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: