JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Heute: Java

www.uni-frankfurt.de

- Programmstruktur
- Datentypen
- Schlüsselwörter
- Operatoren
- Kontrollstrukturen
- Klassen, Methoden, Attribute
- Strings, Arrays, Pakete

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

www.uni-frankfurt.de

Teil 4 - Java

- Programmstruktur
- Datentypen
- Schlüsselwörter
- Operatoren
- Kontrollstrukturen

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Kommentare in Java

In Java gibt es **drei** Möglichkeiten zur Kommentierung:

`// Kommentar`	Alle Zeichen nach dem „//“ werden ignoriert. ➔für „normale“ einzeilige Kommentare.
`/* Kommentar` `über mehrere Zeilen */`	Alle Zeichen zwischen dem „/*“ und dem „*/“ werden ignoriert. ➔ auskommentieren von **Programmteilen.** Die Schachtelung der „Kommentar-klammern“ ist **nicht** erlaubt.

`/**`
`* javadoc-Kommentar`
`* @since JDK1.0`
`* /`
	Alle Zeichen zwischen dem „/**“ und dem „*/“ werden ignoriert.
	➔ für das **javadoc**-Programm des JDK. Mit ihrer Hilfe wird eine einfache **Online-Dokumentation** erstellt.

Primitive Datentypen in Java

Die übliche Struktur in Java ist das **Objekt**.

Es gibt in Java aber auch **einfache (primitive) Datentypen**.

Typ	*Inhalt*	*Default*	*Bereich*
`boolean`	Wahrheitswert true oder false	false	true oder false
`char`	Unicode-Zeichensatz	\u0000	\u0000 bis \uFFFF
`byte`	Ganze Zahl mit Vorzeichen (8 Bit)	0	-128 bis 127
`short`	Ganze Zahl mit Vorzeichen (16 Bit)	0	-32768 bis 32767
`int`	Ganze Zahl mit Vorzeichen (32 Bit)	0	-2^{31} bis $2^{31}-1$
`long`	Ganze Zahl mit Vorzeichen (64 Bit)	0	-2^{63} bis $2^{63}-1$
`float`	Gleitkommazahl (32 Bit)	0.0	±1.4E-45 bis ±3,4E+38
`double`	Gleitkommazahl (64 Bit)	0.0	±4.9E-324 bis ±1.8E+308

Einfachen Datentypen haben also einen **vorgegeben Standardwert**.

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Primitive Datentypen

Ganzzahltypen

- `byte, short, int, long`
- werden intern im 2er-Komplement dargestellt (vorzeichenbehaftet).

Gleitkommatypen

- `float` – Genauigkeit von 6 Nachkommastellen
- `double` – Genauigkeit von 15 Nachkommastellen
- werden intern nach IEEE dargestellt

Primitive Datentypen

Zeichentyp

- `char` (ein Zeichen aus dem Unicode)
- ASCII Code ist ein Untermenge vom Unicode

- Unicode (Norm zur Zeichendarstellung)
  2 Byte (65536 Zeichen)
  http://de.selfhtml.org/inter/unicode.htm
- ASCII Code
  \u0000 bis \u007F
  http://www.unicode.org/charts/PDF/U0000.pdf
  Beispiel: `char wasIstDas= '\u0040';`

Primitive Datentypen

Wahrheitstyp
(für logische Abfragen)

- `boolean`
  Der Wert kann nur `false` oder `true` annehmen.

- Anmerkung: In C/C++ können die Wahrheitswerte `false` und `true` durch die Zahlen 0 und 1 ausgedrückt werden. Das ist aber in JAVA nicht möglich.

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Variablen

Jede Variable muß deklariert werden

(Deklarationen sind Anweisungen)

```
byte b;
long l;
char einZeichen;
boolean warOderFalsch;
int i1, i2;
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Schlüsselwörter in Java

In Java gibt es eine Reihe von **reservierten Schlüsselworten**.

Sie dürfen **nicht** für eigene Bezeichner verwendet werden.

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

Die mit * gekennzeichneten Wörter sind zwar **reserviert**,
werden aber in der Version 1.5 von Java (noch) **nicht** verwendet.

Variablen

Zuweisungen und Initialisierungen

- Eine Integer Zahl

```
int i;          // Deklaration

i = 37;         // Wert zuweisen
```

- Ein einzelnes Zeichen

```
char jaZeichen;

jaZeichen = 'J';     // nicht 'j'
```

Variablen

- Die Zuweisung und Initialisierung ist auch in einem Schritt möglich

