ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Grundelemente der Sprache Java

Beispiel: Hallo Welt

```
/**
 * Die Klasse HalloWelt implementiert eine Anwendung, die den Text
 * "Hallo Welt" ausgibt
 */
public class HalloWelt {

    /** Start-Methode zum Starten eines Java-Programms */
    public static void main(String[] args) {
        System.out.println("Hallo Welt!");
    }
}
```

Beispiel: Hallo WeltBesser

```
/**
 * Die Klasse HalloWeltBesser implementiert die HalloWelt-
 * Anwendung objektorientierter
 */
public class HalloWeltBesser {
    /** Konstruiert ein Objekt der Klasse HalloWeltBesser  */
    public HalloWeltBesser() {}
    /** start-Aktion des Objekts      */
    public void start() {
        System.out.println("Hallo bessere Welt!");
    }
    /**  Start-Methode zum Starten eines Java-Programms      */
    public static void main(String[] args) {
        new HalloWeltBesser().start();
        // Erzeuge ein Objekt der Klasse HalloWeltBesser und rufe
        // die Methode  start auf
    }
}
```


Beispiel: HalloWeltSchlecht

```
public class HalloWeltSchlecht {public
HalloWeltSchlecht(){}public void
start(){System.out.println("Hallo schlechtere
Welt!");}public static void main(String[]
args){new HalloWeltSchlecht().start();}}
```

Programme, Klassen, Objekte und Pakete

- **Programm**
  - ⇨ Ein Java-"Programm" besteht aus einer Menge von Klassen. Beim Übersetzen wird für jede Klasse eine .class-Datei erzeugt.
  - ⇨ Klassen, die eine `main()`-Methode besitzen, können zum Starten eines Programms verwendet werden.
- **Klasse**
  - ⇨ Klassen bestehen aus Daten, den sogenannten Attributen, und Methoden.
  - ⇨ In Methoden können wiederum Daten in Form von lokalen Variablen definiert sein und Anweisungen stehen.

Programme, Klassen, Objekte und Pakete

- **Paket**
  - ⇨ Jede Klasse muss einem Paket zugeordnet werden. Dies wird mit der package-Anweisung, die die erste Anweisung sein muss, durchgeführt. Große Programmsysteme werden in Pakete untergliedert.

```
package de.htw.saarland.stl;
```

  - ⇨ Paketnamen werden mit `.` strukturiert und entsprechen Pfadnamen im Dateisystem

```
java.lang           →    java/lang
java.awt.peer       →    java/awt/peer
de.htw.saarland.stl →    de/htw/saarland/stl
```

Programme, Klassen, Objekte und Pakete

- **Paket**
  - ⇨ Der Paketname wird üblicherweise gebildet aus dem umgekehrten Domain-Namen der entwickelnden Institution. Z. B.
    - `com.sun.security`
    - `org.omg.CORBA`
  - ⇨ Der Paketname ist Bestandteil des Klassennamens, braucht aber innerhalb des Paketes nicht mit angegeben zu werden.
  - ⇨ Das Paket `java.lang` ist voreingestellt, d. h. bei den Klassen `java.lang.System` und `java.lang.String` braucht der Paketname generell nicht mit angegeben zu werden.

Module

- Seit Java 9 gibt es ein neues Konzept: **Module**
- Die komplette Java API ist nun in Module untergliedert, die wiederum aus Packages bestehen.
  - ⇨ Z. B. umfasst das Modul **java.bas**e die fundamentalen Packages der Java SE Plattform
  - ⇨ Siehe auch die Dokumentation von Java 9
- Das Konzept wird vorerst noch nicht besprochen.

Programme, Klassen, Objekte und Pakete

- **Objekt**
  - ⇨ Wenn ein Java-Programm über die main-Methode einer Klasse gestartet wird, dann werden im Normalfall Objekte der beteiligten Klassen erzeugt.
  - ⇨ In diesen Objekten läuft dann die eigentliche Funktionalität des Programms ab.
  - ⇨ Objekte in einem Programm entsprechen dem, was man in der realen Welt auch unter Objekten versteht, bilden jedoch ein stark abstrahiertes Abbild des realen Objekts im Hauptspeicher.

