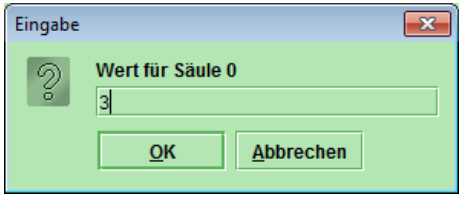
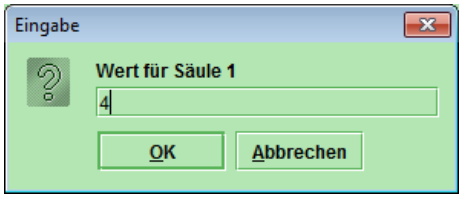
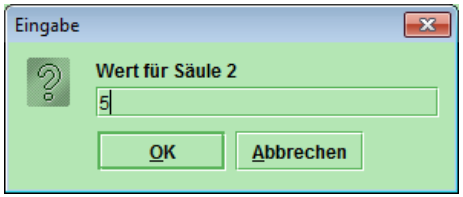
Vom Umgang mit Daten

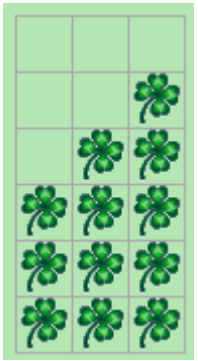
JavaKara programmieren: Arrays: Ein ganzes Feld mit Daten

Kara soll ein Benutzer-definiertes Säulendiagramm zeichnen









```
public void myProgram() {
  int[] saeulenWerte = new int[world.getSizeX()];
  for (int i = 0; i < saeulenWerte.length; i++) {
    saeulenWerte[i] = tools.intInput("Wert für Säule " + i);
  }
  zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeulenHoehen) {
  for (int x = 0; x < saeulenHoehen.length; x++) {
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```

Säulendiagramm mit Kara: Array-Variable für Säulenhöhen

```
public void myProgram() {

  int[] saeulenWerte =     Variable saeulenWerte deklarieren: Feld mit int Werten
    new int[world.getSizeX()];          Variable saeulenWerte initialisieren:
                                  so viele Felder, wie Welt breit ist (world.getSizeX())

  // im Beispiel der vorigen Seite kann jetzt auf die Feld-
  // Elemente saeulenWerte[0], saeulenWerte[1],
  // saeulenWerte[2] zugegriffen werden

  for (int i = 0; i < saeulenWerte.length; i++) {          Grösse des Feldes auslesen

    saeulenWerte[i] = tools.intInput("Wert für Säule " + i);  Wert in Feld speichern
  }

  zeichneSaeulenDiagramm(saeulenWerte);
}
```

Säulendiagramm mit Kara: Array-Variable für Säulenhöhen

```
void zeichneSaeulenDiagramm(int[] saeulenHoehen) {
```
Parameter saeulenWerte deklarieren: Feld mit int Werten

```
  for (int x = 0; x < saeulenHoehen.length; x++) {
```
Grösse des Feldes auslesen

```
    for (int y = 0; y < saeulenHoehen[x]; y++) {
```
Wert eines Elements des Feldes auslesen

```
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```

Säulendiagramm mit Kara: Array-Variable für Säulenhöhen

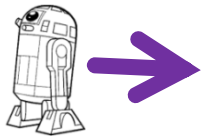
- **int[] saeulenWerte = new int[world.getSizeX()];**

- Variablen speichern Daten. Sie sind von einem bestimmten Typ. **saeulenWerte** ist eine Variable vom Typ **«Array von Ganzzahlen».**
- **Array bedeutet: Eine feste (nicht-veränderliche) Anzahl von Daten vom gleichen Typ, die über einen Index von 0..<Anzahl-1> angesprochen werden.**

- Variablen müssen **definiert** werden, d.h. ihr Typ muss festgelegt werden: **int[] saeulenWerte**.
- Variablen müssen **initialisiert** werden, d.h. es muss ihnen ein erster Wert zugewiesen werden: **new int[world.getSizeX()]**. Anschliessend haben alle Elemente den Wert 0: saeulenWerte[0] = saeulenWerte[1] = … = saeulenWerte[world.getSizeX()-1] = 0.

- Variablen können beliebig **verändert werden:** saeulenWerte[0] = 10; // ändert den Inhalt des ersten Arrayelements

Säulendiagramm mit Kara: Programmausführung

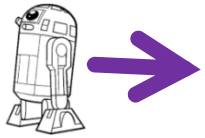


```
public void myProgram() {
 int[] saeulenWerte = new int[world.getSize
 for (int i = 0; i < saeulenWerte.length; i++) {
  saeulenWerte[i] = tools.intInput("Wert für
 }
 zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
 for (int x = 0; x < saeulenHoehen.length; x++) {
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

Methode **myProgram**

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung



```
public void myProgram() {
 int[] saeulenWerte = new int[world.getSize
 for (int i = 0; i < saeulenWerte.length; i++) {
  saeulenWerte[i] = tools.intInput("Wert für
 }
 zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
 for (int x = 0; x < saeulenHoehen.length; x++) {
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

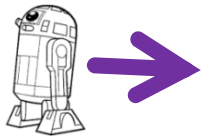
int[]
saeulenWerte
Methode
myProgram

int[3] =
{ 0, 0, 0 }

Globaler Speicher für das ganze Programm

Auf dem «Notizzettel» der Methode ist nur mehr ein Verweis auf die Daten, die unabhängig von der Methode auf einem «globalen Notizzettel» existieren.

