JAVA BASICS

1. Warum Java?

- zunehmend weit verbreitet

- einfach und (relativ) sicher
  keine Adressrechnung, aber Pointer
  keine gotos
  kein Präprozessor
  keine globalen Variablen
  garbage collection

- objekt-orientiert

- interpretiert, d.h.
  plattformunabhängig

- Programme mit Parallelverarbeitung möglich

- “Internet - Sprache” -> Java-Applets

Achtung

Die folgende Übersicht ist mit Absicht nicht vollständig.

2. Primitive Datentypen

a) Boolean (logische Werte wahr & falsch)

Deklarationen

```
boolean b1;
boolean b2 = true;
final boolean b3 = false;
```
(-> Konstantendeklaration)

Literale

```
true, false
```

Operationen

```
=    (Zuweisung),   ! (Negation),
&& (und),           || (oder),
== (gleich),        != (ungleich)
```

Beispiele

```
true && false
true || false
! true
boolean b = false;
!b
```

b) Character (Zeichen)

Deklarationen

```
char a1;
char a2 = 'a';
final char a3= 'a';
```

Literale

```
ASCII: 'a', 'b', ..., 'z',
   'A', 'B', ..., 'Z',
   '0', '1', ..., '9'
escape char's:  '\t'   (-> tab),
                '\n'   (-> return),
                '\\'   (-> \),
                '\"'   (-> ")
```

Operationen

```
=, ==, !=
```

Beispiele

```
'a' == 'b'
'a' != 'b'

char a = 'a';
char b;
b = a;
```


c) Integrale Typen

- byte -128 .. 127 8 Bits
- short -32.768 .. 32.767 16 Bits
- int -2.147.483.648 .. 2.147.483.647 32 Bits
- long (rund) - 10^{18} .. 10^{18} 64 Bits

Deklarationen

```
byte b;              short k = -128;
final int i = 043;   long l = 0xFF;
```

Operationen

```
=, ==, !=, +, -, *,
/    (ganzzahlige Division),
%    (modulo division),
++   (pre/post increment),
--   (pre/post decrement)
```

Beispiele

```
dies = 9; das = 2;

dies = das++    (-> dies == 2, das == 3)
dies = ++das    (-> dies == 3, das == 3)
x = dies / das  (-> x == 4)
x = dies % das  (-> x == 1)
```

Zuweisungskompatibilität

byte < short < int < long,

d.h byte kann short- oder int-Variablen zugewiesen werden, jedoch nicht umgekehrt.

d) Reelle Typen

- float (rund) - 10^{38} .. 10^{38} 32 Bits
- double (rund) - 10^{308} .. 10^{308} 64 Bits

Deklarationen

```
float f;
double d = 0.5;
final double PI = 3.14156;
```

Literale

```
1.3   .3f     1e1    1e-10
```

Operationen

```
=, ==, !=, +, -, *, /,
```

Beispiele

```
float f = 1.0f;
double d = 1e10;
double e = 1e100;

d = f;          (-> d == 1.0)
f = d;          (Fehler!)
f = (float)d;   (-> f == 1e10)
f = (float)e;   (-> ??)
```

Zuweisungskompatibilität

. . . long < float < double

e) Zeichenketten (Strings)

Deklarationen:

```
String gruss = new String ("hallo");
String gruss1 = "hallo";
String gruss2 = new String(gruss);
```

Literale

```
"...", "42", "hallo\n", "\""
```

Operationen

```
length()      (Achtung: mit Klammern)
+                    (Verkettung)
charAt(index)
equals(zk)
```

Beispiele

```
(sei int x = 10)

gruss + "xyz"          (-> "halloxyz")
gruss + 42             (-> "hallo42")
gruss + x + true + "x" (-> "hallo10truex")
gruss.length() == 5    (-> true)
gruss2.charAt(4)       (-> 'o')
gruss.equals(gruss2)   (-> true)
```

Anmerkung

`String` ist eigentlich kein primitiver Datentyp. Das spielt aber für einfache Programme keine Rolle.

