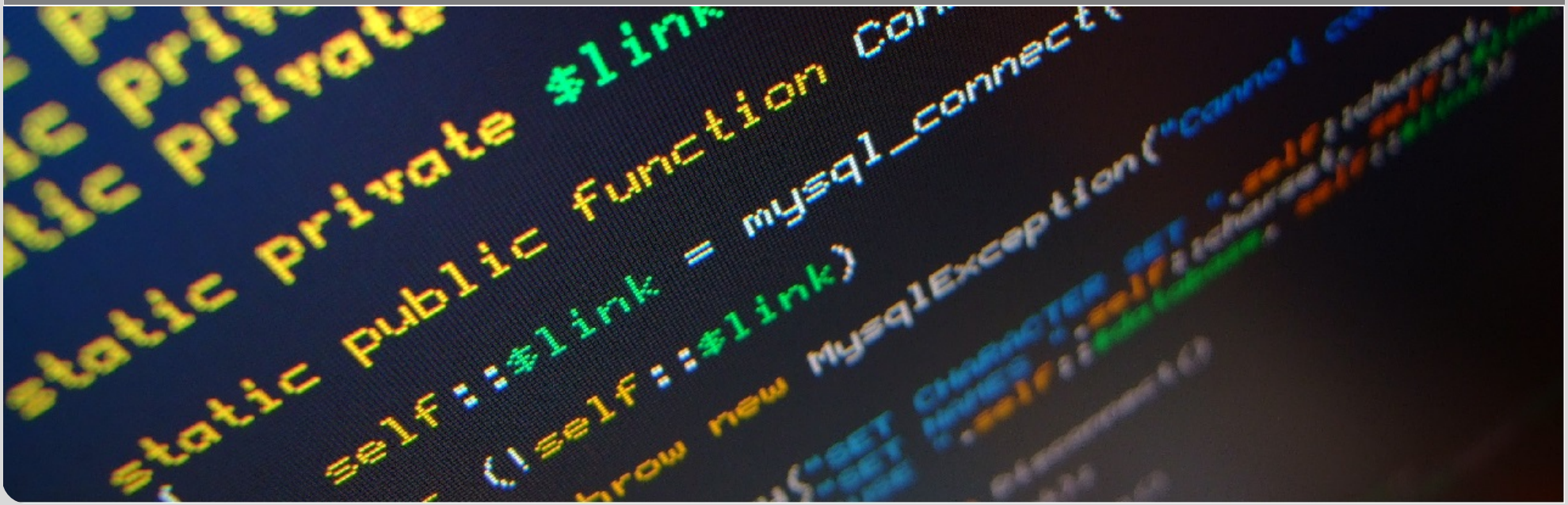
Karlsruher Institut für Technologie

Tutorium Programmieren

Tut Nr.5: Arrays Javadoc

Michael Friedrich | 26. / 28.11.2013

INSTITUT FÜR THEORETISCHE INFORMATIK



Outline/Gliederung

WICHTIG

- jedes Übungsblatt hat seine eigene Checkstyle XML
- nur **grüne** final solutions haben Chance auf **volle Punktzahl**
- die Abgaben **MÜSSEN** gewissen Kriterien genügen, um überhaupt gewertet zu werden
  $\Rightarrow$ **mindestens gelb**
  Vergleich: Leserlichkeit bei handschriftlichen Abgaben
- Ich war mit der ersten Abgabe zufrieden, daher keine weitere allg. Kommentare

WICHTIG

Was mir bisher beim Korrigieren aufgefallen ist...

- Parameter **NIE** final. Überlegt euch immer, was die Modifikatoren machen
- private konsequent benutzen
- Attribute **NIE** static
- redundanter Code immer vermeiden, wenn möglich
  $\Rightarrow$ leserlicher, strukturierter. Eine der Gründe warum es Methoden gibt.
- `main` methode immer als letztes!
  - Reihenfolge: Attribute, Konstruktoren, Methoden, Getter/Setter, print/toString, main
- Kommentare brauchen einen **Sinn**
- haltet euch an die Rahmenbedingungen der Blätter, sprich Pakete etc.

Javadoc

alter Foliensatz

Deklarieren von Arrays

- Datenstruktur für eine Sammlung an Daten
  - Über Indizes einzelne Elemente direkt erreichbar

Beispiele

```
int[] array;

int[] array = {1,2,3};

int[] array = new int[4];
```

- Indizes beginnen immer bei **0** !
- Arraygröße abfragen über `array.length`

Deklarieren von Arrays

- Datenstruktur für eine Sammlung an Daten
  - Über Indizes einzelne Elemente direkt erreichbar

Beispiele

```
int[] array;

int[] array = {1,2,3};

int[] array = new int[4];
```

- Indizes beginnen immer bei **0** !
- Arraygröße abfragen über `array.length`

Deklarieren von Arrays

- Datenstruktur für eine Sammlung an Daten
  - Über Indizes einzelne Elemente direkt erreichbar

Beispiele

```
int[] array;

int[] array = {1,2,3};

int[] array = new int[4];
```

- Indizes beginnen immer bei **0** !
- Arraygröße abfragen über `array.length`

Deklarieren von Arrays

- Datenstruktur für eine Sammlung an Daten
  - Über Indizes einzelne Elemente direkt erreichbar

Beispiele

```
int[] array;

int[] array = {1,2,3};

int[] array = new int[4];
```

- Indizes beginnen immer bei **0** !
- Arraygröße abfragen über `array.length`

Deklarieren von Arrays

- Datenstruktur für eine Sammlung an Daten
  - Über Indizes einzelne Elemente direkt erreichbar

