HAW Hamburg
Fakultät TI
Technik und Informatik

Programmiermethodik 1
Programmiertechnik

Variablen

Wiederholung

- Programmieren
- Organisation
- Java
- Erstes Programm
- Eclipse

Ausblick für heute

Use Cases

- Ich berechne das Ergebnis einer Rechnung (eine Zahl) und möchte es zur späteren Verwendung zwischenspeichern.
- Ich frage von der/m BenutzerIn eine Eingabe ab, verwende Sie in einer Rechnung und melde das Ergebnis einer Berechnung zurück.
- Ich stelle etwas durch eine ganze Zahl dar (z.B. Anzahl Gefährten im Herrn der Ringe).

Agenda

- Bezeichner
- Variablen
- Ein- und Ausgabe
- Ganzzahlen und Literale

Bezeichner

Bezeichner

- alternative Benennung: Identifier
  - Namen für verschiedene "Dinge" im Quellcode
  - sind an vielen Stellen frei wählbar
- erlaubte Bestandteile:
  - große und kleine Buchstaben
  - Ziffern
  - Unterstrich (`'_'`) und Dollar (`'$'`)
- große und kleine Buchstaben werden unterschieden
  - (z.B. `Summe` vs. `summe`, `Hallo` vs. `hallo` vs. `halLo`)
- erstes Zeichen darf keine Zahl sein

Bezeichner

- die meisten Bezeichner muss man selbst festlegen
  - für eigene Dinge
  - z.B. eigene Variablen
- wird Code anderer Programmierer genutzt, werden auch vorgegebene Identifier benutzt
  - z.B. `println`

Bezeichner

- etwa 50 reservierte Wörter dürfen nicht als Name benutzt werden
  - siehe *Java-KeyWords.pdf* auf EMIL

abstract	default	goto	package	this
assert	do	if	private	throw
boolean	double	implements	protected	throws
break	else	import	public	true
byte	enum	instanceof	return	transient
case	extends	int	short	try
catch	false	interface	static	void
char	final	long	strictfp	volatile
class	finally	native	super	while
const	float	new	switch	
continue	for	null	synchronized	

not used
added in 1.2
added in 1.4
added in 5.0

Übung: Bezeichner

Welche der folgenden Bezeichner sind gültig?

- a) `$byte`
- b) `__1`
- c) `name-1`
- d) `1Name`
- e) `operator`
- f) `True`

Variablen

Variablen

- Ablage von Daten
- Variablen haben
  - Namen (siehe Bezeichner)
  - Datentyp (kurz Typ) legt Art der Werte fest, die Variable aufnehmen kann
    - Menge der erlaubten Werte
  - Wert
- Adresse
  - Ort wo Variable gespeichert ist bzw. wo sich der „Behälter“ befindet
  - in JAVA ist die Adresse (selbst) bewusst vorm Programmierer verborgen, als Referenz jedoch anzutreffen
  - Name ist Synonym für Adresse

Beispiel:

```
int geld;
```

Variablen

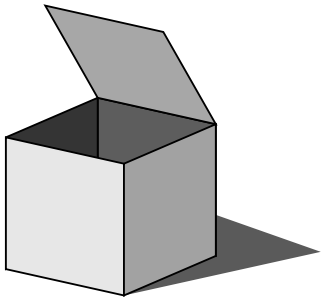
- Achtung!
  - in der Mathematik bezeichnen Variablen Werte
  - in der Informatik ist eine Variable eine Art „Behälter“ für einen Wert

```
x = y + z
```

  - hat (meist) unterschiedliche Bedeutung

Variablen

- Variablen speichern Werte zur späteren Verwendung
- Modell-Vorstellung:
  - Variable = Schachtel, in der ein Wert aufgehoben werden kann
- technische Realisierung:
  - Variable = Speicherplatz im Hauptspeicher

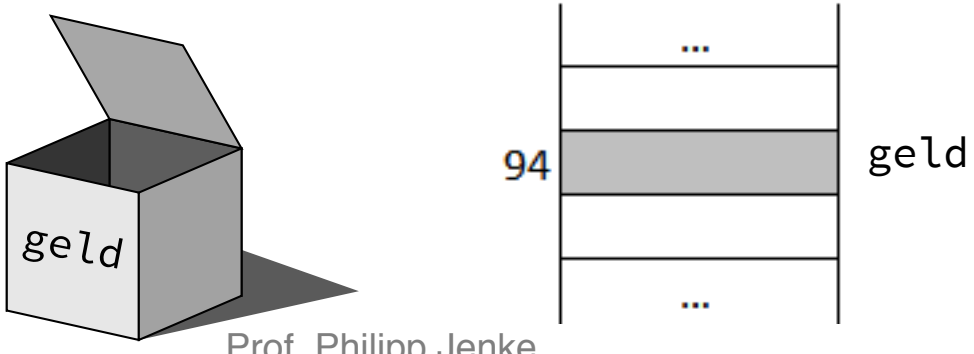


Variablen

- Variablennamen beginnen per Konvention mit kleinem Buchstaben

```
index
geld
meineErsteVariable
```

