

## Übungen zu Einführung in die Informatik: Programmierung und Software-Entwicklung: Lösungsvorschlag

### Aufgabe 9-1

### Erzeugen von Arrays

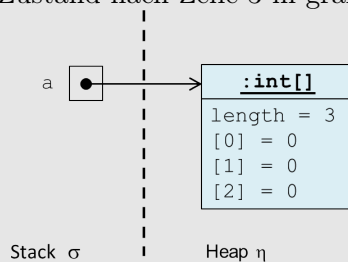
Präsenz

Gegeben ist folgendes Programm, welches nach verschiedenen Prinzipien Arrays erzeugt. Geben Sie für dieses Programm den Zustand des Speichers (Stack und Heap) nach den Zeilen 3, 6, 8, 10 und 17 in grafischer Repräsentation an.

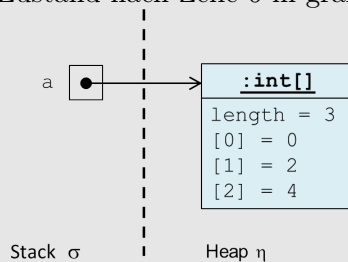
```
1 public class CreateArrays {  
2     public static void main(String[] args) {  
3         int[] a = new int[3];  
4         for (int i = 0; i < a.length; i++) {  
5             a[i] = 2 * i;  
6         }  
7  
8         int[] b = { 1, 3 };  
9  
10        int[] c = new int[a.length + b.length];  
11        for (int i = 0; i < c.length; i++) {  
12            if (i < a.length) {  
13                c[i] = a[i];  
14            } else {  
15                c[i] = b[i - a.length];  
16            }  
17        }  
18    }  
19 }
```

### Lösung:

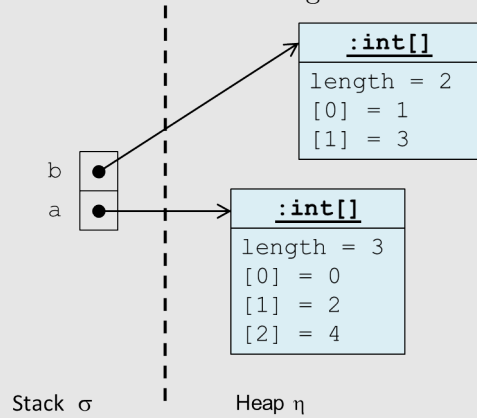
Zustand nach Zeile 3 in grafischer Notation:



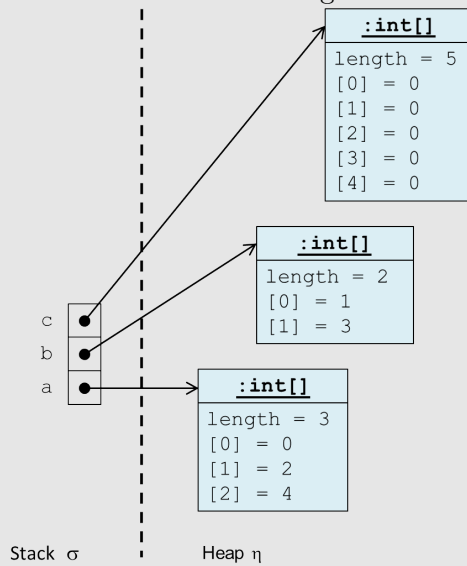
Zustand nach Zeile 6 in grafischer Notation:



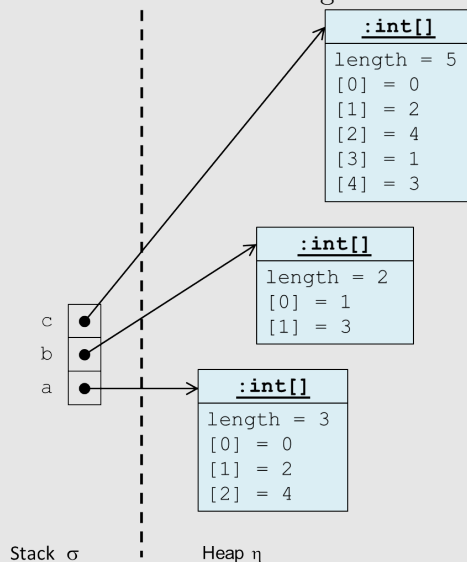
Zustand nach Zeile 8 in grafischer Notation:



Zustand nach Zeile 10 in grafischer Notation:



Zustand nach Zeile 17 in grafischer Notation:



## Aufgabe 9-2

## Erzeugen von Arrays

## Hausaufgabe

Gegeben ist folgendes Programm, welches nach verschiedenen Prinzipien Arrays erzeugt. Geben Sie für dieses Programm den Zustand des Speichers (Stack und Heap) nach den Zeilen 3, 5, 10,

und 12 in grafischer Repräsentation an.

