Array-Operationen

Das blueJ-Projekt enthält eine Klasse `zufallsreihe`, die ein Array von int-Zahlen speichert und mit Zufallswerten füllt. Beim Anlegen von zufallsreihe-Objekten musst du angeben, wie viele Elemente die Zufallsreihe enthalten soll. Eventuell muss man für einzelne Aufgaben den Parameter der Methode getZufallszahl(...) anpassen oder den Quellcode geschickt ändern.

- BlueJ Projekt Arrays

Aufgaben

1)

Verschaffe dir einen Überblick über die gegebenen Methoden.

- Was bewirkt die Zeile `this.anzahl = anzahl;` im Konstruktor der Klasse Zahlenreihe?
- Wie funktioniert die Methode `getZufallszahl(max)`.
  - Welche Werte liefert `getZufallszahl(100)` zurück?
  - Wie muss der Aufruf lauten, um Zufallszahlen zwischen 1 und 50 zu erhalten?
  - Wie kann man Vorgehen, um auch den Wert 0 in der Zufallsreihe zu erhalten?

2)

Schreibe jeweils eine Methode, die... Teste deine Methoden durch entsprechende Aufrufe in der main Methode.

1. die Summe aller Werte im Array bestimmt und zurückgibt.
2. die Anzahl der Nullen im Array bestimmt und zurückgibt.
3. den Index bestimmt, an dem sich die letzte 0 des Arrays befindet und diesen zurückgibt. Falls keine 0 enthalten ist, soll -1 zurückgegeben werden.
4. den Index bestimmt, an dem sich die erste 0 des Arrays befindet und diesen zurückgibt. Falls keine 0 enthalten ist, soll -1 zurückgegeben werden.
5. prüft, ob das Array mindestens eine 1 enthält.
6. prüft, ob das Array mindestens eine 2 und mindestens eine 5 enthält.
7. prüft, ob das Array einen Fixpunkt enthält. Ein Fixpunkt ist ein Element, das gleich seinem Index ist, d.h. es muss `daten[i] == i` gelten.
8. die Anzahl der Wiederholungen bestimmt und zurückgibt. Eine Wiederholung liegt vor, wenn zwei aufeinanderfolgende Elemente des Arrays den gleichen Wert haben.
9. die Anzahl der Dreier-Wiederholungen bestimmt und zurückgibt. Hier müssen drei aufeinanderfolgende Elemente gleich groß sein.
10. die Länge der längsten Serie des Arrays bestimmt und zurückgibt. Eine Serie sind mehrere direkt aufeinanderfolgende Elemente, die den gleichen Wert haben.
11. die zweitgrößte Zahl des Arrays bestimmt und zurückgibt.
12. Jedes Element des Arrays soll um 1 erhöht werden.
13. Jede 0 im Array soll durch eine 100 ersetzt werden.
14. Jedes Element des Arrays soll um eine Stelle nach vorne gerückt werden. Das erste Element soll

an die letzte Stelle gesetzt werden. Diese Operation nennt man "Rotation".