Zeichenvorrat

- Große und kleine Buchstaben werden von Java als unterschiedlich betrachtet.
- Theoretisch sind alle Buchstaben des Unicode-Standards (www.unicode.org) erlaubt
- Nationale Sonderzeichen wie z.B. deutsche Umlaute sollen aber generell nicht für Namen innerhalb von Java-Programmen verwendet werden.

Buchstaben	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z _ $
Ziffern	0 1 2 3 4 5 6 7 8 9
Sonderzeichen	{ } [ ] ( ) < > = : + - * / % & | , ; ? ~ ^ " '
Leerzeichen und Sonderzeichen	Leerzeichen (Blank), Zeilenendezeichen (Carriage Return und Linefeed), horizontaler Tabulator, Seitenvorschub

Kommentare

Einzeilenkommentar:

```
// der Rest der Zeile ist Kommentar
```

Klassischer C-Kommentar

```
/* Dies ist ein C-Kommentar
   der ueber mehrere Zeilen
   geht
*/
```

Javadoc-Kommentar

```
/** Dieser Kommentar wird von
 * dem Tool javadoc
 * in automatisch generierte
 * Programmdokumentationen
 * übernommen
 */
```


Bezeichner

- In einem Programm braucht man zur Bezeichnung von gewissen Größen Namen, auch Bezeichner (engl. *identifier*) genannt.
- Bezeichner werden benötigt, um Konstanten, Variablen, Klassen und Methoden Namen zu geben, unter denen man diese ansprechen können soll.
  - ⇨ Namen können (fast) frei erfunden werden, müssen aber aus Buchstaben, Ziffern und dem Unterstrich-Zeichen (_) zusammengesetzt werden. Das erste Zeichen darf keine Ziffer sein.

Bezeichner

- Korrekte Beispiele für Bezeichner sind:
  - `a`
  - `DasIstEinZwarZiemlichLangerAberTrotzdemZulaessigerVariablenbezeichner`
  - `Das_ist_ein_zwar_ziemlich_langer_aber_trotzdem_zulaessiger_Bezeichner`
  - `a0000001`
  - `_`
- Unzulässige Beispiele sind:
  - `1teVariable`
  - `Erste-Variable`
  - `Erste Variable`
  - `DasIstEinLangerUnzulaessigerVariablenbezeichner!`
  - `dies+das`

Namenskonventionen

- Generell werden nur ASCII-Zeichen verwendet, kein '$', meist auch kein '_'
- Klassennamen beginnen immer mit einem Großbuchstaben, z.B. `String`
- Methodennamen beginnen immer mit einem Kleinbuchstaben. Besteht der Name aus mehr als einem Wort so werden die Wörter zusammen geschrieben und jedes außer dem ersten Wort (ein Verb) beginnt mit einem Großbuchstaben, z.B. `getString()`
- Attributnamen bzw. Datenfeldnamen werden wie Methodennamen gebildet, z.B. `int aPrivateVar;`
- Konstantennamen sollen üblicherweise in Großbuchstaben gebildet werden. Einzelne Wörter werden dabei durch '_' voneinander getrennt, z.B. `Integer.MIN_VALUE`, `Double.POSITIVE_INFINITY`.

Reservierte Wörter

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

- `true`, `false` und `null` sind sogenannte literale Konstanten und können daher auch nicht als Namen verwendet werden.
- Die Schlüsselwörter `const` und `goto` sind reserviert, werden aber nicht benutzt.