3. Komposite Datentypen

a) Arrays (Reihungen, Felder)

-> alle Komponenten haben denselben Datentyp

Deklarationen

```
byte b[] = new byte[1024];
char[] c, d = new char[5];
int [] p = {1, 2, 3, 4};
```

Index

```
0 .. size - 1
```

Operationen

```
[]   (Indizieren einer Komponente)
length (Achtung: keine Klammern)
```

Beispiele

```
b[0] = 3;          (-> b[0] == 3)
3 == b[0]          (-> true)
b[1] = b[0] + 4;   (-> b[1] == 7)
b[1024] = 5;       (-> FEHLER!!)
b.length           (-> 1024)
```


b) Rekords (Strukturen, Verbund)

-> Komponenten können verschiedene Datentypen haben

Deklarationen

```
class StudentT {
   String name;
   int alter;
   double notendurchschnitt;
} // StudentT

StudentT paul = new StudentT ();

StudentT paula;
paula = new StudentT();
```

Operationen

```
.    (Selektieren einer Komponente)
```

Beispiele

```
paul.name = "Paul";
paul.alter = 23;
paul.notendurchschnitt = 2.3;

paula.name = paul.name + 'a';
paula.alter = paul.alter - 2;
paula.notendurchschnitt = 2.0;
```

4. Anweisungen

Einfache Anweisungen

a) Zuweisung

```
<Zielvariable> = <Quelle>
<Quelle> -> Literal, Variable, Ausdruck

z. Bsp. x = 3; y = x + 123;
```

b) Eingabe

```
import dssz.io.*;
stdin in = new stdin();

<int-Bezeichner> =
   in.getInt("Eingabeaufforderung");

<char-Bezeichner> =
   in.getChar("Eingabeaufforderung");

. . .
```

c) Ausgabe

```
System.out.println(zk1+zk2+zk3+ ... +zkn);
   . . .
System.out.println(zk1);
System.out.println();

System.out.print(zk1+zk2+zk3+ ... +zkn);
```

Achtung:
Jede einfache Anweisung wir mit “;” abgeschlossen

Strukturierte Anweisungen

d) if-Anweisung

```
if ( <Bedingung> ) {
   <Anweisungsfolge>
} else {
   <alternative Anweisungsfolge>
} // if
```

der else-Teil kann auch entfallen:

```
if ( <Bedingung> ) {
   <Anweisungsfolge>
} // if
```

e) switch-Anweisung

```
switch ( <Auswahl> ) {

   case <wert1> :<Anweisungsfolge> break;
   case <wert2> :<Anweisungsfolge> break;

   . . .
   default:     <Anweisungsfolge>

} // switch
```

Achtung:

- Immer `break` & `default` verwenden!!!
- Mögliche Typen für die Auswahl:
  boolean, byte, char, short, int

f) while-Schleife

```
while ( <Bedingung> ) {
   <Anweisungsfolge>
} // while
```

g) do-while-Schleife

```
do {
   <Anweisungsfolge>
} while ( <Bedingung> )
```

h) for-Schleife (Zählschleife)

```
for (<Init>; <Bedingung>; <Schritt>) {
   <Anweisungsfolge>
} // for
```

zum Beispiel:

```
for (int i = 0; i < 10; i++) {
   <Anweisungsfolge>
} // for
```

entspricht

```
int i = 0;
while (i < 10) {
   <Anweisungsfolge>
   i++;
} // while
```


5. Das erste Java-Programm

Datei `Hello.java`

```
public class Hello {
   // Kommentar:
   // Jedes Java-Programm besteht aus einer
   // Ansammlung von "Klassen", wobei jede
   // Klasse in ihrer eigenen Datei
   // untergebracht ist. Diese Datei muss
   // ebenso wie die Klasse heissen!

   public static void main (String argv[]) {

     /* Noch ein Kommentar:
       Diese Klasse enthaelt eine Funktion
       main() mit genau diesem Kopf!
       Diese Funktion ist der Einstiegs-
       punkt in das Programm.
     */

   // Vereinbarungen
   String s = "hello";

   // Ausgabe
   System.out.println(s + " World!");

   } // main

} // Hello
```

6. Java-Programm Max

Problem: Eingelesen werden k Zahlen,
ausgegeben wird deren Maximum!

```
import dssz.io.*;
public class Max {

private static int maximum(int[] numbers) {
   ...
} // maximum

public static void main (String argv[]) {

   // Vereinbarungen
   int k = 0;
   int[] numbers;
   stdin in = new stdin();

   // Eingabe
   k = in.getInt("Wieviel Zahlen: ");
   numbers = new int[k];

   for(int i = 0; i < k; i++) {
      numbers[i] = in.getInt(" ");
   } // for

   // Verarbeitung & Ausgabe
   System.out.println(
      "Maximum ist" + maximum(numbers));

} // main


} // Max
```



```
private static int maximum(int[] fnumbers) {

   int max;// Ergebnisvariable

   if (fnumbers.length == 0) {
      max = 0;
   } else if (fnumbers.length == 1) {
      max = fnumbers[0];
   } else {
      max = fnumbers[0];

      for(int i=1; i<fnumbers.length; i++) {

         if (max < fnumbers[i]) {
            max = fnumbers[i];
         } // if

      } // for

    } // if

    return max;

} // maximum
```