```
int i = 37;

char c = 'X';

boolean stimmts = true;

double db = 0.3245;
```

Variablen

- Eine Variable kann überall im Quellcode deklariert werden
- Globale Variablen sind im ganzen Programm sichtbar
- Lokale Variablen sind nur im Unterprogramm (Block) sichtbar
- Konvention: Variablendeklaration nur am Anfang oder am Ende des Quellcodes (bessere Lesbarkeit)

Variablen

Typumwandlung

```
byte > short > int > long > float > double
```

- Binäre Operationen auf numerischen Werten unterschiedlichen Typs sind möglich und werden intern umgewandelt.
- Bei der Umwandlung in einen kleineren Typ können Informationen verloren gehen.

Variablen

Beispiel:

```
float f  = 2.5F;
int i    = 3;

int j   = i * f;
//Fehler, da Ergebnis float

float g = i * f; // OK
```

Variablen

Explizite Typumwandlung (casting)

```
double x = 9.97;
int i;

i=(int)x;
//Informationen gehen verloren
```

- Gültigkeitsbereich sollte vorher überprüft werden

Konstanten

Konstanten - einmalige Zuweisung

```
final double g = 9.81;
```

- eine weitere Zuweisung ist nicht mehr möglich (z.B. g = 10.03;)
- es gibt globale und lokale Konstanten.

Wrapper-Klassen

Alle primitive Datentypen haben eine korrespondierende Klasse.

Z.B. kapselt die Klasse Integer einen int Wert:

```
int a = 5;

Integer intOb = new Integer(a);
```

siehe API: java.lang.Integer

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Operatoren in Java

Operatorzeichen	*Art der Operatoren*	*Auswertungsreihenfolge*
++ -- + - ~ !	Unäre Operatoren	-
* / %	Multiplikationsoperatoren	von links nach rechts
+ -	Additionsoperatoren	von links nach rechts
<< >> >>>	Verschiebeoperatoren	von links nach rechts
< <= > >=	Vergleichsoperatoren	von links nach rechts
== !=	Gleichheitsoperatoren	von links nach rechts
&	Bitoperator UND	von links nach rechts
^	Bitoperator exklusives ODER	von links nach rechts
|	Bitoperator inklusives ODER	von links nach rechts
&&	Logisches UND	von links nach rechts
?:	Bedingungsoperator	von rechts nach links
= += -= *= /=	Zuweisungsoperatoren	von rechts nach links

- Treten **mehrere** Operatoren **zusammen** auf, werden sie in der **Reihenfolge ihrer Priorität** ausgeführt. (➔ vgl. mit der Mathematik: *„Punkt vor Strich“*)
- Die **Priorität** nimmt in der Tabelle von oben nach unten ab.

Operatoren - Beispiele

- bei ganzen Zahlen (byte, short, int, long)

```
15 / 4 = 3    liefert ganzzahligen Anteil

15 % 2 = 1    liefert den Rest
```

- Bei Gleitkommazahlen

```
9.5 / 2 = 4.75    normale Division
```

- Normale Division erzwingen:

```
(double)10 / 4 = 2.5
```

Abkürzungen

```
x = x+3;   ←→   x += 3;
```

- Äquivalent für die anderen Operatoren

```
x = x / 9;   ←→   x /= 9;
x = x * 9;   ←→   x *= 9;
x = x - 9;   ←→   x -= 9;
x = x % 9;   ←→   x %= 9;
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Inkrement und Dekrement

- Inkrementieren

```
int m = 7;
m++;
```

`m` hat nun den Wert 8

- Dekrementieren

```
int m = 7;
m--;
```

`m` hat nun den Wert 6

Operatoren

Vorsicht!:

```
int m = 7;
```

Präfix:

```
int a = 2 * ++m;          // a=16, m=8
```

oder Postfix:

```
int b = 2 * m++;          // b=14, m=8
```

Funktionen

- Für weitere Operationen, die man in der Regel auf einem Taschenrechner findet, wie z.B.
  - `log(double x)`
  - `pow(double x, doubel y)`
  - `sqrt(double x)`

  gibt es die Klasse `java.lang.Math`, in der diese Funktionen vorgefertigt sind.