Beispiel: SummeTest

```
import java.util.Scanner;

public class SummeTest {
    /** eigentliche Startmethode                    */
    public void start() {
        int a, b;
        int summe;
        Scanner input = new Scanner(System.in);
        System.out.print("a = ");
        a = input.nextInt();
        System.out.print("b = ");
        b = input.nextInt();
        summe = a + b;
        System.out.println("Summe = " + summe);
    }

    public static void main(String[] args) {
        new SummeTest().start();
    }
}
```

Standardklasse zum Lesen von Datenströmen

Scanner-Objekt anlegen, das die Standardeingabe liest

Zeichenkette ausgeben ohne Zeilenwechsel

Nächste ganze Zahl lesen

Lokale Variablen

- Werden Variablen innerhalb einer Methode oder eines Blocks (Begriff später) definiert, so handelt es sich um **lokale Variablen**, die nur so lange angelegt bleiben bis die Methode bzw. der Block durchlaufen sind.
- Variablen belegen einen Platz im Hauptspeicher und sind nach ihrer Definition noch ohne konkreten Wert.
- Variablen müssen vor ihrer Verwendung einen definierten Wert erhalten. Dies kann entweder über eine Initialisierung

```
int summe = 0;
double flaeche = 0.0;
```

oder eine Zuweisung geschehen

```
summe = 100;
flaeche = 100.0;
```


Beispiel: KreisBerechnung (1)

```
import java.util.Scanner;
public class KreisBerechnung {
    public void start() {
        final double PI;
        PI = 3.14159265358979323846;
        System.out.println("Kreisberechnung:");
        System.out.println("PI = " + PI);

        double radius;
        Scanner input = new Scanner(System.in);
        System.out.print("Radius = ");
        radius = input.nextDouble();
        double flaeche = radius * radius * PI;
        double umfang  = 2 * PI * radius;

        System.out.println("Flaeche : " + flaeche);
        System.out.println("Umfang  : " + umfang);
    }
```

Beispiel: KreisBerechnung (2)

```
    public static void main(String[] args) {
        new KreisBerechnung().start();
    }
}
```


Lokale Konstanten

- Konstanten sind Variablen, denen nur genau einmal ein Wert zugewiesen werden darf, entweder durch Initialisierung oder durch eine erste Zuweisung. Danach können sie nicht mehr verändert werden.
- Zur Definition von Konstanten wird das Schlüsselwort final verwendet.

```
final double MAX;
final double PI = 3.14159265358979323846;
MAX = 300.0;    // ok
MAX = 400.0;    // Fehler !!
```

Operatoren und Ausdrücke

- Um mit Zahlen zu rechnen oder Variableninhalte zu manipulieren, gibt es in Java eine Vielzahl von Operatoren. Wir betrachten zunächst nur die arithmetischen Operatoren.

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulo-Operator

- Variablen, Konstanten, Ganzzahl- und Gleitpunktzahl-Werte können nun mit Hilfe von Operatoren und Klammern zu Ausdrücken kombiniert werden.

Einfache Ein-/Ausgabe

```
System.out.println("Flaeche : " + flaeche);
// Ausgabe eines Textes und eines Variableninhaltes
// auf die Standardausgabe mit Zeilenwechsel

System.out.print("Radius = ");
// Ausgabe eines Textes ohne Zeilenwechsel

radius = input.nextDouble();
// Einlesen eines double-Wertes von der
// Standardeingabe

a = input.nextInt();
// Einlesen eines int-Wertes von der
// Standardeingabe
```

Beispiel: Zinsberechnung

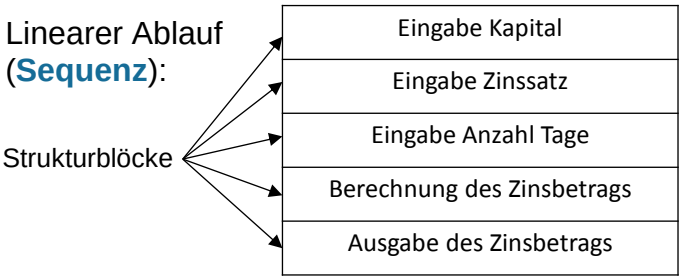
- Die Formel für die Zinsberechnung sieht folgendermaßen aus

$$Zinsen = \frac{Kapital \cdot Zinssatz \cdot Tage}{100 \cdot 360}$$

- Vorgehensweise bei der Programmerstellung:
  - ⇨ erst nachdenken, dann programmieren!
  - ⇨ Strukturieren des Problems z. B. mit Hilfe von Pseudocode oder Struktogrammen
  - ⇨ Umsetzen in die Programmiersprache
  - ⇨ Testen

