1 Arbeiten mit Arrays

Ziel: Standard Algorithmen für Arrays programmieren und kennenlernen

Erstellen Sie in Eclipse ein neues Package `oopArrayAufg` und implementieren Sie darin die Klasse `ArrayTools` in der einige Klassenmethoden zum Bearbeiten von Arrays programmiert werden.

1.1. Die Methode mit der Signatur
`public static int sucheSequenziell(int[] aiListe, int iVonInd, int iBisInd, int iSuchwert)`

- sucht in dem unsortierten Array `aiListe` im Indexbereich `iVonInd` bis `IBisInd` nach dem Wert `iSuchwert`
- Wird `iSuchwert` gefunden, so gibt die Methode den Index vom ersten Vorkommen von `iSuchwert` zurück, sonst (-1).
- Ist der Indexbereich eine leere Menge, so ist der Rückgabewert (-1).

```
13 public class ArrayTools
14 {
15     public static int sucheSequenziell(int[] aiListe, int iVonInd, int iBisInd, int iSuchwert)
16     {
17         int iInd = -1;
18 
19         if (iVonInd >= 0 && iVonInd <= iBisInd && iBisInd <= aiListe.length - 1)
20         {
21             iInd = iVonInd;
22             while (iInd <= iBisInd && aiListe[iInd] != iSuchwert)
23             {
24                 iInd++;
25             }
26         }
27 
28         if (iInd > iBisInd)
29             iInd = -1;
30 
31         return iInd;
32     }
```

1.2. Erstellen Sie eine Startklasse zum Testen der Methode aus 1.1. Das Array zum Testen initialisieren Sie direkt mit festen Werten.

1.3. Die Methode mit der Signatur
`public static int bestimmeMaxWert(int[] aiListe, int iVonInd, int iBisInd)`

- sucht in dem unsortierten Array `aiListe` im Indexbereich `iVonInd` bis `IBisInd` nach dem Maximalwert und gibt diesen als Rückgabewert zurück.
- Ist der Indexbereich eine leere Menge, so ist der Rückgabewert undefiniert.

```
62      public static int bestimmeMaxWert(int[] aiListe, int iVonInd, int iBisInd)
63      {
64          int iMax = 0;
65
66          if (iVonInd >= 0 && iVonInd <= iBisInd && iBisInd <= aiListe.length - 1)
67          {
68              iMax = aiListe[iVonInd];
69              for (int i = iVonInd + 1; i <= iBisInd; i++)
70              {
71                  if (aiListe[i] > iMax)
72                  {
73                      iMax = aiListe[i];
74                  }
75              }
76          }
77          return iMax;
78      }
```

1.4. Die Methode mit der Signatur
`public static int` `bestimmeMinWert(`**`int`**`[] aiListe, `**`int`**` iVonInd, `**`int`**` iBisInd)`

- sucht in dem unsortierten Array `aiListe` im Indexbereich `iVonInd` bis `IBisInd` nach dem Minimalwert und gibt diesen als Rückgabewert zurück.
- Ist der Indexbereich eine leere Menge, so ist der Rückgabewert undefiniert.

1.5. Die Methode mit der Signatur
`public static int` `sucheBinaer(`**`int`**`[] aiListe, `**`int`**` iVonInd, `**`int`**` iBisInd, `**`int`**` iSuchwert)`

- sucht in dem *sortierten* Array `aiListe` im Indexbereich `iVonInd` bis `IBisInd` nach dem Wert `iSuchwert`
- Da das Array sortiert ist, kann der binäre Suchalgorithmus angewendet werden (siehe Lsg Struktogramme4_1Bis4_17 Aufg. 4.10)
- Wird `iSuchwert` gefunden, so gibt die Methode den Index vom ersten Vorkommen von `iSuchwert` zurück, sonst (-1).
- Ist der Indexbereich eine leere Menge, so ist der Rückgabewert (-1).

```java
public static int sucheBinaer(int[] aiListe, int iVonInd, int iBisInd, int iSuchwert)
{
    int iInd = -1;
    int iMitte;

    if (iVonInd >= 0 && iBisInd <= aiListe.length - 1)
    {
        while (iVonInd <= iBisInd && iInd == -1)
        {
            iMitte = (iVonInd + iBisInd) / 2;
            if (iSuchwert < aiListe[iMitte]) // links weitersuchen
            {
                iBisInd = iMitte - 1;
            }
            else
                if (iSuchwert > aiListe[iMitte]) // rechts weitersuchen
                {
                    iVonInd = iMitte + 1;
                }
                else
                {
                    iInd = iMitte; // gefunden
                }
        }
    }
    return iInd;
}
```