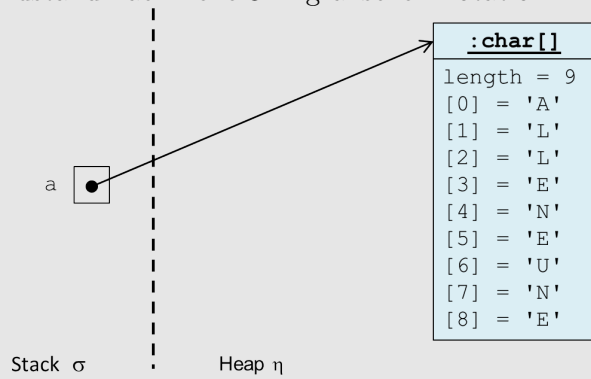
```

1 public class CreateArrays2 {
2     public static void main(String[] args) {
3         char[] a = {'A','L','L','E','N','E','U','N','E'};
4
5         char[] b = new char[a.length/2 + 1];
6         for(int i = 0; i < a.length/2+1; i++) {
7             b[i] = a[i+a.length/2];
8         }
9         b[3] = 'E';
10        b[4] = 'S';
11
12        char[] c = {'G', 'U', 'T', 'E', 'S'};
13    }
14 }

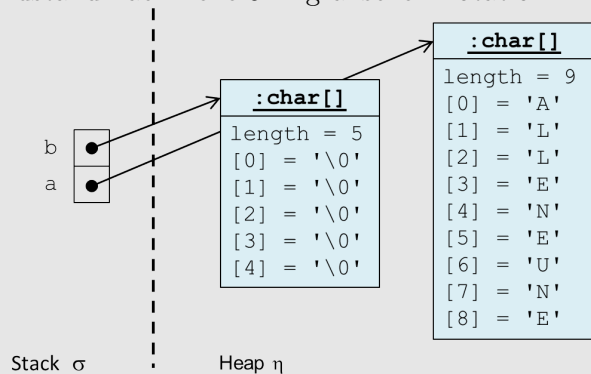
```

### Lösung:

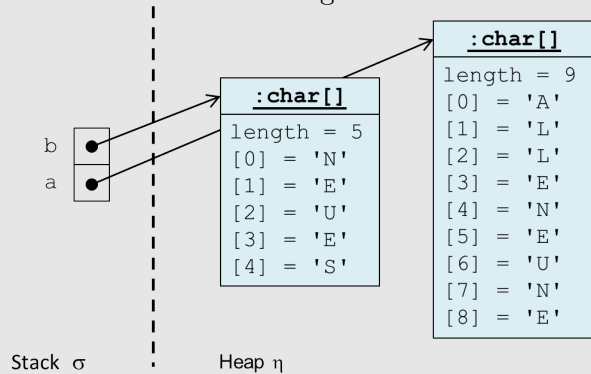
Zustand nach Zeile 3 in grafischer Notation:



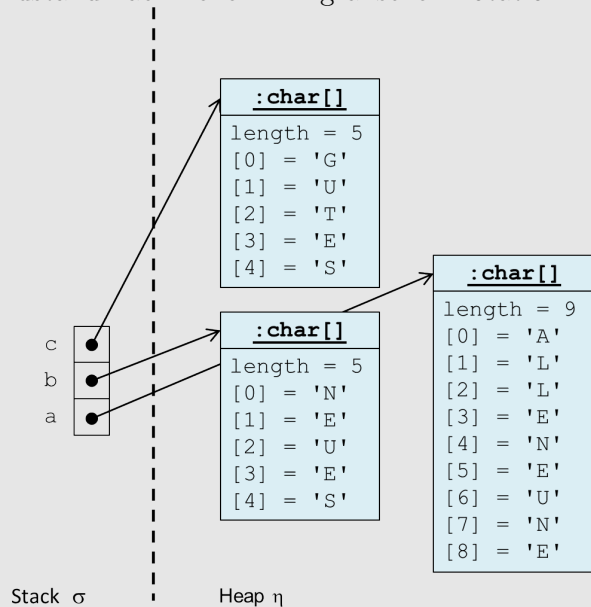
Zustand nach Zeile 5 in grafischer Notation:



Zustand nach Zeile 10 in grafischer Notation:



Zustand nach Zeile 12 in grafischer Notation:



### Aufgabe 9-3

### Operationen auf Arrays

Präsenz

In dieser Aufgabe sollen Sie ein Programm mit einer grafischen Benutzeroberfläche implementieren, welches verschiedene Operationen auf Arrays zur Verfügung stellt.

- a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll zwei Buttons, einen zum Revertieren eines Arrays und einen zum lexikographischen Vergleich zweier Arrays, geben. Darunter soll ein Ausgabebereich platziert werden, in dem Rückmeldung über die Ergebnisse gegeben wird.

Schreiben Sie eine Klasse **ArrayFrame**, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse **ArrayFrameMain**, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse **ArrayFrame** abspeichern.

### Lösung:

Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse **ArrayFrame**, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse **JFrame** erbt. Ihre Klasse **ArrayFrame** soll drei Attribute haben:

- je ein Attribut vom Klassentyp `JButton` für Buttons zum Revertieren eines Arrays und zum lexikografischen Vergleich zweier Arrays,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über Revertierung und Vergleich dient.

