Übungen zu Kapitel 12

12.1 Transponierte Matrix:

Entwickeln Sie (anhand des Beispiels `RZERechner`) ein Programm zum Transponieren von Matrizen. Dabei werden die Zeilen und Spalten einer Matrix miteinander vertauscht: Aus der ersten Zeile der Ursprungsmatrix **m** wird die erste Spalte der transponierten Matrix **mT**, aus der zweiten Zeile die zweite Spalte, etc. Dabei wird die Zahl der Zeilen und Spalten vertauscht: Aus der Matrix **m** mit `p` Zeilen und `q` Spalten wird die Matrix **mT** mit `q` Zeilen und `p` Spalten. Ein Element in **m** an der Position `[i][j]` befindet sich in **mT** an der Position `[j][i]`.

Verwenden Sie die Methoden `eingabe()` und `ausgabe()` aus `RZERechner` und eine weitere Methode: `public int[][] transponiere(int[][] matrix)`

Diese bekommt die zu transponierende Matrix übergeben, legt intern ein passendes neues Array an, füllt dieses mit den transponierten Werten und gibt das gefüllte Array zurück.

12.2 Klasse Arrays:

Verwenden Sie die Klasse `Arrays` aus dem Paket `java.utils`. Fügen Sie dazu im `.java`-File der Übung als erste Anweisung:

```
import java.util.Arrays;
```

ein. Die Klasse `Arrays` besitzt u.a. die überladenen statischen Methoden `sort()` und `binarySearch()`.

Legen Sie ein eindimensionales `int`-Array `arr` mit vom Benutzer vorgegebener Länge an und füllen Sie es mit ganzzahligen Zufallszahlen zwischen 0 und 9. Lassen Sie den Inhalt des Arrays auf der Konsole ausgeben.

Lassen Sie das Array mittels `Arrays.sort(arr)` aufsteigend sortieren und ausgeben, um sich von der Sortierung zu überzeugen.

Bestimmen Sie sich den Index des ersten Vorkommens der Zahl `n` mittels `Arrays.binarySeach(arr, n)`. Studieren Sie die J2SDK-Dokumentation der Klasse `Arrays`.

12.3 Test des Zufallszahlengenerators:

Wandeln Sie die Klasse `CheckArray` aus Kapitel 12.1.4 ab, so dass bei *einem* Programmdurchlauf die relative Häufigkeit *aller* möglicher Zufallszahlen ermittelt wird.

Der Benutzer kann die Zahl `m` der Zufallszahlen vorgeben; mittels `(int) (Math.random()*10)` werden `m` Zahlen zwischen 0 und 9 erzeugt und in einem `int`-Array gespeichert.

Eine separate Methode `int[] tester(int[] z)` soll dieses Array übergeben bekommen und zur Auswertung durchlaufen. In einem internen Auswertarray der Länge 10 wird an den Indexpositionen 0 bis 9 die absolute Häufigkeit des Vorkommens der entsprechenden Zufallszahl durch Hochzählen gespeichert – orientieren Sie sich dazu am Coding des Lottobeispiels.

Die `Methode tester()` gibt das Array der absoluten Häufigkeiten zurück. Aus diesen absoluten Häufigkeiten sollen anschließend die relativen Häufigkeiten als Prozent-Werte berechnet und ausgegeben werden.

12.4 Speichern von Objekten:

Speichern Sie `Konto`-Objekte in einem `Konto`-Array und greifen Sie auf die gespeicherten Objekte zu.

Legen Sie eine Klasse `Konto` mit typischen Attributen und Methoden an: Zumindest sollen die privaten Attribute `String name`, `int kontoNummer` und `double kontoStand` vorhanden sein, sowie die öffentlichen Methoden `einzahlen()`, `abheben()` und ein Konstruktor. Ferner die öffentlichen Methoden `getKonto`-Nummer() zur Abfrage von `kontNummer` und `datenAusgabe()` zur Ausgabe aller `Konto`-Attributwerte. Durch das statische (im Konstruktor hochgezählte) Attribut `anzahl` wird die Zahl der erzeugten Konten vermerkt. Die Kontonummer wird im Konstruktor automatisch auf den Wert `anzahl+1` gesetzt.

In der ausführbaren Klasse `Bank` soll in `main()` das `Konto`-Array **`konten`** mit Platz für 100 `Konto`-Objekte angelegt werden. Innerhalb einer Schleife soll der Benutzer unter folgenden Optionen wählen können:

→ Konto anlegen → Einzahlen → Abheben → Kontodatenausgabe.

Beim Anlegen eines Kontos werden Namen und Kontostand abgefragt und ein neues `Konto`-Objekt erstellt. Jedes `Konto`-Objekt wird in dem Array `konten` an der *Indexposition* gespeichert, die der *Kontonummer des* `Konto`*-Objekts* entspricht:

```
Konten[ k.getKontoNummer()] = k;   // z.B. für Konto-Objekt k
```

Bei Manipulation oder Datenabfrage bestehender `Konto`-Objekte wird vom Benutzer die Kontonummer abgefragt. Diese ist identisch mit dem Index des gesuchten `Konto`-Objekts im Array `konten`. Vor einem `Konto`-Zugriff an der Indexposition *pos* soll mit der Bedingung `konten[pos]!=null` abgeprüft werden, ob dort wirklich ein `Konto`-Objekt abgelegt ist, um eine `NullPointerException` zu vermeiden.

12.5 Speicherbedarf von Arrays:

Die folgende Demonstration macht deutlich, dass nicht unbegrenzt Speicherplatz für ein Array im Hauptspeicher zur Verfügung steht. In einer Endlosschleife wird einer Arrayvariable ein immer größeres Array vom Typ `double` mit der Länge `m` zugewiesen: Beginnend bei `m`=100000 wird die Arraylänge jeweils um jeweils 100000 Ele-

mente vergrößert. Schließlich kommt es zu einem `java.lang.OutOfMemory`-Abbruch des Programms:

```
class Ueberlauf {
   public static void main( String[] args ){
      double[] a;    int m = 100000;
      while (true) {
         a = new double[m];
         m = m + 100000;    IO.writeln( "m = " + m);
      }
   }
}
```

Auf einem PC mit 256MB kommt es bei einer Länge von ca. 8 Millionen Datenelementen zum Abbruch. Wird das Array als 4-Byte-Typ `float` oder 4-Byte-Typ `int` deklariert, so erhält man einen Abbruch erst bei ca. 16 Millionen, bei Deklaration als 2-Byte-Typ `short` entsprechend erst bei ca. 32 Millionen Datenelementen. Wiederholen Sie das Experiment auf Ihrem Rechner!

12.6 Expliziter Aufruf der `main()`-Methode:

Auch der Aufruf der `main()`-Methode einer *anderen* ausführbaren Klasse ist technisch möglich. Somit kann ein Java-Programm aus einem anderen heraus aufgerufen werden. Nach der Abarbeitung des aufgerufenen Programms wird das aufrufenden Programm hinter der Aufrufstelle fortgesetzt:

```
class ProgA {  // In ProgA.class
   public static void main( String[] args ) {
      for( int i=0; i<args.length; i++ ) {
         IO.write( args[i] + " / " );
      }
   }
}
class ProgB {  // In ProgB.java
   public static void main( String[] args ) {
      String[] arg = { "Hallo", "ProgA!" };
      IO.writeln( "Wir sind in Programm B." );
      ProgA.main( arg );
      IO.writeln( "\n Wieder zurück in Programm B." );
   }
}
```

Wie lautet die Konsolenausgabe des Programms ?