UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Programmierkurs Java

Variablen und Datentypen

Raphael Allner
Institut für Telematik
22. Oktober 2019

Rückblick
Kapitel 1: Grundlagen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Algorithmen - *Handlungsanweisungen* zur Lösung von Problem

Höhere Programmiersprachen – nah am menschlichen Denken

- Werden in Maschinensprache übersetzt (vom Compiler)
- Haben eigene Syntax und Semantik

Es gibt eine Vielzahl verschiedener Programmiersprachen

- Java ist weit verbreitet und viele Firmen suchen Javaentwickler
- Java ist plattformunabhängig

Programme laufen in der Java Virtual Maschine (JVM)

- JVM ist im Java Runtime Environment (JRE) enthalten (kostenlos)

Entwickler brauchen das Java Development Kit (JDK)

- Programme schreiben wir in einem Texteditor oder in einer Integrated Development Environment (IDE)

Rückblick
Kapitel 1: Grundlagen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Wichtige Kommandozeilen Befehle:

- `javac` – Kompilieren des Quellcodes in Byte-Code
- `java` – Ausführen des Java-Programmes

In Java ist alles in Klassen strukturiert

- Der Einstieg in ein Programm ist die main-Methode

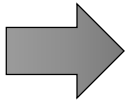
Java ist Turing-mächtig:

- Wenn genug Speicher vorhanden → kann alles berechnet werden
- Müssen über bestimmte Sprachmerkmale verfügen

Rückblick
Sprachmerkmale

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Java ist Turing-mächtig:



Dafür erforderliche Sprachmerkmale

- Sequenz (Hintereinander-Ausführung)
- Zuweisungen (zu Variablen)
- Elementare Rechenoperationen (Addition, Subtraktion, …)
- Bedingte Ausführung (If-Anweisung)
- Wiederholungsanweisung (Schleifen)

Überblick

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Einführung

Variablen: Speicherplätze für Daten

- Primitive Datentypen in Java
- Deklaration, Initialisierung und Wertzuweisung
- Identifier
- Konstanten

Literale

- Notation von Literalen
- Literale der Basisdatentypen

Felder (Arrays)

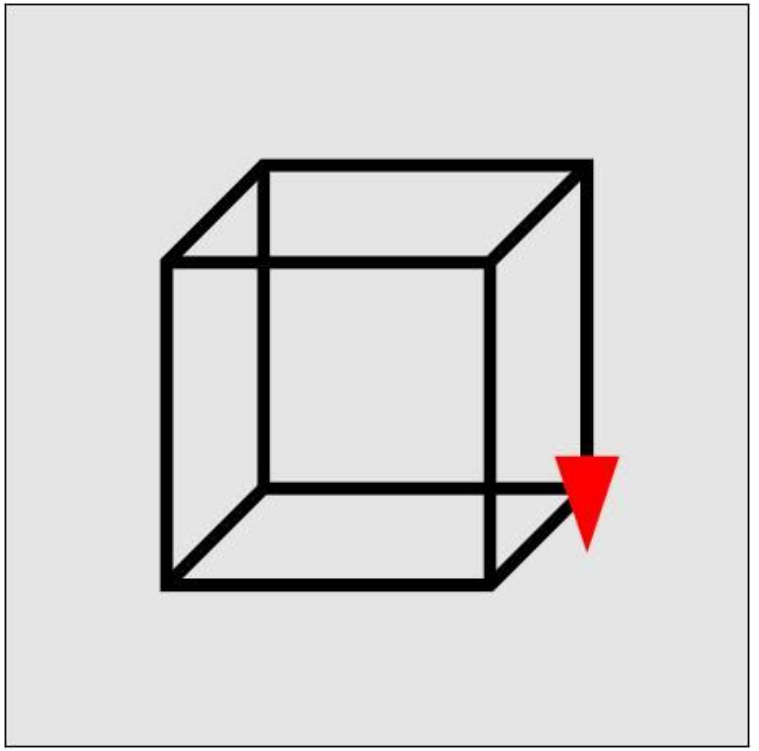
Klassen

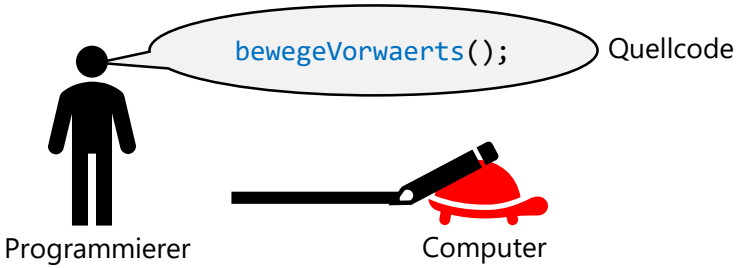
- Zeichenketten (Strings)