Schreiben Sie einen Konstruktor, der den `ArrayFrame` mit einem entsprechenden Titel und Größe (wir wählen hier 500x200 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Die Buttons werden zunächst in einem `JPanel` mit einem `GridLayout` mit zwei Zeilen und einer Spalte gruppiert. Dann wird das Layout des `ContentPane` des Frames auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und die Gruppe der Buttons sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der `ArrayFrame` geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

**Weiter unten finden Sie eine Implementierung der Klassen `ArrayFrame` und `ArrayFrameMain` für die gesamte Aufgabe.**

- b) Erweitern Sie Ihre Klasse `ArrayFrame` um eine Ereignisbehandlung für den Button zum Revertieren eines Arrays. Wird der Button zum Revertieren eines Arrays gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach dem zu revertierenden `int`-Array gefragt werden und anschließend dieses `int`-Array revertiert werden, d.h. die Reihenfolge der Einträge umgedreht werden. Nach dem Revertieren soll der Benutzer im Ausgabebereich darüber informiert werden, was seine Eingabe war und was die Ausgabe (d.h. das revertierte Array) ist.

*Hinweis:* Im ZIP-Archiv finden Sie eine Klasse `Konverter.java`, mit der Sie in der Methode `public static int[] konvertiereZuIntArray(String string)` einen Komma-separierten String (ohne Leerzeichen) in ein `int`-Array konvertieren können (d.h. die Eingabe `1,2,3` wird konvertiert in ein Array `[1,2,3]`). Umgekehrt konvertiert die Methode `public static String konvertiereZuString(int[] intArray)` ein `int`-Array in einen Komma-separierten String.

#### **Lösung: Algorithmus für das Revertieren eines Arrays:**

Gehe das Array `arr` bis zur Mitte, d.h. zur Position `(arr.length/2)-1`, elementweise nacheinander durch, und tue für jedes Element:

- Speichere das aktuelle Element mit dem Index `i` in einer Platzhaltervariable.
- Speichere das Array-Element mit dem Index `(arr.length - 1) - i` im Array an der aktuellen Position `i`.
- Speichere das zwischengespeicherte Array-Element im Array an der Position `(arr.length - 1) - i`.

Es werden also alle Elemente der ersten Hälfte des Arrays spiegelbildlich mit den Elementen der zweiten Hälfte des Arrays vertauscht.

**Weiter unten finden Sie eine Implementierung der Klassen `ArrayFrame` und `ArrayFrameMain` für die gesamte Aufgabe.**

- c) Erweitern Sie Ihre Klasse `ArrayFrame` um eine Ereignisbehandlung für den Button zum lexikografischen Vergleich zweier `int`-Arrays, die nach Knopfdruck einzulesen sind. Ein `int`-Array  $a = [a_1, \dots, a_m]$  ist genau dann kleiner als ein `int`-Array  $b = [b_1, \dots, b_n]$  falls gilt:
- Entweder  $m < n$  und für alle  $i = 1, \dots, m$  gilt  $a_i = b_i$ ,
  - oder es gibt ein  $k \in \mathbb{N}$  mit  $k \leq m$  und  $k \leq n$ , so dass  $a_k < b_k$  gilt und für alle  $i \in \mathbb{N}$  mit  $i < k$  gilt  $a_i = b_i$  (also z. B.  $[1, 2, 3] < [1, 3, 1]$ , hier ist  $k = 2$ ).

Beispielsweise ist das `int`-Array `[1, 2, 8, 7, 3]` kleiner als `[1, 2, 8, 7, 3, 1]`, kleiner als `[1, 2, 8, 7, 4]` sowie kleiner als `[1, 2, 9]`. Nach dem lexikografischen Vergleich soll der Benutzer im Ausgabebereich darüber informiert werden, was seine Eingabe war und ob das erste Array lexikografisch kleiner als das zweite Array ist.

### Lösung: Algorithmus für das Testen zweier Arrays auf lexikographische Ordnung:

Bestimme zuerst die minimale Länge  $n$  der beiden zu vergleichenden `int`-Arrays `a` und `b`. Gehe dann die Arrays elementweise parallel durch und führe für jede Position  $i < n$  folgende Tests durch:

- Ist das Element des `int`-Arrays `a` mit dem Index `i` kleiner als das Element des `int`-Arrays `b` an der gleichen Position, gebe `true` zurück.
- Ist andernfalls das Element des `int`-Arrays `a` mit dem Index `i` größer als das Element des `int`-Arrays `b` an der gleichen Position, gebe `false` zurück.

Sind beide Arrays auf der gesamten minimalen Länge verglichen worden, ohne dass einer der vorhergehenden Fälle eingetreten ist, müssen beide Arrays auf der vergleichbaren Länge gleich sein. Prüfe in diesem Fall, ob die Länge des `int`-Arrays `a` kürzer ist als die Länge des `int`-Arrays `b` und damit das `int`-Arrays `a` lexikografisch kleiner als das `int`-Arrays `b` ist.