Säulendiagramm mit Kara: Programmausführung



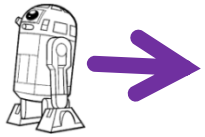
```
public void myProgram() {
 int[] saeulenWerte = new int[world.getSize
 for (int i = 0; i < saeulenWerte.length; i++) {
  saeulenWerte[i] = tools.intInput("Wert für
 }
 zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
 for (int x = 0; x < saeulenHoehen.length; x++) {
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

int i = 0
int[]
saeulenWerte
Methode
myProgram

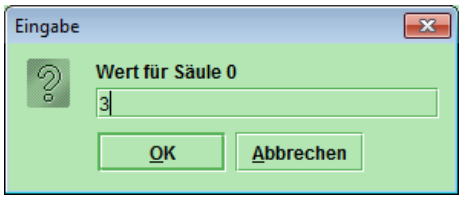
int[3] =
{ 0, 0, 0 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung



```
public void myProgram() {
 int[] saeulenWerte = new int[world.getSize
 for (int i = 0; i < saeulenWerte.length; i++) {
  saeulenWerte[i] = tools.intInput("Wert für
 }
 zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
 for (int x = 0; x < saeulenHoehen.length; x++) {
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

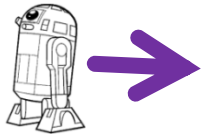


int i = 0
int[]
saeulenWerte
Methode
myProgram

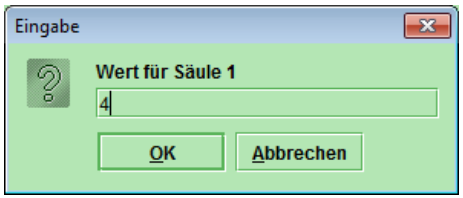
int[3] =
{ 3, 0, 0 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung



```
public void myProgram() {
  int[] saeulenWerte = new int[world.getSize
  for (int i = 0; i < saeulenWerte.length; i++) {
    saeulenWerte[i] = tools.intInput("Wert für
  }
  zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
  for (int x = 0; x < saeulenHoehen.length; x++) {
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```

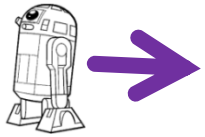


int i = 1
int[]
saeulenWerte
Methode
myProgram

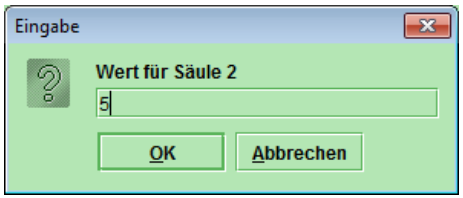
int[3] =
{ 3, 4, 0 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung



```
public void myProgram() {
 int[] saeulenWerte = new int[world.getSize
 for (int i = 0; i < saeulenWerte.length; i++) {
  saeulenWerte[i] = tools.intInput("Wert für
 }
 zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
 for (int x = 0; x < saeulenHoehen.length; x++) {
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

