Prof. Dr. Verena Majuntke - Prof. Dr. Ulrich Meissen

Grundlagen der Programmierung

Arrays

01.12.2020

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Themen

- Einführung in Arrays
- `int`-Array
- Erweiterte `for`-Schleife
- Mehrdimensionale Arrays

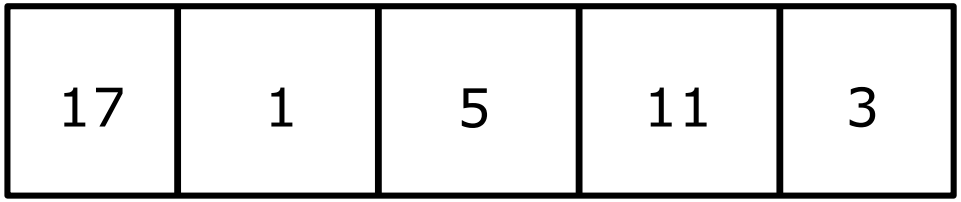
Lernziele

- Was ist ein Array?
- Wie deklariere ich einen Array?
- Wie initialisiere ich einen Array?
- Wie arbeite (lesen, schreiben) ich mit einen Array?
- Wie nutze ich Arrays in for-Schleifen? (Erweiterte for-Schleife)
- Wie können mehrdimensionale Arrays dargestellt und genutzt werden?

Einführung Arrays

Arrays (I)

- Array (auch Feld genannt) – Datentyp, der mehrere Werte eines Datentyps in nummerierter Reihenfolge zusammenfasst
  - Array mit int/double/boolean/...-Werten
  - Array mit Objekten (lernen wir später)



Beispiel:
Array aus int-Werten

Arrays (II)

- Datenfelder, mit denen man beliebig (aber endlich) viele Daten des gleichen Datentyps verwalten kann
- Erkennbar an den eckigen Klammern []

- Deklaration: `Typ[] Bezeichner;`
- Bsp: `int[] meinArray;`

Arrays - Beispiele

```
int[] primes;              // array mit dem Namen primes,
                           // Elemente sind vom Typ int
char[] alphabet;           // array mit dem Namen alphabet,
                           // Elemente sind vom Typ char
boolean[] bvalues;         // array mit dem Namen switch,
                           // Elemente sind vom Typ boolean
String[] args;             // array mit dem Namen args,
                           // Elemente sind vom Typ String
```

Erzeugen von Arrays

- **Deklaration** analog zu Variablen primitiver Datentypen
  - `int[] primes;`
  - `boolean[] bvalues;`

```
int x1;
int x2 = 2;
```

- **Erzeugung** mit dem Schlüsselwort new
  - `primes = new int[5];`
  - `bvalues = new boolean[9];`

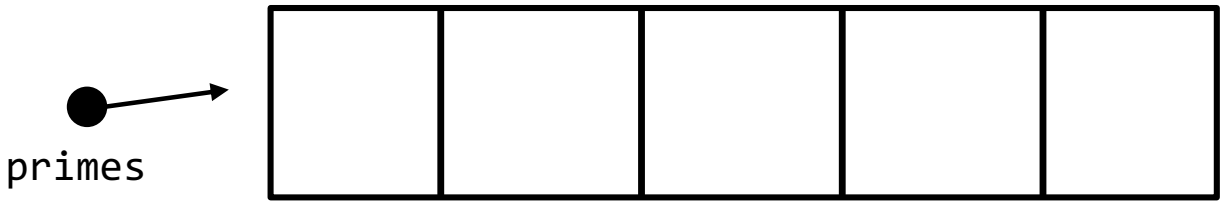
Arrays sind Objekte

- sie werden über das Schlüsselwort new erzeugt (anders als bei Variablen primitiver Datentypen)

(mehr dazu im Thema OO)

Arrays

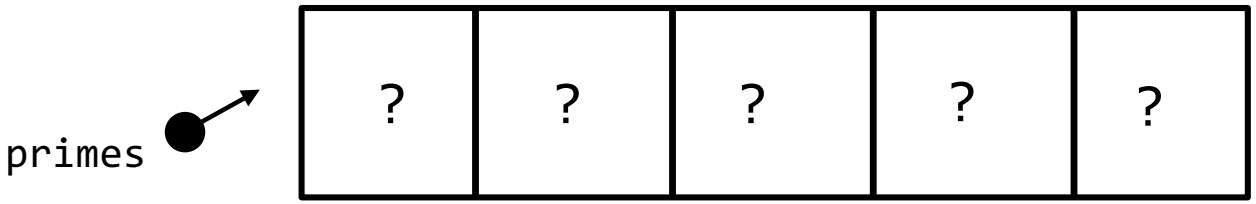
- Deklarieren und Erzeugen geht auch in einem Schritt:
  - `int[] primes = new int[5];`
  - `boolean[] bvalues = new boolean[9];`



`int`-Arrays

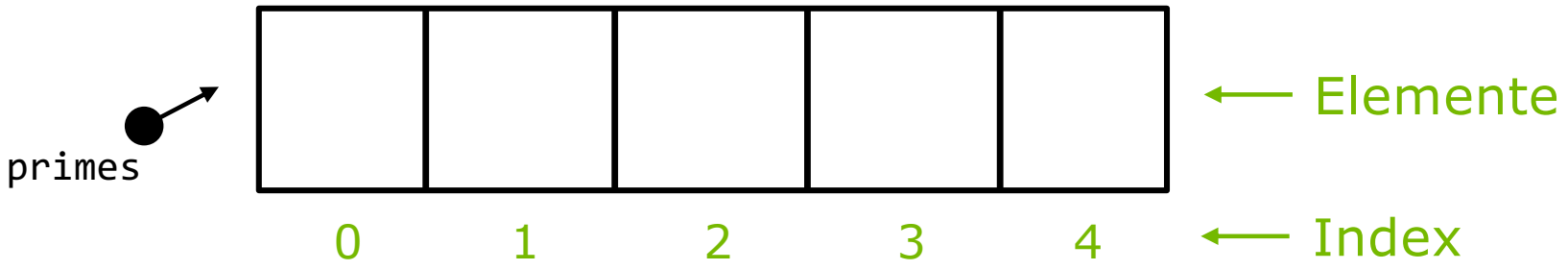
int-Array (I)

- Deklaration und Erzeugung eines int-Arrays mit 5 „Plätzen“
  - `int[] primes = new int[5];`



int-Array (II)

- `int[] primes = new int[5];`

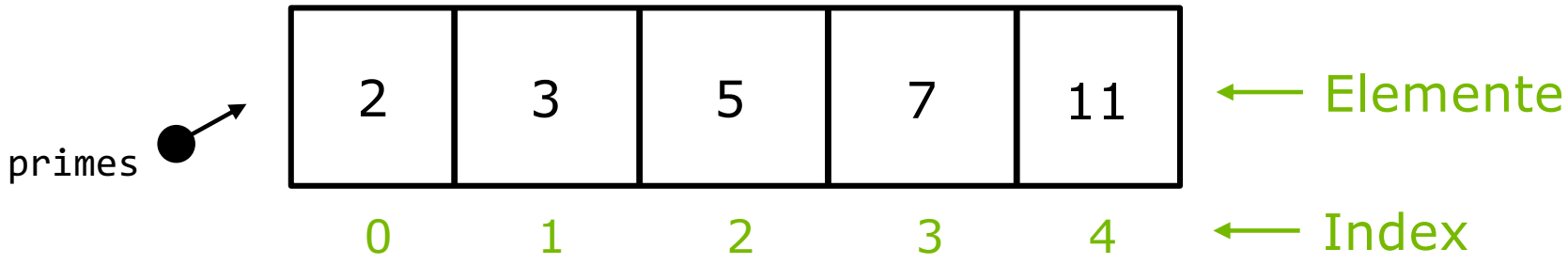


... erzeugt ein Objekt mit Namen *primes*,
welches 5 Elemente vom Typ `int` speichern kann

Initialwerte

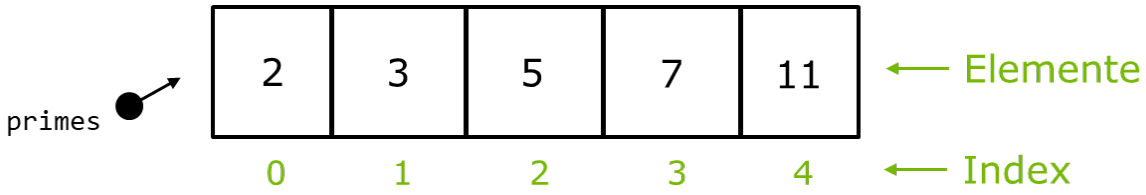
Vorgabe von Initialwerten bei der Definition des Arrays

- `int[] primes = {2, 3, 5, 7, 11};`



Zugriff auf Array-Elemente (I)

