausdruck>) {
   case <k1> : <A1> ; break;
     …
   case <kN> : <AN> ; break;
   default : <Anweisung>
}
```

Wiederholung

- Welche Operatoren auf dem Datentyp boolean gibt es in Java?
- Wie werden in Java bedingte Anweisungen implementiert?
- Wie werden in Java Wiederholungen von Anweisungen implementiert?

2.3 Steueranweisungen

Übung

Schreiben Sie ein Java Programm mit einer while-Schleife, das alle ungeraden Zahlen zwischen 1 und 1000 addiert

```
…
int summe = 0;
//TODO
System.out.println("Die Summe der ungeraden
   Zahlen von 1 bis 1000 ist "+summe);
```


2.3 Steueranweisungen

Kontrollstrukturen in Java

Schleifen (forts.)

```
for ( <Init>; <Bed.>; <Zähl-Anw.>)
   <Anweisung1>
```

```
for( <var-decl> : <collection>)
   <Anweisung>
```


2.3 Steueranweisungen

Beispiel

```
for( int a = 0; a < 10; a++){
    System.out.println("Hallo"+a);
}
```

Übung

Formulieren Sie das gleiche Programm mit while-Schleife

2.3 Steueranweisungen

Übung (Einzelarbeit)

a) Schreiben Sie ein Java Programm mit for-Schleife, das alle Zahlen zwischen 1 und 1000 ausgibt, die durch 37 ganzzahlig teilbar sind
b) Schreiben Sie das gleiche Programm mit einer while-Schleife.

2.3 Steueranweisungen

Murmelgruppe

Klausuraufgabe (leicht verändert)

- (20 Punkte) Schreiben ein Java Programm das prüft ob die eingegebene Zahl eine Primzahl ist. Primzahlen sind nur durch 1 und sich selbst ganzzahlig teilbar.
- Tipp: Es ist zu prüfen, ob es eine Zahl **y** zwischen 2 und **x**-1 gibt, so dass bei der Division **x/y** kein Rest bleibt.

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

2.4 Arrays

Arrays in Programmiersprachen

- Arrays fassen Werte gleichen Typs zu einer Einheit zusammen
- Es ergibt sich ein Array-Typ
- Es muss (i.d.R.) vorher definiert werden wie viele Werte (Dimension d. Arrays)
- Zugriff auf Einträge über Index (in konstanter Zeit)

2.4 Arrays

Arrays in Java

- Arrays sind über allen Typen erlaubt
- Deklaration: `<Typ>[] myArray;`
- `zB int[] x; String[] args;`

- Definition:
  `myArray =new <Typ>[<dim>];`
- `zB x = new int[10];`
  `int[] z = new int[20];`

2.4 Arrays

Arrays in Java

- Definition der Inhalte: `myArray[0] = x;`
- Lesen der Inhalte: `int x = y[1];`
- Zugriff auf Inhalte erst nach der Definition des Arrays möglich:

`String[] x; String y = x[2];` → *Fehler*

- Zugriff nur im definierten Bereich möglich:

`int[] x = new int[5] ; x[6]=0;` → *Fehler*

2.4 Arrays

Dimension von Arrays in Java

- Die tatsächliche Dimension eines arrays kann man mit der Anweisung `length` abfragen

Bsp:

```
int[] arr = x;
for(int i=0; i< arr.length; i++){…}
```

Oder auch:

```
for(int i : arr){...}
```


2.4 Arrays

Arrays, Übung gemeinsam

Schreiben Sie ein Java Programm, das alle Parameter der Kommandozeile in ein Array vom Typ int[] überträgt.

Sie können davon ausgehen, dass nur Zahlen angegeben werden

2.4 Arrays

Übungsaufgabe 4

- Array mit 87 zufälligen int-Werten zwischn 0 und 1000→ Recherchieren Sie, woher Sie Zufallszahlen bekommen
- Prüfen Sie, ob das funktioniert (kommt auch die 0 und die 1000 vor?) bevor Sie weitermachen
- Überlegen Sie sich Algorithmen, wie Sie min, max, mean und median herausfinden
- Definition Median: de.wikipedia.org/wiki/Median

2.4 Arrays

Arrays in Java

- Definition durch Aufzählung (nur in Kombination mit Deklaration, ohne Angabe der Dimension)

