

Einführung in die Programmierung
WS 2018/19

Übungsblatt 8: Arrays, Strings

Aufgabe 8-1 *Arrays*

Diese Aufgabe beschäftigt sich mit Arrays und einigen ihrer Grundoperationen. Daher dürfen Sie nur Basisoperationen verwenden, d.h. Sie müssen auch auf die Objekt-Methoden von Arrays (z.B. `Arrays.sort()`) verzichten. `length` ist allerdings ein Array-Attribut und darf verwendet werden.

- (a) Implementieren Sie eine Methode `int arrayGet(int[] array, int i)`, die für ein `int`-Array den Wert an der `i`-ten Position im Array zurück gibt. Achten Sie darauf, dass auch Positionen sinnvoll behandelt werden, die nicht im Array vorkommen.
- (b) Implementieren Sie eine Methode `int sum(int[] array)`, die alle Einträge des Arrays aufaddiert und diese Summe zurückgibt.
- (c) Implementieren Sie eine Methode `int mean(int[] array)`, die den Mittelwert aller Einträge eines Arrays zurückgibt.
- (d) Implementieren Sie eine Methode `void square(int[] array)`, die jeden Eintrag eines Arrays quadriert.
- (e) Implementieren Sie eine Methode `int max(int[] array)`, die den größten Eintrag eines Arrays zurückgibt.

Lösungsvorschlag:

```
1 public class ArrayMethods {
2     /** Gibt den Wert einer bestimmten Position von einem Array zurueck.
3      * @param array: das Array
4      * @param i: die Position im Array, deren Wert bestimmt wird
5      * @return der Wert an der Position i in einem Array
6      */
7     public static int arrayGet(int[] array, int i) {
8         if(i < 0 || i > array.length-1)
9             return Integer.MIN_VALUE;
10        else
11            return array[i];
12    }
13
14    /** Bestimmt die Summe aller Eintraege des Arrays.
15     * @param array, das Array, dessen Eintraege summiert werden.
16     * @return die Summe aller Eintraege des Arrays.
17     */
18    public static int sum(int[] array) {
19        int sum = 0;
20        for(int i = 0; i < array.length; i++) {
21            sum += array[i];
22        }
23        return sum;
24    }
25
26    /**Die Methode bestimmt den Mittelwert aller Eintraege eines int-Arrays
27     * @param array: das Array, von dem der Mittelwert berechnet wird.
28     * @return der Mittelwert aller Eintraege des Arrays.
29     */
30    public static int mean(int[] array) {
31        return sum(array)/array.length;
32    }
33
34    /**Die Methode quadriert jeden Eintrag eines int-Arrays
35     * @param array: das Array, dessen Eintraege quadriert werden.
36     */
37    public static void square(int[] array) {
38        for(int i = 0; i < array.length; i++) {
39            array[i] = array[i]*array[i];
40        }
41    }
42
43    /**Die Methode findet fuer ein Array den groessten Wert aller Eintraege
44     * @param array: das Array, dessen Eintraege nach dem Groessten
45     *                durchsucht werden.
46     * @return der groesste Eintrag des Arrays.
47     */
48    public static int max(int[] array) {
49        int max = Integer.MIN_VALUE;
50        for(int i = 0; i < array.length; i++) {
51            if(array[i] > max) {
52                max = array[i];
53            }
54        }
55        return max;
56    }
57 }
```

Aufgabe 8-2 *Arrays 2*

Hier gibt es noch mehr Aufgaben zu Arrays. Auch hier dürfen Sie nur Basisoperationen verwenden, d.h. Sie müssen auch auf die Objekt-Methoden von Arrays verzichten.

- (a) Implementieren Sie eine Methode `void swap(int[] array, int i, int j)`. Diese soll die Einträge `i` und `j` des Arrays `array` miteinander vertauschen.
- (b) Implementieren Sie den Sortieralgorithmus <https://de.wikipedia.org/wiki/Selectionsort> mit dem folgenden Methodenrumpf `void sort(int[] array)`. Selectionsort sortiert die Einträge eines Arrays der Größe nach aufsteigend. Legen Sie dazu ein neues leeres Array der gleichen Größe an. Durchlaufen Sie das Array von links nach rechts und tauschen Sie das Element an der `i`-ten Position mit dem `i`-t kleinsten Element.
- (c) Implementieren Sie eine Methode `int median(int[] array)`, die den Median eines Arrays bestimmt. Bei Arrays mit gerader Anzahl an Elementen entspricht hier der Median dem größeren der beiden Kandidaten. Beachten Sie, dass der Median nicht das Gleiche ist wie der Mittelwert.
- (d) Implementieren Sie eine Methode ~~`void`~~ `int[] resize(int[] array, int length)`. Diese Methode soll die Länge eines Arrays wie folgt ändern:
 - ist die neue Länge kürzer als die Bisherige, wird das Array an den letzten Positionen beschnitten.
 - ist die neue Länge länger als die Bisherige, werden die zusätzlichen Positionen am Ende des Arrays mit dem Wert 0 angefügt.

Korrektur: `resize` muss natürlich den Rückgabewert `int[]` haben. Überlegen Sie für sich selbst, warum!