Relationale und bool´sche Operationen

```
(3 == 7)  Gleichheit, Ergebnis false
(3 != 7)  Ungleichheit, Ergebnis true
(3 >= 7)  false
(3 <= 7)  true
```

bool´sche Werte können auch verknüpft werden:

```
(3 == 7) && (4 > 2) logisches UND, Ergebnis false
(3 == 7) || (4 > 2) logisches ODER, Ergebnis true
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Bitoperationen

können auf die **Ganzzahltypen** angewendet werden

```
>>  shift right
<<  shift left
&   AND
|   OR
^   XOR
~   NOT
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Bitoperationen

Beispiel:

```
int foo = 16;
int viertesBit = (foo & 8) / 8;
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Kontrollstrukturen in Java

- if
- switch
- while
- for

Bedingungsanweisungen (`if`)

```
if (Bedingung) Anweisung1;

if (Bedingung) {Block}

if (Bedingung) Anw1 else Anw2;

if (Bedingung) {Block} else {Block}
```

- Abkürzung: `Bedingung ? Anw1 : Anw2`
  Bsp: `(x<y) ? x : y`

Mehrfachauswahl (`switch`)

- Soll eine Variable auf mehrere Werte überprüft werden, dann kann die `If`-Anweisung unübersichtlich werden.

- Eine `switch`-Anweisung ist für solche Fälle besser geeignet.

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Mehrfachauswahl (switch)

```
int wahl =
 Console.readInt("Eine Option wählen (1 bis 4)");

switch (wahl)
{
      case 1:   . . . break;
      case 2:   . . . break;
      case 3:   . . . break;
      case 4:   . . . break;
      default:   . . . break;
}
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Mehrfachauswahl (`switch`)

- Die Variable `wahl` muß vom Typ
  - `char`
  - `byte`
  - `short`
  - `int`
- Nicht erlaubt sind:
  - `long`
  - `float`
  - `double`
  - `boolean`

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Unbestimmte Schleifen (`while`)

Nur wenn die Bedingung wahr ist, wird der Block ausgeführt:

```
while (Bedingung)
{
        Block
}
```

Der Block wird auf jeden Fall einmal durchlaufen:

```
do
{
        Block
}
while (Bedingung);
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Bestimmte Schleifen (`for`)

Anzahl der Durchläufe muss vorher bekannt sein:

```
int i;

for (i = 1; i <= 10; i++)
{
     System.out.println(i);
}
```

Bestimmte Schleifen (`for`)

```
for (x=0; x !=10; x+=0.01) {
   ....
}
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Übersicht der Klammerarten

- () Methoden

```
person.getName();
ma.getGehalt(int jahr);
```

- {} Blöcke

```
{Anw1 {Anw2;Anw3} Anw4}
```

- [] Index der Felder

```
neZahl = stapel[3];
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

www.uni-frankfurt.de

Teil 5 - Java

NOCHMAL:
- Klassen
- Methoden, Attribute

Noch mal im Detail: Grundlegende Programmstruktur

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

package ...	Sichtbarkeit / Ordnersystem
import ...	Verfügbarmachen anderer Klassen
class X {	**Name** der Klasse
int nummer=0; **float zahl;** **...**	**Klassenkörper** Enthält Variablen, ...
X {...}	**Konstruktor** (heißt wie die Klasse, erzeugt das Objekt, hat kein Rückgabewert)
Methode1 {...} **Methode2 {...}**	Die verschiedenen **Methoden**, welche die eigentliche Funktionalität einer Klasse ergeben.
}	**Ende** der Klasse (des ~körpers)

Klassen

Syntax für eine Klasse

```
      ZugriffsSpezifikator class NameDerKlasse
[extends Oberklasse] [implements Schnittstelle]
      {
            // Attribute/Merkmale der Klasse;

            // Konstruktor der Klasse;

            // Methoden der Klasse;
      }
```

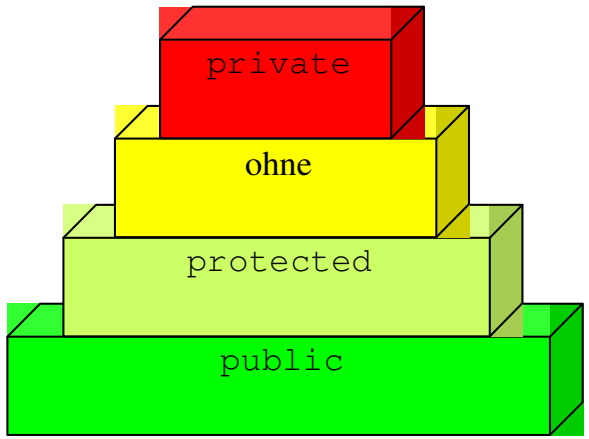
Zugreifbarkeit auf Methoden und Variablen

private – nur innerhalb der Klasse sichtbar und somit zugreifbar.

ohne – nur innerhalb des Pakets, das die Deklaration der Klasse enthält (*Standardzugriffsrecht*).

protected – zugreifbar von Methoden der eigenen Klasse, der davon abgeleitete Klassen und von Klassen im selben Paket.

public – für Methoden aller Klassen zugreifbar (welche die Klasse importieren).



Zugriffsmodifizierer

Klassen

In einer Datei darf nur eine Klasse `public` sein.

➔ Konvention : Pro Datei eine Klasse

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Klassen, Methoden und Variablen

Objekt student1 erzeugen:

```
Student student1 = new Student();
```

Methode aufrufen und Rückgabewert speichern:

```
int semanzahl = student1.getSemester();
```

Konstruktor:

```
Student() { … }
Student(int semester) { … }
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Attribute

Syntax für ein Attribut:

ZugriffsSpezifierer ***Typ*** *NameDesAttributs*

```
private int eineZahl = 11;
public int nochneZahl;  // default=0
```

Attribute

Der **Typ** eines Attributs kann ein einfacher/primitiver Datentyp, ein Feld oder eine Klasse sein.
Damit hat man die Möglichkeit, von einem Objekt auf ein anderes zu verweisen.

Der Name muß innerhalb des Klassenrumpfes eindeutig sein.

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Attribute

einfache/primitive Typen

```
private int neZahl;
```

komplexe Typen (auch eigene Klassen)

```
private Anschrift seineAnschrift;
public Button druckKnopf;
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Attribute: Klassenattribut **(static)**

Ein **Klassenattribut** ist nur für die Klasse gegeben, sein Wert ist unabhängig von den vorhandenen Exemplaren

```
private static int anzahlKunden;
```

Konstante: wie Klassenattribut und zusätzlich nicht änderbar

```
public static final MAX_KUNDEN = 100;
```

Klassenattribut (**static**)

Mit Hilfe von **static** kann man Klassenvariablen und Klassenmethoden erzeugen, die für alle Objekte der Klasse gleich sind.

```
public class Person {
            …
        static int countPerson;
        public Person() {
          countPerson++;
        }
        public static int getCountPersonen() {
            return countPerson;
        } …
```

Die Variable *countPerson* ist für alle Instanzen der Klasse Person gleich!

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Attribute

`private` = Datenkapselung!

nur das Objekt kann auf seine Variablen zugreifen (Black Box).

Empfehlung: Alle Attribute privat

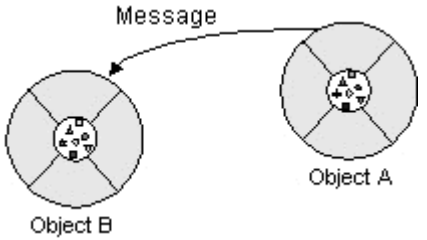
JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Attribute

```
public class Kunde
{
    // Attribute
   private String name;
   private Anschrift seineAnschrift;
   private static int anzahlKunden;
   public  static final MAX_KUNDEN = 100;
   ...
   // Methoden
   ...
}
```

Methoden

- Eine Methode/Operation ist eine Funktion, die auf die internen Daten (Attribute) eines Objekts Zugriff hat.
- Sie kann Botschaften an andere Objekte senden



- Auf alle Objekte einer Klasse sind dieselben Methoden anwendbar
- Abstrakte Methoden besitzen nur eine Signatur

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Methoden

Syntax der Methoden:

ZugriffsSpezifikator ***Rückgabetyp*** *NameDerMethode*
(Parametertyp Parametername, ...)

```
{
        // Anweisungen
}
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Methoden

Die Signatur einer Methode besteht aus dem Namen der Operation, den Namen und Typen aller Parameter, dem Typ des Ergebnistyps und der Bezeichnung der *Ausnahmebehandlung* (siehe nächste Vorlesung).

```
double getGehalt(String name) // kein -
{                             // Semikolon!
}
```

getter-, setter-Methoden

Accessor-Methode

- meldet den Zustand des Objektes

```
String name = Kunde.getName();
```

Mutator-Methode

- ändert den Zustand eines Objektes

```
Kunde.setName(„Mueller“);
```

Beispiel:

```
package mypackage;
import java.util.*;  /* nicht nötig */
/**
 *      Klasse die Studenten darstellen soll. */
public class Student {
        private int semester;
        public Student() {}   // default Konstruktor optional!

        /**
         *      Setzt die Semesterzahl. */
        public void setSemester(int sem) {
                semester=sem;
        }
        /**
         *      Gibt die Semesterzahl zurück. */
        public int getSemester() {
                return semester;
        } ...
}
```

Methoden

Der Rückgabetyp `void` bedeutet, dass nichts zurückgegeben wird.

```
public void nixZurueck()
```

Eine Klassenmethode wird nur auf die Klasse angewendet (auf die Klassenattribute), nicht auf deren Exemplare.

