Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

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).

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.

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).

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.)

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.

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.

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.

2.4. Zusatzaufgabe: Nach der Eingabe des Tipps, können die Zahlen auch noch mit Hilfe einer Klassenmethode in folgender Form 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
```

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:

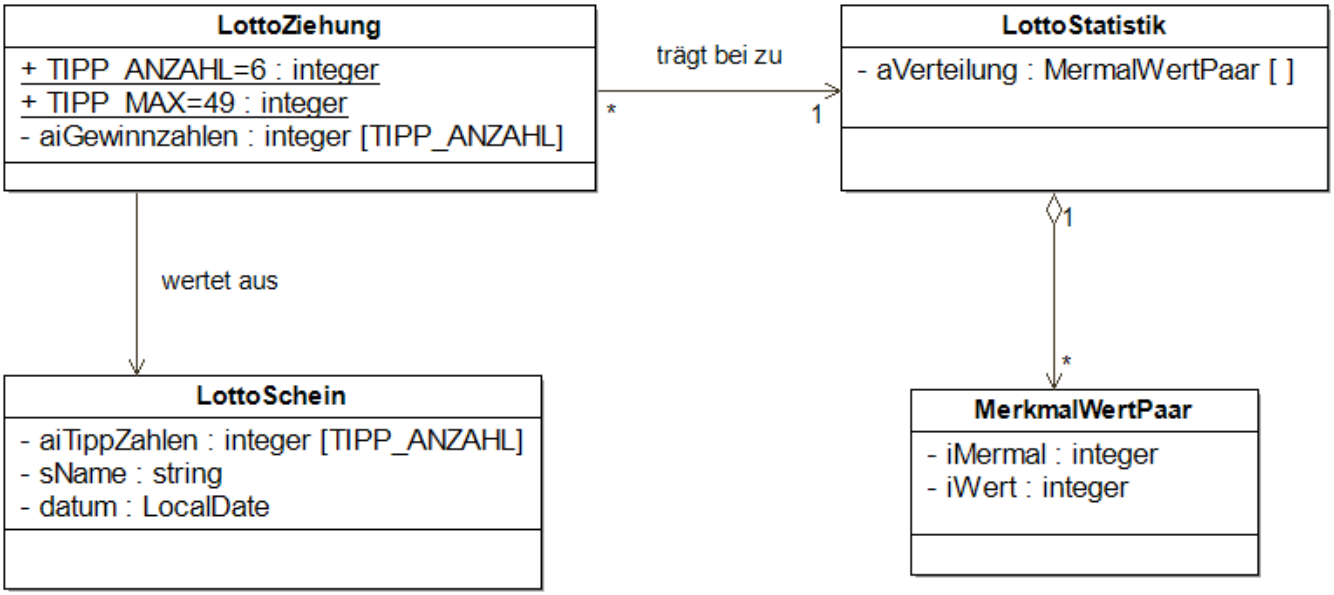
```
Nochmal spielen? (j/n)n
Die  1 wurde   4 mal gezogen
Die  2 wurde   1 mal gezogen
Die  3 wurde   0 mal gezogen
Die  4 wurde   2 mal gezogen
Die  5 wurde   1 mal gezogen
Die  6 wurde   4 mal gezogen
Die  7 wurde   3 mal gezogen
```
und so weiter …

- Komplexere Variante:

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

2.6. Simulation Lotto-Tipp „objektorientiert“

Die Lotto Simulation soll jetzt objektorientiert implementiert werden. Das folgende Klassendiagramm zeigt die erforderlichen Fachklassen.

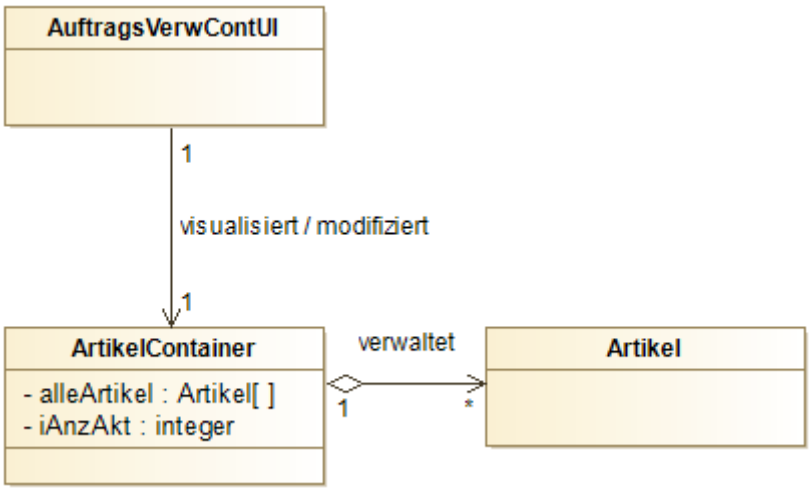


- Erstellen Sie in Eclipse ein neues Package für das Programm
- Holen Sie von Moodle die Zip Datei `DatenLottoSimulation` und entpacken diese.
- Die Startklasse LottoStart kann aus `DatenLottoSimulation` kopiert und in das Package der Anwendung eingebunden werden. (Einfach die Datei LottoStart.java in das Package in Eclipse ziehen.)
- Die Beschreibung der notwendigen Objektmethoden findet sich ebenfalls in `DatenLottoSimulation`. Dort in LottoDok die Datei index.html doppelklicken. In der Dokumentation sind die Objektmethoden der einzelnen Klassen beschrieben.