- (e) Implementieren Sie eine Methode `String show(int[] array)`, die die Einträge des Array elementweise ausgibt.
Für `int[] bsp = {6,4,9}`; liefert `show(bsp)` den String `"[6-4-9]"` `"[6 4 9]"`.

Lösungsvorschlag:

```
1 public class ArrayMethods {
2
3     /** Die Methode vertauscht die Eintraege zweier Positionen eines Arrays
4      * miteinander.
5      * @param array: das Array, in dem Positionen eins mit Position
6      *              zwei vertauscht wird.
7      * @param i: Position eins im Array.
8      * @param j: Position zwei im Array.
9      */
10    public static void swap(int[] array, int i, int j) {
11        if(i == j)
12            return;
13        int temp = array[i];
14        array[i] = array[j];
15        array[j] = temp;
16    }
17
18    /** Sortiert die Eintraege eines Arrays der Groesse nach aufsteigend.
19     * @param array: das Array, das sortiert wird
20     */
21    public static void sort(int[] array) {
22        for(int i = 0; i < array.length; i++) {
23            int min = Integer.MAX_VALUE;
24            int minPos = 0;
25            for(int j = i; j < array.length; j++) {
26                if(min > array[j]) {
27                    min = array[j];
28                    minPos = j;
29                }
30            }
31            int temp = array[i];
32            array[i] = array[minPos];
33            array[minPos] = temp;
34        }
35    }
36
37    /** Diese Methode bestimmt den Wert des Medians eines Arrays.
38     * @param array: das Array, dessen Median bestimmt wird.
39     * @return der Median des Arrays array.
40     */
41    public static int median(int[] array){
42        sort(array);
43        return array[array.length/2];
44    }
45
46    /** Diese Methode wandelt ein Array beliebiger Laenge in ein Array
47     * anderer Laenge um.
48     * @param array: das Start-Array, dessen Laenge geaendert wird.
49     * @param length: die neue Laenge des Arrays
50     * @return Das laengenveraenderte Array
51     */
52    public static int[] resize(int[] array, int length) {
53        if(length == array.length)
54            return array;
55        int[] tempArray = new int[length];
56        if(length < array.length)
57            for(int i = 0; i < length; i++)
```

```

58         tempArray[i] = array[i];
59     else {
60         for(int i = 0; i < array.length; i++)
61             tempArray[i] = array[i];
62         for(int i = array.length; i < length; i++)
63             tempArray[i] = 0;
64     }
65     return tempArray;
66 }
67
68 /** Die Methode gibt die Eintraege eines Arrays aus.
69 * @param    array: das Array, das als String ausgegeben wird
70 * @return   ein String, der die Eintraege des Arrays in richtiger
71 *           Reihenfolge enthaelt.
72 */
73 public static String show(int[] array) {
74     String str = "[";
75     for(int i = 0; i < array.length; i++) {
76         str += " " + array[i];
77     }
78     str += " ]";
79     return str;
80 }
81 }

```

Aufgabe 8-3 *Caesar-Chiffre*

Zum Verschlüsseln von Daten gibt es viele Methoden. Eine sehr alte Variante ist die Rotation des Alphabets um eine bestimmte Anzahl von Stellen. Die bekannteste davon ist vermutlich die Caesar-Chiffre, bei der eine Rotation von 13 Positionen ausgeführt wird. Dies ermöglicht bei 26 Buchstaben, die gleiche Methode zum Ver- und Entschlüsseln zu verwenden. Hier werden wir beliebige Rotationen berücksichtigen. Im Folgenden sehen Sie eine Rotation um 5 Stellen.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e

Implementieren Sie ein Programm, das das Argument `args` des Benutzers verwendet und alle 26 möglichen Rotationen ausführt und ausgibt. Wenn der Nutzer noch eine Zahl vorgibt, wird nur diese Rotation angezeigt. Das Grundgerüst können Sie herunterladen. Sie benötigen für diese Aufgabe sehr wahrscheinlich die `String`-Methoden `charAt()` und `length()`. Es genügt, wenn Ihr Tool nur mit Kleinbuchstaben arbeiten kann. Belassen Sie alle anderen Zeichen bei ihrer Klardarstellung. Entziffern Sie mit Ihrem Tool folgende Textzeile:

robjvsmrox qveomugexcmr. ne rkcd nso kepqklo qovyocd.

Lösungsvorschlag:

```
1 public class Passwort {
2     public static void main(String[] args){
3         String eingabe = "";
4         int index = 0;
5
6         if(args.length > 0)
7             eingabe = args[0];
8         else
9             return;
10
11        if(args.length > 1)
12            index = Integer.parseInt(args[1]);
13
14        if(index == 0)
15            for(int i = 0; i < 26; i++)
16                System.out.println(code(eingabe, i));
17        else
18            System.out.println(code(eingabe, index));
19    }
20
21    // A: 65, Z: 90
22    // a: 97, z: 122
23    public static String code(String eingabe, int index){
24        // TODO
25        String ausgabe = "";
26        for(int i = 0; i < eingabe.length(); i++){
27            char c = eingabe.charAt(i);
28            if(c > 96 && c < 123) { //falls c Kleinbuchstabe
29                c = (char)(c + index);
30                if(c > 122)
31                    c = (char)(c - 26);
32            }
33            ausgabe += c;
34        }
35        return ausgabe;
36    }
37 }
38 }
```