Beispiele

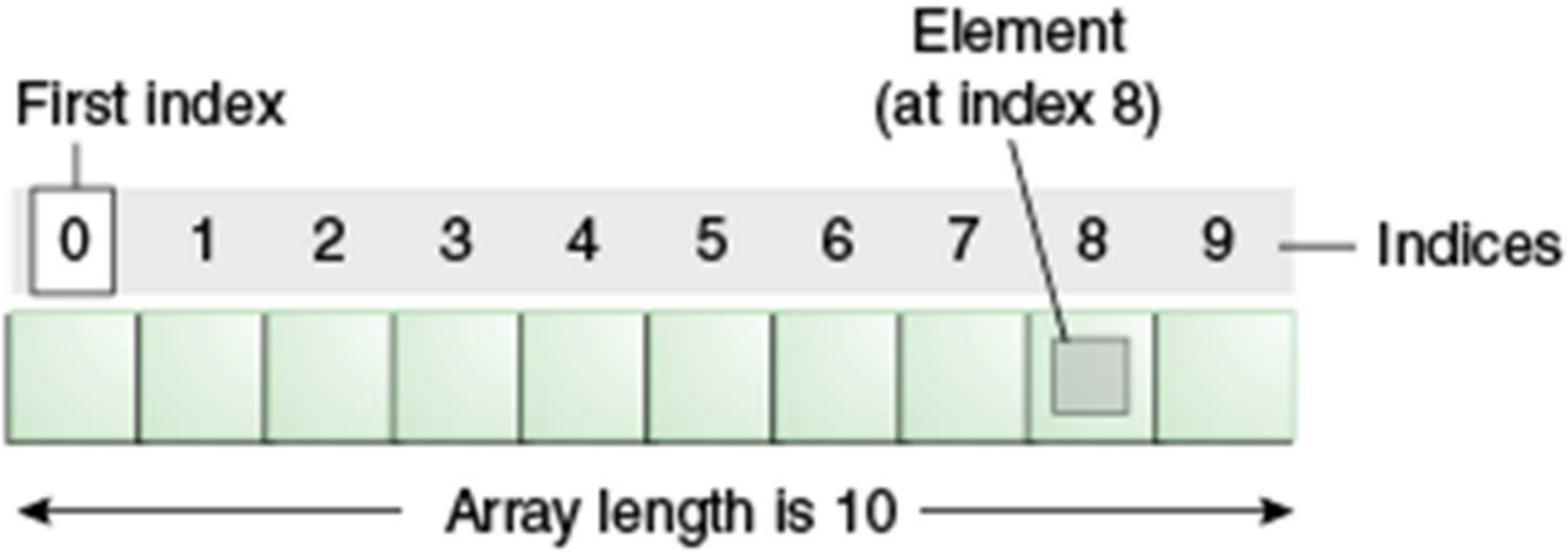
```
int[] array;

int[] array = {1,2,3};

int[] array = new int[4];
```

- Indizes beginnen immer bei **0** !
- Arraygröße abfragen über `array.length`

Besonderheiten



- **Überschreiten** der Grenzen eines Arrays führt zu Exception und damit **Absturz** eures Programms!

Tutoriumsaufgabe

Arrays - A

- Schreiben Sie eine Methode
  `public static int arraySum(int[] array)`,
  die die Summe der Zahlen des übergebenen Arrays als Rückgabewert hat.
  Schreiben Sie in einer Klasse namens Loops die Methoden
- Schreiben Sie eine Methode
  `public static double average(int[] array)`,
  die den durchschnittlichen Wert der Zahlen des übergebenen Arrays als Rückgabewert hat.
- Schreiben Sie eine Methode
  `public static double[] sum(double[] vectorA, double[] vectorB)`,
  die eine Vektoraddition auf den beiden übergebenen Vektoren durchführt und das Ergebnis als Rückgabewert hat. Achten Sie darauf, dass die beiden Vektoren hierzu dieselbe Länge haben müssen und dass Sie beim Berechnen der Summe weder vectorA noch vectorB verändern.

Tutoriumsaufgabe

Arrays - B

- Schreiben Sie eine Methode
  `public static double[] scalarMult(double[] vectorA, double scalar)`,
  die den Vektor vectorA mit dem Skalar scalar multipliziert und das Ergebnis als Rückgabewert hat. Achten Sie auch hier darauf, dass vectorA unverändert bleibt.
- Schreiben Sie eine Methode
  `public static double [][] sum(double[][] matrixA, double[][] matrixB)`,
  die die Matrizen matrixA und matrixB addiert und das Ergebnis als Rückgabewert hat. Beachten Sie, dass hierzu die Dimensionen der Matrizen gleich sein müssen und dass auch hier weder matrixA und matrixB verändert werden sollen.

Lösung

```
static double[] sum(double[] vectorA, double[] vectorB) {

if (vectorA.length != vectorB.length) {
System.out.println( "Vektoren nicht gleich lang" );
return null;
}
double[] vectorRes = new double[vectorA.length];

for (int i = 0; i < vectorA.length; i++) {
vectorRes[i] = vectorA[i] + vectorB[i];
}
return vectorRes;
}
```

Lösung

```
public static double[][] sum(double[][] matrixA, double[][] matrixB) {
   double[][] matrixRes;

   //Hinweis: Interne Darstellung mehrdimensionale Arrays: "Ein Array von Arrays"
   if (matrixA.length != matrixB.length && matrixA[1].length != matrixB.length[2]){
     System.out.println( "Matrizen mit verschiedenen Dimensionen" );
   return null;
}

 for (int i = 0; i < a.length; i++) {
   for (int j = 0; j < a.length; j++) {
     matrixRes[i][j] = matrixA[i][j] + matrixB[i][j];
   }
}
 return matrixRes;
}
```