

## Übung 05: Arrays

Abgabetermin: xx.xx.xxxx

**Name:** Name, Vorname

**Matrikelnummer:** 0XXXXXX

**Gruppe:**  G1 (Prähofer)     G2 (Wolfinger)     G3 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Pkte
Aufgabe 05.x	12	<input checked="" type="checkbox"/>	Prosabeschreibung Ablaufdiagramm Java-Programm Testplan Testergebnisse	Java-Programm	<input type="checkbox"/>	
					<input type="checkbox"/>	

### Aufgabe 05.x: Mischen

Gegeben sind zwei beliebig große Felder *a* und *b*, die positive ganze Zahlen (vom Datentyp *int*) in aufsteigend sortierter Reihenfolge enthalten.

Gesucht ist ein Java-Programm, das ein aufsteigend sortiertes Feld *c* aus *a* und *b* berechnet, sodass *c* alle Zahlen in aufsteigend sortierter Reihenfolge enthält, die in *a* und/oder in *b* vorkommen.

Beachten Sie, dass in jedem Feld (also auch im Ergebnisfeld *c*) Werte mehrfach vorkommen können.

Beispiel:

```
a =  2   4   6   10  15  15
b =  3   4   5   10
```

---

```
c =  2   3   4   4   5   6   10  10  15  15
```

Konstruieren Sie Ihr Java-Programm so, dass die Werte für die Felder *a* und *b* zuerst eingelesen werden, dann das Feld *c* berechnet wird und schließlich der Inhalt von *c* ausgegeben wird. Einlesen und Ausgeben soll von und auf die Konsole erfolgen. Um das Einlesen zu vereinfachen, soll vor der Zahlenfolge die Länge der Folge eingegeben werden.

Beispieldialog:

```
Folge a:
  Länge: 6
  Folge: 2 4 4 10 15 15
Folge b:
  Länge: 4
  Folge: 3 4 5 10
```

```
Ergebnisfolge c: 2 3 4 4 4 5 10 10 15 15
```

Gehen Sie bei der Lösung wie folgt vor:

1. Stellen Sie das Verfahren in Prosa dar.
2. Stellen Sie die Methode zum Mischen (!) in einem Ablaufdiagramm dar.
3. Realisieren Sie das Verfahren in Java.
4. Stellen Sie einen Testplan auf, d.h. stellen Sie interessante Testfälle auf und geben Sie die erwarteten Ergebnisse an.
5. Testen Sie Ihr Programm nach diesem Testplan.

## Lösung

### 1) Lösungsidee:

Beim Mischen der beiden Arrays  $a$  und  $b$  geht man folgend vor:

Man erzeugt sich ein Ergebnisarray  $c$  mit der Länge von  $a$  plus der Länge von  $b$ . Dann übernimmt man jeweils die nächste Zahl aus dem Array  $a$  oder  $b$ , welche kleiner ist. Hat man von einem Array alle Zahlen bereits übernommen, muss man noch die restlichen des anderen übernehmen.

Folgende Skizze verdeutlicht das Verfahren.

$a = 2 \ 4 \ 6 \ 10 \ 15 \ 15$   
 $b = \blacktriangledown \ 4 \ 5 \ 10$   
 $c = 2$

$a = 2 \ 4 \ 6 \ 10 \ 15 \ 15$   
 $b = \textcircled{3} \ 4 \ 5 \ 10$   
 $c = 2 \ 3$

$a = 2 \ \textcircled{4} \ 6 \ 10 \ 15 \ 15$   
 $b = 3 \ 4 \ \blacktriangledown \ 10$   
 $c = 2 \ 3 \ 4$

$a = 2 \ 4 \ 6 \ 10 \ 15 \ 15$   
 $b = 3 \ 4 \ \textcircled{5} \ 10$   
 $c = 2 \ 3 \ 4 \ 4$

$a = 2 \ 4 \ 6 \ 10 \ 15 \ 15$   
 $b = 3 \ 4 \ \textcircled{5} \ 10$   
 $c = 2 \ 3 \ 4 \ 4 \ 5$   
 ...