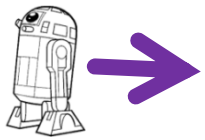


int i = 2
int[]
saeulenWerte
Methode
myProgram

int[3] =
{ 3, 4, **5** }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung



```
public void myProgram() {
  int[] saeulenWerte = new int[world.getSize
  for (int i = 0; i < saeulenWerte.length; i++) {
    saeulenWerte[i] = tools.intInput("Wert für
  }
  zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
  for (int x = 0; x < saeulenHoehen.length; x++) {
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```

int[]
saeulenWerte
Methode
myProgram

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
 int[] saeulenWerte = new int[world.getS
 for (int i = 0; i < saeulenWerte.length; i+
  saeulenWerte[i] = tools.intInput("Wert
 }
 zeichneSaeulenDiagramm(saeulenWert
}
```

```
void zeichneSaeulenDiagramm(int[] saeu
 for (int x = 0; x < saeulenHoehen.length;
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

int[] saeulenHoehen
Methode **zeichneSaeulen Diagramm**

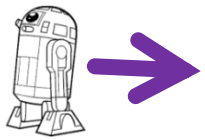
int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Die Methode zeichneSaeulenDiagramm hat einen eigenen Verweis auf die eigentlich Daten. Kann daher die gleichen Daten verändern wie myProgram.

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
 int[] saeulenWerte = new int[world.getS
 for (int i = 0; i < saeulenWerte.length; i+
  saeulenWerte[i] = tools.intInput("Wert
 }
 zeichneSaeulenDiagramm(saeulenWert
}
```



```
void zeichneSaeulenDiagramm(int[] saeu
 for (int x = 0; x < saeulenHoehen.length;
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

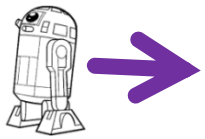
int x = 0
int[]
saeulenHoehen
Methode
zeichneSaeulen Diagramm

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
 int[] saeulenWerte = new int[world.getS
 for (int i = 0; i < saeulenWerte.length; i+
  saeulenWerte[i] = tools.intInput("Wert
 }
 zeichneSaeulenDiagramm(saeulenWert
}

void zeichneSaeulenDiagramm(int[] saeu
 for (int x = 0; x < saeulenHoehen.length;
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```



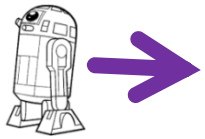
int y = 0
int x = 0
int[]
saeulenHoehen
Methode
zeichneSaeulen Diagramm

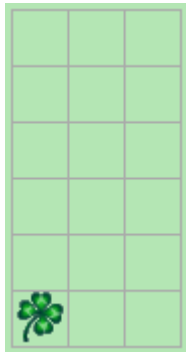
int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
 int[] saeulenWerte = new int[world.getS
 for (int i = 0; i < saeulenWerte.length; i+
  saeulenWerte[i] = tools.intInput("Wert
 }
 zeichneSaeulenDiagramm(saeulenWert
}

void zeichneSaeulenDiagramm(int[] saeu
 for (int x = 0; x < saeulenHoehen.length;
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```





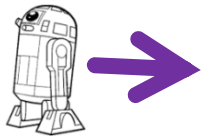
int y = 0
int x = 0
int[]
saeulenHoehen
Methode
zeichneSaeulen Diagramm

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
 int[] saeulenWerte = new int[world.getS
 for (int i = 0; i < saeulenWerte.length; i+
  saeulenWerte[i] = tools.intInput("Wert
 }
 zeichneSaeulenDiagramm(saeulenWert
}

void zeichneSaeulenDiagramm(int[] saeu
 for (int x = 0; x < saeulenHoehen.length;
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```



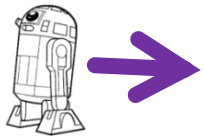
int y = 1
int x = 0
int[] saeulenHoehen
Methode **zeichneSaeulen Diagramm**

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
  int[] saeulenWerte = new int[world.getS
  for (int i = 0; i < saeulenWerte.length; i+
    saeulenWerte[i] = tools.intInput("Wert
  }
  zeichneSaeulenDiagramm(saeulenWert
}

void zeichneSaeulenDiagramm(int[] saeu
  for (int x = 0; x < saeulenHoehen.length;
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```



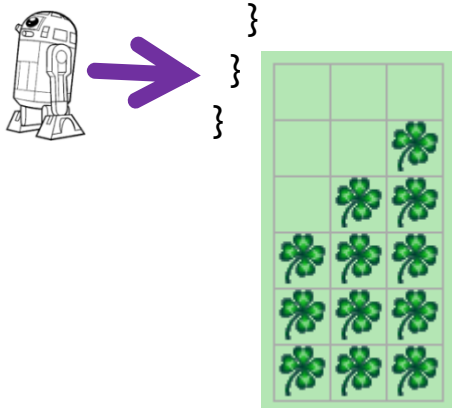
int y = 2
int x = 0
int[]
saeulenHoehen
Methode
zeichneSaeulen Diagramm

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
  int[] saeulenWerte = new int[world.getS
  for (int i = 0; i < saeulenWerte.length; i+
    saeulenWerte[i] = tools.intInput("Wert
  }
  zeichneSaeulenDiagramm(saeulenWert
}

void zeichneSaeulenDiagramm(int[] saeu
  for (int x = 0; x < saeulenHoehen.length;
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```



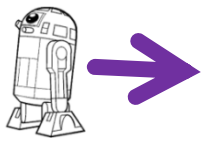
int y = 4
int x = 2
int[]
saeulenHoehen
Methode
zeichneSaeulen Diagramm

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
 int[] saeulenWerte = new int[world.getS
 for (int i = 0; i < saeulenWerte.length; i+
  saeulenWerte[i] = tools.intInput("Wert
 }
 zeichneSaeulenDiagramm(saeulenWert
}

void zeichneSaeulenDiagramm(int[] saeu
 for (int x = 0; x < saeulenHoehen.length;
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

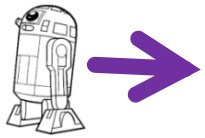


Methode **zeichneSaeulen Diagramm**

int[3] =
{ 3, 4, 5 }

Globaler Speicher für das ganze Programm

Säulendiagramm mit Kara: Programmausführung



```
public void myProgram() {
 int[] saeulenWerte = new int[world.getSize
 for (int i = 0; i < saeulenWerte.length; i++) {
  saeulenWerte[i] = tools.intInput("Wert für
 }
 zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
 for (int x = 0; x < saeulenHoehen.length; x++) {
  for (int y = 0; y < saeulenHoehen[x]; y++) {
   world.setLeaf(x, world.getSizeY() - 1 - y, true);
  }
 }
}
```

Methode **myProgram**

int[3] =
{ 3, 4, 5 }

Globaler

Endet eine Methode, werden alle Daten ihres Notizzettels abgeräumt. Die Daten des «globalen Notizzettel» werden automatisch abgeräumt, irgendwann nachdem sie nicht mehr benötigt werden («Garbage Collection»).

Säulendiagramm mit Kara: Programmausführung

