FIONA NÜESCH

BRÜCKENKURS PROGRAMMIEREN

WAS IST EIN PROGRAMM

FIONA NÜESCH

- Vom Computer ausführbare Datei
- beinhaltet Maschinencode
- kann vom Computer als Abfolge von Maschinen-/ Prozessorbefehlen ausgeführt werden
- Die ausführbare Datei entsteht im Softwareentwicklungspro-zess aus Quellcode

```
FC 90 F7 60 B1 3C

36 00     JMP   ($0
A0        CMP   #$A
02        BCC   $FD
32        AND   $32
35        STY   $35
          PHA
78 FB     JSR   $FB
          PLA
35        LDY   $35
          RTS
34        DEC   $34
9F        BEQ   $FD
          DEX
16        BNE   $FE
BA        CMP   #$B
BB        BNE   $FD
31        STA   $31
3E        LDA   $3E
40        STA   ($4
40        INC   $40
```

- Quellcode ist typischerweise in einer höheren Programmiersprache geschrieben
- höhere Programmiersprachen sind für den Menschen leichter verständlich
- der Quelltext kann automatisiert über einen Compiler oder Interpreter in Maschinensprache übersetzt werden

Programmiersprache C [Bearbeiten | Quelltext bearbeiten]

Gegeben sei das folgende Programm in der Programmiersprache C, das die Summe der Zahlen *a*=2 und *b*=3 berechnet und das Ergebnis *c* an den Aufrufer zurückliefert:

```
int main() {
    int a = 2;
    int b = 3;
    int c = a + b;
    return c;
}
```

Das Kompilieren dieses Programms kann folgenden Maschinencode ergeben:

Maschinencode (hexadezimal)	zugehöriger Assemblercode	zugehöriger C-Code	Erläuterung
55 48 89 E5	`push rbp` `mov rbp, rsp`	`int main() {`	Sichere Register RBP auf dem Stack und setze RBP auf den Wert von Register RSP, dem Stackpointer (gehört nicht zur eigentlichen Berechnung). Diese Vorbereitung ist notwendig, um die Werte der Variablen *a*, *b* und *c* auf dem Stack speichern zu können.
C7 45 FC 02	`mov DWORD PTR [rbp-4], 2`	`int a = 2;`	Setze Variable *a*, die durch Register RBP adressiert wird, auf den Wert 2.
C7 45 F8 03	`mov DWORD PTR [rbp-8], 3`	`int b = 3;`	Setze Variable *b*, die durch Register RBP adressiert wird, auf den Wert 3.

8B 45 F8
8B 55 FC
01 D0
89 45 F4
	`mov eax, DWORD PTR [rbp-8]`
	`mov edx, DWORD PTR [rbp-4]`
	`add eax, edx`
	`mov DWORD PTR [rbp-12], eax`
		`int c = a + b;`	Setze Register EAX auf den Wert von Variable *b*.
			Setze Register EDX auf den Wert von Variable *a*.
			Addiere den Wert von EDX zum Wert von EAX.
			Setze Variable *c*, die durch RBP adressiert wird, auf den Wert von EAX.

8B 45 F4	`mov eax, DWORD PTR [rbp-12]`	`return c;`	Setze Register EAX auf den Wert von Variable *c*. Weil Register EAX diesen Wert bereits enthält, könnte diese Anweisung in einem optimierten Programm entfallen.
5D C3	`pop rbp` `ret`	`}`	Setze RBP wieder auf seinen ursprünglichen Wert. Springe zurück an die Stelle des Aufrufs von *main*. Register EAX enthält den Rückgabewert.

Der Compiler schreibt diesen Maschinencode, gemeinsam mit weiteren zur Ausführung notwendigen Informationen, in eine sogenannte ausführbare Datei. Zur Ausführung wird der Maschinencode vom Lader des Betriebssystems in den Arbeitsspeicher geladen. Anschließend ruft es die Funktion *main* des Programms auf, und die CPU beginnt mit der Abarbeitung der Maschinenbefehle.

QUELLE: WIKIPEDIA -> MASCHINENSPRACHE

FIONA NÜESCH

JAVA

FIONA NÜESCH

- ist eine objektorientierte höhere Programmiersprache
- wird in Java-Bytecode übersetzt
- dieser wird kann von JVM interpretiert werden und garantiert so Plattformunabhängigkeit

