Übungsaufgabe Email-Verteiler

Die folgende Aufgabe soll noch einmal einige Themen der letzten Stunden zusammen führen sowie en passant ein paar kleine Nebenthemen einführen: Schreiben von Textdateien, Java Varargs, XML-Dateien.

Die Aufgabe

Schreibe ein konsolenbasiertes Programm zur Verwaltung eines Emailverteilers. Die Emails und die Namen der Inhaber werden in einer Textdatei gespeichert. Das Programm liest bei Programmstart die komplette Liste aus der Datei ein und ermöglicht hinzufügen, löschen und editieren der Datensätze. Bei Beenden des Programms werden die Daten zurück in die Datei geschrieben. Außerdem gibt es die Möglichkeit, die Daten in eine XML-Datei zu exportieren.

Beachte das Vorlageprojekt (entweder mit Netbeans öffnen oder die .java-Datei rauskopieren, wenn Du eine andere IDE verwendest) und den Beispielprogrammablauf im Anhang.

Vorgaben

- Jede einzelne Kombination Personenname <-> Email wird als String-Array gespeichert. Die Gesamtliste ist eine ArrayList dieser String-Arrays:

```
    private static ArrayList<String[]> emails = new ArrayList<>();
```

- die Textdatei hat den Namen emails.csv, wird ohne expliziten Pfad angelegt. Das Dateiformat ist klassisches CSV (→ „comma-separated-values“) :ein Datensatz pro Reihe, Name und Email werden durch Komma getrennt:

```
Tick,tick@entenhaus.en
Trick,trick@entenhaus.en
Jin,jin@sakai.tsu.jp
```

- bei Beenden des Programms wird die CSV Datei komplett neu geschrieben

Varargs

In Java-Methoden darf der jeweils letzte Parameter ein „Vararg“ sein – beim Aufruf können für diesen Parameter beliebig viele Werte übergeben werden. Vararg-Parameter werden durch drei Punkten markiert. Aus Sicht der Methode handelt es sich um eine Array, der Unterschied liegt im Aufruf:

Normales Array

```
private int addiere (int[] zahlen){
  int summe = 0;
  for (int z : zahlen)
         summe += z;
  return summe;
}

// Es muss ein Array übergeben werden

private void beispiel(){
  int[] z = {45,3,12};
  int summe2 = addiere(z);
}
```

Varargs

```
private int addiere (int... zahlen){
  int summe = 0;
  for (int z : zahlen)
         summe += z;
  return summe;
}

// Es kann eine Array übergeben werden
// oder man zählt einfach Werte auf
private void beispiel(){
  int[] z = {45,3,12};
  int summe1 = addiere(45,3,12);
  int summe2 = addiere(z);
}
```

Übungen (direkt in der Vorlage lösen)

Schreibe eine Funktion

```
private static void outputNumberedList(String... entries)
```

die alle `entries` nummeriert ausgibt:

```
outputNumberedList("Un","Dos");
```

```
1. Un
2. Dos
```

Programmiere jetzt eine Funktion „menu“ die ein nummeriertes Menü anzeigt, den User eine Zahl eingeben lässt und diese Zahl als Funktionswert zurückgibt:

```
int wahl = menu("Kaffee","Tee");
```

```
1. Kaffee
2. Tee

Deine Wahl: 2
```

Text-Dateien schreiben

Eine Textdatei könnte man schreiben, indem man ein Objekt der Klasse „FileWriter“ anlegt und dessen append-Funktion benutzt:

```
       try {
            FileWriter test = new FileWriter("beispiel.txt");
            test.append ("Jo");
            test.close();
        } catch (Exception ex){
            ex.printStackTrace();
        }
```

Da der Zugriff auf das Dateisystem zu Laufzeitfehlern führen kann (Dateisystem nicht schreibbar/voll etc), muss das Ganze per try-catch abgesichert werden. Den FileWriter kann man mit einem zusätzlichen Parameter aufrufen, um zu steuern, ob eine eventuel existierende Datei überschrieben oder an sie angehängt wird:

```
…
 FileWriter test = new FileWriter("beispiel.txt", true); // Es wird angehängt
…
```