```
public void myProgram() {
  int[] saeulenWerte = new int[world.getSize
  for (int i = 0; i < saeulenWerte.length; i++) {
    saeulenWerte[i] = tools.intInput("Wert fü
  }
  zeichneSaeulenDiagramm(saeulenWerte);
}

void zeichneSaeulenDiagramm(int[] saeule
  for (int x = 0; x < saeulenHoehen.length; x++) {
    for (int y = 0; y < saeulenHoehen[x]; y++) {
      world.setLeaf(x, world.getSizeY() - 1 - y, true);
    }
  }
}
```

Methode **myProgram**

«Kurzzeit-gedächtnis» der Methoden, lokaler Speicher («Stack»)

int[3] =
{ 3, 4, 5 }

«Langzeit-gedächtnis» des gesamten Programms, globaler Speicher («Heap»)

Säulendiagramm zeichnen: Arrays als Parameter

- Der **Parameter saeulenHoehen** ist innerhalb der Methode zeichneSaeulenDiagramm eine **normale Variable.**

- **Die Methode zeichneSaeulenDiagramm erhält das einen Verweis auf die Daten, auf die saeulenWerte verweist.** Würde die Methode zum Beispiel saeulenHoehen[x] = 0; setzen, würde das den Array der aufrufenden Methode ändern. Das gilt für alle Parameter von komplexen Datentypen (Arrays, Objekte, …).

Säulendiagramm zeichnen: Arrays als Parameter

```
public myProgram() {
  int[] saeulenWerte = new int[world.getSizeX()];
  // … saeulenWerte mit Benutzerwerten initialisieren …
  zeichneSaeulenDiagramm(saeulenWerte);
  // saeulenWerte[0] hat jetzt Wert 7, nicht mehr 0!
}

void zeichneSaeulenDiagramm(int[] saeulenHoehen) {
        saeulenHoehen[0] = 7;
}
```

Die Bezeichnung des Arrays (es gibt in diesem Programm nur einen Array!) spielt dabei überhaupt keine Rolle:

- Im Hauptprogramm wird der Array saeulenWerte definiert.
- Er wird als Parameter mit dem Namen saeulenHoehen an die Methoden zeichneSaeulenDiagramm übergeben.
- saeulenHoehnen «verweist» aber auf den gleichen Array, auch wenn es ein anderer Name ist als saeulenWerte!

Vom Umgang mit Daten

JavaKara programmieren: Arrays: Bildbearbeitung

Die Welt von Kara: Zwei-dimensionaler Array

```
public void myProgram() {
  boolean[][] neueFelder =
    new boolean[world.getSizeX()][world.getSizeY()];
  berechneNeueFelder(neueFelder);
  schreibeNeueFelder(neueFelder);
}
```

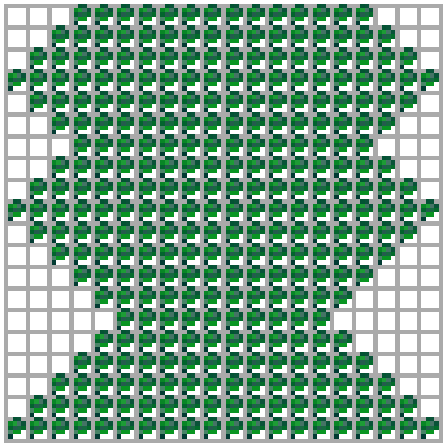
Das Programm soll eine Welt berechnen – oder: Bildbearbeitung in 2 Schritten:

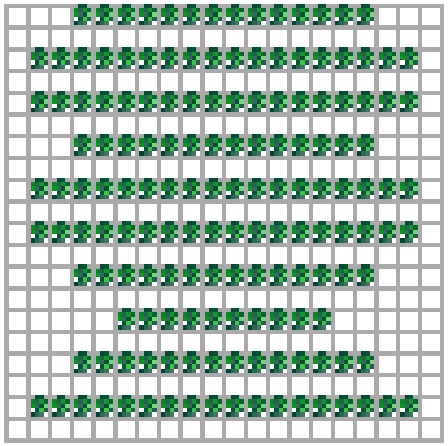
1. Die neue Welt soll zunächst zwischengespeichert werden (Variable neueFelder, Methode **berechneNeueFelder**).

2. Anschliessend soll die neue Welt dargestellt werden (Methode **schreibeNeueFelder**).

Ein Kleeblattbild «aufhellen»: Jede zweite Zeile auf weiss setzen

Ein schwarz-weiss Bild (wobei in Kara «schwarz» als Feld mit Kleeblatt verstanden wird, «weiss» als leeres Feld) kann zum Beispiel dadurch aufgehellt werden, dass jede zweite Zeile auf weiss gesetzt wird:





Ein Kleeblattbild «aufhellen»: Jede zweite Zeile auf weiss setzen

```
void berechneNeueFelder(boolean[][] neueFelder) {
  for (int y = 0; y < world.getSizeY(); y++) {
    for (int x = 0; x < world.getSizeX(); x++) {
      if (y % 2 == 0) {
        neueFelder[x][y] = world.isLeaf(x, y);
      } else {
        neueFelder[x][y] = false;
      }
    }
  }
}
```

Die Bedingung «y % 2 == 0» prüft, ob der Rest der Ganzzahldivision von y durch 2 0 ist.

- Falls ja (d.h. in den Spalten 0, 2, 4, ...), dann wird für das neue Bild an Koordinate (x,y) der Wert aus dem Originalbild an der gleichen Koordinate (x,y) verwendet.
- Falls nein (d.h. in den Spalten 1, 3, 5, ...), dann wird für das neue Bild an Koordinate (x,y) kein Kleeblatt gesetzt (unabhängig vom Originalbild).

Ein Kleeblattbild «aufhellen»: Jede zweite Zeile auf weiss setzen