- Modell-Vorstellung:
  - Variablenname = Beschriftung der Schachtel für gezielten Zugriff
- technische Realisierung:
  - Variablenname = symbolische Bezeichnung der Hauptspeicheradresse



Deklaration

- Variablen müssen vor ihrer Verwendung deklariert werden!
  - Grund: Verwaltung Variablenname ↔ Hauptspeicheradresse durch den Compiler
- vor dem Variablennamen steht der Datentyp der Werte, die in der Variablen gespeichert werden können
  - Typ der Variablen
  - Syntax: `<Typ> <Variablenname>;`

Datentypen: Ganzzahlen

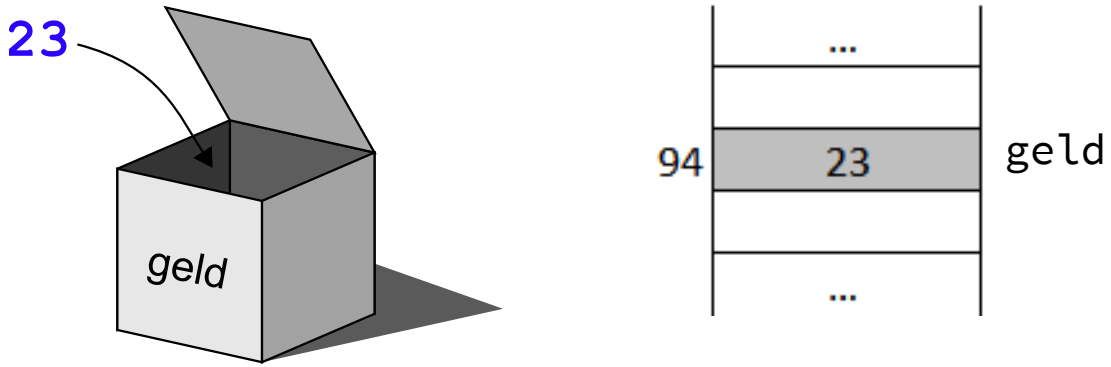
- Typ `int` bezeichnet ganze Zahlen („integer“)
- Beispiele für Variablendeklarationen:

```
int index;
int geld;
int meineErsteVariable;
```

- eine Variable darf nur einmal innerhalb eines Sichtbarkeitsbereichs deklariert werden

Wertzuweisung

- Wertzuweisung gibt einer Variablen einen Wert
  - engl. assignment
  - Syntax: `<Variablenname> = <Ausdruck>;`
  - Beispiel: `geld = 23;`
- Modell-Vorstellung:
  - Ergebnis des Ausdrucks wird in Schachtel gelegt
- technische Realisierung:
  - Ergebnis des Ausdrucks wird unter der Adresse, die der Variablenname bezeichnet, in den Hauptspeicher geschrieben



Wertzuweisungen

- Beispiele:

```
absoluterNullpunkt = -273;
tageImJahr = 31+28+31+30+31+30+31+31+30+31+30+31;
dieUltimativeAntwort = 179%18*5/2;
```

- mehrfache Wertzuweisungen überschreiben vorhergehende Werte

```
int geld;
geld= 1;
geld= -(6 - 8);
geld= 11%4;
```

Verwendung in Ausdrücken

- Variablen können in Ausdrücken verwendet werden
  - daher auf beiden Seiten einer Wertzuweisung stehen
  - links: Zuweisen eines neuen Wertes („Schreiben“ einer Variablen)

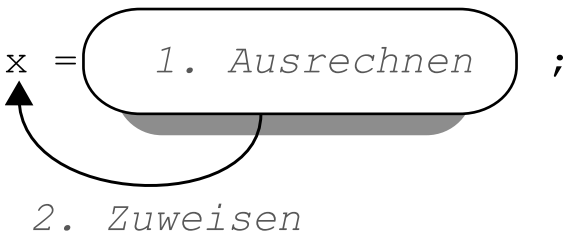
    ```
    fahrenheit = 91;
    ```
  - rechts: Verwenden des alten Wertes („Lesen“ einer Variablen)

