Übungsaufgaben Extra

Array

Einfaches Array (Eindimensional)

Aufgabe 47

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayFunktionsWeise.java.** Legen Sie ein Array mit der Anweisung **int wurf[] = new int[6];** an und erzeugen sie nachfolgende Bildschirmausgabe:

wurf[0] = 1
wurf[1] = 7
wurf[2] = 9
wurf[3] = 4
wurf[4] = 2
wurf[5] = 8

Hinweis: Mit Arrays kann man viele Variablenwerte verwalten. Häufig kommen auch Schleifen zum Einsatz.

Aufgabe 48

Erstellen Sie ein ausführbares Programm mit dem Namen **ArraySchnellInitialisiert.java.** Das Programm erzeugt unten stehende Bildschirmausgabe. Verwenden Sie nachfolgende Quelltextzeile um das Array wurf schnell zu initialisieren **int wurf[]={1,7,9,4,2,8};**

wurf[0] = 1
wurf[1] = 7
wurf[2] = 9
wurf[3] = 4
wurf[4] = 2
wurf[5] = 8

Hinweis: Beachten Sie die geschweiften Klammern: int wurf[]={1,7,9,4,2,8};

Aufgabe 49

Erstellen Sie ein ausführbares Programm mit dem Namen **ArraySumme.java.** Das Programm gibt zunächst die Zahlen des Arrays durch das Zeichen ‚+' getrennt auf dem Bildschirm aus und hängt die berechnete Summe mit dem Zeichen ‚=' dran.

1 + 7 + 9 + 4 + 2 + 8 = 31

Hinweis: Beachten Sie, dass hinter der letzten Zahl kein + Zeichen mehr steht. Mit wurf.length kann die Array Größe erkannt werden, mit -1 verringert werden.

Aufgabe 50

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayMinMax.java**. Ihr Programm ermittelt die kleinste und größte Zahl in einem Array. Das Programm erzeugt unten stehende Bildschirmausgabe.

12,14,-9,7,34,843,23,107,29,40,576,
Minimum: -9 Maximum: 843

Aufgabe 51

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayAnWelcherStelleKommtZahlVor**.java. Das Programm enthält die Quelltextzeile

```
int suche=23, anzahl=0;
```

Ihr Programm sucht und zählt die Zahl 23 in Ihrem Array und gibt die Stellen (nicht den Index) und die Anzahl des Vorkommens auf dem Bildschirm aus:

Die gesuchte Zahl 23 kommt an den Stellen: 2, 7, 9 vor.
Die gesuchte Zahl 23 kommt insgesamt 3 mal vor.

Hinweis: Mit .length kann die Größe des Arrays festgestellt werden. Achten Sie darauf, dass der Index immer 1 darunter liegen muss. Andernfalls wird eine java.lang.ArrayIndexOutOfBoundsException ausgelöst.

Aufgabe 52

Erstellen Sie ein ausführbares Programm mit dem Namen **LottoZahlenZaehlen.java**. Das Programm gibt zunächst 10000 Lottozahlen durch Komma getrennt auf dem Bildschirm aus. Danach wird gezählt, wie oft jede Zahl im Lottoarray vorkommt und auf dem Bildschirm ausgegeben.
Aus Platzgründen wurden nicht alle 10000 Lottozahlen dargestellt sondern durch … angedeutet. Auch die Anzahlen zwischen den Zahlen 4 und 47 wurden aus Platzgründen mit … gekürzt dargestellt.

7, 27, 35, 37, 11, 31, 6, 40, 11, 34, 10, 11, ...
1 = 213
2 = 213
3 = 196
.....
48 = 193
49 = 197

Hinweis: Die Anzahl des Vorkommens einer Lottozahl wird in einem eigenen Array gezählt. Der Index 0 wird nicht genutzt. Mit der Quelltextzeile: `anzahl[feld[i]]++;` wird gezählt. In feld[i] befindet sich eine Lottozahl z.B. 5. Somit wird im Array anzahl[5] der Wert hochgezählt (anzahl[5]++).

Aufgabe 53

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayUmdrehen.java.** Das Programm gibt zunächst das Original-Array aus (vorher) und speichert dann das Array umgedreht ab und gibt es erneut (nachher) auf dem Bildschirm aus:

vorher: 1, 16, 23, 99, 153, 683, 987, 993, 998,
nachher: 998, 993, 987, 683, 153, 99, 23, 16, 1,

Hinweis: Beachten Sie, dass die Schleife lediglich bis zur Mitte zählt, weil die erste Zahl mit der letzten Zahl, die zweite mit der vorletzten Zahl, .. getauscht werden. Die entsprechenden Quelltextzeilen wurden fett hervorgehoben.

Aufgabe 54

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayFibonacci.java.** Das Programm speichert 30 Zahlen der Fibonacci-Reihe in einem Array namens fibonacci ab und gibt sie danach durch Komma getrennt auf dem Bildschirm aus:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377,
610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657,
46368, 75025, 121393, 196418, 317811, 514229

Hinweis: In der Fibonacci-Reihe sind die ersten beiden Zahlen gegeben: 0 und 1. Diese zwei Zahlen werden addiert und ergeben die nächste Zahl der Reihe = 1. Nun werden wiederum die beiden letzten Zahlen (1 und 1) addiert und ergeben die nächste Zahl der Reihe = 2. Der Algorithmus geht nun so weiter: Die Fibonacci-Reihe lautet: 0 1 1 2 3 5 8 13 21 34 55 89 ….
Da die ersten beiden Zahlen der fibonacci Zahlen feststehen und an der dritten Stelle addiert werden muss der Zähler Wert für den index mit i = 2 initialisiert werden.
Also fibonacci[2] = fibonacci[0] + fibonacci[1] daher fängt i mit 2 an

Aufgabe 55

Erstellen Sie ein ausführbares Programm mit dem Namen **EingabeVieleZahlenAbbruchMitNull.**java. Das Programm verlangt die Eingabe von Zahlen. Bei Eingabe der Zahl 0 endet die Eingabe. Ihr Programm berechnet die Summe der eingegebenen Zahlen (ohne 0 !!) und erzeugt nachfolgenden Bildschirmausgabe. Bei den fett und kursiv dargestellten Zahlen handelt es sich um eine Benutzereingabe.

Eingabe 1. Zahl: 15
Eingabe 2. Zahl: 13
Eingabe 3. Zahl: 7
Eingabe 4. Zahl: 0
15+13+7 = 35

Hinweis: Das Array wurde im Beispiel auf 100 Werte begrenzt.

Aufgabe 56

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayKopieren.java**. Das Programm soll ein Array kopieren. Ändern Sie die letzte Zahl Ihres kopierten Arrays, um zu prüfen, dass es sich tatsächlich um eine Kopie handelt. Beachten Sie den unten stehenden Hinweis. Das Programm erzeugt folgende Bildschirmausgabe:

Wurf 1
wurf1[0] = 1
wurf1[1] = 2
wurf1[2] = 3
wurf1[3] = 4
wurf1[4] = 5
wurf1[5] = 6
Wurf 2
wurf2[0] = 6
wurf2[1] = 5
wurf2[2] = 4
wurf2[3] = 5
wurf2[4] = 6
wurf2[5] = 1

Hinweis:
VORSICHT: Man erzeugt keine Kopie eines Arrays durch eine einfache Zuweisung!!! Die Quelltextzeile wurf8 = wurf7 bewirkt, dass die Referenz von wurf8 nun auf das Array wurf7 zeigt. Alle Änderungen die in Zukunft an wurf8 vorgenommen werden, werden auch an wurf7 vorgenommen und umgekehrt. Dies ist vergleichbar mit einer Verknüpfung oder einem Hyperlink!

Aufgabe 57

Erstellen Sie ein ausführbares Programm mit dem Namen **ArrayVergleichen.java**. Das Programm vergleicht, ob die Werte der Arrays gleich sind und gibt eine entsprechende Meldung auf dem Bildschirm aus. Überprüfen Sie auch, ob die Referenzen (die Speicherbereiche) der Arrays identisch sind.

wurf1 und wurf2 haben die gleichen Werte.
wurf1-Referenz und wurf2-Referenz zeigen auf zwei unterschiedliche Speicherbereiche.

