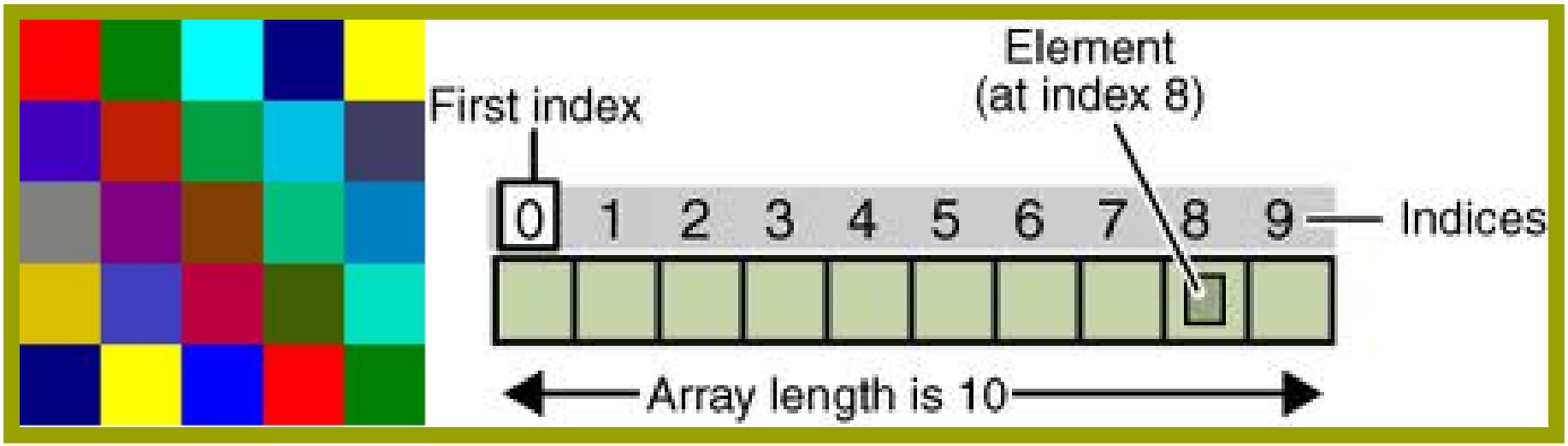
Felder

M. Jakob

Gymnasium Pegnitz

28. April 2015



Inhaltsverzeichnis

1. Begriffsbildung

2. Verwendung von Feldern

Gliederung

1 Begriffsbildung

2 Verwendung von Feldern

Beispiel — Ergebnistabelle 100m-Lauf

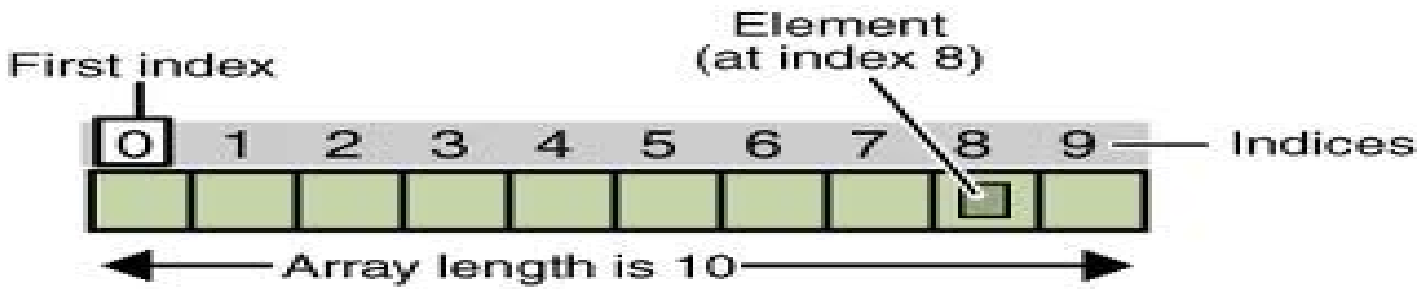
Platzierung	1	2	3	...	8
Zeit/s	10.1	11.2	12.3	...	15.7

