

Arrays: Übungen 1

Gegeben ist eine Klasse „**messreihe1**“ mit einigen Methoden. Bei der Erzeugung einer Instanz des Typs **Messreihe** wird ein Array mit zufällig generierten „Messwerten“ vom Typ **double** erzeugt. bearbeite die folgenden Aufgaben.¹⁾

- [BlueJ Projekt Arrays](#)

Aufgaben:

- **(1)** Erprobe die Methode `gibGewicht()`. Wie muss sie aufgerufen werden? Klappt der Aufruf immer? Was wird in der Abfrage in Zeile 36 geprüft? Was versucht man hier abzufangen? Erläutere diese Prüfabfrage im Detail.
- **(2)** Vervollständige die Methode `setzeAn5und9` zum Setzen eines neuen Wertes für die Elemente `gewicht[5]` und `gewicht[9]` diesem Array.
- **(3)** Schreibe eine Methode zum Setzen eines neuen Wertes für ein Element dieses Arrays mit wählbarem Index.
- **(4)** Ermittle das Durchschnittsgewicht der gesamten Messreihe. Notiere zuerst deine Idee und setze sie dann in Quelltext um. Warum sollte dies eine eigenständige Funktion(Methode) werden?
- **(5)** Schreibe eine Methode, die je eine Integer Zahl als Start- (s) und Endindex (e) erhält und damit den Durchschnitt aller Werte mit Indizes (i) zwischen s und e ermittelt.

Lösungsvorschläge

Aufgabe 1: Das ist ein klassischer „Getter“: `gibGewicht(index)` gibt das Gewicht des am Index `index` stehenden Arrayelements zurück. Die Überprüfungen sollen sicherstellen, dass man nicht den zulässige Wertebereich für den Index verlässt, als „Fehlermeldung“ wird -8.888 zurückgegeben.

Hier ein paar Vorschläge zur Lösung der Aufgaben. Die Methoden können meist noch verbessert werden, die Lösungen machen keine Eingabeüberprüfung.

```
/**
 * Aufgabe 3
 *
 * @param index    Index des zu setzenden Elements
 * @param value    Wert, auf den das Element gesetzt werden soll
 */
public void insertAtIndex(int index, double value)
{
    gewicht[index] = value; // das geht natürlich auch besser...
}

/**
 * Aufgabe 4
 *
 * @return    Durchschnittsgewicht
 */
```

```
public double getAverage()
{
    double sum = 0;

    for (int i=0; i<anzahl;i++) {
        sum = sum + gewicht[i];
    }
    return sum/anzahl;
}

/**
 * Aufgabe 5
 *
 * @param start Startindex (inklusive dieses Elements)
 * @param end Endindex (inklusive dieses Elements)
 * @return Durchschnittsgewicht
 */
public double getAverageFromTo(int start, int end)
{
    // Hier gibt es zahlreiche Verbesserungsmöglichkeiten
    double sum = 0;

    for (int i=start; i<=end;i++) {
        sum = sum + gewicht[i];
    }
    return sum/(end-start+1);
}
```

