Übungsblatt 13

Abgabe: nein

Auf diesem Übungsblatt werden kleine Aufgaben gestellt, die zum Üben genutzt werden können. Es ist sinnvoll, alle Aufgaben tatsächlich zu programmieren, ohne sich Lösungsvorschläge aus dem Internet zusammen zu suchen, denn das würde ja eher nicht üben.

Aufgabe 1 Ungerade Zahlen zählen

Schreibt eine Methode *int howManyOdds(int[] numbers)*, die zählt, wie viele ungerade Zahlen in dem übergebenen Array enthalten sind.

Aufgabe 2 Gerade Zahl finden

Schreibt eine Methode *int findEven(int[] numbers)*, die die erste gerade Zahl in dem übergebenen Array findet und zurückliefert. Enthält das Array keine gerade Zahl, liefert die Methode -1.

Aufgabe 3 Zeile mit nur geraden Zahlen finden

Schreibt eine Methode *int[] findAllEven(int[][] numbers)*, die die erste Zeile in dem übergebenen Array zurückliefert, in der alle Spalten gerade Werte enthalten. Gibt es keine solche Zeile, wird *null* zurückgeliefert. Z.B. liefert *findAllEven* für dieses Array:

```
{{2, 1},
 {3, 1, 2, 9},
 {6, 4, 2}}
```

das folgende Ergebnis:

```
{6, 4, 2}
```

Aufgabe 4 Parität bestimmen

Schreibt eine Methode *boolean evenParity(int number)*, die prüft, ob die Anzahl der gesetzten Bits (also Einser-Bits) in der Eingabe gerade ist. Dies wird auch als Parität bezeichnet.

Aufgabe 5 Matrix transponieren

Schreibt eine Methode *void transpose(final int[][] matrix)*, die eine quadratische Matrix transponiert, d.h. Zeilen und Spalten vertauscht. Z.B. macht *transpose* aus dieser Matrix:

```
{{1, 2, 3},
 {4, 5, 6},
 {7, 8, 9}}
```

diese Matrix:

```
{{1, 4, 7},
 {2, 5, 8},
 {3, 6, 9}}
```

Aufgabe 6 Text transponieren

Schreibt eine Methode *void transpose(final String[] matrix)*, die einen quadratischen Text transponiert, d.h. Zeilen und Spalten vertauscht. Z.B. macht *transpose* aus diesem Text:

```
{"123",
 "456",
 "789"}
```

diesen Text:

```
{"147",
 "258",
 "369"}
```

Aufgabe 7 Doppelliste transponieren

Schreibt eine Methode *void transpose(final List<List<Integer>> matrix)*, die eine quadratische Matrix transponiert, d.h. Zeilen und Spalten vertauscht. Z.B. macht *transpose* aus dieser Matrix:

```
// import static java.util.Arrays.asList;
asList(asList(1, 2, 3),
       asList(4, 5, 6),
       asList(7, 8, 9));
```

diese Matrix:

```
asList(asList(1, 4, 7),
       asList(2, 5, 8),
       asList(3, 6, 9));
```

Aufgabe 8 Bergarbeiter

Vervollständigt die Klasse *Miner* im gleichnamigen Projekt. Dazu muss im Konstruktor eine Wegbeschreibung zum Ziel aus einer Datei eingelesen werden (z.B. mit einem *Scanner*) und dann in der Methode *act* Schritt für Schritt ausgeführt werden. Die Beschreibung besteht einfach aus der Abfolge der Richtungen, in die die Figur gehen muss. Kann die Datei mit der Wegbeschreibung nicht gefunden werden, bewegt sich der Bergarbeiter nicht.

Aufgabe 9 Logische Ausdrücke auswerten

Erweitert die Klasse *Logic* im gleichnamigen Projekt so, dass die Methode *boolean eval(String)* Ausdrücke auswertet, die der folgenden Grammatik entsprechen:

```
or  = and , { '|' , and } ;
and = val , { '&' , val } ;
val = 'f' | 't' | '!' val | '(' or ')' ;
```

Dabei stehen *'f'* für *false* und *'t'* für *true*. Die restlichen Zeichen haben dieselbe Bedeutung wie in Java. Der Ausdruck `"(t|!t)&(f|!f)"` sollte also zu *true* und `"(t&!t)|(f&!f)"` zu *false* ausgewertet werden. Ausdrücke, die nicht der Grammatik entsprechen, erzeugen eine *IllegalArgumentException*.

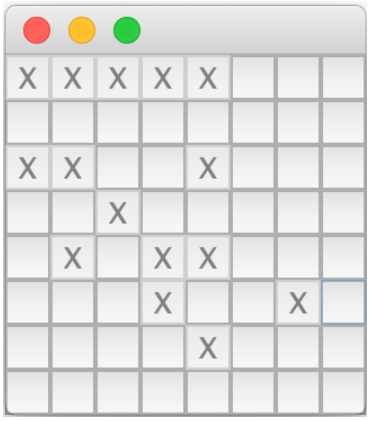

Abbildung 1: *Minesweeper*, wenn die *preferredSize* von Knöpfen auf 20×20 Pixel gesetzt wird.

Aufgabe 10 Minesweeper

Erzeugt ein Fenster, in dem ihr mit Hilfe des passenden Layouts 8×8 Knöpfe platziert, deren Beschriftung zu Beginn aus einem Leerzeichen besteht. Wird ein Knopf angeklickt, ändert sich seine Beschriftung in ein 'X' und er wird deaktiviert, so dass er nicht noch einmal angeklickt werden kann (s. Abb. 1). Da alle Knöpfe dasselbe Verhalten haben, reicht es, einen einzigen *ActionListener* (z.B. als Lambda-Ausdruck) zu definieren, der für alle Knöpfe verwendet wird. Welchen Knopf der *ActionListener* manipulieren soll, kann er über die Methode *getSource* beim übergebenen *ActionEvent* erfragen. Um deren Rückgabe nutzen zu können, müsst ihr sie in den richtigen Typ umwandeln (also den des Knopfes).