Beispiel — Ergebnistabelle 100m-Lauf

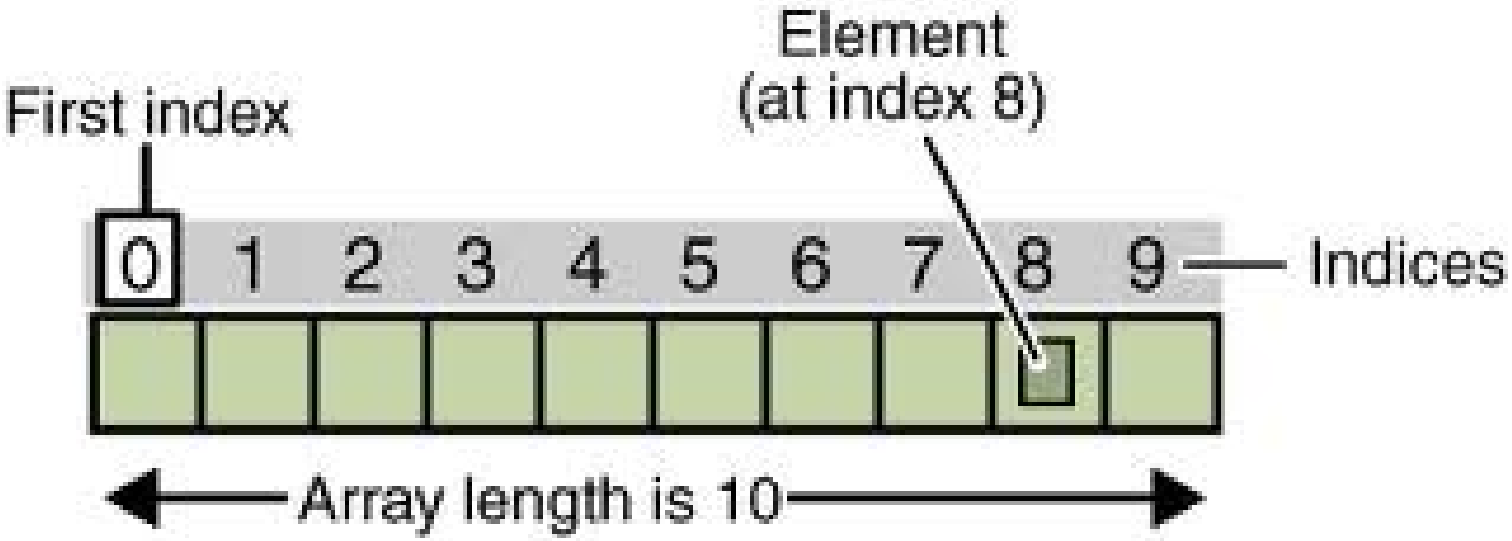
Platzierung	1	2	3	...	8
Zeit/s	10.1	11.2	12.3	...	15.7

Felder

auch Arrays genannt, benutzt man um Attribute zu einem Paket mit nummerierten Zellen zusammenzufassen. Die Anzahl der Zellen hießt Länge des Arrays. Die Nummer einer Zelle wird auch Index genannt. Die erste Zelle besitzt in Java den Index 0.



Beispiel — Ergebnistabelle 100m-Lauf



Platzierung	1	2	3	...	8
Index	0	1	2	...	7
Variablenname	`zeiten[0]`	`zeiten[1]`	`zeiten[2]`	...	`zeiten[7]`
Zeit	10.1	11.2	12.3	...	15.7

Das Erzeugen von Feldern in Java . . .

. . . ist ein dreistufiger Prozess.
Beispiel

```
1 private double[] zeiten;
2 zeiten = new double[8];
3 zeiten[0] = 10.1;
4 zeiten[1] = 11.1;
5 ...
```

Das Erzeugen von Feldern in Java . . .