Ohne BlueJ

App.java

```
/** Fachklasse: Messreihe (=eine Reihe von nummerierten Messdaten)
 * @author: thh
 * @author: fs
 * @version: 20200115
 */

public class Messreihe {
    // Objektvariablen deklarieren
    int anzahl = 45;
    double[] gewicht = new double[anzahl];

    /** Konstruktor fuer Objekte der Klasse Messreihe
     * Jede Messreihe enthaelt eine Reihe von positiven Messdaten
     (Gewichten);
     */
}
```

```
public Messreihe() {
    for (int i=0; i<anzahl; i++) {    // Alle Gewichte
        gewicht[i] = erzeugeZZahl(); // der Reihe nach festlegen
    }
}

/** das Element der Reihung mit dem Index i zurueckgeben
 * Der gewuenschte Index i muss eingegeben werden
 * Bei Eingabe eines nicht vorhandenen Index wird
 * -8.888 als Fehlersignal zurueckgemeldet */
public double gibGewicht(int i) {
    if (i<0 || i>anzahl) {            //<-- 2.
        return -8.888;                // als Fehlersignal!
    }
    else {
        return gewicht[i];
    }
}

/** setzt fuer zwei Elemente der Messreihe neue Werte fest.
 * Das Element mit dem Index 5 in Reihung gewicht[ ] wird auf
555.55 gesetzt
 * Das Element mit Index 9 auf den Wert 99.99 */
public void setzeAn5und9() {
    // deine Aufgabe                    //<-- 3.a) b)
}

/*# <-- 4. Aufgabe */

// ----- Hilfsfunktionen
/** dient zum Anzeigen der Reihung am Bildschirm;
 * kann durch GUI oder INSPECT ersetzt werden */
public void anzeigen() {
    System.out.println("\n Aktuelle Messreihe:");
    for (int i=0; i< anzahl; i++) {
        schreibe(i, gewicht[i]);
    }
}

//----- interne Hilfsfunktionen
/** interne Methode, um eine Zufallszahl im Bereich 200.0 - 799.999
 * mit 3 Nachkommastellen zu erzeugen;
 * Math.random() liefert eine Zahl von 0 (inkl.) bis 1 (exkl.) */
private double erzeugeZZahl() {
    double zufZahl = 200 + 600*Math.random();
    return Math.round((zufZahl*1000))/1000.0;
}

/** interne Hilfsfunktion zur Anzeige;
```

```
    *  setzt ein- bis zweistelligen Zahlen stellenrichtig ein. */
private void schreibe(int i, double wert) {
    String erg = "Index";
    if (i<10) {
        erg = "Index  " + i;    // Zwei Leerzeichen drin !!
    }
    else {
        erg = "Index " + i;    // hier nur eines !!
    }
    System.out.println(erg+" : "+wert);
}
}

/* App Klasse: Steuert den Programmablauf */
public class App {

    public static void main(String[] args)
    {
        Messreihe reihel = new Messreihe();
        reihel.anzeigen();
        // Erzeuge eine zweite Messreihe reihe2 und gebe sie
aus
        // Teste weitere Methoden/bearbeite die Aufgaben
unten/im Wiki
        double g=reihel.gibGewicht(20);
        System.out.println("Gewicht " + g);
    }
}

/** Aufgaben:
 *
 * 1. Erprobe die Methode gibGewicht(). Wie muss sie aufgerufen werden.
 *    Klappt der Aufruf immer?
 *    Was wird in der Abfrage Z.28 geprueft? Was versucht man hier
abzufangen?
 *    Erlaeutere diese Pruefabfrage im Detail.
 *
 * 2.a) Vervollstaendige diese Methode zum Setzen eines neuen Wertes
fuer
 *    die Elemente gewicht[5] und gewicht[9] dieser Reihung.
 * b) Schreibe eine Methode zum Setzen eines neuen Wertes fuer ein
 *    Element dieser Reihung mit waehlbarem Index.
 * c) Teste deine Methoden mit entsprechenden Anweisungen in main()
 *
 *
 * 3. Ermittle das Durchschnittsgewicht der gesamten Messreihe.
 *    Notiere zuerst deine Idee und setze sie in Quelltext um.
 *    Warum sollte dies eine eigenstaendige Funktion(Methode) werden?
```

```
*  
* 4. Schreibe eine Methode, die je eine Integer Zahl als Start- (s)  
* und Endindex (e) erhält  
* und damit den Durchschnitt aller Werte mit Indizes (i) zwischen s  
* und e ermittelt.  
*  
*/
```

1)

Diese Übungen stehen unter einer CC-BY-SA Lizenz, sie wurden erstellt in enger Anlehnung an das Material der ZPG BW/Heußer

From:
<https://wiki.qg-moessingen.de/> - QG Wiki

Permanent link:
<https://wiki.qg-moessingen.de/faecher:informatik:oberstufe:java:algorithmen:arrays:uebungen1:start?rev=1633455811>

Last update: **05.10.2021 19:43**