Einführung
Beispiel

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
 1 beginneZeichnen();
 2 bewegeVorwaerts(1);
 3 dreheNachLinks(90);
 4 bewegeVorwaerts(1);
 5 dreheNachLinks(90);
 6 bewegeVorwaerts(1);
 7 dreheNachLinks(90);
 8 bewegeVorwaerts(1);
 9 dreheNachLinks(90 + 45);
10 bewegeVorwaerts(0.3);
11 unterbrecheZeichnen();
12 bewegeNach(0.3,1.3);
13 beginneZeichnen();
14 bewegeVorwaerts(0.3);
15 ...
```



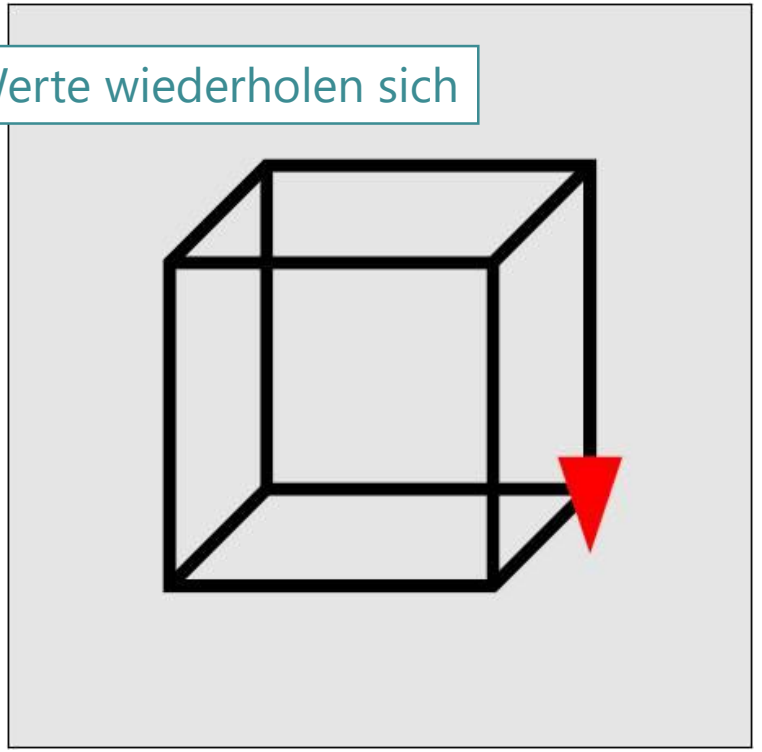


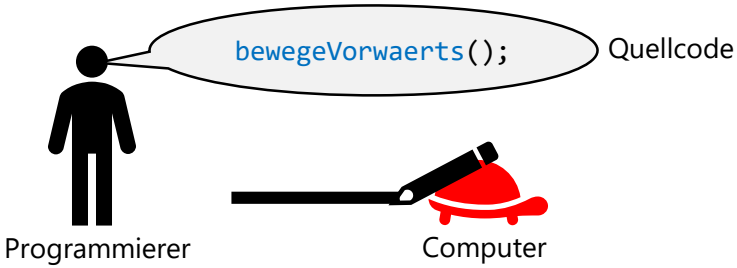
Einführung
Beispiel

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
 1 beginneZeichnen();
 2 bewegeVorwaerts(1);
 3 dreheNachLinks(90);
 4 bewegeVorwaerts(1);
 5 dreheNachLinks(90);
 6 bewegeVorwaerts(1);
 7 dreheNachLinks(90);
 8 bewegeVorwaerts(1);
 9 dreheNachLinks(90 + 45);
10 bewegeVorwaerts(0.3);
11 unterbrecheZeichnen();
12 bewegeNach(0.3,1.3);
13 beginneZeichnen();
14 bewegeVorwaerts(0.3);
15 ...
```