. . . ist ein dreistufiger Prozess.
Beispiel

```
1 private double[] zeiten;
2 zeiten = new double[8];
3 zeiten[0] = 10.1;
4 zeiten[1] = 11.1;
5 ...
```

1. Deklaration eines Feldes (erkennbar an den []) mit dem Namen `zeiten` das mehrere **`double`**-Werte speichern kann
2. Festlegen der Länge des Feldes (hier 8). Dies erfolgt meist im Konstruktor.
3. Belegung des Feldes mit Werten.

Erzeugen von Feldern — kompakt

- in drei Stufen:

```
1 private double[] zeiten;
2 zeiten = new double[8];
3 zeiten[0] = 13.1;
4 ...
```

- in zwei Stufen:

```
1 private double[] zeiten = new double[8];
2 zeiten[0] = 10.1;
3 ...
```

- in einer Stufe:

```
1 private double[] zeiten = {10.1, 11.1, 12.3,
      12.4, 12.6, 13.0, 13.5, 15.7};
```

Gliederung

1 Begriffsbildung

2 Verwendung von Feldern

Einführungsbeispiele

```
1  private double[] zeiten = {10.1, 11.1, 12.3,
       12.4, 12.6, 13.0, 13.5, 15.7};
2  ...
3  System.out.println("Beste Zeit: "+zeiten[0]);
4
5  System.out.println(
6     "Geschwindigkeit: "+zeiten[0]/100+"m/s"
7  );
8
9  System.out.println(
10    "Teilnehmerzahl: "+zeiten.length
11 );
```

Ausgabe aller Zeiten — Holzhackermethode

```
1 ...
2 System.out.println("Platz 1: "+zeiten[0]);
3 System.out.println("Platz 2: "+zeiten[1]);
4 System.out.println("Platz 3: "+zeiten[2]);
5 ...
6 System.out.println("Platz 8: "+zeiten[7]);
```

Ausgabe aller Zeiten — Deluxe

Platzierung	1	2	3	...	8
Index	0	1	2	...	7
Variablenname	`zeiten[0]`	`zeiten[1]`	`zeiten[2]`	...	`zeiten[7]`
Zeit	10.1	11.2	12.3	...	15.7

Ausgabe aller Zeiten — Deluxe

Platzierung	1	2	3	...	8
Index	0	1	2	...	7
Variablenname	`zeiten[0]`	`zeiten[1]`	`zeiten[2]`	...	`zeiten[7]`
Zeit	10.1	11.2	12.3	...	15.7

```
1 wiederhole für die Zellen 0,..,7 des Feldes
2     Ausgabe: Platzierung und Zeit;
3 endewiederhole;
```

Ausgabe aller Zeiten — Deluxe

Platzierung	1	2	3	...	8
Index	0	1	2	...	7
Variablenname	`zeiten[0]`	`zeiten[1]`	`zeiten[2]`	...	`zeiten[7]`
Zeit	10.1	11.2	12.3	...	15.7

```
1 wiederhole für die Zellen 0,..,7 des Feldes
2     Ausgabe: Platzierung und Zeit;
3 endewiederhole;
```

wird zu

```
1 wiederhole für die Zellen mit Index 0,..,7
2     Ausgabe: index +1 und zeiten[index];
3 endewiederhole;
```

Ausgabe aller Zeiten — Deluxe

Index	0	1	2	...	7
Variablenname	`zeiten[0]`	`zeiten[1]`	`zeiten[2]`	...	`zeiten[7]`

```
1 index=0;
2 wiederhole zeiten.length mal
3     Ausgabe: index +1 und zeiten[index];
4     index = index+1;
5 endewiederhole;
```

Ausgabe aller Zeiten — Deluxe

Index	0	1	2	...	7
Variablenname	`zeiten[0]`	`zeiten[1]`	`zeiten[2]`	...	`zeiten[7]`

```
index=0;
wiederhole zeiten.length mal
    Ausgabe: index +1 und zeiten[index];
    index = index+1;
endewiederhole;
```

```
for ( int i=0 ; i<zeiten.length ; i=i+1 ){
    System.out.println("Platz "+
        (i+1) +": "+ zeiten[i] +"s");
}
```