Methoden

CALL BY VALUE

In Java können Methoden die Werte ihrer Parameter nicht ändern, also kein call by reference wie in C:

Zeiger in C++:

```
void swap(int &a,int &b) {
   int tmp=a;
   a=b;
   b=tmp;
};
void main() {
   int a=1, int b=2;
   swap(a,b);
   cout<<"a: "<<a<<endl;  cout<<"b: "<<b<<endl;
};
```

Methoden

Funktioniert nicht:

```
static void kundeTauschen(Kunde a, Kunde b)
{
      Kunde temp = b;     // Objektvariable : erster
      b = a;               // Buchstabe klein
      a = temp;
}
```

Methoden

ABER: das funktioniert:

```
static void kundeTauschen(Kunde a, Kunde b)
{
        String name_a = a.getName();
        String name_b = b.getName();

        a.setName(name_b);
        b.setName(name_a);
}
```

Konstruktor

Konstruktor

- Operation, die ein neues Objekt einer Klasse erzeugt.
  Der Konstruktor heißt wie die Klasse. Die Syntax ist wie bei Methoden, jedoch **ohne** Rückgabewert!

```
public Kunde()
  {
        kundenId = getNewId();
  }
```

- versetzt bei Aufruf das Objekt in einen Anfangszustand.

```
einKunde = new Kunde();
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Konstruktor

Eine Klasse kann mehrere Konstruktoren besitzen (Überladen).

```
public Mitarbeiter(String n, double g){
       name   = n;
       gehalt = g;}

public Mitarbeiter(){}
```

Die Parametertypen und deren Anzahl können unterschiedlich sein.

Regeln für Namensgebung

- es wird zwischen Groß- und Kleinschreibung unterschieden,
- das erste Zeichen muß ein Buchstabe sein,
- Rest kann aus Zahlen, dem '_' und Buchstaben bestehen,
- Leerzeichen und andere Sonderzeichen (`+`, `©` , `#` , `~`, etc) sowie reservierte Wörter (Schlüsselwörter, siehe nächste Folie) sind nicht erlaubt.

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Bezeichner - Konvention

	Anfang	Rest
Paket	klein	durchgehend klein, Teilwort
Klasse	gross	Teilworte aneinandergereiht ohne Unterstriche, z.B. TextConverter
Variable	klein	
Methode	klein	Teilworte aneinandergereiht ohne Unterstriche, z.B. setText
Konstante	gross	durchgehen groß mit Unterstrichen

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

www.uni-frankfurt.de

Teil 6 - Java

Strings
Arrays
Pakete

Strings

Zeichenketten (`String`)

- Folgen von Zeichen

```
String s = "Hallo";
// char  'J' , String "J"

String b = "Du";
System.out.println(s + b + "!!!");
// Ausgabe c:\>HalloDu!!!
```

Strings

- Die Klasse String liefert viele Methoden, um eine Zeichenkette zu bearbeiten.

```
int i  = s1.length();
// Länge des Strings s1

char ch = s1.charAt(2);
// Zeichen des Strings s1 an der Position 2
```

Strings

Weitere Methoden der Klasse String

```
int compareTo(String anotherString)
String concate(String str)
String toUpperCase()
String trim()  //entfernt Leerzeichen
String valueOf(char c)
String replace(char oldChr,char newChr)
```

- und weitere

Strings

Um eine Zeichenkette zu analysieren kann die Klasse StringTokenizer verwendet werden.

```
StringTokenizer(String str, String delim)
```

Strings

Beispiel

```
String str ="Dies ist ein String!";
StringTokenizer stkn;
stkn = new StringTokenizer(str," ");
while (stkn.hasMoreTokens())
{
  System.out.println(stkn.nextToken());
}
```

String Vergleich

```
int a=1, b=2;
String x="abc", y="abc";

if(a=b) …
if(x=y) …
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Felder (Arrays)

Felder werden benötigt, um temporäre Listen von primitiven Typen oder Objekten zu verwalten.

z.B.: Eine Liste der Mitarbeiter…

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Arrays

Von jedem Typ lassen sich Felder definieren.