```
int[] werte = {1,2,3,4};
```

- Matrizen sind Arrays von Arrays

```
int[][] matrix = new int[3][2];
matrix[0][0] = x;
int[][] matrix2 = {{1,2},{3,4}}
```


2.4 Arrays

Exkurs: Kommandozeilen-Parameter in Java

- die Methode `main(String[] args){…}` wird bei Programmstart aufgerufen
- `args` ist eine Variable vom Typ Array von Zeichenketten
- Beim Aufruf können Werte in args geschrieben werden:

```
java myProg arg0 arg1 arg2
```


Wiederholung

1. Wie werden in Java Arrays deklariert?
2. Wie werden sie definiert?
3. Wie werden Arrayinhalte definiert?
4. Implementieren Sie eine for-Schleife, in der alle Elemente von int[] a in ein neues Array b kopiert werden.

Wer wünscht feedback zur Primzahlen-Klausuraufgabe?

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

2.5 Algorithmen

Was könnte das sein, „ein Algorithmus“

Welche Algorithmen haben Sie schon in dieser LV kennengelernt?

2.5 Algorithmen

Beispiele

- Tauschen der Werte zweier Variablen
- Alle Elemente eines Arrays traversieren
- Zähler
- Aufsummieren
- Minimum/Maximum finden
- Sortieren → sorting-algorithms.com
- Größte gemeinsame Teiler finden
- U.v.m.

2.5 Algorithmen

Jetzt ist alles klar:

Was ist also ein Algorithmus?

2.5 Algorithmen

Was ist ein Algorithmus?

Diverse „Definitionen“ im web, z.B.

- „…ein Verfahren zur Lösung eines Problems, das für eine Realisierung in Form eines Programms geeignet ist.“ [R.Sedgewick, Algorithms]
- „…eine eindeutige Handlungsvorschrift zur Lösung eines Problems[…]. A. bestehen aus endlich vielen wohldefinierten Einzelschritten“[de.wikipedia, 3.11.2015]

2.5 Algorithmen

Übung: Kleingruppe (8 min.)

Beschreiben Sie umgangssprachlich einen Algorithmus, der die Elemente eines Arrays „umdreht“

z.B. wird aus a = [1,2,3,4] durch den Algorithmus b = [4,3,2,1]

Definieren Sie dann diesen Algorithmus in Java

2.5 Algorithmen

Eigenschaften von Algorithmen

- **Terminieren**: Der Algorithmus endet nach endlich vielen Schritten
- **Determiniertheit**: Der Algorithmus liefert zu jeder Eingabe immer die selbe Ausgabe
- **Determinisums**: Der Algorithmus läuft immer gleich ab
- **Komplexität**: Die Anzahl der Befehle und Bedingungen im Algorithmus

2.5 Algorithmen

Im Plenum

Nennen Beispiele für einen Algorithmus, der…

- nicht terminiert
- nicht deterministisch ist
- nicht determiniert ist

2.5 Algorithmen

Zustand, Zusicherung, Zustandsübergang

- Der **Zustand** eines Programms ist die aktuelle Belegung all seiner Variablen
- Eine **Zusicherung** beschreibt den Zustand eines Programms
- Ein Zustandsübergang beschreibt die Veränderung von Zuständen durch Operationen