Hinweis:
Nutzen Sie zuerst aus der Klasse Array die Funktion Arrays.equals.
Danach verwenden Sie den Vergleichsoperator '==' Anschließend vergleichen Sie mit einer Schleife die Werte innerhalb der Arrays

Aufgabe 58

Erstellen Sie ein ausführbares Programm mit dem Namen **QuadratZahlenArray.java**. Das Programm berechnet die Quadratzahlen von 1 bis 40, speichert diese in einem Array, ermittelt zusätzlich die Summe der Quadratzahlen und gibt diese auf dem Bildschirm aus.

1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400,441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600,

Summe = 22140

Hinweis:
Verwenden Sie ein Array zum Speichern der Quadratzahlen.

Aufgabe59

Erstellen Sie ein ausführbares Programm mit dem Namen **ZufallzahlSucheOhneKomma**.java. Initialisieren Sie ein Array mit 100 Stellen und generieren die Zufallswerte zwisch 1 und 49. Das Programm sucht die eingegebene Zahl und gibt die Stellen (nicht den Index) auf dem Bildschirm aus. Bei der fett und kursiv dargestellten Zahl handelt es sich um eine Benutzereingabe.

Geben sie die gesuchte Zahl ein: ***45***
Die gesuchte Zahl kommt an den Stellen 27, 32, 158, 163, 216 vor.

Hinweis:
Das Array wird mit 100 Lottozahlen (Zahlen von 1-49) gefüllt. Deshalb ist es wahrscheinlich, dass die Bildschirmausgabe nach jedem Start anders aussieht. Achten Sie darauf, dass nach der letzten Zahl kein Komma mehr steht.

Aufgabe 60

Erstellen Sie ein ausführbares Programm mit dem Namen **MultiplikationsAufgaben.java.** Ihr Programm generiert zufällige Multiplikationsaufgaben und speichert das eingegebene Ergebnis. Mit der Eingabe Zahl 0 wird die Eingabe weiterer Zahlen beendet.
Danach bekommt der Benutzer ein Feedback, welche Aufgaben richtig oder falsch waren.
Das Programm erzeugt unten stehende Bildschirmausgabe. Bei den fett und kursiv dargestellten Zahlen handelt es sich um Benutzereingaben.

*Was ergibt 4 * 7 : 28*
*Was ergibt 6 * 3 : 18*
*Was ergibt 2 * 9 : 17*
*Was ergibt 6 * 5 : 30*
*Was ergibt 4 * 2 : 8*
*Was ergibt 2 * 4 : 0*
*Aufgabe: 4 * 7 = 28 Richtig!*
*Aufgabe: 6 * 3 = 18 Richtig!*
*Aufgabe: 2 * 9 = 17 Falsch! Richtige Ergebnis: 18*
*Aufgabe: 6 * 5 = 30 Richtig!*
*Aufgabe: 4 * 2 = 8 Richtig!*

Hinweis:
Das Aufgaben- und Lösungs-Array wurde auf 100 Werte begrenzt.

Zweidimensionales Array (Array)

Aufgabe 61

Erstellen Sie ein ausführbares Programm mit dem Namen **ZweiDimensionalesArray**.java. Verwenden Sie in Ihrem Programm die Quelltextzeile:

```
int zahlen[][] = new int[3][2];
```

Ihr Programm erzeugt untenstehende Bildschirmausgabe:

```
zahlen[0][0] = 1
zahlen[0][1] = 2
zahlen[1][0] = 3
zahlen[1][1] = 4
zahlen[2][0] = 5
zahlen[2][1] = 6
```

Hinweis: Das zweidimensionale Array kann man sich wie eine Tabelle vorstellen: zahlen[spalte][zeile]=wert;

Aufgabe 62

Erstellen Sie ein ausführbares Programm mit dem Namen **ZweiDimensionalesArraySchleife.java.** Verwenden Sie dieses Mal eine Schleife für die Bildschirmausgabe:

```
zahlen[0][0] = 1
zahlen[0][1] = 2
zahlen[1][0] = 3
zahlen[1][1] = 4
zahlen[2][0] = 5
zahlen[2][1] = 6
```

Hinweis: Verwenden Sie eine Schleife, um den Zeilenindex hochzuzählen und eine Schleife um den Spaltenindex hochzuzählen.

