




## Verwendung von Arrays

|                                  | eindimensional   | zweidimensional   |
|----------------------------------|--|---|
| Deklaration:                     | <code>int[] zahlen;</code>                                       | <code>Color[][] pixel;</code>   |
| Initialisierung:                 | <code>zahlen = new int[20];</code>                               | <code>pixel = new Color[100][50];</code>  |
| Zugriff auf ein<br>Arrayelement: | <code>zahlen[5] = 10;</code><br><code>zahlen[i] = 12;</code>     | <code>pixel[4][7] = Color.BLACK;</code><br><code>pixel[x][y] = Color.BLUE;</code>                           |
| Direkte<br>Initialisierung:      | <code>int[] buchstaben =</code><br><code>{'i', 'n', 'f'};</code> | <code>double[][] faktoren =</code><br><code>{{ 4.0, -1.2, 2.3},</code><br><code> { -2.1, 2.1, 0.0}};</code> |

**Achtung:** Bei zweidimensionalen, direkt initialisierten Arrays gibt die erste Position die Zeile und die zweite die Spalte an: `faktoren[0][2]` hat im Beispiel oben den Wert 2,3.

### Auftrag:

1. Gib die notwendigen Java-Befehle an, um ...

-  a) ein Array zu deklarieren und zu initialisieren, um 35 Kommazahlen zu speichern. Welche Position hat das erste bzw. das letzte Element des Arrays?
- b) ein zweidimensionales Array für ein Bild der Größe 800x350 Pixel zu deklarieren und zu initialisieren.
- c) den 4. Wert des eindimensionalen Arrays auf 45,2 zu setzen und den 3. Wert auf 13,2.
- d) den 5. Wert des eindimensionalen Arrays auf die Summe des 3. und des 4. Wertes zu setzen.
- e) das Pixel an der Stelle  $x = 17$  und  $y = 34$  auf Rot zu setzen.
- f) dem Pixel an der Stelle  $x = 123$  und  $y = 12$  die Farbe des Pixels an der Stelle  $x = 76$  und  $y = 99$  zuzuweisen.

Folgender Java-Code kopiert die Pixel eines Bildes in ein neues Bild.

```
int breite = originalbild.getWidth();
int hoehe = originalbild.getHeight();

Color[][] pixel = originalbild.getPixelArray();
Color[][] pixelNeu = new Color[breite][hoehe];

for(int x=0; x < breite; x++) {
    for(int y=0; y < hoehe; y++) {
        pixelNeu[x][y] = pixel[(breite-1)-x][y];
    }
}

Picture neuesBild = new Picture();
neuesBild.setPixelArray(pixelNeu);
return neuesBild;
```

Die doppelte For-Schleife ermöglicht es, jedes Pixel des neuen Bildes zu definieren.



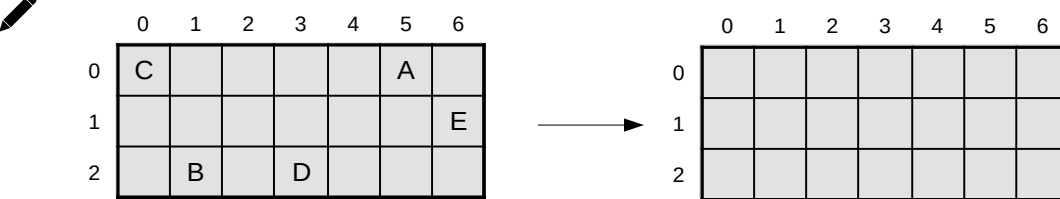
## Auftrag:

2. Gib an, welche Werte die Schleifenvariablen  $x$  und  $y$  beim jeweiligen Schleifendurchlauf erhalten, falls das Bild  $7 \times 3$  Pixel groß ist:

| Durchgang | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|-----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| x         |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |
| y         |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |

3. Gib an, welches Pixel des Originalbildes an die Position  $(5|2)$  wandert. Woher kommt das Pixel mit der Position  $(0|0)$ ? Wohin geht das Pixel, das ursprünglich an der Position  $(3|1)$  im Originalbild war?

4. Zeichne ein, an welcher Stelle die Pixel A- E im neuen Bild zu finden sind.



5. Beschreibe mit 3-4 Worten, wie das neue Bild aus dem alten hervorgeht.

## Bildbearbeitungsalgorithmen 1: Geometrische Bildoperationen

### Programmierauftrag

6. Öffne das Projekt "01\_geometrische\_bildoperationen" in BlueJ. Erzeuge ein neues Beispielbild. (Rechte Maustaste -> new Beispielbild()). Lade ein anderes Bild aus dem Ordner "images".  
Achtung: Strings mit Anführungszeichen eingeben: z.B. "beispiel2.png".