- Auf Werte, die im Array gespeichert sind, kann man über die Index zugreifen
  - `variable[index]`
  - Beispiel:
    - `primes[3]` → 7
    - `primes[0]` → 2

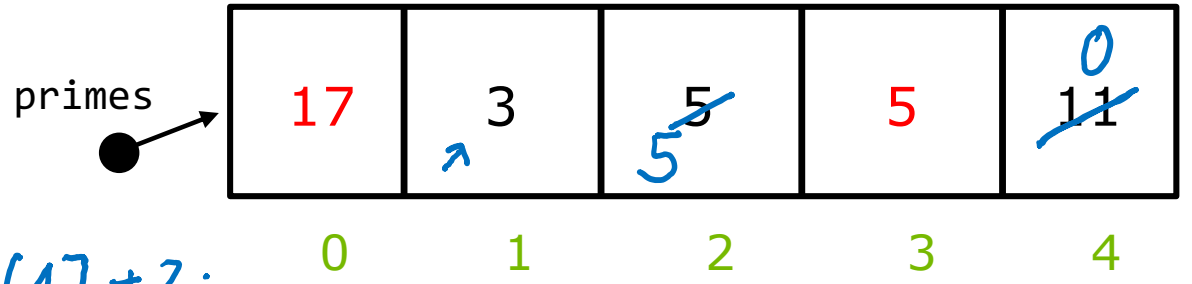


Zugriff auf Array-Elemente (II)

- Array-Werte lesen:
  - `int a = primes[3]` → a = 5
  - `int b = primes[0]` → b = 17

- Array-Werte schreiben
  - `primes[3] = 5;`
  - `primes[0] = 17;`

primes[4] = 0;
primes[2] = primes[1] + 2;
3 + 2



Länge eines Array

- Anzahl der Elemente, die ein Array aufnehmen kann, wird *Länge* (oder auch Größe) genannt
- Die Länge ist im Attribut **`length`** gespeichert
- Die Größe eines Arrays lässt sich nach der Erzeugung NICHT mehr ändern

Arrays sind Objekte
- sie verfügen über vordefinierte Attribute (Charakteristika) und Methoden (Operationen), die bei der Entwicklung genutzt werden können

(mehr dazu im Thema OO)

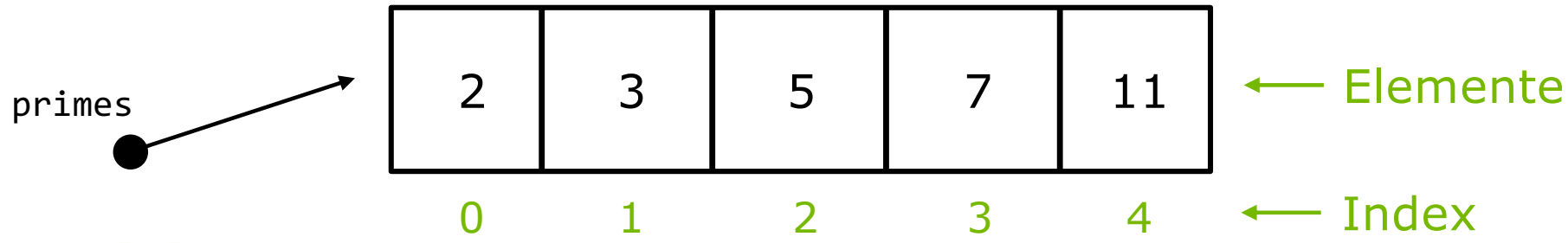
```
int[] primes =  {2, 3, 5, 7, 11}
System.out.println(primes.length);
```

Zugriff auf Array-Elemente (III)

- sowohl beim Lese- als auch beim Schreibzugriff muss der gewählte Index existieren (Index: 0 bis a.length-1)
- sonst erhält man eine Fehlermeldung (**Laufzeitfehler**) `ArrayIndexOutOfBoundsException`
- Laufzeitfehler können von Eingabedaten abhängen, d.h. sie können zur Entwicklungszeit nicht erkannt werden
- Fehler, die der Compiler findet, hängen nicht von Eingabedaten ab und sind daher erkennbar

int-Array (II)

- `int[] primes = {2, 3, 5, 7, 11};`




```
primes[0] = 2;
primes[1] = 3;
primes[2] = 5;
...
```

Initialisierung entspricht ...