    ```
    celsius = 4 * fahrenheit / 7 - 32;
    ```

- Variablen spielen je nach Kontext unterschiedliche Rollen
  - linke oder rechte Seite einer Wertzuweisung: Wert speichern oder in einem Ausdruck zur Verfügung stellen

Java vs. Mathematik

- Variablen in Java sind nicht vergleichbar mit Variablen in der Mathematik
- Beispiel: `x = x + 1`
  - Mathematik: Widerspruch
  - Java-Programm: Wertzuweisung
- Gleichheitszeichen:
  - Mathematik: Relation ohne zeitliche Dimension (Aussage)
  - Java-Programm: Zuweisungs-Operator, der eine Abfolge nacheinander abgewickelter Teilschritte auslöst:
    - 1. Rechte Seite komplett ausrechnen
    - 2. Rechenergebnis an die Variable links zuweisen



Initialisierung

- eine neu definierte Variable hat noch keinen Wert, sie ist nicht initialisiert
- eine nicht initialisierte Variable kann nicht gelesen werden
- Beispiel (fehlerhaft):

```
int indexI;
int indexJ;
indexJ = 2 * indexI;
```

- Compiler prüft die Initialisierung
  - kein Laufzeit-Fehler möglich

Initialisierung

- Die Initialisierung (erste Wertzuweisung) kann bei der Deklaration erfolgen
  - Syntax: `<Typ> <Variablenname> = <Ausdruck>;`
  - Beispiele:

```
int fahrenheit = 91;
int celsius = 4 * fahrenheit / 7 - 32;
```

- getrennte Deklaration und Wertzuweisung ...
  - Beispiel

```
int wert;
wert = 1;
```

  - … ist äquivalent zu:

```
int wert = 1;
```

- Daumenregel: Variablen sollten möglichst immer initialisiert werden!

Übung: Zuweisungsoperatoren

Welchen Wert hat die Variable Wert nach jeder Zeile?

```
int wert = 1;
wert += 1;
wert *= 3;
wert -= 2;
wert %= 3;
```

Ein- und Ausgabe

Ausgabe

- Ausgabe einer Zeichenkette auf der Konsole (ohne Zeilenumbruch)

```
System.out.print("Hallo Welt!");
```

- Ausgabe einer Zeichenkette auf der Konsole (mit Zeilenumbruch)

```
System.out.println("Hallo Welt!");
```

- Ausblick: Formatierte Ausgabe

```
System.out.format("Wert: %d", 23);
```

Konsole:

```
Hallo  Welt!Hallo  Welt!
Wert: 23
```

Eingabe

- Abfragen einer Zeichenkette über die Konsole

```
Scanner scanner = new Scanner(System.in);
String eingabeText = scanner.next();
int eingabeZahl = scanner.nextInt();
scanner.close();
```

Wenn beim Einlesen etwas schief geht, wird eine Exception geworfen. In dem Fall ist ein Problem aufgetreten. Dies kann an verschiedenen Stellen in einem Java-Programm passieren. Später dazu mehr.

Import-Anweisungen

- Klassen (zusammengefasst in Packages) können dem Compiler am Anfang einer Quellcode-Datei durch eine import-Anweisung bekannt gegeben werden
- Vorteil: es muss nicht bei jeder Verwendung das Package mit angegeben werden!
- Beispiel (Verwendung des Scanners):

```
import java.util.Scanner;
```

- *oder*

```
import java.util.*;
```

- alle Klassen aus dem Package `java.lang` sind automatisch bekannt

Ganzzahlen und Literale

Ganzzahlen (int)

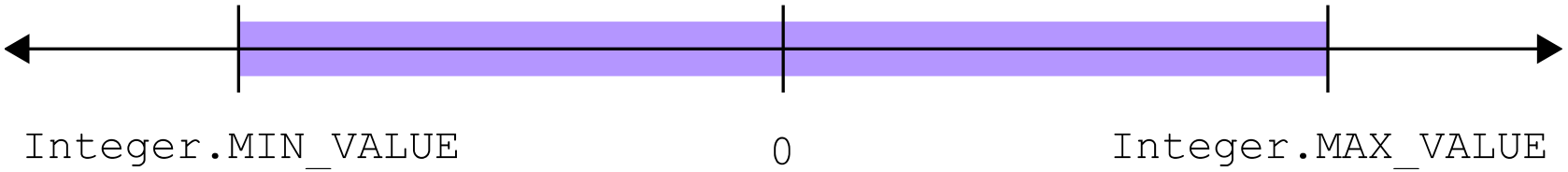
- Repräsentation ganzer Zahlen

- Beispiele
  - Anzahl Äpfel an einem Baum
  - Anzahl Atome in der Milchstraße
  - Leistungspunkte als Ergebnis der Klausur
  - Geburtsjahr