1.6. Die Methode mit der Signatur
`public static void sortiereZahlen(int[] aiListe, int iVonInd, int iBisInd)`

- sortiert das Array `aiListe` im Indexbereich `iVonInd` bis `IBisInd` nach aufsteigenden Werten.
- Zum Sortieren kann der Bubble-Sort Algorithmus angewendet werden. (Siehe Arbeitsblatt a_BubbleSort.pdf.)

```
 98    public static void sortiereZahlen(int[] aiListe, int iVonInd, int iBisInd)
 99    {
100        int k, iHilf;
101        boolean bMerker;
102
103        do
104        {
105            bMerker = false;
106            for (k = iVonInd; k <= iBisInd - 1; k++)
107            {
108                if (aiListe[k] > aiListe[k + 1])
109                {
110                    iHilf = aiListe[k];
111                    aiListe[k] = aiListe[k + 1];
112                    aiListe[k + 1] = iHilf;
113                    bMerker = true;
114                }
115            }
116            iBisInd--;
117
118        } while (bMerker);
119    }
```

2 Simulation Lotto-Tipp

Es soll eine Startklasse mit Namen TippzettelStart erstellt werden. In dieser Anwendung soll die Eingabe eines Lottotipps (6 aus 49) mit anschließender Lottoziehung und Auswertung simuliert werden. Der Dialog in der Main-Methode soll in etwa wie folgt aussehen:

```
TippzettelStart [Java Application] C:\Program Files\Java\jdk
Bitte die 1. Zahl eingeben:
12
Tippzettel: [12, 0, 0, 0, 0, 0]
Bitte die 2. Zahl eingeben:
3
Tippzettel: [3, 12, 0, 0, 0, 0]
Bitte die 3. Zahl eingeben:
27
Tippzettel: [3, 12, 27, 0, 0, 0]
Bitte die 4. Zahl eingeben:
45
Tippzettel: [3, 12, 27, 45, 0, 0]
Bitte die 5. Zahl eingeben:
33
Tippzettel: [3, 12, 27, 33, 45, 0]
Bitte die 6. Zahl eingeben:
7
Tippzettel: [3, 7, 12, 27, 33, 45]

Ziehung   : [1, 2, 12, 14, 23, 28]
1 Treffer
Nochmal spielen? (j/n)[j]:
```

2.1. Wie man erkennt, enthält die Main-Methode eine Schleife, die es ermöglicht, dass der Vorgang Tippzahlen eingeben und anschließende Ziehung und Auswertung auf Benutzerwunsch wiederholt werden kann.

Im ersten Schritt soll eine Klassenmethode
`private static int``[] leseLottoZahlen()`
die Benutzereingabe des Lotto-Tipps ermöglichen.