Viele Werte wiederholen sich

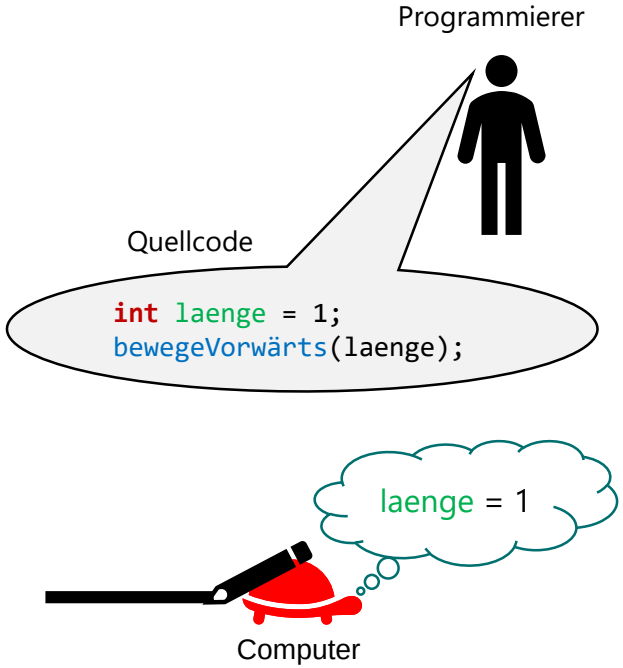




Einführung
Beispiel

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
1  int laenge = 1;
2  int rechterWinkel = 90;
3  float kuerzereLaenge = 0.3f;
4  beginneZeichnen();
5  bewegeVorwaerts(laenge);
6  dreheNachLinks(rechterWinkel);
7  bewegeVorwaerts(laenge);
8  dreheNachLinks(rechterWinkel);
9  bewegeVorwaerts(laenge);
10 dreheNachLinks(rechterWinkel);
11 bewegeVorwaerts(laenge);
12 dreheNachLinks(rechterWinkel + 45);
13 bewegeVorwaerts(kuerzereLaenge);
14 ...
```



Einführung
Variablen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

一个变量是一个储存位置

Eine *Variable* ist ein **Speicherplatz**.

- Ein temporärer Container für Daten im Programm
- Benennt eine (Start-)Adresse im **Arbeitsspeicher**

一个地址就是一个运行储存器

Programme verwenden **Namen statt Adressen**

- Beispiele: suesseKatzeXD, jahr, i, hallo
- Compiler bzw. Betriebssystem kümmern sich um konkrete Zuordnung von Name zu Speicheradresse

Der Datentyp bestimmt Größe des belegten Speichers

Adresse	Wert
00000001	1
00000010	‚a‘
00000011	true
00000100	2.8
00000101	5
00000110	
00000111	
00001000	
00001001	
00001010	
00001011	
00001100	
00001101	
00001110	
00001111	

Primitive Datentypen in Java

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Datentyp	Bemerkung	Min	Max
byte	8 Bit, 2er-Komplement	-128	+127
short	16 Bit, 2er-Komplement	-32.768	+32.767
int*	32 Bit, 2er-Komplement	-2^{31} (-2.147.483.648)	$+2^{31}-1$ (+2.147.483.647)
long*	64 Bit, 2er-Komplement	-2^{63} (-9.223.372.036.854.775.808)	$+2^{63}-1$ (+9.223.372.036.854.775.807)
float	32-bit IEEE 754 Fließpunktzahl	+/-1,4E-45 (+/-$1{,}4\cdot10^{-45}$)	+/-3,4E+38 (+/-$3{,}4\cdot10^{38}$)
double	64-bit IEEE 754 Fließpunktzahl	+/-4,9E-324 (+/-$4{,}9\cdot10^{324}$)	+/-1,7E+308 (+/-$1{,}7\cdot10^{308}$)
boolean	Größe nicht spezifiziert	false (0 nicht möglich)	true (1 nicht möglich)
char	Ein 16-bit Unicode Zeichen	'\u0000' (oder 0)	'\uffff' (oder 65.535)

* Seit Java 8 auch vorzeichenlose Darstellung möglich, siehe auch [1]

Primitive Datentypen in Java
Überlauf

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Was passiert, wenn ich in Java die Grenzen des Wertebereichs durch Rechenoperationen überschreite?

```
1000000 * 100000 / 100000 =
```
(Datentyp ist **int**)

```
1000000l * 1000001 / 1000001 =
```
(Datentyp ist **long**)

```
meinByte + meinByte =
```
(`byte meinByte = 127;`)

Es kommt zum Überlauf!

Siehe `Ueberlauf.java`

Variablen und Datentypen

Variablen

Übungs- oder Hausaufgabe

1. Laden Sie die Datei VariablenUndDatentypen.java aus dem Moodle herunter
2. Verfolgen Sie die Vorlesung um die Aufgaben zu erfüllen

```java
public class VariablenUndDatentypen {

    public static void main(String[] args){

        // 0. Deklarieren Sie eine Variable vom Typ long mit dem Identifier meineLieblingszahl