Struktogramm

- Ein **Struktogramm** oder auch **Nassi-Shneiderman-Diagramm** ist ein Diagrammtyp zur Darstellung von Programmabläufen.
- Struktogramme sind geeignet um Algorithmen zu visualisieren, werden aber nicht mehr so häufig eingesetzt.
- Linearer Ablauf (**Sequenz**):



Beispiel: Zins1 (1)

```
public class Zins1 {
    private static final int TAGE_PRO_JAHR = 360;
    private static final int HUNDERT = 100;
    private Scanner input = new Scanner(System.in);

    public void start() {
        double kapital, zinssatz, zinsen;
        int tage;
        System.out.print("Kapital  = ");
        kapital = input.nextDouble();
        System.out.print("Zinssatz = ");
        zinssatz = input.nextDouble();
        System.out.print("Tage     = ");
        tage = input.nextInt();

        zinsen = kapital * zinssatz * tage
               / (TAGE_PRO_JAHR * HUNDERT);

        System.out.println("Die Zinsen betragen € " + zinsen);
    }
```

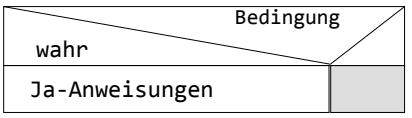

Beispiel: Zins1 (2)

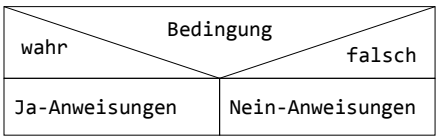
```
    public static void main(String[] args) {
        new Zins1().start();
    }
}
```

Verzweigung (if-Anweisung)

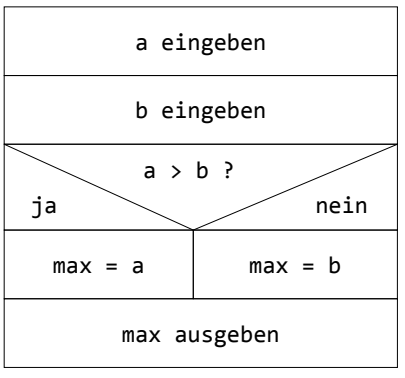
- Die if-Anweisung dient zur Verzweigung in Alternativabläufe eines Programms.

	Syntax	***Struktogrammdarstellung***

einseitige Alternative	`if (Bedingung) {`
	`  Ja-Anweisungen`
	`}`
		

zweiseitige Alternative	`if (Bedingung) {`
	`  Ja-Anweisungen`
	`} else {`
	`  Nein-Anweisungen`
	`}`
		

Beispiel: Maximum zweier Zahlen

Pseudocode	***Struktogramm***
• Eingabe der ersten Zahl a • Eingabe der zweiten Zahl b • Falls a größer als b ist, setze das Maximum max auf a, sonst auf b • Gebe das Maximum aus	

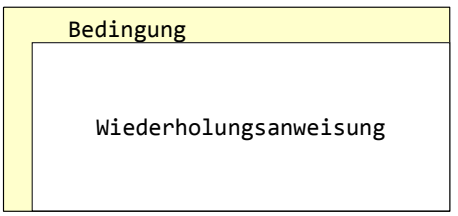
Beispiel: MaximumTest

```
public class MaximumTest {
    private Scanner input = new Scanner(System.in);

    public void start() {
        int a, b, max;
        System.out.print("a = ");
        a = input.nextInt();
        System.out.print("b = ");
        b = input.nextInt();

        if (a > b) {
            max = a;
        } else {
            max = b;
        }
        System.out.println("max = " + max);
    }
    public static void main(String[] args) {
        new MaximumTest().start();
    }
}
```


Die while-Schleife

- Die while-Anweisung ist eine Schleifenanweisung, d. h. abhängig von der Erfüllung einer Bedingung wird ein Anweisungsblock immer wieder wiederholt.