Die Implementierung der Klassen `ArrayFrame` und `ArrayFrameMain` finden Sie auch im ZIP-Archiv.

```

1  /**
2   * Diese Klasse startet den ArrayFrame
3   *
4   */
5  public class ArrayFrameMain {
6
7      /**
8       * Dieses Programmstueck startet das Programm.
9       *
10      * @param args
11      */
12     public static void main(String[] args) {
13         ArrayFrame arrayFrame = new ArrayFrame();
14         arrayFrame.setVisible(true);
15     }
16
17 }
```

```

1  import java.awt.Container;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JOptionPane;
9  import javax.swing.JPanel;
10 import javax.swing.JTextArea;
11
12 public class ArrayFrame extends JFrame implements ActionListener {
13     private JButton revertierenButton;
14     private JButton lexVergleichButton;
15
16     private JTextArea ausgabeBereich;
17
18     /**
19      * In diesem Programmstueck wird das Fenster erzeugt
20      */
21     public ArrayFrame() {
```

```

22      /* In der Kopfleiste des Fenster steht "ArrayFrame" */
23      this.setTitle("ArrayFrame");
24
25      /* Das Fenster ist 500 Pixel breit und 200 Pixel hoch. */
26      this.setSize(500, 200);
27
28      /* Hier werden alle Buttons erzeugt. */
29      this.revertierenButton = new JButton("Array revertieren");
30      this.lexVergleichButton = new JButton(
31          "Zwei Arrays lexikographisch vergleichen");
32
33      /* Hier wird der Ausgabe-Bereich erzeugt. */
34      this.ausgabeBereich = new JTextArea(10, 100);
35
36      /* Hier werden alle Buttons zusammengruppiert. */
37      JPanel buttonPanel = new JPanel();
38      buttonPanel.setLayout(new GridLayout(2, 1));
39      buttonPanel.add(this.revertierenButton);
40      buttonPanel.add(this.lexVergleichButton);
41
42      /*
43       * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
44       * Interaktionselemente (wie eine TextArea oder ein Button) platziert
45       * werden koennen.
46       */
47      Container contentPane = this.getContentPane();
48      contentPane.setLayout(new GridLayout(2, 1));
49      /* Hier wird die Gruppe von Buttons platziert. */
50      contentPane.add(buttonPanel);
51      /* Hier wird der Ausgabebereich platziert. */
52      contentPane.add(this.ausgabeBereich);
53
54      /*
55       * Hier wird der ArrayFrame als Listener fuer Knopfdruck-Ereignisse bei
56       * jedem der Buttons registriert.
57       */
58      this.revertierenButton.addActionListener(this);
59      this.lexVergleichButton.addActionListener(this);
60
61      /*
62       * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
63       * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
64       */
65      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66  }
67
68  @Override
69  public void actionPerformed(ActionEvent e) {
70      // je nach Button entsprechende Methode ausfuehren
71      Object source = e.getSource();
72      if (source == this.revertierenButton) {
73          this.revertierung();
74      }
75      else if (source == this.lexVergleichButton) {
76          this.lexikografischeOrdnung();
77      }
78  }
79
80  /**
81    * implementiert die Funktionalitaet des Revertierungsbuttons
82    */
83  private void revertierung() {
84      String einlesenArray = JOptionPane.showInputDialog("Array: ");
85      int[] array = Konverter.konvertiereZuIntArray(einlesenArray);
86

```

```

87         for (int i = 0; i < array.length / 2; i++) {
88             int temp = array[i];
89             array[i] = array[(array.length - 1) - i];
90             array[(array.length - 1) - i] = temp;
91         }
92
93         String eingabe = "Eingabe: " + einlesenArray + "\n\n";
94         String ergebnis = eingabe + "Ausgabe: "
95             + Konverter.konvertiereZuString(array);
96         this.ausgabeBereich.setText(ergebnis);
97     }
98
99     /**
100      * implementiert die Funktionalitaet des Lexikografische-Ordnung-Button
101      */
102     private void lexikografischeOrdnung() {
103         String einlesenArrayA = JOptionPane
104             .showInputDialog("Erstes Array: ");
105         int[] arrayA = Konverter
106             .konvertiereZuIntArray(einlesenArrayA);
107
108         String einlesenArrayB = JOptionPane
109             .showInputDialog("Zweites Array: ");
110         int[] arrayB = Konverter
111             .konvertiereZuIntArray(einlesenArrayB);
112
113         String eingabe = "Erstes Array: " + einlesenArrayA + "\n";
114         eingabe = eingabe + "Zweites Array: " + einlesenArrayB
115             + "\n\n";
116
117         boolean istLexKleiner = istLexikographischKleiner(arrayA,
118             arrayB);
119
120         String ergebnis = eingabe;
121         if (istLexKleiner) {
122             ergebnis = eingabe
123                 + "Das erste Array ist lexikografisch kleiner als das zweite.";
124         }
125         else {
126             ergebnis = eingabe
127                 + "Das zweite Array ist lexikografisch kleiner als das erste.";
128         }
129         this.ausgabeBereich.setText(ergebnis);
130     }
131 }
132
133 /**
134  * testet, ob das erste Array lexikografisch kleiner als das zweite ist.
135  *
136  * @param arrayA
137  * @param arrayB
138  * @return
139  */
140 private boolean istLexikographischKleiner(int[] arrayA,
141     int[] arrayB) {
142     int minLength = 0;
143     if (arrayA.length < arrayB.length) {
144         minLength = arrayA.length;
145     }
146     else {
147         minLength = arrayB.length;
148     }
149     // alternativ: minLength = (int)Math.min(arr1.length, arr2.length);
150     for (int i = 0; i < minLength; i++) {
151         if (arrayA[i] < arrayB[i]) { // arr1 ist kleiner als arr2