- Das int-Array für den Lotto-Tipp kann in der Methode erzeugt werden und wird am Ende mit return an Main zurückgegeben.
- Es muss sichergestellt sein, dass die Zahlen im richtigen Zahlenbereich sind und keine Zahl doppelt vorkommt. (Dabei bietet es sich an, die Methode ArrayTools.*sucheSequenziell*() zu verwenden.)
- Damit der Benutzer gut überblicken kann, was er bereits für Zahlen getippt hat, werden die Zahlen nach jeder Eingabe sortiert (`ArrayTools.`*`sortiereZahlen()`*) und am Bildschirm angezeigt.

```java
public class TippzettelStart
{
    public static final int TIPP_ANZAHL = 6;
    public static final int LOTTO_BEREICH = 49;

     * @param args
    public static void main(String[] args)
    {
        int[] aiTippZettel;
        int[] aiZiehung;
//      int[] aiStatistik = null; // Verweis auf Häufigkeitsverteilung

        //int iZahl;
        int iTreffer;
        String sWeiter;

        do
        {
            // Tipp einlesen und anzeigen
            aiTippZettel = leseLottoZahlen();

//          zeigeLottoZahlen(aiTippZettel);

            // Ziehung...
            aiZiehung    = zieheLottoZahlen();

            // Auswertung...
            iTreffer = ermittleTreffer(aiTippZettel, aiZiehung);

            // Statistik...
//          aiStatistik = akkumuliereHaeufigkeit(aiStatistik, 1, LOTTO_BEREICH, aiZiehung);

            System.out.println();
            System.out.println("Ziehung   : " + Arrays.toString(aiZiehung));
            System.out.println(iTreffer + " Treffer");

            sWeiter = Eingabe.getString("Nochmal spielen? (j/n)[j]:");
        } while (sWeiter == null || sWeiter.equals("j"));

        // Statistik ausgeben...
//      for (iZahl = 0; iZahl < aiStatistik.length; iZahl++)
//      {
//          // Der Wertebereich ist gegenüber dem Indexbereich um 1 verschoben
//          System.out.printf("Die %2d wurde %3d mal gezogen\n", (iZahl + 1), aiStatistik[iZahl]);
//      }
//      System.out.println(schreibeHHistogr(aiStatistik, 1, LOTTO_BEREICH));
    }
```

```
private static int[] leseLottoZahlen()
{
    int iTipp;

    int iZahl = 0;
    int[] aiTippZettel = new int[TIPP_ANZAHL];

    while (iZahl < aiTippZettel.length)
    {
        System.out.println("Bitte die " + (iZahl + 1) + ". Zahl eingeben:");
        iTipp = Eingabe.getInt();
        if (iTipp >= 1 && iTipp <= LOTTO_BEREICH)
        {
            if (ArrayTools.sucheSequenziell(aiTippZettel, 0, iZahl - 1, iTipp) == -1)
            {
                aiTippZettel[iZahl] = iTipp;
                iZahl++;
            }
        }
        Arrays.sort(aiTippZettel, 0, iZahl);
        //ArrayTools.sortiereZahlen(aiTippZettel, 0, iZahl-1);
        System.out.println("Tippzettel: " + Arrays.toString(aiTippZettel));
    }
    return aiTippZettel;
}

private static void zeigeLottoZahlen(int[] aiTippzettel)
{
    int i;
    int j = 0;

    for (i = 1; i <= LOTTO_BEREICH; i++)
    {
        if (i != aiTippzettel[j])
        {
            System.out.printf("%3d", i);
        }
        else
        {
            System.out.printf("%s", "  X");
            if (j < aiTippzettel.length - 1)
                j++;
        }
        if (i % 7 == 0)
            System.out.println();
    }
}
```

2.2. Nachdem die Zahlen eingegeben wurden kann die Ziehung von Lottozahlen erfolgen; Klassenmethode **`private static int`**`[] zieheLottoZahlen()`

- Das int-Array für die Lotto-Ziehung kann in der Methode erzeugt werden und wird am Ende mit return an Main zurückgegeben.
- Die Zahlen werden zufällig ermittelt. Es muss sichergestellt sein, dass keine Zahl doppelt vorkommt.
- Vor der Rückgabe des Arrays an Main werden die Zahlen noch sortiert.

```
private static int[] zieheLottoZahlen()
{
    int iTipp;

    int iZahl = 0;
    int[] aiZiehung = new int[TIPP_ANZAHL];

    while (iZahl < aiZiehung.length)
    {
        iTipp = A0402Zufall.getZufallInt(1, LOTTO_BEREICH);
        if (ArrayTools.sucheSequenziell(aiZiehung, 0, iZahl - 1, iTipp) == -1)
        {
            aiZiehung[iZahl] = iTipp;
            iZahl++;
        }
    }
    //Arrays.sort(aiZiehung);
    ArrayTools.sortiereZahlen(aiZiehung, 0, aiZiehung.length-1);

    return aiZiehung;
}
```

2.3. Nach dem Tipp und der Ziehung, kann die Auswertung erfolgen. Dazu wird die Methode mit der Signatur
`private static int` `ermittleTreffer(`**`int`**`[] aiTippZettel, `**`int`**`[] aiZiehung)`
programmiert.

