Musterlösung und Erläuterungen S. 205|7

Wie im letzten pdf sollen hier zuästzliche Erläuterungen zu den einzelnen Teilaufgaben stehen. Bevor Teilaufgabe a) überhaupt begonnen werden kann, muss das Grundgerüst der Klasse gebaut werden mit dem gewünschten Konstruktor. Das Array ist dabei ein Klassenattribut (so muss das array nicht jeder Methode immer übergeben werden!) Die Prüfung, ob eine Zahl bereits im Array ist, wird in eine extra Methode ausgelagert:

```
class Feldarbeiten {
    private int[] array;

    public Feldarbeiten(int n) {
        array = new int[n];
        //Wir füllen alle Felder
        for(int i = 0; i < n; i++) {
            //Eine Zufallszahl zwischen 0 und 100 wird erzeugt
            int zufallszahl = (int) Math.random()*100;
            /*Die Wiederholung prüft in jedem Schritt, ob die Zufallszahl in unserem
            Array bereits enthalten ist, falls ja, wird neu gewürfelt. Gibt "enthält"
            das erste Mal "false" zurück, wird die Zahl gespeichert.*/
            while(enthält(zufallszahl, array)) {
                zufallszahl = (int) Math.round(Math.random()*100);
            }
            array[i] = zufallszahl;
        }
    }
    /*Diese Methode durchläuft das übergebene Array vollständtdig und gibt
    true zurück, falls die übergebene Zahl gefunden wird.
    Wir übergeben das array hier und greifen nicht auf das Attribut zurück,
    um die Methode auch in Teilaufgabe f) verwenden zu können.*/
    private boolean enthält(int zahl, int[] übergeben) {
        for(int i = 0; i < übergeben.length; i++) {
            if(übergeben[i] == zahl) {
                return true;
            }
        }
        return false;
    }
}
```

Teilaufgabe a)

```
public int minimum() {
    // Wir nehmen den ersten Eintrag als Minimum an.
    int minimum = array[0];
    // Das gesamte Feld wird durchlaufen
    for(int i = 1; i < array.length; i++) {
        //Falls eine kleinere Zahl gefunden wird, wird ein neues Minimum gespeichert.
        if(array[i] < minimum) {
            minimum = array[i];
        }
    }
    return minimum;
}
```

Teilaufgabe b)

```
public int summe() {
    int summe = 0;
    for(int i = 0; i < array.length; i++) {
```

```
        //Kurzschreibweise für summe = summe + array[i]
        summe += array[i];
    }
    return summe;
}
```

Teilaufgabe c)
Bei dieser Methode können wir abkürzen und direkt auf unsere Summenmethode zurückgreifen. Es muss nur darauf geachtet werden, dass das Ergebnis des Funktionsaufrufs *summe()* als **double** interpretiert wird, damit von java nicht gerundet wird.

```
public double durchschnitt() {
    return (double) summe() / array.length;
}
```

Teilaufgabe d)

```
public int anzahlKleinerAls(int k) {
    int anzahl = 0;
    for(int i = 0; i < array.length; i++) {
        //Wenn eine kleinere Zahl b eim Durchlauf gefunden wird, wird jedes Mal der Zähler
        //um eins hochgesetzt.
        if(array[i] < k) {
            //Kurzschreibweise für anzahl = anzahl + 1
            anzahl++;
        }
    }
    return anzahl;
}
```

Teilaufgabe e)

```
public String positionMinimum() {
    //Wir nutzen zuerst unsere bereits geschriebene Methode, um das Minimum zu bestimmen...
    int minimum = minimum();
    for(int i = 0; i < array.length; i++) {
        //dann suchen wir die richtige Position und geben die Position als String zurück.
        if(array[i] == minimum) {
            return "Minimum: " + minimum + " an Position: " + i;
        }
    }
    //Falls das Minimum aus irgendwelchen Gründen nicht gefunden wird,
    // wird einfach ein leerer String zurückgegeben.
    return "";
}
```

Teilaufgabe f)
Hier müssen wir wieder etwas nachdenken, es gibt verschiedene Möglichkeiten die geforderte Methode zu implementieren, z.B. könnte jedes Mal abgeprüft werden, ob das Element bereits im neuen Array verwendet wurde, oder es wird geprüft, ob die entsprechende Position schon benutzt wurde. Für den ersten Fall können wir unsere enthält()-Methode benutzen, im anderen Fall müssen wir uns die Positionen zusätzlich in einem eigenen Array merken. Eine mögliche Implementierung wäre:

```
public int[] gemischtesFeld() {
    int[] gemischt = new int[array.length];
    for(int i = 0; i < array.length; i++){
        //Wir verwenden hier Math.floor() . also immer abrunden, damit wir den bereich
        //des arrays nicht verlassen!
        int zufallszahl = (int) Math.floor(Math.random()*array.length);
```

```
        while(enthält(array[zufallszahl], gemischt)) {
            zufallszahl = (int) Math.floor(Math.random()*array.length);
        }
        gemischt[i] = array[zufallszahl];
    }
    return gemischt;
}
```

Teilaufgabe g)

```
public void ausgeben() {
    System.out.println("Feld: ");
    for(int i = 0; i < array.length; i++) {
        System.out.print(array[i] + " ");
    }
    System.out.println();
    System.out.println("Minimum: " + minimum());
    System.out.println("Summe: " + summe());
    System.out.println("Durchschnitt: " + durchschnitt());
    System.out.println("Anzahl kleiner als 5: " + anzahlKleinerAls(5));
    System.out.println(positionMinimum());
    int[] gemischt = gemischtesFeld();
    System.out.println("Gemischt: ");
    for(int i = 0; i < gemischt.length; i++) {
        System.out.print(gemischt[i] + " ");
    }
    System.out.println();
}
```