```



```

152         return true;
153     }
154     else if (arrayA[i] > arrayB[i]) { // arr1 ist groesser als arr2
155         return false;
156     }
157     // die Elemente sind gleich, pruefe naechstes
158 }
159 // beide Arrays sind, bei den vergleichbaren
160 // Elementen, gleich
161 // dann ist arr1 nur kleiner, wenn kuerzer
162 return arrayA.length < arrayB.length;
163 }
164 }

```

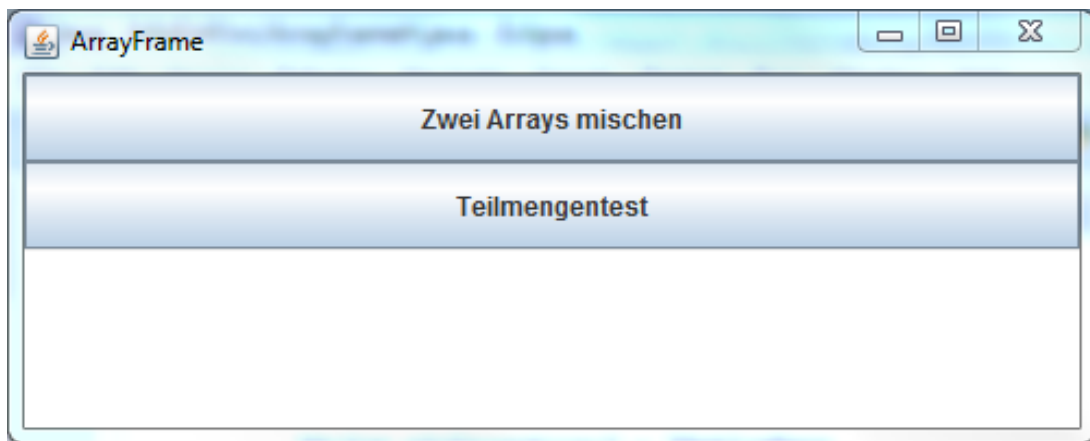
### Aufgabe 9-4

### Operationen auf Arrays

*Hausaufgabe*

In dieser Aufgabe sollen Sie ein Programm mit einer grafischen Benutzeroberfläche implementieren, welches die unten beschriebenen Operationen auf Arrays zur Verfügung stellt.

- a) Die grafische Benutzeroberfläche soll wie folgt aussehen:



Es soll zwei Buttons, einen zum Mischen zweier `int`-Arrays und einen zum Teilmengentest zweier `int`-Arrays, geben. Darunter soll ein Ausgabebereich platziert werden, in dem Rückmeldung über das Mischen und den Teilmengentest gegeben wird.

Schreiben Sie eine Klasse `ArrayFrameH`, die die Hauptklasse dieser grafischen Benutzeroberfläche sein soll und das Fenster erzeugt. Um Ihr Programm ausführen zu können, schreiben Sie eine weitere Klasse `ArrayFrameHMain`, die Sie wie gewohnt im gleichen Ordner wie Ihre Klasse `ArrayFrameH` abspeichern.

### Lösung:

Bei der Implementierung der grafischen Benutzeroberfläche geht man wie folgt vor:

Deklarieren Sie eine Klasse `ArrayFrameH`, die die Hauptklasse Ihrer grafischen Benutzeroberfläche wird und deshalb von der Klasse `JFrame` erbt. Ihre Klasse `ArrayFrameH` soll drei Attribute haben:

- je ein Attribut vom Klassentyp `JButton` für Buttons zum Mischen zweier Arrays und zum Teilmengentest zweier Arrays,
- ein Attribut vom Klassentyp `JTextArea`, das als Ausgabebereich für die spätere Rückmeldung über Mischen und Teilmengentest dient.

Schreiben Sie einen Konstruktor, der den Frame mit einem entsprechenden Titel und Größe (wir wählen hier 500x200 Pixel) initialisiert. In dem Konstruktor sollen weiterhin alle Attribute korrekt initialisiert werden. Die Buttons werden zunächst in einem `JPanel` mit einem `GridLayout` mit zwei Zeilen und einer Spalte gruppiert. Dann wird das Layout des `ContentPane` des Frames auf ein `GridLayout` mit zwei Zeilen und einer Spalte initialisiert und die Gruppe der Buttons sowie der Ausgabebereich darauf platziert. Fügen Sie abschließend noch ein, dass das Programm ordnungsgemäß beendet wird, falls der Frame geschlossen wird. Benutzen Sie dazu die Methode `setDefaultCloseOperation`.