        // 1. Initialisieren Sie die Variable meineLieblingszahl mit Ihrer Lieblingszahl



        // 2. Deklarieren Sie eine Variable vom Typ String mit dem Identifier wuhu und initialisieren sie diesen mit dem Wert "Wuhu"



        // 3. Deklarieren und Initialisieren Sie ein Array mit dem Identifier woerter für 5 String Variablen



        // 4. Weisen Sie dem dritten Element des Arrays den String wuhu zu



        // 5.   Überprüfen Sie ob die folgenden Kommandozeilenausgaben indem sie diese .java compilieren und ausführen
        // 5.1. Überprüfen Sie ob die Ausgaben Ihren Erwartungen entsprechen

        System.out.println("Meine Lieblingszahl ist die " + meineLieblingszahl);
        System.out.println("Das 3te Element des Array woerter ist: " + woerter[3]);
        System.out.println("Die Laenge des Array woerter betraegt: " + woerter.length);

        // 6. Starten Sie dieses Programm mit einem beliebigen Argument und Überprüfen Sie ob folgende Konsolenausgabe Ihr Argument enthält

        if(args.length > 0){
            System.out.println("Ich habe dieses Programm mit folgendem Argument gestartet:" + args[0]);
        }
    }
}
```

Variablen
Deklaration von Variablen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Deklaration - eine Variable wird deklariert, wenn **Speicher** für sie **reserviert** wird 定义一个变量

- Man spricht auch alternativ von „erzeugt“ oder „definiert“

Syntax: 句法 `<Typ> <Name>;`

Semantik: 内容 Deklariere Variable vom Datentyp `<Typ>` mit dem Identifier `<Name>`

Beispiele:

- `int myInt;`
- `char c1;`
- `boolean flag;`
- `short myShort, myShort2;` ← mehrere Variablen gleichen Typs

Variablen
Beispielprogramm

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

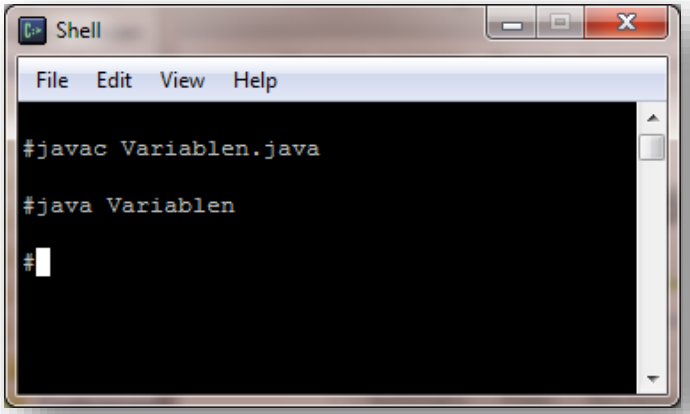
```
1  public class Variablen {
2      public static void main(String[] args) {
3          byte eineToppigeByteVariable;
4          short dannyDeVito;
5          int eineIntVariable;
6          long inspektorLongAndEvanLonger;
7          float elbe;
8          double schuldenDeutschlands;
9          boolean wahrOderFalsch;
10         char einZeichenGottes;
11     }
12 }
```

Was tut dieses Programm?

Es reserviert Speicher für Variable
它为变量保存了储存器

Variablen
Was tut dieses Programm?

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK



Auf den ersten Blick: *Nichts*

- **Keine Eingabe** erwartet
- **Keine Ausgaben**
- Es **berechnet** offensichtlich **nichts**

Immerhin: es **reserviert Speicher** für Variablen

- Und gibt ihn sofort wieder frei

Mit den Variablen können wir „arbeiten“:

- (Zwischen-)Ergebnisse von **Berechnungen ablegen**
- Diese Ergebnisse **über den Namen** der Variablen **ansprechen**

Variablen

Initialisierung und Wertzuweisungen

初始化和赋值

Initialisierung - eine Variable wird bei der bei der ersten Wertzuweisung initialisiert

初始化就是给变量第一次赋值

Syntax: `<Typ> <Name> = <Wert>;`

Semantik: Deklariere Variable vom Datentyp `<Typ>` mit dem Identifier `<Name>` und weise ihr den Wert `<Wert>` zu

Beispiel:

```
1  int myInt;
2  myInt = 1;
3  long myLong = 9000;
4  int otherInt = 2;
5  myInt = otherInt;
```

Deklaration

Initialisierung

Wertzuweisung mittels **Literal** (gleich mehr)

Wertzuweisung mit dem **Wert einer anderen Variable**

Variablen
Identifier

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Der `<Name>` einer Variable ist ein „Identifier“ 标识符

Über Identifier werden Variablen **eindeutig bezeichnet**

- Mehrere **Variablen können nicht den gleichen Identifier** haben
- Zumindest nicht im gleichen Kontext (Scope) 许多的变量的标识不能一样
  - (Siehe Kapitel 8 - Gültigkeitsbereiche)

Identifier dürfen nur aus gewissen Zeichen bestehen

Variablen
Legale Identifier

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Zulässige Zeichen für Identifier:

- Unicode Zeichen a b c
- Nummern 1 2 3
- Währungssymbolen € $
- Verbindungszeichen _ -

Dürfen theoretisch **beliebig lang** sein

Einschränkungen:

开头
- **Beginnen mit:** Zeichen, Währungssymbol oder Verbindungszeichen
  - Also auch nicht mit einer Nummer

没有空格
- **Keine Leerzeichen**

大小写
- Identifier sind **case-sensitive** (Unterscheidung von Groß- und Kleinschreibung)

无关键词
- **Keine reservierten Schlüsselwörter** (class, public etc., siehe [2]).

Variablen
Schlüsselwörter in Java

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

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

Identifier
Beispiele

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Legale Identifier

`Test`

- Besteht nur aus Alphazeichen und ist daher korrekt.

`__DAS_IST_TOTAL_TOLL_____`

- Unterstriche sind erlaubt (auch am Anfang)

`bóolêáñ`

- Gültige Unicode Zeichen

`DAGOBERT$$$$`

- Dollar-Zeichen ist ein gültiger Java-Buchstabe.

`€¥$`

- Währungssymbole sind gültige Zeichen

Ungültige Identifier

`2good4u`

- Erstes Zeichen muss ein Buchstabe sein und keine Ziffer

`das ist toll`

- Leerzeichen sind nicht erlaubt

`Krass!`

- Ausrufezeichen ist, wie viele Sonderzeichen, ungültig

`class`

- Der Name ist schon von Java belegt (Class wäre ok)

Variablen
Identifier Beispielprogramm

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
1  public class Identifier {
2     public static void main(String[] args) {
3           char Test;
4           char _DAS_IST_TOTAL_TOLL_______;
5           char bóolêáñ;
6           char DAGOBERT$$$$;
7           char €¥$;
8
9           char 2good4u;
10          char das ist toll;
11          char Krass!;
12          char class;
13    }
14 }
```

```
#javac Identifier.java

#javac Identifier.java
Identifier.java:9: error: not a statement
                char 2good4u;
                ^
Identifier.java:9: error: ';' expected
                char 2good4u;
                    ^
Identifier.java:9: error: not a statement
                char 2good4u;
                     ^
Identifier.java:10: error: ';' expected
                char das ist toll;
                        ^
Identifier.java:11: error: ';' expected
                char Krass!;
                          ^
Identifier.java:12: error: not a statement
                char class;
                ^
Identifier.java:12: error: ';' expected
                char class;
                    ^
Identifier.java:12: error: <identifier> expected
                char class;
                          ^
Identifier.java:14: error: reached end of file while parsing
}
 ^
9 errors

#
```

Variablen
Konventionen

Java Code Style

- Man hält sich an gewisse Konventionen
- https://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

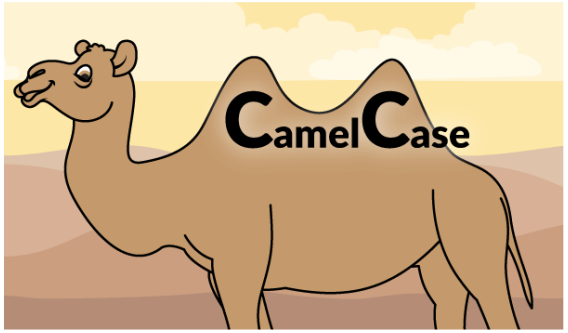
Beispiel für Identifier (Kurzfassung):

- **Klein beginnen** und im „**Camel Case**“ weiterschreiben
- Beispiel: eineVariableMitEinemLangenNamen

Für internationale Zusammenarbeit.

- **Auf ASCII beschränken** (kein ä, ü, ß, ...)

Details unter [3]



Variablen
Typsicherheit

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Alle Daten sind immer von einem bestimmten Typ

- Der **Datentyp der Werte**, die wir einer Variablen zuweisen, muss dem **Typ der Variablen** entsprechen. 每个数据都有数据类型
- Alternativ muss konvertiert werden (später)

Funktioniert das?