2.5 Algorithmen

Beispiel im Plenum

Schreiben Sie einen Algorithmus mit while-Schleife, der alle Elemente eines Arras addiert

Geben Sie nun zwischen allen Programmzeilen eine Zusicherung an

2.5 Algorithmen

Übung: Partnerinterview

3 Minuten (einzeln): Überlegen Sie sich einen Algorithmus, den Sie in ihrem Alltag anwenden

- Was ist der Anfangszustand?
- Was ist der Zielzustand?
- Welche Zwischenzustände gibt es?

2 Minuten (Interview) : erklären Sie Ihrem Partner den Algorithmus: Zustände und Zustandsübergänge, Terminierung, Determiniertheit

Übung 5

Bruchrechner für die Konsole

Eingabe: java Bruch add 1 2 3 4

Ausgabe: 1/2 + 3/4 = 10/8

- Für alle Grundrechenarten
- Erstmal ohne Kürzen
- Mit Fehlerbehandlung

„java Bruch div 1 2 0 4“ oder „java Bruch add 1 0 1 2“

Ausgabe: „Fehler, Division durch 0“

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

2.5 Prozeduren

Was ist eine Prozedur?

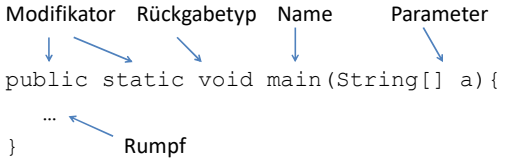
- Eine Zusammenfassung von Anweisungen
- Sie besteht in Java aus
  - Modifikatoren (z.B. `static`)
  - Rückgabe-Typ (`void` oder ein einfacher oder ein zusammengesetzter Typ)
  - Name
  - Parameter-Liste (Variablen-Deklarationen)
  - Rumpf (eine Block von Anweisungen)

Prozedurkopf

- Sie dient der Strukturierung von Programmen und Verhinderung von dupliziertem Code

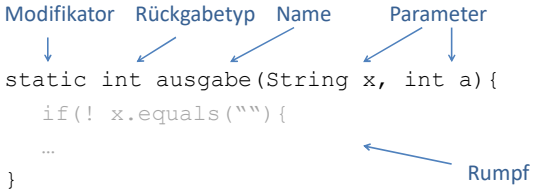
2.5 Prozeduren

Beispiele Prozedur



2.5 Prozeduren

Beispiele Prozedur



2.5 Prozeduren

Übung, 6 Gruppen:

Was sind die Typen der Eingabe (Parameter) und der Rückgabetyp folgender Prozeduren. Schreiben Sie Prozedurköpfe in Java

1. Addition ganzer Zahlen
2. Berechnen einer Quadratwurzel
3. Ausgabe von drei Zahlen
4. Berechnen einer Zufallszahl (0.0 - 1.0)
5. Zählen der Buchstaben in einer Zeichenkette
6. Umwandeln einer Zahl in eine Zeichenkette

2.5 Prozeduren

Der Modifikator `static`

- In Programmen ohne Objekte müssen alle Prozeduren diesen Modifikator haben
- (Also bei uns vorläufig alle)
- Mehr dazu in wenigen Wochen….

Wiederholung

1. Was ist ein Algorithmus?
2. Was bedeuten folgende Eigenschaften von Algorithmen:
   a) Terminierung
   b) Determinismus
   c) Determiniertheit
   d) Komplexität
3. Wie ist der Zustand eines Programms definiert?
4. Wie werden in Java Prozeduren deklariert?

2.5 Prozeduren

Rückgabe-Werte von Prozeduren

- Jede Prozedur kann ein Ergebnis liefern
- der **Typ** dieses Ergebnisses wird im Prozedur-Kopf deklariert
- der **Wert** wird mit dem Befehl `return` geliefert
- Prozeduren ohne Rückgabe haben den Rückgabetyp `void`

2.5 Prozeduren

Prozedur, Beispiele

