Problemorientiertes Programmieren RoboCode

2. Termin

Eva Eibenberger und Christian Rieß

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Übersicht

- Variables
  - Primitive Datentypen
  - Arrays
- Die Klasse String
- Parameterübergabe beim Programmaufruf

Primitive Datentypen

- Bei Java werden acht Grundtypen unterschieden:
  - Ganzzahlige Werte (integer): `byte, short, int, long`
  - Gleitpunktwerte (Fließkommazahlen): `float, double`
  - Zeichen (character): `char`
  - Wahrheitswerte (boolesche Werte): `boolean`

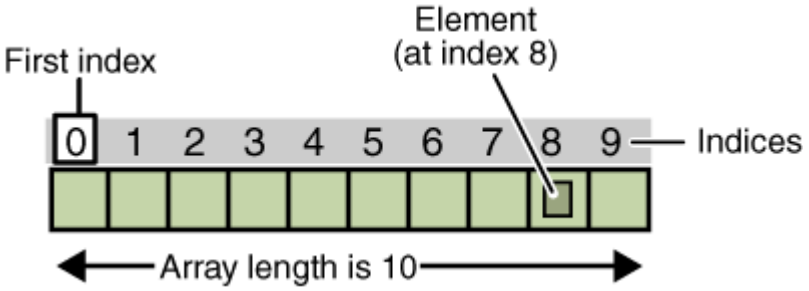
- Beispiel:

```
int zaehler1 = 1;
double kommaZahl = 1.5;
char zeichen = 'a';
```

- Bisher:
  - Jede Variable kann genau einen Wert speichern

Arrays

- Arrays fassen eine **feste Anzahl von Werten** eines bestimmte Datentyps zu einer Einheit zusammen



- Die **Länge** des Arrays wird beim Erzeugen des Arrays festgelegt
- Eine Einheit wird **Element** genannt
- Der Zugriff auf eine Element erfolgt über den **Index** (Nummernindex)
- Der Index für das erste Element ist `0`
  Das fünfte Element hat also den Index `4`
  Das letzte Element hat den Index `length-1`
- Arrays sind statisch: Die Anzahl der Elemente kann nach der Erzeugung des Feldes nicht mehr verändert werden

Bild entnommen von http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Arrays – Anlegen und Zugriff

- Anlegen von Arrays

```
int[] zahlen = new int[10];
double[] values = new double[20];
```

oder

```
double[] gutenoten = {1.7, 2.0, 2.3};
```

- Zuweisen/Zugriff auf einzelne Elemente mit dem `[]`-Operator

```
zahlen[0] = 10;      // erstes Element auf 10 setzen
zahlen[1] = 20;      // zweites Element auf 20

// drittes Element ausgeben
System.out.println(gutenoten[2] + "ist eine gute Note");
```

Arrays – Beispiel

- Beispiel: Array mit den Zahlen von 10 bis 15 anlegen und auslesen

```
int[] values = new int[6];
values[0] = 10;              // schreiben
values[1] = 11;
values[2] = 12;
values[3] = 13;
values[4] = 14;
values[5] = 15;
System.out.println("0. Wert: " + values[0]);   // lesen
System.out.println("1. Wert: " + values[1]);
System.out.println("2. Wert: " + values[2]);
System.out.println("3. Wert: " + values[3]);
System.out.println("4. Wert: " + values[4]);
System.out.println("5. Wert: " + values[5]);
```

- Oft wiederholen sich Berechnungen/Zuweisungen für alle/viele Elemente eines Arrays → Verwendung von Schleifen

Arrays – Beispiel

- Beispiel: Array mit den Zahlen von 10 bis 15 anlegen und auslesen

```
int[] values = new int[6];

// Werte in ein Array schreiben
for (int i=0; i<values.length; i++) {
        values[i] = 10 + i;
}

// Werte aus dem Array auslesen
for (int i=0; i<values.length; i++) {
        System.out.println(i + ". Wert: " + values[i]);
}
```

Arrays – Beispiel

- Beispiel: Array mit den Zahlen von 10 bis 15 anlegen und auslesen

```
int[] values = new int[6];

// Werte in ein Array schreiben
for (int i=0; i<values.length; i++) {
        values[i] = 10 + i;
}

// Werte aus dem Ar
for (int i=0; i<val
        System.out.p
}
```

Beachte die Werte, die von Index i angenommen werden!