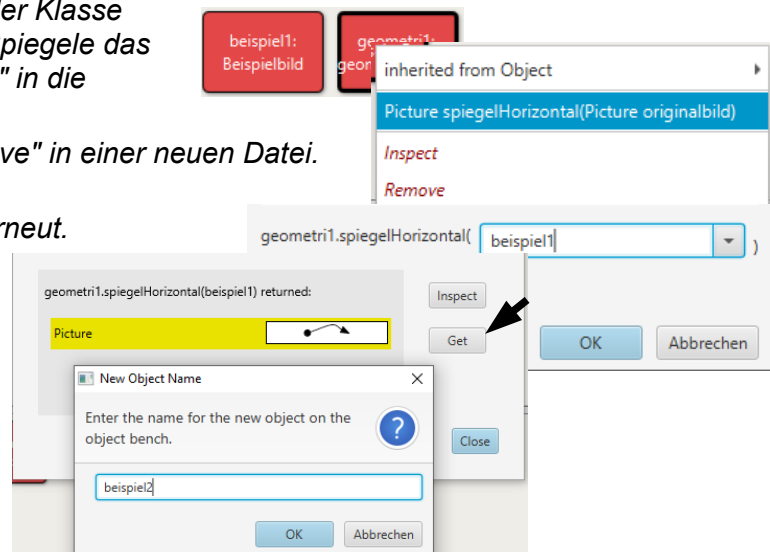


7. Erzeuge außerdem ein Objekt der Klasse `geometrischeBildoperationen`. Spiegle das Bild horizontal. Hole es mit "Get" in die Objektleiste.

8. Speichere das neue Bild mit "save" in einer neuen Datei.

9. Spiegle das gespiegelte Bild erneut.

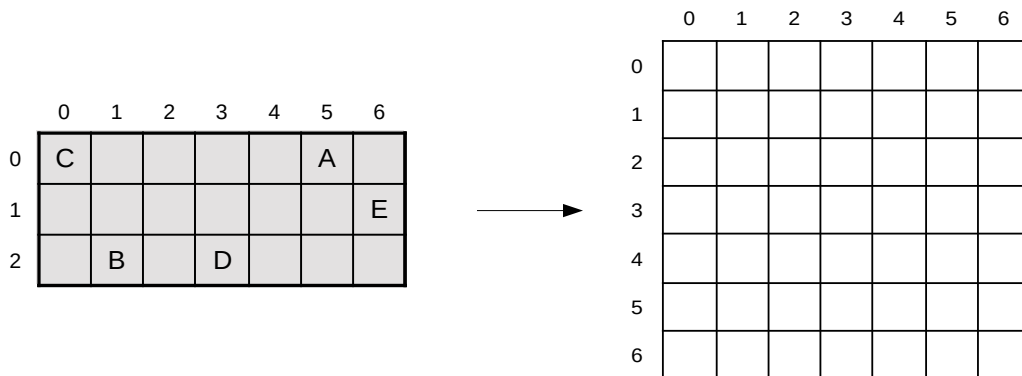
10. Implementiere die Methode `public Picture spiegelVertikal(Picture originalbild)` analog zur `spiegelHorizontal`-Methode. Teste deine neue Methode.





## Drehung des Bildes um 90°

Ein Bild soll um 90° nach links gedreht werden.



### Auftrag:

11. Markiere im Raster den Bereich, den das gedrehte Bild abdeckt. Wie kann die Breite und die Höhe des neuen Bildes bestimmt werden?
12. Zeichne ein, an welcher Stelle die Pixel A-E im neuen Bild zu finden sind.
13. Bestimme die Koordinaten der Punkte A-E im neuen Bild und im Originalbild.

|                       | A | B | C | D | E |
|-----------------------|---|---|---|---|---|
| $x_{\text{neu}}$      | 0 |   |   |   |   |
| $y_{\text{neu}}$      | 1 |   |   |   |   |
| $x_{\text{original}}$ | 5 |   |   |   |   |
| $y_{\text{original}}$ | 0 |   |   |   |   |

14. Ermittle Formeln, mit denen die Koordinaten des ursprünglichen Pixel bestimmt werden können:

$x_{\text{original}} =$

$y_{\text{original}} =$

In Java: `pixelNeu[x][y] = pixel[ ][ ];`

### Programmierauftrag:

15. Implementiere die Methode `Picture dreheLinks(Picture originalbild)`.



16. Stelle die gleichen Überlegungen (Aufgaben 11-14) für eine Drehung um 90° nach rechts an. Implementiere die Methode `Picture dreheRechts(Picture originalbild)`.



17. Überlege, wie du mit den bestehenden Methoden eine Drehung um 180° realisieren könntest. Teste dies in BlueJ durch geeignete Methodenaufrufe. Implementiere eine Methode `Picture drehe180(Picture originalbild)` unter Verwendung der bestehenden Methoden.



Hinweis: Das Zwischenergebnis darf auch als Bild angezeigt werden.