- Implementieren Sie das Klassendiagramm, indem Sie die angegebenen Attribute der Klassen anlegen und die in der Dokumentation beschriebenen Methoden in den Klassen programmieren.

3 Containerklassen und 3-Schichten-Architektur

Ziel: Entsprechend der 3-Schicht-Architektur soll eine Containerklasse zum Verwalten von Artikel Objekten erstellt werden.

3.1. Gegeben ist die bereits früher erstelle Fachkonzeptklasse Artikel und eine einfache UI-/Startklasse. Die Beschreibung der Methoden liegt in Form JavaDoc vor.
(Siehe `Moodle: DatenArtikelContainer`)



Kopieren Sie sich die Klassen `Artikel` und `AuftragsVerwContUI` in eine neu erstelltes Package. Implementieren Sie die Klasse ArtikelContainer und testen Sie die Methoden. Gehen Sie schrittweise vor indem Sie Teile von `AuftragsVerwContUI` auskommentieren.

4 Zwei-dimensionale-Arrays

Ziel: Der Umgang mit zwei-dimensionalen Arrays soll am Beispiel eines Memory-Spiels geübt werden.

4.1. Erstellen Sie ein neues Package `memorySpiel`. Laden Sie von Moodle die Datei `DatenMemorySpiel` herunter und entpacken Sie die Dateien in den Ordner des Packages. Nach einem „Refresh“ sollten die Klassen sichtbar sein.

Ergänzt werden muss nur in der Klasse `Memory` der Inhalt des Konstruktors, entsprechend der dort angegebenen Kommentare.
Das Array `aMaske` hat folgenden Inhalt: Das Array `aVorlage` z.B. folgenden:
(jeweils ohne Ränder):

```
 -1-2-3-4-5-6-7-8-
1 #|#|#|#|#|#|#|#|
2 #|#|#|#|#|#|#|#|
3 #|#|#|#|#|#|#|#|
4 #|#|#|#|#|#|#|#|
5 #|#|#|#|#|#|#|#|
6 #|#|#|#|#|#|#|#|
7 #|#|#|#|#|#|#|#|
8 #|#|#|#|#|#|#|#|
-----------------
```

```
 -1-2-3-4-5-6-7-8-
1 2|B|9|8|2|1|)|-|
2 .|=|>|;|C|&|,|@|
3 5|?|/|>|+|'|;|0|
4 6|,|'|4|@|A|8|(|
5 %|3|*|)|$|0|5|(|
6 7|%|6|&|.|<|?|B|
7 C|A|/|9|:|1|=|:|
8 4|7|*|-|<|3|$|+|
-----------------
```