15. Das Array soll umgedreht werden.

Lösungsvorschläge Aufgaben 1-7

```
    /**
     * aufaufgabe01Summe() berechnet die Summe aller Arrayelemente
     *
     * @return      Summe aller Arrayelemente
     */
    public int aufgabe01Summe()
    {
        int summe = -1;
        for (int i=0; i<this.anzahl; i++) {
            summe = summe + daten[i];
        }
        return summe;
    }

    /**
     * aufgaufgabe02ZaehleNullen() Gibt die Zahl den Nullen im Array zurück
     *
     * @return      Zahl der Nullen
     */
    public int aufgabe02ZaehleNullen()
    {
        int numNull = 0;
        for (int i=0; i<this.anzahl; i++) {
            if ( daten[i] == 0 ) {
                numNull++;
            }
        }
        return numNull;
    }

    /**
     * aufgabe03FindeLetzteNull() Gibt den Index des Elements mit der
letzten Null zurück
     *
     * @return      Index des Elements mit der letzten Null
     */
    public int aufgabe03FindeLetzteNull()
    {
        int letzteNullIndex = -1;
        for(int i=0; i<this.anzahl; i++) {
            if ( daten[i] == 0 ){
                letzteNullIndex = i;
            }
        }
```

```
        return letzteNullIndex;
    }

    /**
     *  aufgabe04FindeErsteNull() Gibt den Index des Elements mit der ersten
Null zurück
     *
     * @return      Index des Elements mit der ersten Null
     */
    public int aufgabe04FindeErsteNull()
    {

        for(int i=0; i<this.anzahl; i++) {
            if ( daten[i] == 0 ){
                return i;
            }
        }
        return -1;
    }

    /**
     *  aufgabe05Enthaelt1() Wahr, wenn die Zahlenreihe mindestens eine 1
enthaelt
     *
     * @return      Wahr, wenn 1 vorhanden, sonst falsch
     */
    public boolean aufgabe05Enthaelt1()
    {
         for(int i=0; i<this.anzahl; i++) {
            if ( daten[i] == 1 ){
                return true;
            }
        }
        return false;
    }

    /**
     *  aufgabe06Enthaelt2Und5()a Wahr, wenn die Zahlenreihe mindestens eine
2
     *  und eine 5 enthaelt
     *
     * @return      Wahr, wenn 1 u. 5 vorhanden, sonst falsch
     */
    public boolean aufgabe06Enthaelt2Und5()
    {
        boolean enthaelt2 = false;
        boolean enthaelt5 = false;

         for(int i=0; i<this.anzahl; i++) {
            if ( daten[i] == 2 ) {
                enthaelt2 = true;
```

```
            }

            if ( daten[i] == 5 ) {
                enthaelt5 = true;
            }
        }

        return (enthaelt2 && enthaelt5);

    }

    /**
     *  aufgabe07EnthaeltFixpunkt() Wahr, wenn die Zahlenreihe mindestens
einen
     *  Fixpunkt enthält
     *
     * @return      Wahr, wenn Fixpunkt vorhanden, sonst falsch
     */
    public boolean aufgabe07EnthaeltFixpunkt()
    {
        for(int i=0; i<this.anzahl; i++) {
            if ( daten[i] == i ){
                return true;
            }
        }
        return false;

    }
```

Ohne Bluej

App.java

```
/**
 * Erzeugt eine Zufallsreihe und ermöglicht Abfragen darüber.
 *
 * @author Rainer Helfrich
 * @author Frank Schiebel
 * @version 1.0
 */
class Zufallsreihe
{
    private int[] daten;
    int anzahl;

    public Zufallsreihe(int anzahl)
    {
        this.anzahl = anzahl;
```

```
        daten = new int[anzahl];
        for (int i = 0; i < daten.length; i++)
        {
            // Für manche Aufgaben sollte man die 6 durch z.B. 1000
ersetzen
            daten[i] = getZufallszahl(6);
        }
    }

    public int aufgabe01Summe()
    {
        return 0;
    }

    public int aufgabe02ZaehleNullen()
    {
        return 0;
    }

    public int aufgabe03FindeLetzteNull()
    {
        return 0;
    }

    public int aufgabe04FindeErsteNull()
    {
        return 0;
    }

    public boolean aufgabe05Enthaelt1()
    {
        return false;
    }

    public boolean aufgabe06Enthaelt2Und5()
    {
        return false;
    }

    public boolean aufgabe07EnthaeltFixpunkt()
    {
        return false;
    }

    public int aufgabe08ZaehleWiederholungen()
    {
        return 0;
    }

    public int aufgabe09ZaehleDreierWiederholungen()
    {
```

```
        return 0;
    }

    public int aufgabe10LaengsteSerie()
    {
        return 0;
    }

    public int aufgabe11Zweitgroesste()
    {
        return 0;
    }

    public void aufgabe12Plus1()
    {

    }

    public void aufgabe13NullZuHundert()
    {

    }

    public void aufgabe14Rotation()
    {

    }

    public void aufgabe15Umdrehen()
    {

    }


     /** dient zum Anzeigen der Reihung am Bildschirm;
     * kann durch INSPECT ersetzt werden */
    public void anzeigen() {
        for (int i=0; i< anzahl; i++) {
             System.out.println( i + " :  " + daten[i]);
        }
    }

    /**
     * Gibt eine Zufallszahl zwischen 0 und grenze-1 zurück.
     */
    private int getZufallszahl(int grenze)
    {
        return (int)(grenze*Math.random()+1);
    }
```

```
}


/* App Klasse. Steuerklasse für unser Programm */
public class App {

    public static void main(String[] args) {
        Zufallsreihe reihe1 = new Zufallsreihe(100);
        reihe1.anzeigen();
    }

}
```