**Weiter unten finden Sie eine Implementierung der Klassen `ArrayFrameH` und `ArrayFrameHMain` für die gesamte Aufgabe.**

- b) Erweitern Sie Ihre Klasse `ArrayFrameH` um eine Ereignisbehandlung für den Button zum Mischen zweier `int`-Arrays. Wird der Button zum Mischen zweier `int`-Arrays gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach den zu mischenden `int`-Arrays gefragt werden und anschließend diese `int`-Arrays gemischt werden, d.h. die Einträge der beiden Arrays Schritt für Schritt durchgegangen und nach dem Reißverschlussverfahren in einem neuen Array mit der Gesamtlänge der beiden Arrays zusammengefügt werden. Es kann davon ausgegangen werden, dass die beiden Arrays **gleich lang** sind. Das Mischen der Arrays `[0, 2, 4, 6, 8]` und `[1, 3, 5, 7, 9]` ergibt also das Ergebnis `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`. Nach dem Mischen soll der Benutzer im Ausgabebereich über seine Eingabe und Ausgabe informiert werden.

*Hinweis:* Verwenden Sie wieder die Klasse `Konverter.java` mit den beiden Methoden zum Konvertieren.

**Lösung: Algorithmus für das Mischen zweier Arrays:**

Lege ein Zielarray `reissverschluss` mit der Gesamtlänge der beiden einzugebenden Arrays an und gehe dieses Array elementweise nacheinander durch. Initialisiere dabei jedes Element des Arrays `reissverschluss` wie folgt:

- Hat das aktuelle Element eine Position mit geradem Index `i`, belege es mit dem Element mit dem Index `i/2` des ersten Eingabe-Arrays.
- Hat das aktuelle Element eine Position mit ungeradem Index `i`, belege es mit dem Element mit dem Index `(i-1)/2` des zweiten Eingabe-Arrays.

So werden alle Elemente der beiden einzugebenden Arrays nach dem Reißverschlussverfahren zusammenggefügt.

**Weiter unten finden Sie eine Implementierung der Klassen `ArrayFrameH` und `ArrayFrameHMain` für die gesamte Aufgabe.**

- c) Ein Array `a` ist eine Teilmenge eines Arrays `b`, wenn jedes Element von Array `a` auch im Array `b` vorkommt. Erweitern Sie Ihre Klasse `ArrayFrameH` um eine Ereignisbehandlung für den Button zum Teilmengentest von `int`-Arrays. Wird der Button gedrückt, soll der Benutzer zunächst mit Hilfe der Klasse `JOptionPane` nach dem `int`-Array `a` und anschließend nach dem `int`-Array `b` gefragt werden. Nach dem Teilmengentest sollen im Ausgabebereich nochmal die beiden Eingabe-Arrays `a` und `b` gezeigt werden sowie eine Information, ob das `int`-Array `a` Teilmenge des `int`-Arrays `b` ist.

**Lösung:**

Gehe das erste einzugebende Array `a` elementweise nacheinander durch und verfare für jede Position mit Index `i` wie folgt:

- Gehe das zweite einzugebende Array **b** elementweise und vollständig nacheinander durch und prüfe auf Vorkommen von **a[i]**. Merke, wenn das Element **a[i]** gefunden wurde.
- Wenn auf keiner Position von **b** das Element **a[i]** gefunden wurde, gebe **false** zurück.

Falls für alle Elemente von **a** ein Vorkommen in **b** gefunden wurde, gebe **true** zurück.