- `boolean bool = 123;`
- Kann man einem Wahrheitswert (boolean) eine int-Zahl zuweisen?
- **Nein**
  - Dient der „Sicherheit“ von Programmen
  - Sog. statische Typsicherheit
  - Diese wird vom Compiler überprüft
  - Auch nicht 1 oder 0, wie in anderen Programmiersprachen

Konstanten

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Eine Konstante ist ein **Speicherplatz** 一个常量是一个储存位

- Repräsentiert einen Wert aus einem bestimmten Wertebereich
- Wertebereich durch verwendeten Datentyp festgelegt
- Speicherbereich wird über einen Namen referenziert
- **Unterschied zu Variablen: Der Wert ist unveränderlich**

Syntax: `final <Typ> <Name>;`

Beispiel:

```
final int myInt = 8;
```

索引

final代表它是一个常量

ohne „final“ wäre „myInt“ eine Variable,
so ist „myInt“ eine Konstante

Variablen und Datentypen

Literale

Literale
Typsicherheit

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Literale sind **konstante Werte** als Zeichenketten im Quelltext

Beispiele:

- `10`
- `012`
- `0xA`
- `9.81`
- `true`
- `'z'`

Auch „Literale" sind immer von einem bestimmten Datentyp

Welchen Datentyp haben diese Literale?

Literale
Datentypen - Ganzzahlen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Ganzzahlige Typen (standardmäßig vom Typ **`int`**) 整数类型 “int”

- **`10`** (Dezimal)
- **`012`** (**Oktal**, Basis 8, wegen **führender Null**)
- **`0xA`** (**Hexadezimal**, Basis 16 wegen führendem **`0x`**, Groß-/Kleinschreibung egal)
- **`0b1010`** (**Binär**, Basis 2, wegen führendem **`0b`**)
- Suffix “1“ oder “L“ → Datentyp **`long`** statt **`int`**

Beispiele:

- **`12273`**;
- **`12733L`**;
- **`int i = 10;`**
- **`long l = 10L;`**

Ab Version 7 erlaubt Java Unterstriche zur besseren Übersichtlichkeit

- Beispiele: **`1012_1234_1_4, 0xFFFF_ABCD, 0b1111__0101`**
- Unterstriche werden vom Compiler ignoriert

Literale
Datentypen - Fließkommazahlen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Fließkommazahlen sind standardmäßig vom Typ **`double`**

浮动精点数（小数） “double”

- Engl. *Floating Point Number*
- Optional kann ein **`d`** angehängt werden, um **`double`** explizit zu fordern
- Durch Anhängen von **`f`** wird ein Literal vom Typ **`float`**

Schreibweisen:

- Normale Schreibweise: `10.0` (alternativ: `10.`)
- Normale Schreibweise: `0.1` (alternativ : `.1`)
- „Eng.“ Schreibweise: `1E2` (entspricht $1 \cdot 10^2$)
- **`float`** erzwingen: `10.0f`
- **`double`** erzwingen: `10.0d`

double und float sollten nicht genutzt werden, wenn hohe Präzision erforderlich ist (siehe [1])

Literale
Datentypen - Zeichen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Zeichen in Java: **`char`** 一个单独的字母 “char”

In einfachen Anführungszeichen (`'`) eingeschlossen

- Repräsentieren immer genau **ein einzelnes** Zeichen
- Mehr als ein Zeichen - nicht möglich (dafür Strings → kommt später)
- Beispiele: `'a'` (das Zeichen a), `'ü'` (das Zeichen ü)

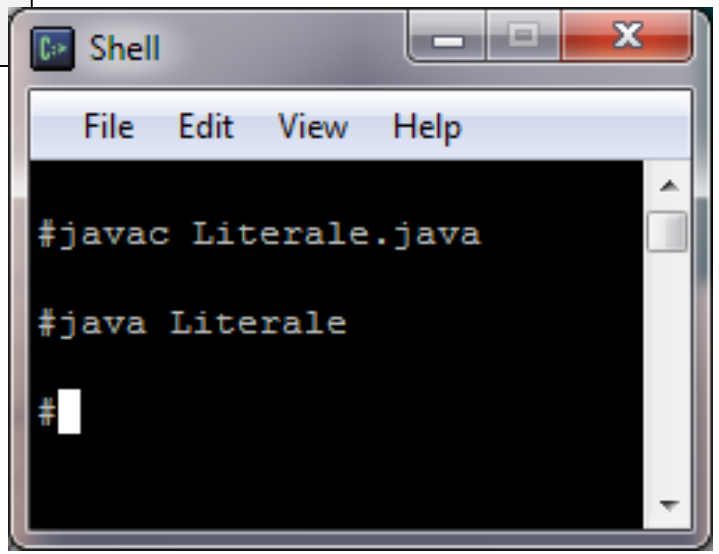
Manche Zeichen werden mit Backslash (`\`) **„escaped“**