Üblicherweise schreibt man nicht direkt mit einem FileWriter sondern verwendet eine BufferedWriter, der zwischen den physikalischen Schreibvorgängen mehrere Ausgaben – der Vorgang wird dadurch effizienter. Außerdem hat der BuffererWriter eine eigene Methode, um in der Datei eine neue Zeile anzufangen, man braucht sich dann keine Gedanken um das (betriebsystemübergreifend) passende Sonderzeichen zu machen.

```
        try {
            BufferedWriter w = new BufferedWriter(new FileWriter("beispiel.txt"););
            w.append("Jo");
            w.newLine();
            w.close();
        } catch (Exception ex){
            ex.printStackTrace();
        }
```

Ich persönlich lege das File-Objekt gerne explizit an, dann kann ich bei Bedarf zum Beispiel den vollständigen Pfad abfragen. Da File-Objekte keine Änderungen am Dateisystem vornehmen können, müssen sie nicht fehlerbehandelt und können auch außerhalb des try-Blocks verwendet werden:

```
        File datei = new File("beispiel.txt");
        boolean append = datei.exists();
        try {
            BufferedWriter w = new BufferedWriter(new FileWriter(datei,append));
            w.append("Jo");
            w.newLine();
            w.close();
            System.out.println("Datei "+datei.getAbsolutePath()+" geschrieben ");
        } catch (Exception ex){
            System.out.println("Kann "+datei.getAbsolutePath()+" nicht schreiben.");
    }
```

Übung

Vervollständige in der Vorlage die Funktion

```
private static void save(String name)
```

die die Daten aus „emails“ als kommaseparierte Liste in eine Datei schreibt.

XML (EXtensible Markup Language)

XML ist eine Datenbeschreibungssprache bei der Daten durch „tags“ (englisch „tag“ → Etikett/Markierung) gekennzeichnet und strukturiert werden. Ein „tag“ besteht aus einem Namen in spitzen Klammern. Um etwas zu umschließen gibt es öffnende XML-tags und schließende XML-tags – sie unterscheiden sich durch einen Slash:

```
<beispiel>
      <text>Ein XML Beispiel</text>
      <wichtigkeit>3</wichtigkeit>
</beispiel>
```

Die Einrückungen sind übrigens nicht notwendig sondern werden nur gemacht, wenn XML für Menschen übersichtlich lesbar sein soll.

XML-Elemente dürfen Attribute haben, diese werden als Name-Wert-Paar im Starttag aufgeführt. Die Werte müssen immer in Anführungszeichen gesetzt werden. Dadurch gibt es in XML mehrere Möglichkeiten, Daten zu kapseln:

```
<beispiel wichtigkeit="3">
      <text>Ein XML Beispiel</text>
</beispiel>
```

Möchte man Tags verwenden, die gar nichts umschließen, kann man sie als selbstschließende Tags verwenden – der Slash kommt dann vor die schließende spitze Klammer:

```
<beispiel wichtigkeit = "3" text="Ein XML Beispiel" />
```

Für eine ganze XML-Datei gibt es noch zwei weitere Regeln:

- sie beginnt immer mit einem „Prolog“, der die XML-Version angibt:

```
<?xml version="1.0" encoding="UTF-8"?>
```

- eine XML-Datei hat genau ein Wurzelelement – alle weiteren Elemente müssen dort drin stecken:

```
<?xml version="1.0" encoding="UTF-8"?>
<beispiele>
      <beispiel wichtigkeit = "3" text="Ein XML Beispiel" />
      <beispiel wichtigkeit = "2" text="Noch ein XML Beispiel" />
</beispiele>
```

In einer XML-Datei könnten unsere Adressen also folgendermaßen gespeichert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<emails>
  <email>
    <name>Tick</name>
    <address>tick@entenhaus.en</address>
  </email>
  <email>
    <name>Trick</name>
    <address>trick@entenhaus.en</address>
  </email>
  <email>
    <name>Jin</name>
    <address>jin@sakai.tsu.jp</address>
  </email>
</emails>
```

Übung

Schreibe die Methode „exportToXml(String name)“ die die Email-Liste in eine XML-Datei entsprechend dem oben gezeigten Format speichert.

Hinweis: Du brauchst hier Anführungszeichen in einem Java-String. Da diese eigentlich dazu dienen, den Java-String zu begrenzen, muss man sie mit einem Escape-Zeichen versehen:

```
System.out.println("He said \"Bye\" and left");
```

Der Rest

Schreibe jetzt die Methoden zum Laden der Emails, Anzeigen der Liste sowie für das Hinzufügen, Löschen und Ändern von Datensätzen.

Anhang – Beispiel für einen Programmablauf

```
Loading from /home/sascha/wara/it/java/collections/loesungen/EmailVerteiler/emails.csv
Loaded 3 Emails

 1 Show Emails
 2 Add Email
 3 Remove Email
 4 Edit Email
 5 Export to XML
 6 Save & Exit Program
Your choice:
3

Stored Emails
---------------

Number |                Name |                          Email
-------+---------------------+------------------------------
     1 |                Tick |             tick@entenhaus.en
     2 |               Trick |            trick@entenhaus.en
     3 |                 Jin |              jin@sakai.tsu.jp
Which one to remove ? (0 for 'Main Menu')
1
Are you sure you want to remove Tick/tick@entenhaus.en ? [y/n]y
Removed...

Stored Emails
---------------

Number |                Name |                          Email
-------+---------------------+------------------------------
     1 |               Trick |            trick@entenhaus.en
     2 |                 Jin |              jin@sakai.tsu.jp
Which one to remove ? (0 for 'Main Menu')
0
```

```
 1 Show Emails
 2 Add Email
 3 Remove Email
 4 Edit Email
 5 Export to XML
 6 Save & Exit Program
Your choice:
2
Enter a name
Huey
Enter the email of Huey
huey@duckburg.edu
Added email huey@duckburg.edu for Huey

 1 Show Emails
 2 Add Email
 3 Remove Email
 4 Edit Email
 5 Export to XML
 6 Save & Exit Program
Your choice:
1

Stored Emails
--------------

Number |                 Name |                          Email
-------+----------------------+-------------------------------
     1 |                Trick |             trick@entenhaus.en
     2 |                  Jin |               jin@sakai.tsu.jp
     3 |                Huey |              huey@duckburg.edu

 1 Show Emails
 2 Add Email
 3 Remove Email
 4 Edit Email
 5 Export to XML
 6 Save & Exit Program
Your choice:
4

Stored Emails
--------------

Number |                 Name |                          Email
-------+----------------------+-------------------------------
     1 |                Trick |             trick@entenhaus.en
     2 |                  Jin |               jin@sakai.tsu.jp
     3 |                Huey |              huey@duckburg.edu

Stored Emails
--------------

Number |                 Name |                          Email
-------+----------------------+-------------------------------
     1 |                Trick |             trick@entenhaus.en
     2 |                  Jin |               jin@sakai.tsu.jp
     3 |                Huey |              huey@duckburg.edu
Which one to edit ? (0 for 'Main Menu')
1
Editing Trick/trick@entenhaus.en
What do you want to change ?

 1 Name
 2 Email
Your choice:
```

```
1
Enter new Name
Dewey
Changed Name to Dewey

Stored Emails
---------------

Number |                Name |                          Email
-------+---------------------+-----------------------------
     1 |               Dewey |            trick@entenhaus.en
     2 |                 Jin |              jin@sakai.tsu.jp
     3 |                Huey |             huey@duckburg.edu
Which one to edit ? (0 for 'Main Menu')
0

 1 Show Emails
 2 Add Email
 3 Remove Email
 4 Edit Email
 5 Export to XML
 6 Save & Exit Program
Your choice:
5
Emails exported to /home/sascha/wara/it/java/collections/loesungen/EmailVerteiler/emails.xml

 1 Show Emails
 2 Add Email
 3 Remove Email
 4 Edit Email
 5 Export to XML
 6 Save & Exit Program
Your choice:
6
Emails saved to /home/sascha/wara/it/java/collections/loesungen/EmailVerteiler/emails.csv
Bye
```