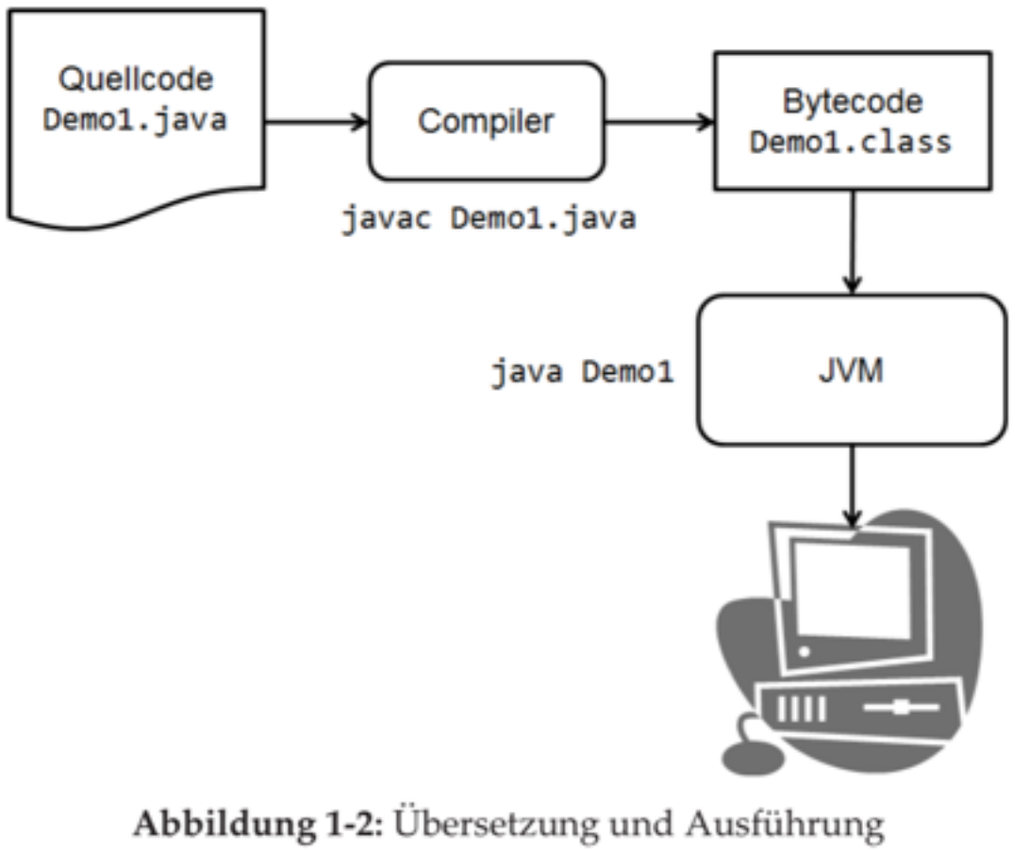


Abbildung 1-2: Übersetzung und Ausführung

Consider the following Java code:

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

A Java compiler might translate the Java code above into byte code as follows, assuming the above was put in a method:

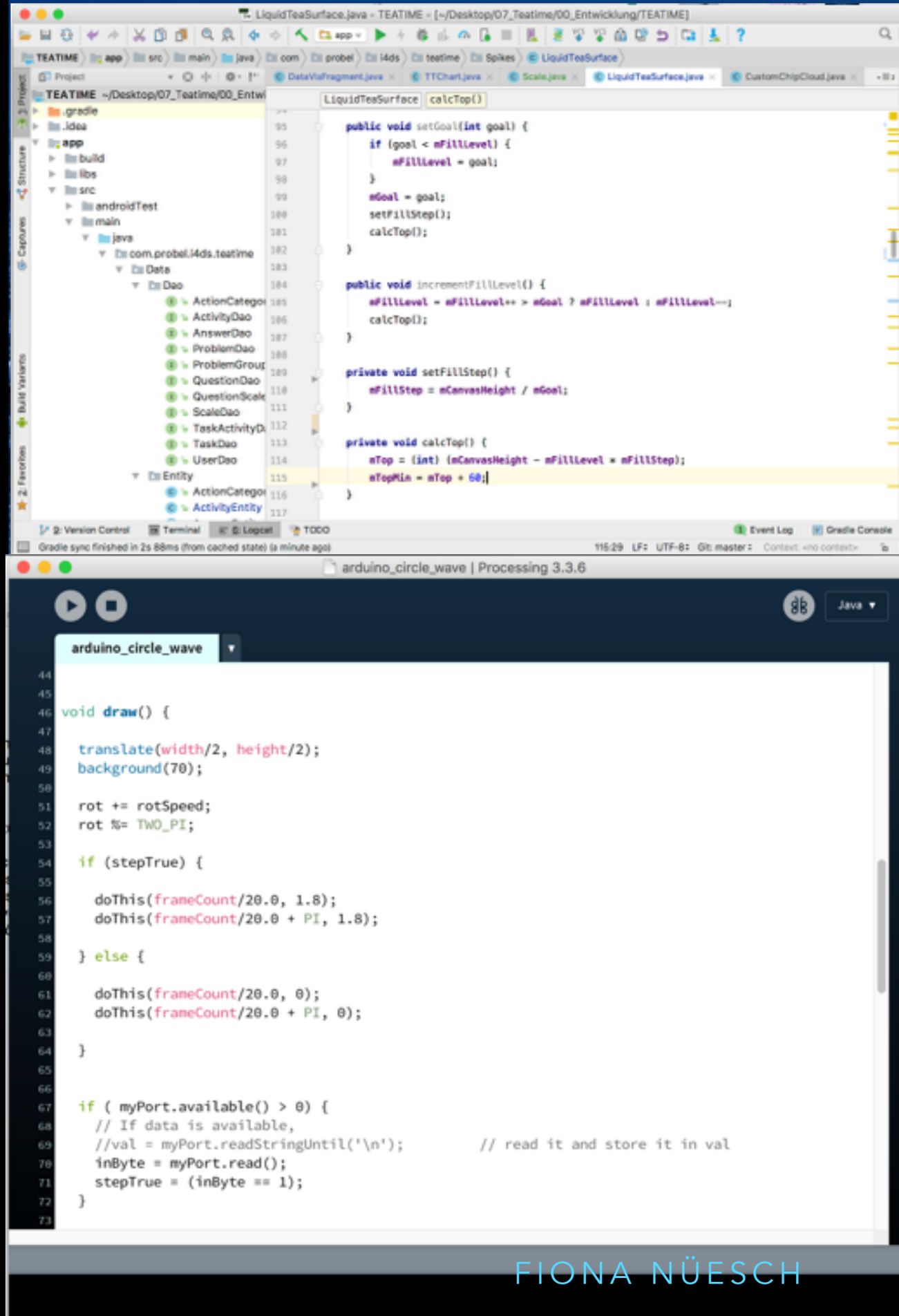
```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge       44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge       31
16:  iload_1
17:  iload_2
18:  irem
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic       #84; // Field java/lang/System.out:Ljava/io/PrintStream;
34:  iload_1
35:  invokevirtual   #85; // Method java/io/PrintStream.println:(I)V
38:  iinc    1, 1
41:  goto    2
44:  return
```

QUELLE: GRUNDKURS JAVA

IDE

FIONA NÜESCH

- integrierte Entwicklungsumgebung
- ermöglicht Softwareentwicklung ohne Medienbrüche
- Unterstützt Programmierende mit Funktionalitäten wie: Syntax-Highlighting oder Kompilierung



VARIABLEN UND PRIMITIVE DATENTYPEN

FIONA NÜESCH

- Variable = Gefäss um Daten zu speichern
- kann in Java nur Daten von einem definiertem Wert aufnehmen

```
        frei gewählter Name dieser Variable
int name = 1;
Datentyp      Wert
```

- Java kennt acht primitive Datentypen
- für **Zahlen**, **Zeichen** und **Wahrheitswerte**
- primitive Datentypen haben einen festen Wertebereich
- Variablen Namen beginnen klein und werden wie alle Namen in der CamelCase-Notation geschrieben

Wahrheitswerte		
boolean	false, true	`boolean b = true;`
Ganze Zahlen		
byte	-128 - 127	`byte b = 1;`
short	-32’768 - 32’767	`short s = 1;`
int	-2’147’483’648 - 2’147’483’647	`int i = 1;`
long	-9.223.372.036.854.775.808 - 9.223.372.036.854.775.807	`long l = 1;`
Fliesskommazahlen		
float	ca 1,4 * 10^{-45} - 3,4 * 10^{38} Genauigkeit ca 7 Stellen	`float f = 1.1f;`
double	ca. 4,9 * 10^{-324} - 1,8 * 10^{308} Genauigkeit ca. 15 Stellen	`double d = 1.1;`
Zeichen		
char	Unicode Zeichen	`char c = 'a';`
String	Zeichenketten	`String s = "Hallo Welt!";`

```
int variableName;          Deklaration
…
variableName = 1;          Initialisierung
```

Übung Variablen & Datentypen

„Der sicherste Weg zum Erfolg ist immer, es doch noch einmal zu versuchen.“

–THOMAS ALVA EDISON

OPERATOREN

FIONA NÜESCH

- Mit Operatoren können Zuweisungen und Berechnungen vorgenommen und Bedingungen formuliert und geprüft werden.

- Es gibt Operatoren für **Berechnungen**, zum **Vergleichen** von numerischen Werten und zum **verknüpfen** von **Logischen** Werten.