- Der Index beginnt bei 0
- Wenn ein Array die Länge n hat, dann ist hat das letzte Element des Arrays den Index (n-1)!
- Im Beispiel:

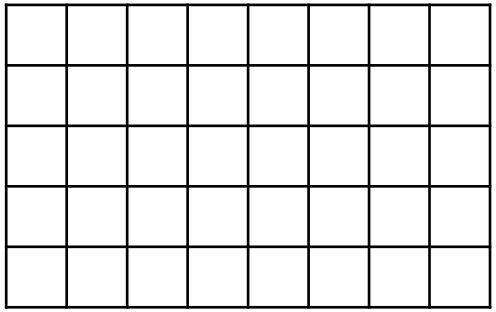
  `(... ; i < values.length() ; ...)`

  oder auch

  `(... ; i <= (values.length()-1) ; ...)`
- Bei Missachtung: Fehler zur Laufzeit!

Mehrdimensionale Arrays

- Typ eines Arrays kann wieder ein Array sein (Verschachtelung)
- Dadurch entstehen mehrdimensionale Felder



```
// Erstellen einer Matrix aus Einsen

int matrix[][] = new int[8][5];

for (int i=0; i<matrix.length; i++) {

    for (int j=0; j<matrix[i].length; j++) {
        matrix[i][j] = 1;
    }

}
```

Zugriff über zwei verschachtelte `for`-Schleifen

Klasse String – Zeichenketten

- Klasse `String` zur Speicherung von Zeichenfolgen

```
String s1 = "Hello";
String s2 = "World";
```

- Strings können aneinandergehängt werden (to concatenate)

```
String s3 = s1.concat(s2);          // s3 == "HalloWorld"
```

- Alternative: mit dem +-Operator

```
String s4 = s1 + " " + s2 + "!";  // s4 == "Hallo World!"
```

Klasse String – Zeichenketten

- Klasse `String` bietet weitere interessante Methoden:

```
String s = "RoboCode";
for (int i=0; i<s.length(); i++) {
        System.out.println(s.charAt(i));
}
```

- Ausgabe:

```
R
o
b
o
C
o
d
e
```

- Weitere Methoden der Klasse String können in der Java API nachgesehen werden

Die main-Methode

- Standardisierter Funktionskopf der main-Methode

```
public static void main(String[] args) {
        // ...
}
```

- `String[] args` bedeutet:
  - Variable mit Namen `args`
  - Datentyp ist `String[]` – also ein Array von Zeichenketten

- Das Array `args` enthält alle Parameter, die wir beim Aufruf des Programms aus der Kommandozeile übergeben

Die main-Methode – Parameterübergabe

- Beispiel:

```
faui00a [~/test]> javac GeometryCalc.java
faui00a [~/test]> javac GeometryCalc.java Rect 3 8
```

- Dann enthält die Variable `args` folgende drei Zeichenketten:

```
"Rect"   "3"   "8"
```

- Der Zugriff auf den Inhalt erfolgt wie bei allen übrigen Arrays:

```
public static void main(String[] args) {
        System.out.println("1. parameter: " + args[0]);
        System.out.println("2. parameter: " + args[1]);
        System.out.println(„3. parameter: " + args[2]);
}
```

Die main-Methode – Parameterübergabe

- Aber:
  Die Übergabeparameter sind vom Typ `String` – also Zeichenketten!

```
public static void main(String[] args) {
    String strGeoType = args[0];
    String strWidth = args[1];
    String strHeight = args[2];
```

- Um mit den Übergabeparametern rechnen zu können ist Typkonvertierung notwendig!

```
    // Konvertierung von string nach integer
    int intWidth = Integer.parseInt(args[1]);
    int intHeight = Integer.parseInt(strHeight);

    int area = intWidth*intHeight;
    System.out.println("Flaeche: " + area);
}
```

Die main-Methode – Parameterübergabe

- Aber:
  Die Übergabeparameter sind vom Typ `String` – also Zeichenketten!

```
public static void main(String[] args) {
    String strGeoType = args[0];
    String strWidth = args[1];
    String strHeight = args[2];
```

- Um mit den Übergabeparametern rechnen zu können ist Typkonvertierung notwendig!

```
    // Konvertierung von string nach double
    double doubWidth = Double.parseDouble(args[1]);
    double doubHeight = Double.parseDouble(strHeight);

    double area = doubWidth*doubHeight;
    System.out.println("Flaeche: " + area);
}
```