```
static int gibDasGroessere( int a, int b){
   if(a > b) return a;
   if(b > a) return b;
   return -1;
}

static void ausgabe(int x){
   System.out.println("Ergebnis = "+x);
}
```


2.5 Prozeduren

Einzelübung (5 min), Kompetenztest

Klausuraufgabe

(5 Punkte) Implementieren Sie eine Prozedur, die den Mittelwert der Zahlen eines arrays berechnet.

Der Mittelwert der Zahlen x1, x2, … xn ist definiert durch (x1 + x2 + … + xn) / n

2.5 Prozeduren

Kontext von Prozeduren

- In einer Klasse können mehre Prozeduren stehen
- Sie stehen dann anderen Prozeduren zum Aufruf zur Verfügung

```
class X{
   public static void main(String[] a){
     ausgabe(„Erg: „+addiere(3,4));
   }
   static int addiere(int x, int y){…}
   static void ausgabe(String x){…}
}
```


2.5 Prozeduren

Aufruf von Prozeduren

- Ein Prozeduraufruf kann jederzeit erfolgen indem der Name der Prozedur als Anweisung verwendet wird
- Es müssen alle Parameter angegeben werden
- Der Rückgabewert sollte ausgelesen werden

2.5 Prozeduren

Beispiel

```
class K{
   public static void main(String[] a){
        int x = proc(2);  // Prozeduraufruf
        // x hat den Wert 7
   }
   static int proc(int a){
        return a+5;
   }
}
```


2.5 Prozeduren

Übung im Plenum

Implementieren Sie mit Hilfe Ihrer Prozedur „mittelwert“ ein Programm, das die Mittelwerte der Arrays [1,2,3,4] und [3,4,5,6] ausrechnet

Übungsaufgabe 6

Umstrukturierung des Bruchrechners ohne die Funktionalität zu verändern

- Alle Rechenoperationen in separate Prozeduren
- Fehlererkennung in separate Prozedur
- Kürzen in separate Prozedur
- Ergebnisausgabe in separate Prozedur

→Gleiche Funktion wie vorher

→Keine globalen Variablen

→Klare Struktur: Jede Prozedur macht genau eine „Sache“

2.5 Prozeduren

Lokale und Globale Variablen

- *Lokale Variablen* werden für Prozeduren, in Kopf oder Rumpf deklariert
- *Globale Variablen* für die gesamte Klasse/Objekt
- Namenskonflikte sind möglich
- es werden dann immer die Variablen verwendet, die im innersten umgebenden Block deklariert wurden

2.5 Prozeduren

Beispiel, lokale vs globale variablen

```
class K{
   int x = 1; int y;
   public static void main(String[] a){
        int b = 2;
        b = x * y; // Zugriff auf globale
   }
   // hier kein Zugriff auf lokale V.
   // x = b + 2; // geht nicht!!
}
```


2.5 Prozeduren

Beispiel, Namenskonflikt

```
class K{
   int x = 1; int y;
   public static void main(String[] a){
        int x = 2;
        // x hat den Wert 2
        proc(x);
   }
   static void proc(int a){
     // x hat den Wert 1, a den Wert 2
     y = x + a;
   }
```


2.5 Prozeduren

Übung:

Welche der Variablen in der Prozedur foo sind global, welche lokal?

```
static double foo(int x){
  int y = x * x;
  z = y * x;
  return a;
}
```


2.5 Prozeduren

Best Practices: Lokale und Globale Variablen

1. Geben Sie lokalen und globalen Variablen unterschiedliche Namen
2. Verwenden Sie so wenig globale Variablen wie möglich

2.5 Prozeduren

Einzelübung: Verbessern Sie dieses Bruchrechne-Programm, indem Sie die Anzahl der globalen Variablen reduzieren

```
class K{
    String op;
    int z1,n1,z2,n2;
    int ergZ,ergN;
    public static void main(String[] args){
      op = args[0];
      z1 =  Integer.parseInt(args[1]);…
      if(op.equals("add")) add();
      System.out.println("Ergebnis: "+ergZ+"/"+ergN);
    }
    static void add(){…}
```


2.5 Prozeduren

Rekursion gibt es auch in Java

- Rekursive Prozeduren rufen sich selbst auf
- Dabei sollten sich die Parameter ändern so dass ein Fixpunkt erreicht werden kann
- Es sollte einen Rekursionsabbruch geben, damit die Prozedur terminiert

2.5 Prozeduren

Beispiel

```
int EUCLID(int a, int b) {
   if(b = 0) return a;
   /*else*/ return EUCLID(b, a mod b);
}
```


2.5 Prozeduren

Übung, gemeinsam

Implementieren Sie eine Prozedur, die die Fakultätsfunktion implementiert. Die Fakultät von x wird berechnet als x* (x-1) * (x-2)…. * 1

Oder rekursiv ausgedrückt:

fac(1) = 1;

fac(x) = x* fac(x-1);

2.5 Prozeduren

Übung in Kleingruppen

Implementieren Sie eine rekursive Prozedur ohne Verwendung von Schleifen die alle Elemente eines Arrays ausgibt.

2.5 Prozeduren

Extrafutter

Seit Java 1.7: variable Parameterlisten

Deklaration /Verwendung:

int proz(int x , String… s){ // VarArg am Schluss
  String s1 = s[0]; // Zugriff wie bei Array

Aufruf:

proz(2,"Hallo") // oder…

proz(5,"A","B","C"); // beliebig viele Parameter

2. Imperative Programmierung

1. „Hello World“
2. Typen, Variablen und Grundoperationen
3. Steueranweisungen
4. Arrays
5. Algorithmen
6. Prozeduren

Ggf in Übung: Exkurs zum Software Testen