### Algorithmus in Prosa:

Einlesen Array  $a$

Einlesen Array  $b$

Erzeuge Array  $c$  mit Länge von  $a$  + Länge von  $b$

Solange nicht alle Zahlen von  $a$  und  $b$  übernommen wurden

    Wenn bereits alle Zahlen von  $a$  übernommen wurden

        Übernimm nächste Zahl von  $b$  in  $c$

    Sonst wenn bereits alle Zahlen von  $b$  übernommen wurden

        Übernimm nächste Zahl von  $a$  in  $c$

    Sonst wenn nächste Zahl von  $a \leq$  nächste Zahl von  $b$

        Übernimm nächste Zahl von  $a$  in  $c$

    Sonst

        Übernimm nächste Zahl von  $b$

Gib  $c$  aus

Verfeinerter Algorithmus in Prosa:

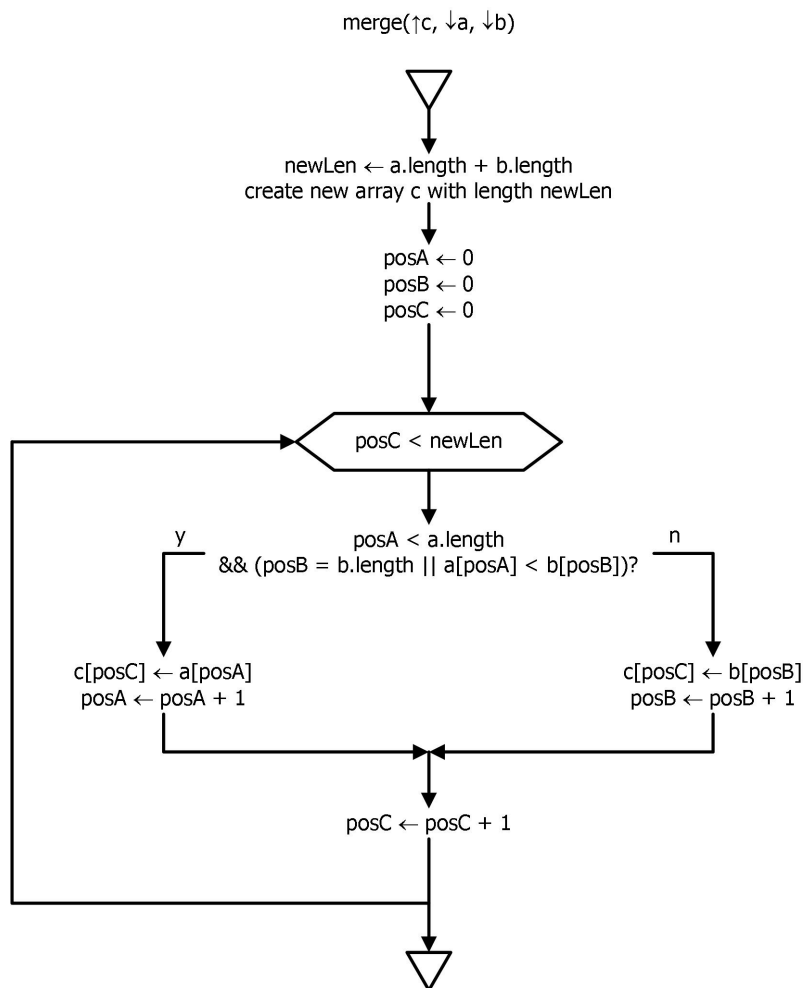
Idee: Verwende Positionsvariablen `posA`, `posB` und `posC`. `posA` und `posB` repräsentieren dabei die Position der nächsten Zahl von `a` bzw. `b`, die übernommen werden soll. `posC` stellt die nächste Einfügestelle in `c` dar.

```
Einlesen Array a (mit Methode readArray())
Einlesen Array b (mit Methode readArray())
Erzeuge Array c mit Länge von a + Länge von b
Verwende Indexvariablen posA, posB und posC; initialisiere diese mit 0
Solange nicht alle Zahlen von a und b übernommen wurden (d.h. posC < c.length)
    Wenn bereits alle Zahlen von a übernommen wurden (d.h. posA == a.length)
        Weise aktueller Stelle posC in c den Wert von b an Stelle posB zu
        Erhöhe Index posB
    Sonst wenn bereits alle Zahlen von b übernommen wurden (d.h. posB == b.length)
        Weise aktueller Stelle posC in c den Wert von a an Stelle posA zu
        Erhöhe Index posA
    Sonst wenn nächste Zahl von a <= nächste Zahl von b (d.h. a[posA] <= b[posB])
        Weise aktueller Stelle posC in c den Wert von a an Stelle posA zu (c[posC] = a[posA])
        Erhöhe Index posA
    Sonst (d.h. a[posA] > b[posB])
        Weise aktuelles Stelle posC in c den Wert von b an Stelle posB zu
        Erhöhe Index posB
    Erhöhe Index posC
Gib c aus (mit Methode printArray)
```

Optimierter Algorithmus in Prosa:

Idee für Optimierung: Die Fallunterscheidungen für die Zuweisung bringen vier Zuweisungen und damit eine Codeverdopplung mit sich. Durch Umformulierung der Bedingung kann auf zwei Fälle reduziert werden.

```
Einlesen Array a (mit Methode readArray())
Einlesen Array b (mit Methode readArray())
Erzeuge Array c mit Länge von a + Länge von b
Verwende Indexvariablen posA, posB und posC; initialisiere diese mit 0
Solange nicht alle Zahlen von a und b übernommen wurden (d.h. posC < c.length)
    Wenn noch nicht alle Zahlen von a übernommen wurden (d.h. posA < a.length)
    und (entweder bereits alle Zahlen aus b übernommen wurden
        oder nächste Zahl von a <= nächste Zahl von b (d.h. a[posA] <= b[posB]))
        Weise aktueller Stelle posC in c den Wert von a an Stelle posA zu
        Erhöhe Index posA
    Sonst (d.h. a[posA] > b[posB] oder bereits alle Zahlen aus a übernommen)
        Weise aktueller Stelle posC in c den Wert von b an Stelle posB zu
        Erhöhe Index posB
    Erhöhe Index posC
Gib c aus (mit Methode printArray)
```

2) Ablaufdiagramm3) Java-Programm

```

1 public class MergeSort
2 {
3     public static void main(String[] args) {
4         int[] a = readIntArray();
5         int[] b = readIntArray();
6
7         printArray("a = ", a);
8         printArray("b = ", b);
9         int[] c = merge(a, b);
10        printArray("c = ", c);
11    }
12
13    static int[] merge(int[] a, int[] b) {
14        int newLen = a.length + b.length;
15        int[] c = new int[newLen];
16        int posA=0, posB=0;
17        for(int posC=0; posC < newLen; posC++) {
18            if(posA < a.length && (posB == b.length || a[posA] < b[posB])) {
19                c[posC] = a[posA];
20                posA++;
21            } else {
22                c[posC] = b[posB];
23                posB++;
24            }
25        }
26        return c;
27    }
28
29    public static int[] readIntArray() {
30        Out.print("Bitte Laenge des Arrays eingeben: ");
31        int len = In.readInt();
32        int[] arr = new int[len];
33        for(int i = 0; i < arr.length; i++) {
34            Out.print(" arr[" + i + "] : ");
35            arr[i] = In.readInt();

```

```
36     }
37     return arr;
38 }
39
40 static void printArray(String prefix, int [] arr) {
41     Out.print(prefix);
42     for(int x: arr) {
43         Out.print(x + " ");
44     }
45     Out.println();
46 }
47 }
```

#### 4) Testplan

##### 1. Fall: Beispiel aus Angabe

Folge a: 2 4 6 10 15 15

Folge b: 3 4 5 10

Ergebnisfolge c: 2 3 4 4 5 6 10 10 15 15

##### 2. Fall: Beispiel aus Angabe mit a und b vertauscht

Folge a: 3 4 5 10

Folge b: 2 4 6 10 15 15

Ergebnisfolge c: 2 3 4 4 5 6 10 10 15 15

##### 3. Fall: a enthält alle kleinen, b alle größeren Zahlen

Folge a: 1 2 3 4

Folge b: 5 6 7 8 9

Ergebnisfolge c: 1 2 3 4 5 6 7 8 9

##### 4. Fall: Fall 3 mit a und b vertauscht

Folge a: 5 6 7 8 9

Folge b: 1 2 3 4

Ergebnisfolge c: 1 2 3 4 5 6 7 8 9

##### 5. Fall: gleiche Zahlenfolge von a und b

Folge a: 1 2 3 4 5

Folge b: 1 2 3 4 5

Ergebnisfolge c: 1 1 2 2 3 3 4 4 5 5

##### 6. Fall: Lauter gleiche Zahlen, a und b gleich

Folge a: 1 1 1

Folge b: 1 1 1

Ergebnisfolge c: 1 1 1 1 1 1

##### 7. Fall: Nur jeweils eine Zahl in a und b

Folge a: 1

Folge b: 2

Ergebnisfolge c: 1 2

##### 8. Fall: Fall 7 mit a und b vertauscht

Folge a: 2

Folge b: 1

Ergebnisfolge c: 1 2

##### 9. Fall: Nur jeweils eine (gleiche) Zahl in a und b

Folge a: 1

Folge b: 1

Ergebnisfolge c: 1 1

##### 10. Fall: Folge a hat Länge 0

Folge a:

Folge b: 1 2 3 4 5

Ergebnisfolge c: 1 2 3 4 5

**11. Fall: Folge b hat Länge 0**

Folge a: 1 2 3 4 5

Folge b:

Ergebnisfolge c: 1 2 3 4 5

**12. Fall: Beide Folgen haben Länge 0**

Folge a:

Folge b:

Ergebnisfolge c:

**5) Test****Testfall #1 (Beispiel aus Angabe)**

Bitte Laenge des Arrays eingeben: 6

arr[0] : 2

arr[1] : 4

arr[2] : 6

arr[3] : 10

arr[4] : 15

arr[5] : 15

Bitte Laenge des Arrays eingeben: 4

arr[0] : 3

arr[1] : 4

arr[2] : 5

arr[3] : 10

a = 2 4 6 10 15 15

b = 3 4 5 10

c = 2 3 4 4 5 6 10 10 15 15

**Testfall #2 (Beispiel aus Angabe mit a und vertauscht)**

Bitte Laenge des Arrays eingeben: 4

arr[0] : 3

arr[1] : 4

arr[2] : 5

arr[3] : 10

Bitte Laenge des Arrays eingeben: 6

arr[0] : 2

arr[1] : 4

arr[2] : 6

arr[3] : 10

arr[4] : 15

arr[5] : 15

a = 3 4 5 10

b = 2 4 4 10 15 15

c = 2 3 4 4 5 6 10 10 15 15

**Testfall #3 (a enthält alle kleinen, b all größeren Zahlen)**

Bitte Laenge des Arrays eingeben: 4

arr[0] : 1

arr[1] : 2

arr[2] : 3

arr[3] : 4

Bitte Laenge des Arrays eingeben: 5

arr[0] : 5

arr[1] : 6

arr[2] : 7

arr[3] : 8

arr[4] : 9

a = 1 2 3 4

b = 5 6 7 8 9

c = 1 2 3 4 5 6 7 8 9

**Testfall #4 (Fall 3 mit a und b vertauscht)**

```
Bitte Laenge des Arrays eingeben: 5
arr[0] : 5
arr[1] : 6
arr[2] : 7
arr[3] : 8
arr[4] : 9
Bitte Laenge des Arrays eingeben: 4
arr[0] : 1
arr[1] : 2
arr[2] : 3
arr[3] : 4
a = 5 6 7 8 9
b = 1 2 3 4
c = 1 2 3 4 5 6 7 8 9
```

**Testfall #5 (gleiche Zallenfolge von und b)**

```
Bitte Laenge des Arrays eingeben: 5
arr[0] : 1
arr[1] : 2
arr[2] : 3
arr[3] : 4
arr[4] : 5
Bitte Laenge des Arrays eingeben: 5
arr[0] : 1
arr[1] : 2
arr[2] : 3
arr[3] : 4
arr[4] : 5
a = 1 2 3 4 5
b = 1 2 3 4 5
c = 1 1 2 2 3 3 4 4 5 5
```

**Testfall #6 (Lauter gleiche Zahlen, a und b gleich)**

```
Bitte Laenge des Arrays eingeben: 3
arr[0] : 1
arr[1] : 1
arr[2] : 1
Bitte Laenge des Arrays eingeben: 3
arr[0] : 1
arr[1] : 1
arr[2] : 1
a = 1 1 1
b = 1 1 1
c = 1 1 1 1 1 1
```

**Testfall #7 (Nur jeweils eine Zahl in a und b)**

```
Bitte Laenge des Arrays eingeben: 1
arr[0] : 1
Bitte Laenge des Arrays eingeben: 1
arr[0] : 2
a = 1
b = 2
c = 1 2
```

**Testfall #8 (Fall 7 mit a und b vertauscht)**

```
Bitte Laenge des Arrays eingeben: 1
arr[0] : 2
Bitte Laenge des Arrays eingeben: 1
arr[0] : 1
a = 2
b = 1
```

```
c = 1 2
```

**Testfall #9 (Nur jeweils eine gleiche Zahl in a und b)**

```
Bitte Laenge des Arrays eingeben: 1
arr[0] : 1
Bitte Laenge des Arrays eingeben: 1
arr[0] : 1
a = 1
b = 1
c = 1 1
```

**Testfall #10 (Folge a hat Länge 0)**

```
Bitte Laenge des Arrays eingeben: 0
Bitte Laenge des Arrays eingeben: 5
arr[0] : 1
arr[1] : 2
arr[2] : 3
arr[3] : 4
arr[4] : 5
a =
b = 1 2 3 4 5
c = 1 2 3 4 5
```

**Testfall #11 (Folge b hat Länge 0)**

```
Bitte Laenge des Arrays eingeben: 5
arr[0] : 1
arr[1] : 2
arr[2] : 3
arr[3] : 4
arr[4] : 5
Bitte Laenge des Arrays eingeben: 0
a = 1 2 3 4 5
b =
c = 1 2 3 4 5
```

**Testfall #12 (beide Folgen haben Länge 0)**

```
Bitte Laenge des Arrays eingeben: 0
Bitte Laenge des Arrays eingeben: 0
a =
b =
c =
```