תוכנה I בשפת ג'אווה
בית הספר למדעי המחשב
אוניברסיטת תל אביב

Assignment No. 7
Address Book – Part I

This assignment consists of two parts. In the first part you will write a class (and additional classes if needed) for maintaining an address book and a textual user-interface application. In the second part you will implement a graphical user interface (GUI) for the address book. Your implementation for both parts should follow the model-view separation paradigm. This paradigm dictates that the model of an application (logic and functionality) should be separated from the visual representation (the user-interface). The rationale behind this approach is that visual representation tends to change a lot. Model-view separation ensures us that view changes will not affect the basic model and enables us to maintain one model for several different views (in our case, textual and graphical user-interfaces).

In this part of the assignment you are required to write two main classes: **`AddressBook`** for maintaining the logic of the address book (the model) and **`TextualAddressBookViewer`** for a textual user-interface application (the view).

The **`AddressBook`** class should adhere to the following specifications:

- Each entry in the address book (**`Contact`**) will consist of the following fields:
  - **name**: a string representing last and first names separated by a comma, e.g. "Smith, John". This field is mandatory and thus cannot be null. Names are case insensitive.
  - **email**: a string, can be null.
  - **telephone**: a string, can be null.
  - **address**: consists of four strings that stand for: street, city, zip code and country, where each can be null.

- The **`AddressBook`** class will support the following operations:

  - **`public void add(Contact c)`** – add a new contact. A new contact is a contact for which the name does not already exist in the address book.

  - **`public Contact get(String name)`** – return the contact of the given name field.

  - **`public void delete(String name)`** – delete the contact of the given name field.

  - **`public void modify(Contact c)`** – replace an existing contact with a new one. A contact with the same name field should appear in the address book before the call.

  - **`public Iterator<Contact> getContacts()`** – return an iterator over the contacts in the address book , sorted by the name fields alphabetically.

  - **`public int getCount()`** – return the number of contacts in the address book

- **`public Iterator<Contact> search(String prefix)`** – return an iterator over the contacts in the address book for which the name field starts with the given prefix. The order is by names, alphabetically.

- **`public void save(String filename)`** – save the whole address book to the given file (using serialization).

- **`public void load(String filename)`** – load an address book from the given file name (using serialization)

- The underlying structure of the class should be a standard **`java.util`** collection or map. You may consult java libraries API (http://java.sun.com/j2se/1.5.0/docs/api/) to find more about the services offered by the different classes.

- You should properly define the contract of the class (e.g. state for the get method what to do if the given contact does not exist) and add appropriate exceptions to its methods

The **`TextualAddressBookViewer`** application will get as an argument a filename. Each line in the given file should be one of the following records:

- **`n`** – for creating a new address book
- **`l <filename>`** - for loading an address book from a file
- **`a <name>;<email>;<telephone>;<street>;<city>;<zip>;<country>`** - for adding a new contact to the address book. Note that only the name filed is mandatory. The rest of the fields can be null.
- **`p <name prefix>`** - for printing to the standard output all the contacts for which the name field starts with the given prefix
- **`x`** – for printing all the contacts in the address book
- **`d <name>`** - for deleting the contact of the given name
- **`m <name>;<email>;<telephone>;<street>;<city>;<zip>;<country>`** - for replacing an existing contact with the given one.
- **`s <filename>`** - for saving the address book to a file

Here is an example for an input file for the textual user-interface application:

```
n
a Smith, John;smith@gmail.com;03-6404324;23 Laskov St.;Tel-Aviv;56743;Israel
a Stein, Rita;rita@gmail.com;03-5524324;;;;
a Altman, Rebecca;rebecca@gmail.com;03-9414324;;Tel-Aviv;42732;Israel
a Altman, David;david@gmail.com;;;;;
p Altman
p Altman, Rebecca
x
d Stein, Rita
m Altman, David; david@gmail.com; 03-9414324;;;;
x
s addresses.data
```

Note that the first line must be "`n`" or "`l <filename>`".

<u>הוראות הגשה:</u>

1. קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
2. הגשת התרגיל תעשה ע"י המערכת VirtualTAU (http://virtual2002.tau.ac.il/).
3. הגשת התרגיל היא ע"י יצירת קובץ zip אשר יכיל את כל הקבצים הדרושים:
   א. יש לקרוא לקובץ ה- zip לפי שם המשתמש שלכם, למשל: zvainer.zip
   ב. יש לצרף קובץ פרטים אישיים בשם details.txt שיכיל את שמכם ומספר תעודת הזהות שלכם
   ג. נוסף על הקבצים הבודדים יש להגיש קובץ טקסטואלי נוסף (doc/txt/rtf) שיכיל את כל תשובותיכם