Die Implementierung der Klassen **ArrayFrameH** und **ArrayFrameHMain** finden Sie auch im ZIP-Archiv.

```

1  /**
2   * Diese Klasse startet den ArrayFrame
3   *
4   */
5  public class ArrayFrameHMain {
6
7      /**
8       * Dieses Programmstueck startet das Programm.
9       *
10      * @param args
11      */
12     public static void main(String[] args) {
13         ArrayFrameH arrayFrame = new ArrayFrameH();
14         arrayFrame.setVisible(true);
15     }
16
17 }
```

```

1  import java.awt.Button;
2  import java.awt.Container;
3  import java.awt.GridLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6
7  import javax.swing.JButton;
8  import javax.swing.JFrame;
9  import javax.swing.JOptionPane;
10 import javax.swing.JPanel;
11 import javax.swing.JTextArea;
12
13 public class ArrayFrameH extends JFrame implements ActionListener {
14     private JButton mischenButton;
15     private JButton teilmengenButton;
16
17     private JTextArea ausgabeBereich;
18
19     /**
20      * In diesem Programmstueck wird das Fenster erzeugt
21      */
22     public ArrayFrameH() {
23         /* In der Kopfleiste des Fenster steht "ArrayFrame" */
24         this.setTitle("ArrayFrame");
25
26         /* Das Fenster ist 500 Pixel breit und 200 Pixel hoch. */
27         this.setSize(500, 200);
28
29         /* Hier werden alle Buttons erzeugt. */
30         this.mischenButton = new JButton("Zwei Arrays mischen");
31         this.teilmengenButton = new JButton(
32             "Teilmengentest");
33
34         /* Hier wird der Ausgabe-Bereich erzeugt. */
35         this.ausgabeBereich = new JTextArea(10, 100);
```

```

36
37      /* Hier werden alle Buttons zusammengruppiert. */
38      JPanel buttonPanel = new JPanel();
39      buttonPanel.setLayout(new GridLayout(2, 1));
40      buttonPanel.add(this.mischenButton);
41      buttonPanel.add(this.teilmengenButton);
42
43      /*
44       * Der ContentPane ist der Ausschnitt des Fensters, auf dem Widgets d.h.
45       * Interaktionselemente (wie eine TextArea oder ein Button) platziert
46       * werden koennen.
47      */
48      Container contentPane = this.getContentPane();
49      contentPane.setLayout(new GridLayout(2, 1));
50      /* Hier wird die Gruppe von Buttons platziert. */
51      contentPane.add(buttonPanel);
52      /* Hier wird der Ausgabebereich platziert. */
53      contentPane.add(this.ausgabeBereich);
54
55      /*
56       * Hier wird der ArrayFrame als Listener fuer Knopfdruck-Ereignisse bei
57       * jedem der Buttons registriert.
58      */
59      this.mischenButton.addActionListener(this);
60      this.teilmengenButton.addActionListener(this);
61
62      /*
63       * Wird das Fenster geschlossen (d.h. auf X gedrueckt), wird mit Hilfe
64       * dieser Programmzeile auch unser Programm ordnungsgemaess beendet.
65      */
66      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
67  }
68
69  @Override
70  public void actionPerformed(ActionEvent e) {
71      // je nach Button entsprechende Methode ausfuehren
72      Object source = e.getSource();
73      if (source == this.mischenButton) {
74          this.mischen();
75      }
76      else if (source == this.teilmengenButton) {
77          this.teilmenge();
78      }
79  }
80
81      /**
82       * implementiert die Funktionalitaet des Mischenbuttons
83      */
84      private void mischen() {
85          String einlesenArray1 = JOptionPane
86              .showInputDialog("Erstes Array: ");
87          int[] array1 = Konverter
88              .konvertiereZuIntArray(einlesenArray1);
89
90          String einlesenArray2 = JOptionPane
91              .showInputDialog("Zweites Array: ");
92          int[] array2 = Konverter
93              .konvertiereZuIntArray(einlesenArray2);
94
95          String eingabe = "Erstes Array: " + einlesenArray1 + "\n";
96          eingabe = eingabe + "Zweites Array: " + einlesenArray2
97              + "\n\n";
98
99          int[] reissverschluss = new int[array1.length + array2.length];
100

```

```

101     for (int i = 0; i < reissverschluss.length; i++) {
102         if (i % 2 == 0) {
103             reissverschluss[i] = array1[i / 2];
104         }
105         else {
106             reissverschluss[i] = array2[(i - 1) / 2];
107         }
108     }
109
110     String ergebnis = eingabe + "Das gemischte Array ist: "
111         + Konverter.konvertiereZuString(reissverschluss);
112
113     this.ausgabeBereich.setText(ergebnis);
114 }
115
116 /**
117  * implementiert die Funktionaliaet des Teilmengentest-Buttons
118  */
119 private void teilmenge() {
120     String einlesenArrayA = JOptionPane
121         .showInputDialog("Erstes Array: ");
122     int[] arrayA = Konverter
123         .konvertiereZuIntArray(einlesenArrayA);
124
125     String einlesenArrayB = JOptionPane
126         .showInputDialog("Zweites Array: ");
127     int[] arrayB = Konverter
128         .konvertiereZuIntArray(einlesenArrayB);
129
130     String eingabe = "Erstes Array: " + einlesenArrayA + "\n";
131     eingabe = eingabe + "Zweites Array: " + einlesenArrayB
132         + "\n\n";
133
134     boolean istTeilmenge = this.istTeilmenge(arrayA, arrayB);
135
136     String ergebnis = eingabe;
137     if (istTeilmenge) {
138         ergebnis = eingabe
139             + "Das erste Array ist Teilmenge des zweiten Arrays.";
140     }
141     else {
142         ergebnis = eingabe
143             + "Das erste Array ist nicht Teilmenge des zweiten Arrays.";
144     }
145     this.ausgabeBereich.setText(ergebnis);
146 }
147
148
149 /**
150  * testet, ob zwei Arrays die selben Elemente enthalten
151  *
152  * @param arrayA
153  * @param arrayB
154  * @return
155  */
156 private boolean istTeilmenge(int[] arrayA, int[] arrayB) {
157     for (int i = 0; i < arrayA.length; i++) {
158         boolean gefunden = false;
159         for (int j = 0; j < arrayB.length; j++) {
160             if (arrayA[i] == arrayB[j]) {
161                 gefunden = true;
162             }
163         }
164         if (!gefunden) {
165             return false;