Deklaration:

```
int[] i;     // ein Integer-Feld
Mitarbeiter[] m; //Mitarbeiterfeld
```

Instanzierung:

```
i = new int[10];
m = new Mitarbeiter[100];
```

Arrays

Zuweisung

```
i[0] = 100;
i[1] = 258;
...
i[9] = 300;

int zahl = i[9];          //zahl = 300
```

Numerierung beginnt bei 0

Arrays

- Es gibt vorgefertigte Methoden, um Felder zu kopieren, zu sortieren, auszugeben etc.:

  Klasse *java.util.Arrays*

- Mehrdimensionale Felder:

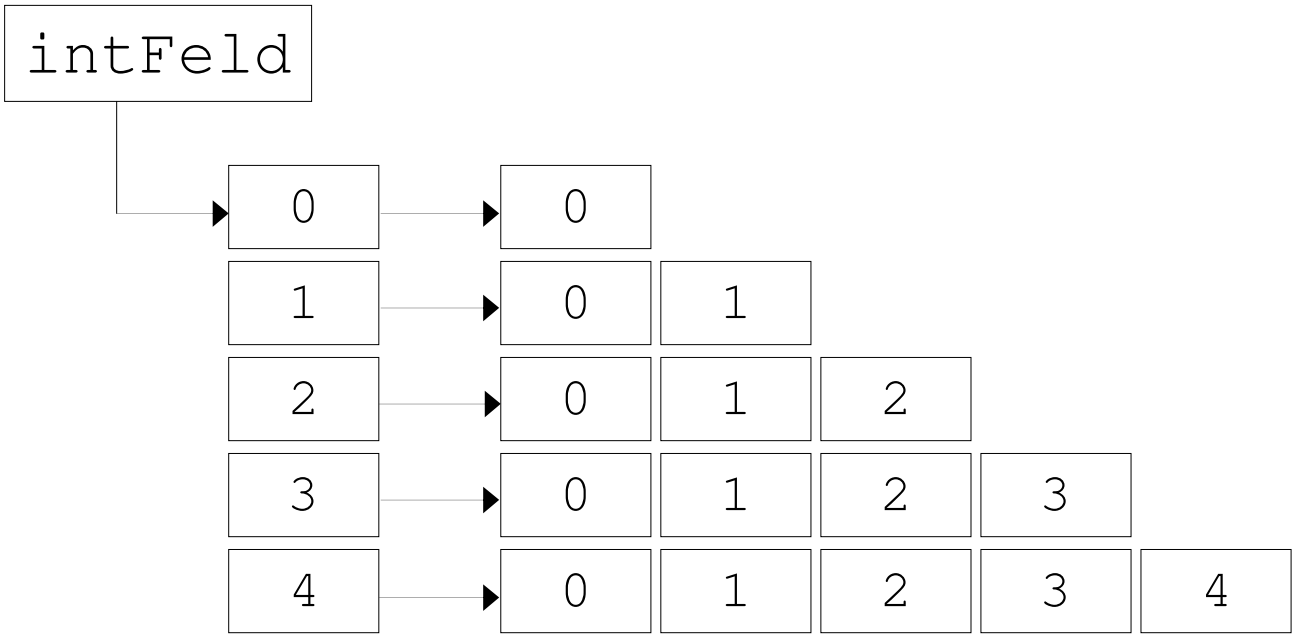
```
int[][] i = new int[5][10]
/* Matrix */
```

Arrays

- Hinweis: auch unregelmäßige Felder sind möglich (z.B. Pyramide)

```
int[][] intFeld = new int[5][];
for (int i=0; i<5; i++)
{
  intFeld[i]= new int[i+1];
}
```

Arrays



Ansprechbar mit `intFeld[3][2]`

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Pakete

- Mehrfachverwendung von Klassen
- Übersichtlichkeit, zusammengehörige Klassen in einem Paket

- Namensaufbau wie ein URL nur umgedreht (Vorschlag)

- Importieren aller Klassen eines Paketes mit dem * (Sternchen)

  Beispiel:

```
import de.uni-frankfurt.cs.dbis.*;
```

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Wichtige Pakete

`java.lang`

- einfache Datentypen, Ausnahmebehandlung

`java.util`

- Datum, Liste, Stack, ZIP, Hashtable, Random

`java.io`

- Ein-/Ausgabe, Filesystem

`java.math`

- Bitoperationen