```
void schreibeNeueFelder(boolean[][] neueFelder) {
  for (int y = 0; y < world.getSizeY(); y++) {
    for (int x = 0; x < world.getSizeX(); x++) {
      world.setLeaf(x, y, neueFelder[x][y]);
    }
  }
}
```

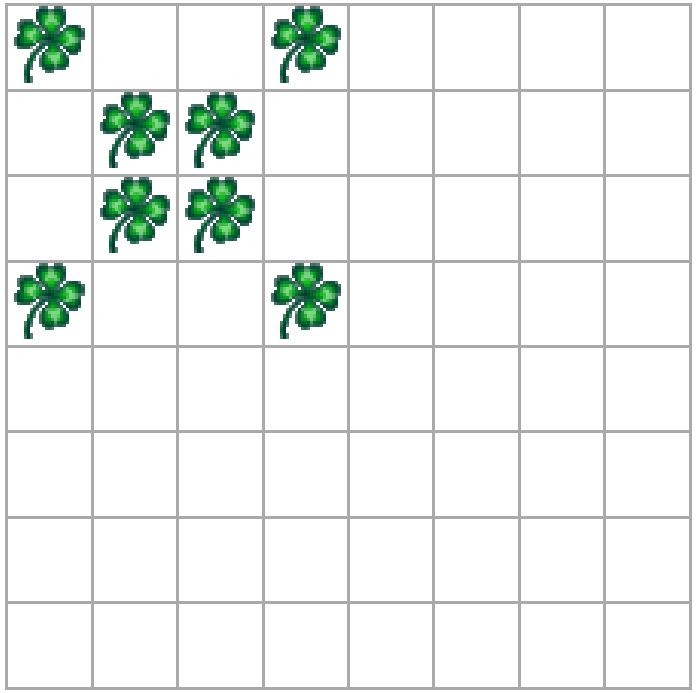
Diese Methode setze setzt Kleeblätter gemäss den Boole'schen Werte im zwei-dimensionalen Array neueFelder.

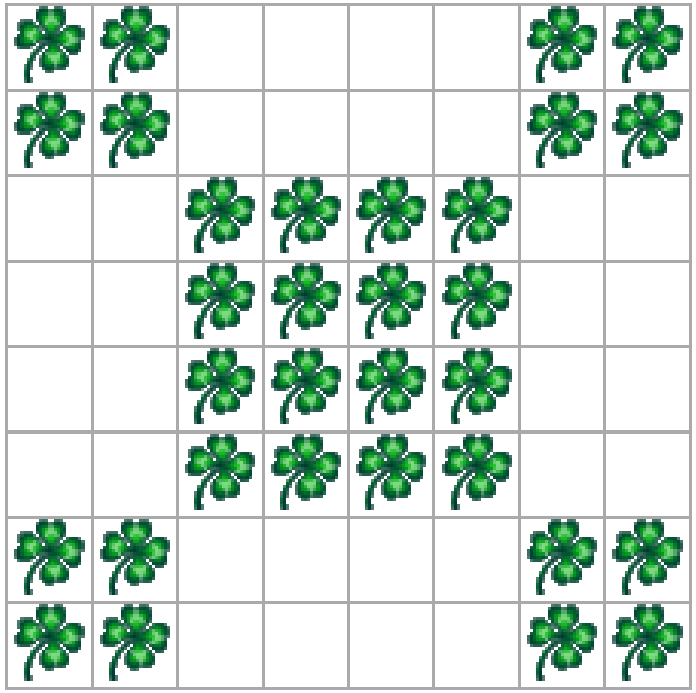
Diese Methode kann für alle «Bildbearbeitungsprogramme» in JavaKara verwendet werden.

Für dieses Beispiel hätten wir einfach die Original-Welt verändern können. Das Beispiel sollte lediglich das «Schema X» der Bildbearbeitung aufzeigen.