```

```

166         }
167     }
168     return true;
169 }
170 }

```

### Aufgabe 9-5

### Weihnachtsknochelei

*Hausaufgabe*

Das Christkind macht an Weihnachten eine Rundreise um Geschenke zu verteilen. Auf der Rundreise liegen 5 Lagerorte, die von 0 bis 4 durchnummeriert sind. An jedem Lagerort  $i \in \{0, \dots, 4\}$  liegen  $g_i$  Geschenke bereit, die das Christkind alle abholt, wenn es den Ort besucht. Auf dem Weg von einem Lagerort  $i$  zum nächsten Lagerort ( $i + 1$  falls  $i < 4$  bzw. 0 falls  $i = 4$ ) besucht das Christkind  $k_i$  Kinder, von denen jedes genau ein Geschenk erhalten soll. Prinzipiell kann das Christkind an jedem Lagerort seine Rundreise beginnen. Das Christkind muss jedoch darauf achten, dass es den Beginn so wählt, dass es auf seiner Rundreise immer genügend Geschenke zum Verteilen dabei hat.

Ihre Aufgabe ist es, dem Christkind bei der Planung zu helfen und ihm alle Lagerorte zu berechnen, von denen aus es seine Rundreise erfolgreich durchführen kann. Dazu soll ein Java-Programm erstellt werden, das zunächst über eine grafische Benutzeroberfläche ein Array einliest, das für alle Lagerorte  $i$  die Anzahl  $g_i$  der dort vorhandenen Geschenke angibt. Danach soll ein Array eingelesen werden, das für alle Lagerorte  $i$  die Anzahl  $k_i$  der auf dem Weg zum nächsten Lagerort zu beschenkenden Kinder angibt. Das Programm soll die Nummern aller Lagerorte ausgeben, von denen aus das Christkind mit einem anfangs leeren (aber genügend großem) Schlitten die Rundreise antreten kann, so dass jedes Kind genau ein Geschenk erhält. Testen Sie Ihr Programm mit einem Array  $[3, 6, 8, 7, 4]$  von bereit liegenden Geschenken an den Lagerorten  $i = 0, \dots, 4$  und einem Array  $[2, 3, 4, 10, 8]$ , das die Anzahl der Kinder zwischen zwei aufeinanderfolgenden Lagerorten angibt.

### Lösung:

Die Implementierung der Klasse Christkind finden Sie auch im ZIP-Archiv.

```

1 import javax.swing.JOptionPane;
2
3 public class Christkind {
4     public static void main(String[] args) {
5         String einlesenArray = JOptionPane
6             .showInputDialog("Anzahl an Geschenken pro Lagerort: ");
7         int[] geschenkeProLagerort = Konverter
8             .konvertiereZuIntArray(einlesenArray);
9
10        String einlesenArray2 = JOptionPane.showInputDialog(
11            "Anzahl an Kindern zwischen " + "zwei Lagerorten: ");
12        int[] geschenkeAbzugeben = Konverter
13            .konvertiereZuIntArray(einlesenArray2);
14
15        for (int startpunkt = 0; startpunkt < geschenkeProLagerort.length;
16            startpunkt++) {
17            boolean rundtourMoeglich = istRundtourMoeglich(startpunkt,
18                geschenkeProLagerort, geschenkeAbzugeben);
19
20            if (rundtourMoeglich) {
21                System.out.println(
22                    "Rundtour ab Startpunkt " + startpunkt + " moeglich.");
23            }
24            else {
25                System.out.println("Rundtour ab Startpunkt " + startpunkt
26                    + " NICHT moeglich.");

```

```

27         }
28     }
29 }
30
31 private static boolean istRundtourMoeglich(int startpunkt,
32     int[] geschenkeProLagerort, int[] geschenkeAbzugeben) {
33
34     int anzahlGeschenkeImSchlitten = 0;
35     int anzahlLagerorte = geschenkeProLagerort.length;
36
37     for (int i = startpunkt; i < (startpunkt + anzahlLagerorte); i++) {
38         anzahlGeschenkeImSchlitten = anzahlGeschenkeImSchlitten
39             + geschenkeProLagerort[i % anzahlLagerorte];
40         anzahlGeschenkeImSchlitten = anzahlGeschenkeImSchlitten
41             - geschenkeAbzugeben[i % anzahlLagerorte];
42         if (anzahlGeschenkeImSchlitten < 0) {
43             return false;
44         }
45     }
46     return true;
47 }
48 }

```

*Besprechung der Präsenzaufgaben in den Übungen am 22.12.2017 und 08.01.2018. Abgabe der Hausaufgaben bis Mittwoch, 17.01.2018, 14:00 Uhr über UniworX (siehe Folien der ersten Zentralübung). Erstellen Sie zu jeder Aufgabe Klassen, die die Namen tragen, die in der Aufgabe gefordert sind. Geben Sie nur die entsprechenden **.java**-Dateien ab. Wir benötigen **nicht** Ihre **.class**-Dateien.*