Syntax	*Struktogramm*

`while (Bedingung) {`
`  Wiederholungsanweisung`
`}`
	

Beispiel: Berechne die Summe der ersten n Zahlen

Pseudocode	*Struktogramm*
• Eingabe der Zahl n • Setze Zähler i auf 1 • Setze Summe auf 0 • Solange i noch kleiner oder gleich n – erhöhe Summe um i – erhöhe i um 1 • Gebe die Summe aus	n eingeben i = 1 summe = 0 solange i <= n summe = summe + i i = i + 1 summe ausgeben

Beispiel: SummeTest2

```
public class SummeTest2 {
     private Scanner input = new Scanner(System.in);

     public void start() {
        int i = 1;
        long summe = 0;
        long n;
        System.out.print("n = ");
        n = input.nextInt();

        while (i <= n) {
           summe = summe + i;
           i = i + 1;
        }

        System.out.println("summe = " + summe);
        System.out.println("Mit der Summenformel:");
        System.out.println("summe = " + n*(n+1)/2);
    }
```

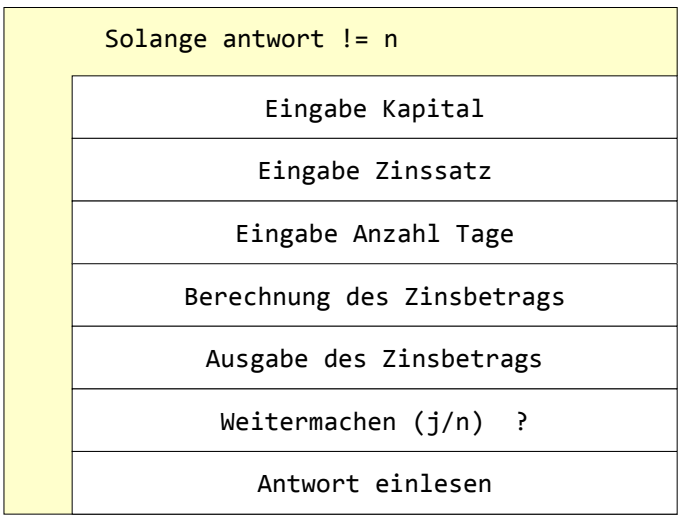
Erweiterung des Beispiels Zinsberechnung

- Die Zinsberechnung soll um eine Schleife erweitert werden, so dass der Benutzer des Programms nach Ausgabe des errechneten Zinsbetrages gefragt wird:

```
Weitermachen (j/n) ?
```

- Für die Eingabe einer Antwort benötigen wir noch einen weiteren Datentyp, den Datentyp `char`.
- Der Datentyp `char` belegt 2 Bytes im Speicherplatz und kann ein Zeichen des UNICODE-Zeichensatzes aufnehmen.

Erweiterung des Beispiels Zinsberechnung



Beispiel: Zins2

```
public void start() {
    double kapital, zinssatz, zinsen;
    int tage;
    char antwort = ' ';

    while (antwort != 'n') {
        System.out.print("Kapital  = ");
        kapital = input.nextDouble();
        System.out.print("Zinssatz = ");
        zinssatz = input.nextDouble();
        System.out.print("Tage     = ");
        tage = input.nextInt();

        zinsen = kapital * zinssatz * tage
           / (TAGE_PRO_JAHR * HUNDERT);

        System.out.println("Die Zinsen betragen € " + zinsen);
        System.out.print("Weitermachen (j/n) ? ");
        antwort = input.next().charAt(0);
    }
}
```

input.next() liefert das nächste Token als String, charAt(0) gibt das erste Zeichen zurück