Operatoren für Berechungen

+	`int c = a + b;`	Addiert die Werte von `a` und `b` und speichert das Resultat in `c`.
-	`int c = a - b;`	Subtrahiert die Werte von `a` und `b` und speichert das Resultat in `c`.
*	`int c = a * b;`	Multipliziert die Werte von `a` und `b` und speichert das Resultat in `c`.
/	`int c = a / b;`	Dividiert die Werte von `a` und `b` und speichert das Resultat in `c`.
%	`int c = a % b;`	Berechnet den Modulo von `a` und `b` und speichert das Resultat in `c`.
++	`a++;`	Erhöht `a` um eins.
--	`b--;`	Verkleinert `b` um eins.

Operatoren zum Vergleichen

<	a < b	kleiner
<=	a <= b	kleiner gleich
>	a > b	grösser
>=	a >= b	grösser gleich
==	a == b	gleich
!=	a != b	ungleich

Logische Operatoren (Verknüpfung von Wahrheitswerten)

!	nicht
&	und (vollständig)
^	xor
|	oder (vollständig)
&&	und (kurz)
||	oder (kurz)

a	**b**	**a & b** **a && b**	**a | b** **a || b**	**a ^b**
true	true	true	true	false
true	false	false	true	true
false	true	false	true	true
false	false	false	false	false

„Lehre bildet Geister; doch Übung macht den Meister.“

– DEUTSCHES SPRICHWORT

ARRAYS

FIONA NÜESCH

Eine Sammlung von Elementen desselben Datentyps

frei wählbarer Name — Anzahl Elemente, die die Sammlung beinhaltet (ganzzahlig)

```
int[] arrayName = new int[10];
```

Datentyp — Ein Referenztyp wird über die Anweisung **new** Erzeugt

Elemente vom Typ int

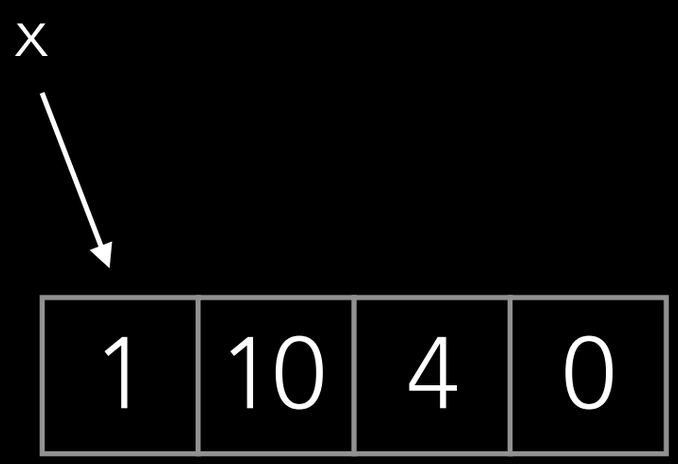
Deklaration und Erzeugung separiert

```
int[] arrayName;
…
arrayName = new int[10];
```

Initialisierung

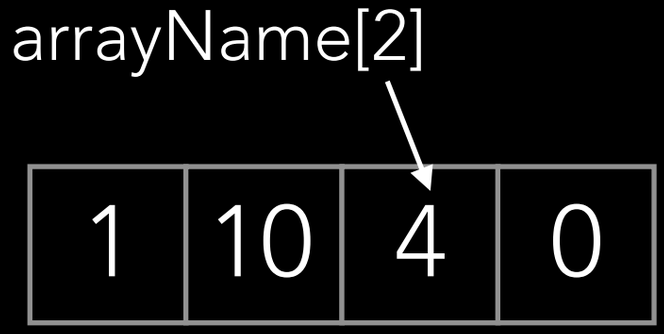
```
int[] x = {1, 10, 4, 0};
```

Array x ist eine Sammlung von 4 Elementen

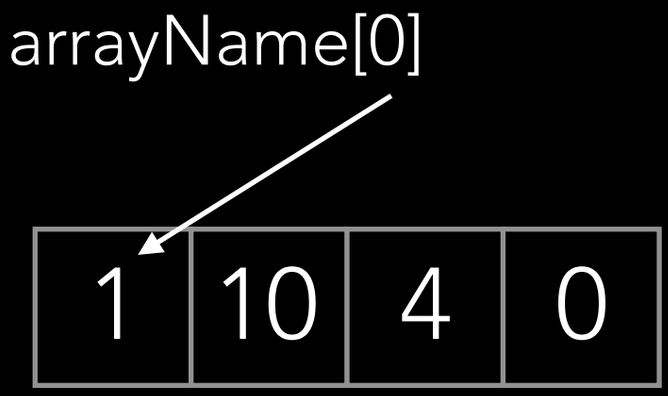


Auf Elemente zugreifen

```
Name                Position
= arrayName[2];
```



MERKE: Es beginnt bei 0.



Übung Arrays

„I haven’t failed. I’ve just found 10’000 ways that won’t work.“

–THOMAS EDISON