Weitere Beispiele

```
1 public void rueckstaendeGeben(){
2     for ( int i=1 ; i<zeiten.length ; i=i+1 ){
3         System.out.println("Rückstand Platz"+
4           (i+1) +": "+ (zeiten[i]-zeiten[0]) +"s"
                );
5     }
6 }
```

```
1 public boolean reihenfolgePruefen(){
2     boolean ok=true;
3     for (int i=1;i<zeiten.length;i=i+1){
4       ok = ok && ( zeiten[i] >= zeiten[i-1] );
5     }
6     return ok;
7 }
```

Übung

Ü 2.1: 100-m-Lauf

(a) Erzeuge ein Feld `Zeiten` der Länge 8 und belege die Zellen mit den oben angegebenen Werten.

(b) Implementiere die Methoden `zeitenAusgeben()`, `rueckstaendeGeben()` und `reihenfolgePruefen()` wie oben beschrieben.

(c) Implementiere die Methode `zeitAendern(int platz, double zeit)`, die die Zeit des Läufers, der auf dem Platz `platz` steht, ändert. Stelle sicher, dass für die Variable `platz` nur Werte zwischen 1 und 8 eingegeben werden können. Prüfe auch, ob die Methode `reihenfolgePruefen()` richtig arbeitet.

Übung

Ü 2.2: Programmcode analysieren

```
1 public int wasMachIch(int[] liste){
2     int x = liste[0];
3     for(int i=1;i<liste.length;i=i+1){
4 
5         if ( liste[i] > x ) {x = liste[i] }
6     }
7     return x;
8 }
```

Gegeben sei die Variable `liste={4,3,7,6,7}`. Erstelle eine Tabelle mit den Spalten `x`, `i` und `liste[i]` und trage ein, welchen Wert diese Variablen in Zeile 4 jeweils haben.
Erkläre damit, welche Aufgabe die Methode erledigt, und wie sie das macht.

Übung

Ü 2.3: Programmcode analysieren

```
1 public boolean UndWasMachIch(int[] liste){
2     int x = liste[0];
3     boolean res = false;
4     for(int i=1;i<liste.length;i=i+1){
5 
6         if ( liste[i] == x ) {res = true; }
7     }
8     return res;
9 }
```

Gegeben sei die Variable `liste={4,3,4,6}`. Erstelle eine Variablenbelegungstabelle mit den Spalten `x`, `i`, `liste[i]` und `res`. Erkläre damit, welche Aufgabe die Methode erledigt, und wie sie das macht.

Übung

Ü 2.4: Programmcode analysieren

```
1  public boolean UndDannNochIch(int a,int[] liste
       ){
2      int x = liste[0];
3      boolean res = false;
4      for(int i=1;i<liste.length;i=i+1){
5
6          if ( liste[i] == a ) {res = true; }
7      }
8      return res;
9  }
```

Erkläre, welche Aufgabe die Methode erledigt, und wie sie das macht. Welche Programmzeilen sind unnötig?

Übung

➥Ü 2.5: Schulklasse als Array implementieren

Vorlage: Paket `Schulklasse`
Es sollen die Namen einer Schulklasse in einem Array gespeichert werden. Außerdem sollen in den Array Schüler eingefügt und gelöscht werden können.
Implementiere die fehlenden Methoden und erkläre bei allen Methoden, wie sie funktionieren.

➥Ü 2.6: Programmcode analysieren

Vorlage: Paket `WuerfelLsg`
Analysiere die Klasse `Wuerfelautomat` und beschreibe bei jeder einzelnen Methode wie sie funktioniert. Ergänze die Methode `haeufigkeitenAusgeben` so, dass die relative Häufigkeit der einzelnen Augenzahlen in Prozent mit angegeben wird.