Aufgabe 63

Erstellen Sie ein ausführbares Programm mit dem Namen **ZweiDimensionalesArraySchnellInitialisiert.java.** Das Programm gibt die Zahlen des Arrays auf dem Bildschirm aus. Initialisieren Sie das zweidimensionale Array durch die Quelltextzeile:

```
int zahlen[][] = { {1,2},{3,4},{5,6} };
```

Beachten Sie die Anordnung der geschweiften Klammern.

```
zahlen[0][0] = 1
zahlen[0][1] = 2
zahlen[1][0] = 3
zahlen[1][1] = 4
zahlen[2][0] = 5
zahlen[2][1] = 6
```

Aufgabe 64

Erstellen Sie ein ausführbares Programm mit dem Namen **VierGewinnt.java.** Das Programm soll ein Vier-Gewinnt-Spiel simulieren. In der ersten Version soll lediglich die Bildschirmausgabe bei jeder der 42 Spalteneingaben neu angezeigt werden. Die Spielsteine der beiden Spieler werden durch die Zahlen 1 und 2 dargestellt. In der nachfolgenden Bildschirmausgabe werden led. 2
Spielzüge simuliert. Die Eingaben für die Spalte sind fett und kursiv
hervorgehoben:

```
1 2 3 4 5 6 7
--------------
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
Eingabe: 5

1 2 3 4 5 6 7
--------------
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 1 0 0
Eingabe: 6

1 2 3 4 5 6 7
--------------
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 1 2 0
Eingabe:
```

Hinweis:
In dieser Version fehlen noch einige Funktionen wie beispielsweise die Gewinnüberprüfung.

Aufgabe 65

Erstellen Sie ein ausführbares Programm mit dem Namen **MultiplikationsAufgaben.java.** Ihr Programm generiert zufällige Multiplikationsaufgaben und speichert das eingegebene Ergebnis. Mit der Eingabe Zahl 0 wird die Eingabe weiterer Zahlen beendet. Danach bekommt der Benutzer ein Feedback, welche Aufgaben richtig oder falsch waren. Das Programm erzeugt unten stehende Bildschirmausgabe. Bei den fett und kursiv dargestellten Zahlen handelt es sich um Benutzereingaben.

*Was ergibt 4 * 7 :* ***28***
*Was ergibt 6 * 3 :* ***18***

*Was ergibt 2 * 9 :* ***17***
*Was ergibt 6 * 5 :* ***30***
*Was ergibt 4 * 2 :* ***8***
*Was ergibt 2 * 4 :* ***0***

*Aufgabe: 4 * 7 = 28 Richtig!*
*Aufgabe: 6 * 3 = 18 Richtig!*
*Aufgabe: 2 * 9 = 17 Falsch! Richtige Ergebnis: 18*
*Aufgabe: 6 * 5 = 30 Richtig!*
*Aufgabe: 4 * 2 = 8 Richtig!*

Hinweis:
Das Aufgaben- und Lösungs-Array wurde auf 100 Werte begrenzt.

Aufgabe 66

Erstellen Sie ein ausführbares Programm mit dem Namen MagischesQuadrat.java. Ihr Programm erzeugt aufgrund nachfolgender Beschreibung die unten stehende Bildschirmausgabe:
Im Folgenden ist der Algorithmus für das magische Quadrat beschrieben.