Ein Bild um Faktor zwei vergrössern

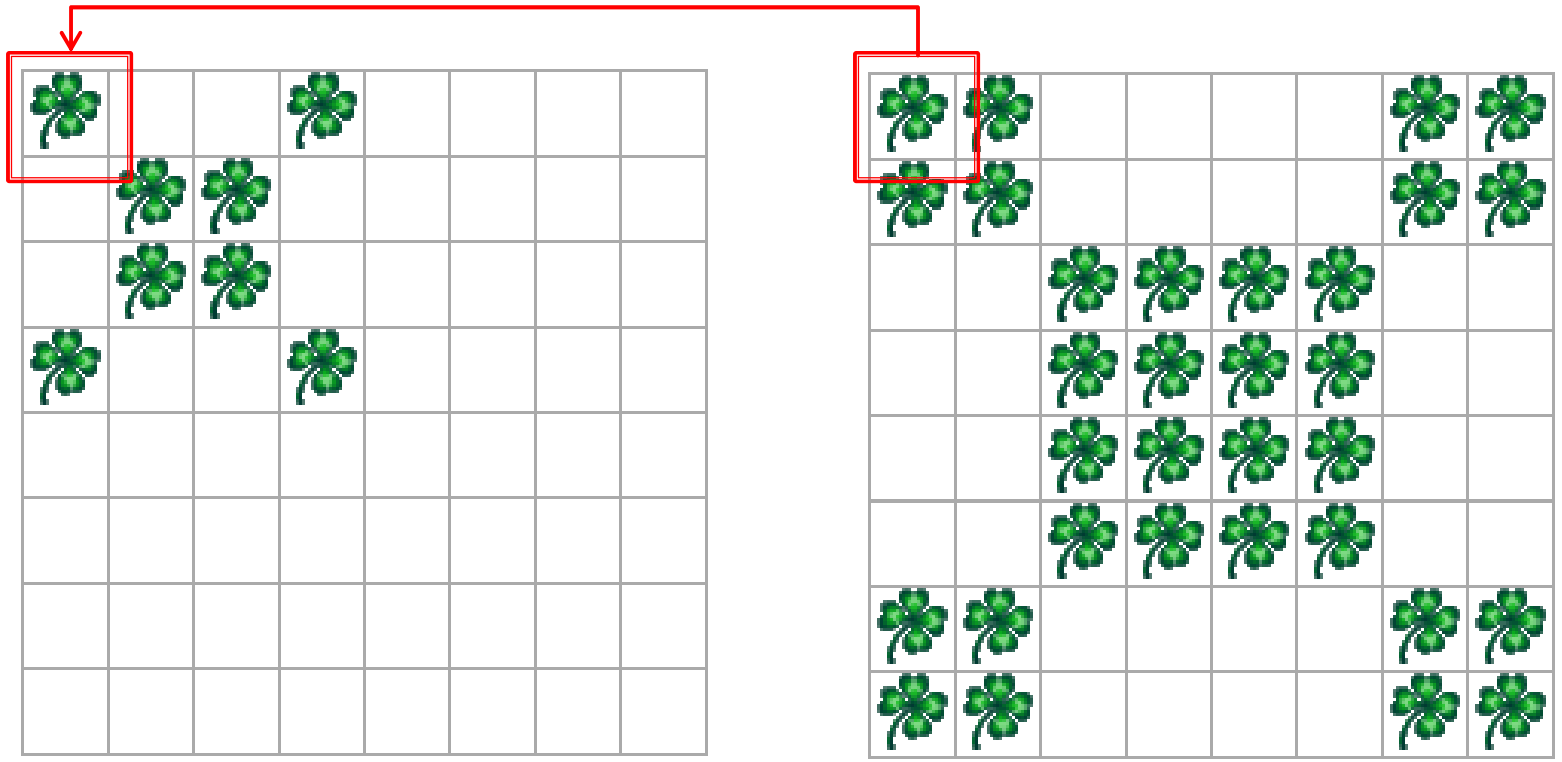
Die Grösse eines Kleeblattbildes soll verdoppelt werden:





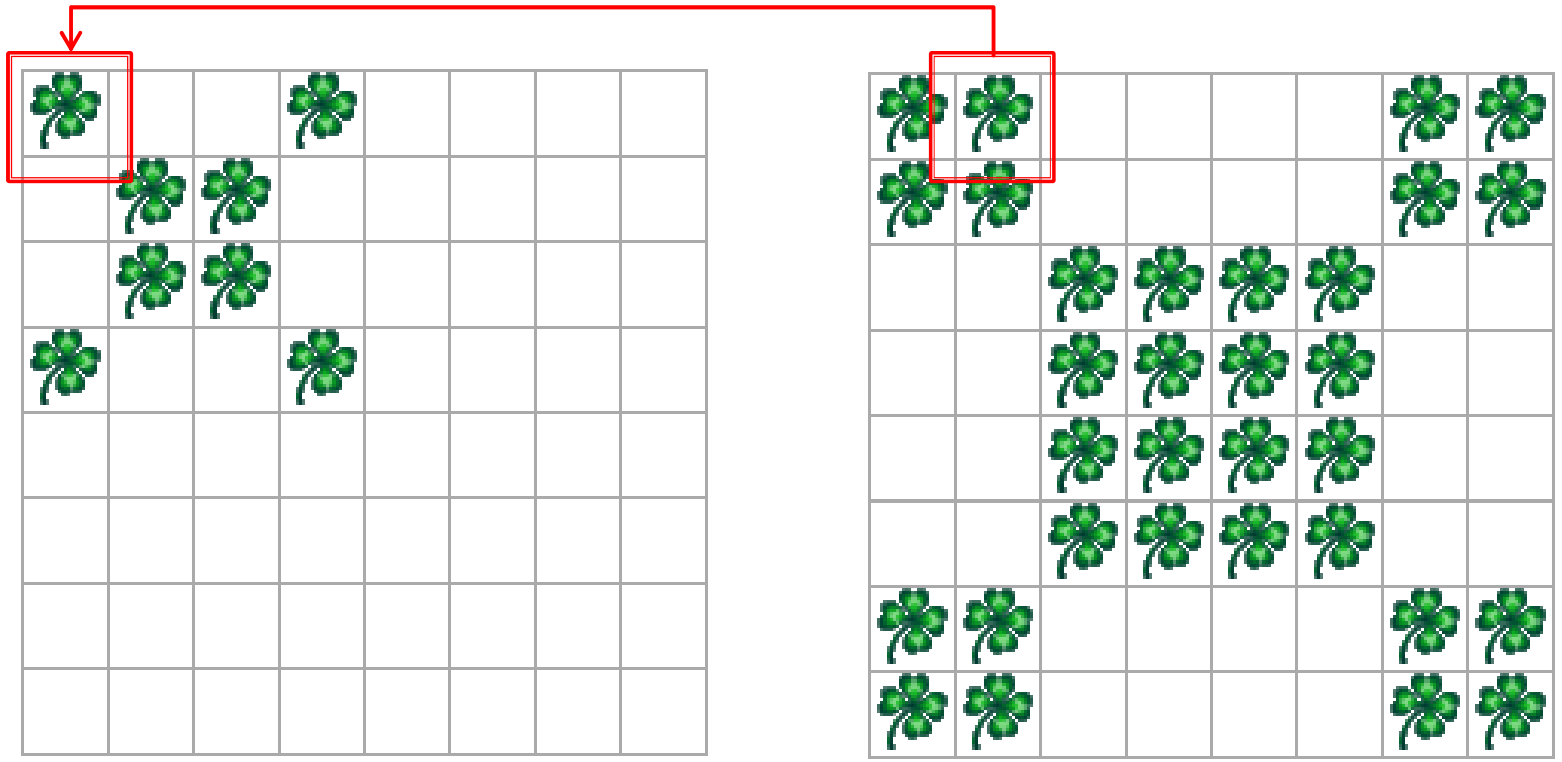
Ein Bild um Faktor zwei vergrössern

Die ersten Felder in Zeitlupe berechnet…



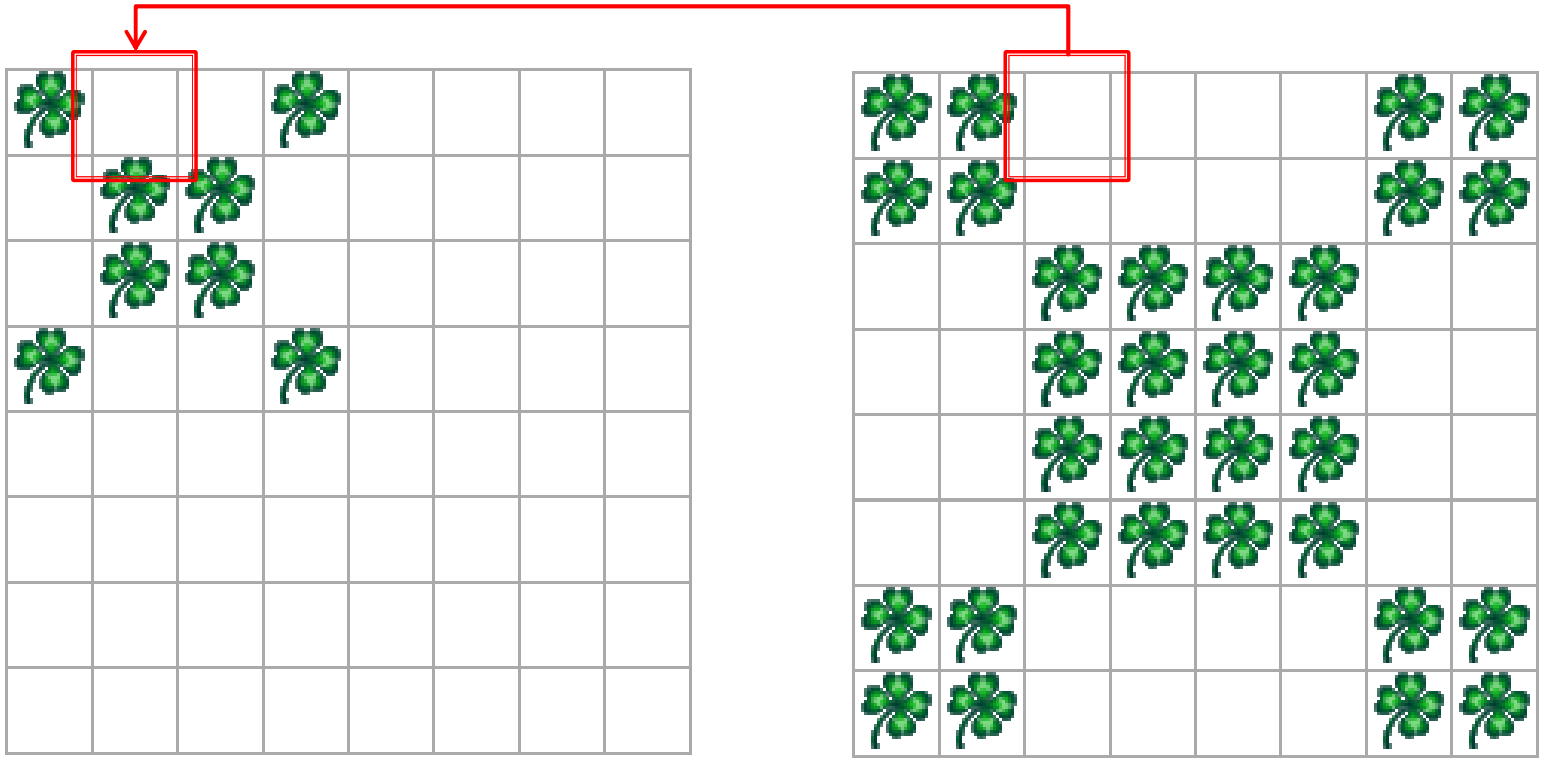
Ein Bild um Faktor zwei vergrössern

Die ersten Felder in Zeitlupe berechnet…



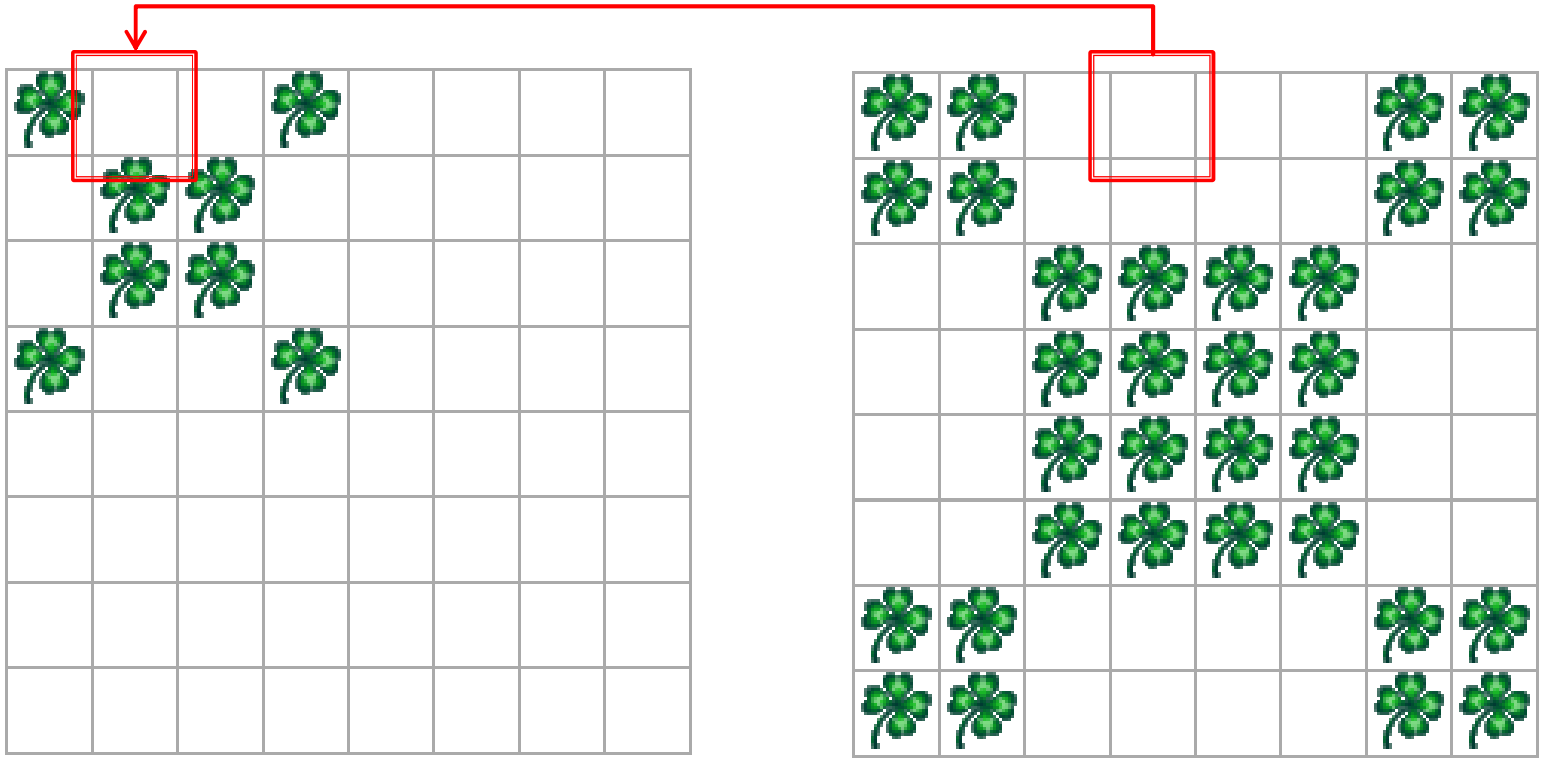
Ein Bild um Faktor zwei vergrössern

Die ersten Felder in Zeitlupe berechnet…