- `'\''` → Das Zeichen '
- `'\\'` → Das Zeichen \ selbst
- `'\n'` → Zeilenumbruch
- `'\u00F1'` → Das Unicode-Zeichen mit der Nummer $F1_{16}$ (241_{10}): ñ
- Details unter [4]

Literale
Beispiel

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
1  public class Literale {
2     public static void main(String[] args) {
3           int i = 1;
4           Long l1 = 1L;
5           float f1 = 1.3f;
6           float f2 = 1E1f;
7           double d1 = 2.3d;
8           double d2 = 2.4;
9           char c1 = '\n';
10          char c2 = 'a';
11    }
12 }
```



Variablen und Datentypen

Arrays

Arrays
Zugriff und Zuweisung

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Wir benötigen zehn Fließkommazahl-Variablen!

→ `double d0, d1, d2, d3, d4, d5, d6, d7, d8, d9;`

Einfacher: Verwendung sog. Arrays (Felder)

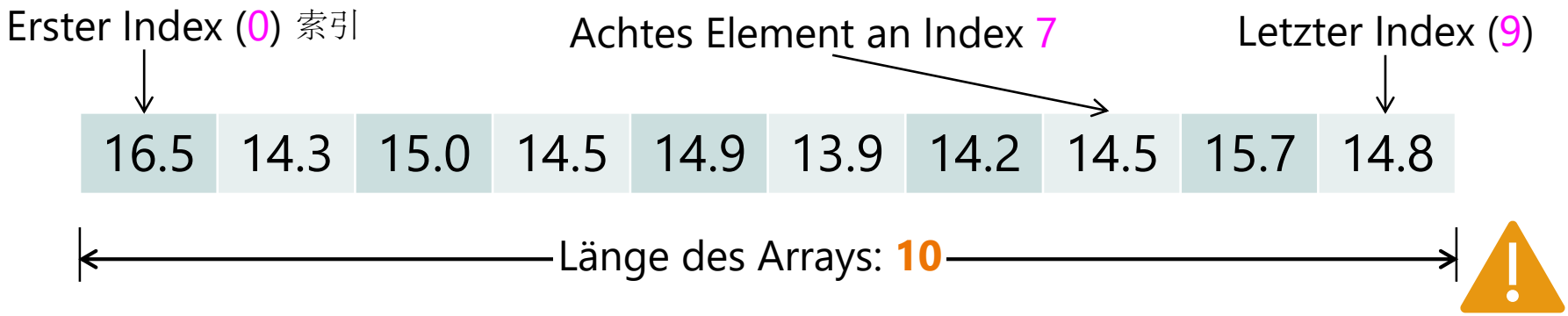
- `double[] d = new double[10];`

Zugriff - Lesend

`d[0], d[1], … , d[9]`

Zugriff - Schreibend

`d[0]=16.5;` `d[1]=meineZahl;`



Arrays
Deklaration und Initialisierung

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Deklaration 给Array定义

Syntax: **<Typ>** []<Name>; oder **<Typ>**[] <Name>;
oder **<Typ>** <Name>[];

Semantik: Deklariere ein Array mit dem Identifier <Name> vom Typ **<Typ>**

Initialisierung 给Array赋值

Syntax: **new** **<Typ>**[<Anzahl>]; oder
{ <Wert1> ,<Wert2> , <Wert3> };

Semantik: Reserviere Speicher für <Anzahl> neue Werte vom Typ **<Typ>** Variablen

Beispiel:

```
3  double[] laufZeiten = new double[10];
4  double []laufzeiten = new double[10];
5  double lz[] = new double[10];
6  double[] d = { 16.5 ,14.3, 15.0 };
```

Arrays
Property Länge

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Die Länge von Arrays ist als sog. Objektvariable abfragbar

- Komfortfeature von Java (nicht in allen Sprachen)
- Länge = Anzahl der Elemente
  - **Nicht vermischen mit Indizes** (Wir fangen bei 0 an zu zählen)
  - Länge = 10 → Index von 0-9;

Syntax: `<arrayName>.length` Array的长度

Semantik: Gibt eine Wert vom Typ **int** zurück, welcher der Anzahl an Elementen des `<arrayName>` entspricht

Beispiel:

```
3  int[] i = new int[10];
4  int[] t = new int[20];
5  i.length → ergibt 10
6  t.length → ergibt 20
```

Arrays
Beispiel: „args“-Array

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

```
1  public class ArgsArray {
2       public static void main(String[] args){
3          System.out.println("Erstes Argument: "+ args[0]);
4       }
5  }
```

„args“-Array, dient zur Übergabe von Parametern an das Java-Programm

- Parameter werden auch Argumente, Kommandozeilenparameter etc. genannt

Aufruf auf der Kommandozeile (auf der Konsole, im Terminal):

- `> java ArgsArray Hallo Welt`

Leerzeichen → zwei Argumente

Konsolenausgabe:

- Nur erstes Argument: `Hallo`

Variablen und Datentypen

Klassen

Klassen
Zeichenketten - Strings

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Nicht alles lässt sich komfortabel mit den primitiven Datentypen abbilden.

Deshalb: Klassen (später dazu mehr)

Beispiel - Zeichenketten in Java:

- Die Klasse **String** 文本“String”
- In doppelten Anführungszeichen (") eingeschlossen
- Wird verwendet wie primitiven Datentypen

Deklaration:

- `String s;`

Deklaration (Wiederholung):
Namen und Datentyp eines Speicherplatzes festlegen

Initialisierung/Wertzuweisung:

- `s = new String("Hallo Welt!");`
- `s = "Hallo Welt!";`
- `s = "Hallo " + "Welt!";`

Initialisierung (Wiederholung):
Erste Zuweisung eines Wertes an einen Speicherplatz

Initialisierung von Objekten mittels Konstruktor:
Eigentlich sind **Strings** Klassen, die mit **new Objekttyp();** initialisiert werden.
Strings haben eine Sonderstellung in Java, da diese wie primitive Datentypen initialisiert werden können.

Klassen
Zeichenketten - Strings

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

String Länge

- Bei **`Strings`** ist dies keine *Property* sondern eine sog. *Zugriffsmethode* welche einen Wert vom Typ **`int`** zurückgibt
- Verhält sich sonst wie `array.length` 文本的长度

Syntax:
```
<stringName>.length();
```

Strings verbinden/anhängen

Syntax:
```
string1.concat(string2);
```
Semantik: `string1` erweitert um `string2`

Beispiel:
```
"My name is ".concat("Zara");
s = "Hallo " + „World!";
```

Klassen
Wrapperklassen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Für jeden **primitiven Datentyp** gibt es eine sog. Wrapperklassen 包装类

- **Byte**, **Short**, **Integer**, **Long**, **Float**, **Double**, **Boolean** und **Character**
- Braucht man z.B. für **Listen**
- Bieten nützliche **Funktionalitäten** (Methoden) und **Konstanten**

Verschiedene Varianten der Initialisierung:

```
Integer zahl = new Integer(5);
Integer zahl = Integer.valueOf(„5“);
Integer zahl = 5;
```

Später hierzu mehr

Zusammenfassung
Variablen und Datentypen

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Eine Variable/Konstante ist ein **Speicherplatz**

- Repräsentiert einen **Wert** aus einem **bestimmten Wertebereich**
- Der Wertebereich wird durch verwendeten **Datentyp** festgelegt
- Speicherplatz wird über einen **Identifier/Namen** referenziert
- Bei Konstanten gilt: Der Wert ist **unveränderlich**

Deklaration und Initialisierung

- Variable: `<Typ> <Name> = <Wert>;`
- Konstante: `final <Typ> <Name> = <Wert>;`

Literale sind **konstante Werte** im Quelltext

Variablen und Literale sind immer von einem bestimmten Datentyp

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Kontakt

Raphael Allner, M. Sc.
Wissenschaftlicher Mitarbeiter
Institut für Telematik

Universität zu Lübeck
Ratzeburger Allee 160
23562 Lübeck

https://www.itm.uni-luebeck.de/mitarbeitende/raphael-allner.html

Literatur und Web-Links

1. Oracle and/or its affiliates: „Primitive Data Types“ https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

2. Oracle and/or its affiliates: „Java Language Keywords“ https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

3. Sun Microsystems, Inc.: „Naming Conventions“ https://www.oracle.com/technetwork/java/javase/documentation/codeconventions-135099.html#367

4. Oracle and/or its affiliates: „Characters“ https://docs.oracle.com/javase/tutorial/java/data/characters.html