Beispiel einer Matrix (3*3)

1.) Bilden Sie ein Quadrat aus ungeraden Zahlen. (z.B. 1*1 :-), 3*3,5*5, 7*7, ...)

x x x
x x x
x x x

2.) Die Erste Zahl (1) wird in die Mitte oben geschrieben.

x **1** x
x x x
x x x

3.) Die nächste Zahl wird oben rechts angeordnet

2
x 1 x
x x x
x x x

4.) Befindet sich die Zahl außerhalb der Matrix, wird sie auf der gegenüberliegenden Seite eingefügt.
Wenn oben rausrutscht -> unten einfügen.
Wenn rechts rausrutscht -> links einfügen.

x 1 x
3 x x
x x **2**

5.) Befindet sich bereits eine Zahl in einem Feld, wird die Zahl unter die letzte eingefügte Zahl geschrieben.

x 1 6
3 5 x
4 x 2

6.) Wenn die Zahl in die obere rechte Ecke rutscht, wird sie unter die letzte eingefügte Zahl eingefügt. 7

8 1 6
3 5 7
4 9 2

Die Diagonale ist immer mit fortlaufenden Zahlen gefüllt.

```
8 1 6
3 5 7
4 9 2
```

Die mittlere Zahl ist demnach immer: die kleinste Zahl (1) + die größte Zahl (hier 9) geteilt durch 2 (hier 5).

```
8 1 6
3 5 7
4 9 2
```

Die Bildschirmausgabe in Abhängigkeit der ungeraden Größe könnte wie folgt aussehen:

Magisches Quadrat für size = 3

8 1 6
3 5 7
4 9 2

Magisches Quadrat für size = 5

17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9

Hinweis:
Der beschriebene Algorithmus setzt eine ungerade Zahl für die Matrix voraus.

Aufgabe 67

Erstellen Sie ein ausführbares Programm mit dem Namen **Sudoku.java.** Das Programm speichert ein beliebiges Sudokurätsel und berechnet die Spalten-, Zeilen- und Blocksummen und erzeugt unten stehende Bildschirmausgabe.

```
-----------------------------------
| 6 | 3 | 4 | 9 | 1 | 5 | 7 | 2 | 8 | 45
-----------------------------------
| 2 | 7 | 8 | 6 | 4 | 3 | 5 | 1 | 9 | 45
-----------------------------------
| 5 | 9 | 1 | 2 | 7 | 8 | 6 | 4 | 3 | 45
-----------------------------------
| 4 | 5 | 7 | 3 | 6 | 9 | 2 | 8 | 1 | 45
-----------------------------------
| 9 | 8 | 6 | 4 | 2 | 1 | 3 | 5 | 7 | 45
-----------------------------------
| 3 | 1 | 2 | 8 | 5 | 7 | 4 | 9 | 6 | 45
-----------------------------------
| 1 | 2 | 5 | 7 | 8 | 6 | 9 | 3 | 4 | 45
-----------------------------------
| 8 | 6 | 3 | 5 | 9 | 4 | 1 | 7 | 2 | 45
-----------------------------------
| 7 | 4 | 9 | 1 | 3 | 2 | 8 | 6 | 5 | 45
-----------------------------------
 45   45  45  45  45  45  45  45  45
Blockkontrolle
45 45 45
45 45 45
45 45 45
```

Hinweis:
Lediglich die Summe der Zeilen, Spalten und Blöcke zu bilden ist noch kein Beweis für ein funktionierendes Sudokurätsel. Um sicherzustellen, dass es sich um ein gültiges Sudokurätsel handelt ist es notwendig für Zeile, Spalte und Block zu prüfen, dass die Zahlen von 1-9 jeweils lediglich einmal vorkommen.
Hier finden sie ein ungültiges Sudokurätsel als Kommentar fett gekennzeichnet. Zeilen und Blocksummen stimmen – nicht aber die Spaltensummen.

```
// int sudoku [][] = {
// {5,3,4,6,7,2,1,9,8},
// {6,7,8,1,9,5,3,4,2},
// {9,1,2,3,4,8,5,6,7},
// {8,5,9,4,2,6,7,1,3},
// {7,6,1,8,5,3,9,2,4},
// {4,2,3,7,9,1,8,5,6},
// {9,6,1,2,8,7,3,4,5},
// {5,3,7,4,1,9,2,8,6},
// {2,8,4,6,3,5,1,7,9},
// };
```