Ein Bild um Faktor zwei vergrössern

Die ersten Felder in Zeitlupe berechnet…



Ein Bild um Faktor zwei vergrössern

Vorgehensweise für Berechnung des neuen Bildes

Vom neuen Bild ausgehend überlegen wir uns für jedes Feld der neuen Welt, wie wir aufgrund vom alten Bild entscheiden, ob ein Kleeblatt gelegt werden soll.

Betrachten wir Beispiel-Koordinaten:

Wert von x(neu)=0 => Wert von x(alt)=0
Wert von x(neu)=1 => Wert von x(alt)=0
Wert von x(neu)=2 => Wert von x(alt)=1
Wert von x(neu)=3 => Wert von x(alt)=1
...

Allgemein formuliert:

Wert für x(neu) = Wert von x(alt)/2
wobei / eine Ganzzahldivision ist

Analoge Herleitung für Umrechnung von **Wert für y(neu) = Wert von y(alt)/2**

Ein Bild um Faktor zwei vergrössern

```
void berechneNeueFelder(boolean[][] neueFelder) {
  for (int y = 0; y < world.getSizeY(); y++) {
    for (int x = 0; x < world.getSizeX(); x++) {
      neueFelder[x][y] = world.isLeaf(x / 2, y / 2);
    }
  }
}
```

Ob **im neuen Bild an Koordinate (x,y)** ein Kleeblatt liegt oder nicht, hängt davon ab, ob **im alten Bild an Koordinate (x/2, y/2)** ein Kleeblatt liegt oder nicht.