Breakout-Session 1

Schreiben Sie ein Programm, dass zuerst einen int-Array mit 5 Elementen anlegt und dann jedes Element dieses Arrays mit den dem Wert 10 füllt. Danach soll es alle Elemente des Arrays auf der Konsole ausgeben wie folgt ausgegeben werden:

```
Der Inhalt des Elements mit dem Index … lautet ….
```

Beispiel: Schreiben und Lesen

```
public class Zahlenreihe {

     public static void main(String[] args)
     {
          int[] zahlenreihe = new int[5];          // Array zahlenreihe erzeugt

          zahlenreihe[0]=0;
          System.out.println(zahlenreihe[1]);
          zahlenreihe[1]=1;
          System.out.println(zahlenreihe[1]);
          zahlenreihe[2]=4;
          System.out.println(zahlenreihe[2]);
          zahlenreihe[3]=9;
          System.out.println(zahlenreihe[3]);
          zahlenreihe[4]=16;
          System.out.println(zahlenreihe[4]);
          }
}
```

Ausgabe:

```
0
1
4
9
16
```

Initialisierung über `for`-Schleife

- Initialisierung der Array-Elemente, hier Belegung der Elemente mit dem Wert 0
- Schleifenbedingung mithilfe von `length`

```
int[] feld = new int[4];
for (int i = 0; i < feld.length; i++) {
        feld[i] = 0;
}
```

Beispiel: Schreiben und Lesen

```
public class Zahlenreihe {

     public static void main(String[] args)
     {
          int[] zahlenreihe = new int[5];          // Array zahlenreihe erzeugt

          for (int index=0; index<5; index++)
          {
                    zahlenreihe[index]=index*index; // mit Werten belegen
                    System.out.println(zahlenreihe[index]);  // Ausgabe
          }
}
```

Breakout-Session 2

Variieren Sie Ihr Programm so, dass das Einlesen des Arrays über eine Methode erfolgt. Die Methode hat zwei Eingabeparamater: Array-Länge und Wert der Elemente. Die Methode gibt einen mit dem vorgegeben Wert gefüllten Array mit der vorgegebenen Länge zurück.

Breakout-Session 3

Nehmen Sie ihr bereits geschriebenes Programm und variieren Sie es:

Variante 1: Füllen Sie die Elemente mit Werten von 0,…,4

Variante 2: Füllen Sie die Elemente mit Werten von 5,…,1

Variante 3: Machen Sie aus dem Array einen Boolean Array und füllen diesen abwechselnd mit true und false

Erweiterte `for`-Schleife

- Komfortables Durchlaufen von Array-Elementen eines Collection Objekts

```
for (Object element : Collection) {
        System.out.println(element);
}
```

`(Object element : Collection)` bedeutet:
für jedes Element in der Collection

Beispiel: Erweiterte for-Schleife

```
int [ ] feld = { 67, 45, -8, 0, 888 };
int summe = 0;

for( int nummer : feld ){
        summe = summe + nummer;
}
```

Statt:

```
        for(int index = 0 ; index < feld.length; index ++){
                summe = summe + feld[ index ];
        }
```

Mehrdimensionale Arrays (I)

- Mehrdimensionale Arrays werden als Arrays von Arrays realisiert
- Deklaration eines zweidimensionalen Arrays:

```
int[][] arr = new int[ 3 ][ 2 ]; //Fasst 6 Elemente
```

- Alternative Arraydeklarationen:

```
int arr[][] = new int[ 3 ][ 2 ];
int[] arr[] = new int[ 3 ][ 2 ];
```

Empfehlung: Alle eckigen Klammern hinter den Typ setzen

Mehrdimensionale Arrays (II)

Anlegen und Initialisieren:

```
int[][] array3x2 = { {1, 2}, {2, 3}, {3, 4} };
int[][] arrayB = { {1, 2}, {2, 3, 4}, {5} };
```

Zugriff auf Array-Elemente:

```
referenzvariable[index1] [index2]
arrayA3x2 [1][2] //Beispiel
arrayB[1][0] //Beispiel
```

Mehrdimensionale Arrays (III)

Zweidimensionale Arrays ablaufen:

```
for ( int line = 0; line < arr.length; line++ ) {
    for ( int column = 0; column < arr[i].length; column++ )
        arr[line][column] = 0;
}
```

Breakout-Session 4

Schreiben Sie ein Programm, dass einen 2-dimensionalen Array vom Typ char wie folgt füllt und ausgibt:

```
a b b b b
b a b b b
b b a b b
b b b a b
b b b b a
```

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

www.htw-berlin.de