Ganzzahlen (int)

- Werte benötigen Hauptspeicherplatz
  - Größe ist für jeden Typ festgelegt
  - `int`: 32 Bit
  - ⇒ darstellbare Wertebereiche sind begrenzt

Grenze	Wert	Vordefinierte Variable
größter positiver Wert	$2147483647 = 2^{31} - 1$	`Integer.MAX_VALUE`
größter negativer Wert	$-2147483648 = -2^{31}$	`Integer.MIN_VALUE`



Datentypen: Ganzzahlen

- Idee: es gibt 2^n Bitfolgen der Länge n
  - jedes Bit verdoppelt die Anzahl
  - → mit n Bit sind 2^n Zahlen darstellbar
- erstes Bit als einfaches Vorzeichenbit
  - 0 positive Zahl, 1 negative Zahl
- Nachteile
  - doppelte Null-Darstellung
  - Fallunterscheidung positive/negative Zahl nötig
    - Rechnen aufwändig!

Zweierkomplement

- besser: Darstellung ganzer Zahlen im Zweierkomplement:
  - erstes Bit ist bei negativen Zahlen 1
  - dreht aber die Wertigkeit der restlichen Bits um
    - anschließend noch + 1
  - Darstellbarer Zahlenbereich:
    $-2^{n-1},..,0,..,2^{n-1}-1$
- daher besonderes Verhalten bei Überlauf!

Zweierkomplement

8-Bit Zweierkomplement

Binärwert	Interpretation als Zweierkomplement	Interpretation als vorzeichenlose Zahl
00000000	0	0
00000001	1	1
...	...	...
01111110	126	126
01111111	127	127
10000000	−128	128
10000001	−127	129
10000010	−126	130
...	...	...
11111110	−2	254
11111111	−1	255

Übung: Zweierkomplement

- Nehmen wir an, wir haben einen Datentyp für Ganzzahlen (positive und negative) mit der Größe 4 Bit im Zweierkomplement erschaffen. Welche (Dezimal-)Zahlen können wir damit darstellen?

Literale

- konstante Werte
  - stehen „wörtlich“ im Quellcode
  - können sich innerhalb eines Programms nicht ändern
- haben vorgeschriebene Schreibweisen
  - abhängig von ihrem Typ

Ganzzahlige Literale

- einfachste Schreibweise ganzzahliger Literale
  - Folge von Dezimalziffern
- positive und negative Werte
  - Vorzeichen "+" oder "-" (unäre Operatoren)
  - Vorzeichen optional: falls nicht vorhanden, implizit ergänzt mit "+"
- Beispiele:
  - `0`
  - `42`
  - `+42`
  - `-7`
  - `+5673456`

Andere Zahlensysteme

- Hexadezimalzahlen (16 Ziffern `0..9, A..F`)
  - müssen mit `0x` beginnen
  - Beispiele: `0x0` (dezimal: `0`) / `0xFFFF` (dezimal: `65535`)
- Oktalzahlen (8 Ziffern `0..7`)
  - müssen mit `0` beginnen *(Vorsicht!)*
  - Beispiele: `05` (dezimal: `5`) / `012` (dezimal: `10`)
- Binärzahlen *(ab Java 7)* (2 Ziffern `0..1`)
  - müssen mit `0b` beginnen
  - Beispiele: `0b1` (dezimal: `1`) / `0b101` (dezimal: `5`)

Umrechnung eines Literals in eine einheitliche interne Darstellung erledigt der Compiler!

Andere Zahlensysteme

- negative Zahlen in allen genannten Zahlensystemen werden mit dem unären Operator "-" erzeugt
- Beispiel:
  - oktal -012 = dezimal -10
- negative Zahlen im Binär- und Hexadezimalsystem können auch durch das Zweierkomplement dargestellt werden
  - z.B: 0b11111111111111111111111111111111 → -1

Übung: Zahlensysteme

- Geben Sie je das Literal für die Dezimalzahl 23 als
  - a) Hexadezimalzahl
  - b) Oktalzahl
  - c) Binärzahl

  an.

Zusammenfassung

- Bezeichner
- Variablen
- Ein- und Ausgabe
- Ganzzahlen und Literale