- Die Methode liefert die Anzahl der Treffer, also die Anzahl der übereinstimmenden Zahlen im Tipp und in der Ziehung.

```
private static int ermittleTreffer(int[] aiTippZettel, int[] aiZiehung)
{
    int iZahl;
    int iTreffer;
    iTreffer = 0;
    for (iZahl = 0; iZahl < aiTippZettel.length; iZahl++)
    {
        if (Arrays.binarySearch(aiZiehung, aiTippZettel[iZahl]) >= 0)
        //if (ArrayTools.sucheBinaer(aiZiehung, 0, aiZiehung.length - 1, aiTippZettel[iZahl]) != -1)
        {
            iTreffer++;
        }
    }
    return iTreffer;
}
```

2.4. Zusatzaufgabe: Nach der Eingabe des Tipps, können die Zahlen auch noch mit Hilfe einer angezeigt werden:

```
Tippzettel: [1, 9, 17, 25, 33, 41]
  X  2  3  4  5  6  7
  8  X 10 11 12 13 14
 15 16  X 18 19 20 21
 22 23 24  X 26 27 28
 29 30 31 32  X 34 35
 36 37 38 39 40  X 42
 43 44 45 46 47 48 49
```

```
private static void zeigeLottoZahlen(int[] aiTippzettel)
{
    int i;
    int j = 0;

    for (i = 1; i <= 49; i++)
    {
        if (i != aiTippzettel[j])
        {
            System.out.printf("%3d", i);
        }
        else
        {
            System.out.printf("%s", "  X");
            if (j < aiTippzettel.length - 1)
                j++;
        }
        if (i % 7 == 0)
            System.out.println();
    }
}
```

2.5. Zusatzaufgabe: Mit Hilfe eines zusätzlichen Arrays der Länge 49 kann noch eine Statistik über die Häufigkeitsverteilung der gezogenen Zahlen mitprotokolliert werden. Das Ergebnis wird nach Beendigung der Schleife in Main am Bildschirm angezeigt.

- Einfache Variante:

```
Nochmal spielen? (j/n)n
Die  1 wurde   4 mal gezogen
Die  2 wurde   1 mal gezogen
Die  3 wurde   0 mal gezogen
Die  4 wurde   2 mal gezogen
Die  5 wurde   1 mal gezogen
Die  6 wurde   4 mal gezogen
Die  7 wurde   3 mal gezogen
```

und so weiter …

- Komplexere Variante:

```
 *                 *
 *                 *  *     *  *
 *        *        *  *     *  *  *           *
 *  *     *  *  *  *  *     *  *  *     *  *  *
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

und so weiter …

```java
private static int[] akkumuliereHaeufigkeit(int[] aiStatistik, int iVonWert,
                                            int iBisWert, int[] aiZiehung)
{
    // Der Wertebereich für die Merkmale ist durch iVonWert und iBisWert gegeben
    // Im Fall von Lotto 6 aus 49 ist der Wertebereich 1 bis 49
    // Beim ersten Aufruf ist aiStatistik == null und es wird ein Array erzeugt,
    // das lang genug ist, um den Wertebereich abzudecken: beim Lotto wäre die Länge 49
    if (aiStatistik == null) aiStatistik = new int[iBisWert - iVonWert + 1];

    for (int i = 0; i < aiZiehung.length; i++)
    {
        // Beispiel: aiZiehung[0] == 14 => aiStatistik[14-1] wird um 1 erhöht
        aiStatistik[aiZiehung[i] - iVonWert]++;
    }
    return aiStatistik;
}
```

```java
privat Close ic String schreibeHHistogr(int[] aiStatistik,
                                        int iVonWert, int iBisWert)
{
    int iMax = aiStatistik[0];

    // Maximalwert bestimmen
    for (int i = 1; i < aiStatistik.length; i++)
    {
        if (aiStatistik[i] > iMax)
        {
            iMax = aiStatistik[i];
        }
    }

    StringBuilder sb = new StringBuilder(1000);
    for (int i = iMax; i > 0; i--)
    {
        for (int j = 0; j < aiStatistik.length; j++)
        {
            if (aiStatistik[j] >= i)
            {
                sb.append("  *");
            }
            else
            {
                sb.append("   ");
            }
        }
        sb.append("\n");
    }
    for (int j = 0; j < aiStatistik.length; j++)
    {
        sb.append(String.format("%3d", iVonWert + j));
    }
    sb.append("\n");
    return sb.toString();
}
```