Assignment No. 11
Address Book

In this assignment you are required to write an address book application. An address book is used for storing contacts sorted in alphabetical order of people's names.
This assignment consists of three parts. In the first part you will implement the core address book functionality (adding / removing contacts, save / load an address book, etc.). The second part will provide a simple – console based – interface for the address book you implemented in part 1. Part 3 will improve on part 2 by providing a Graphical User Interface for the address book.
The three parts 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).

Part 1 – The Model

In this part you will implement the core address book functionality.

You are required to define and implement the classes `AddressBook` and `Contact.` This, however, is the basic requirement. You may define and implement as many other classes as you see fit to aid you in your task.

The class `AddressBook` will support the following operations:

- **`void add(Contact c)`** – add a new contact.
- **`void remove(String name)`** – remove a contact from the address book
- **`Contact get(String name)`** – retrieve an existing contact
- **`boolean exists(String name)`** – check if a contact already exists
- **`void replace(Contact c)`** – replace an existing contact with a new one.
- **`Iterator<Contact> getContacts()`** – return an iterator over the contacts in the address book, sorted by the name fields in alphabetical, case insensitive order.
- **`int size()`** – return the number of contacts in the address book
- **`List<Contact> search(String prefix)`** – return a list of contacts in the address book for which the name field starts with the given prefix. The order is by names, alphabetical, case insensitive order.
- **`void save(String filename)`** – save the whole address book to the given file (using serialization).
- **`void load(String filename)`** – load an address book from the given file name (using serialization)

Remarks:

- Names equality is case insensitive. For example, “Doe, John” is equal to “doe, john”. However, you are not allowed to change the name provided by the user.
- The required methods are under specified, e.g. what should happen if you call replace() for a contact that doesn’t exists? You should decide how to handle such cases and specify the method contract accordingly.
- Add throws clauses where appropriate.
- Use the standard collections framework (sets, maps, list etc.) for the underlying structure of the class.

Stored in the address book are contacts. In addition to the class `AddressBook` you will implement the class `Contact` that holds the following information:

- **name**: a string representing last and first name separated by a comma, e.g. "Smith, John". (mandatory).
- **email**: the contact’s email address stored as a string (optional).
- **telephone**: the contact’s telephone number, stored as a string (optional).
- **address**: the street address, consists of street, city, zip code and country, all stored as strings (optional).

You should define constructor(s), getters / setters and other methods as you see fit.
Mandatory information must be available throughout an object life cycle, whereas optional field may be empty at some times.

Part 2 – Textual View

In this part you will implement a simple textual user interface for the address book. You will implement the class `TextualAddressBookView` and the application `TextualAddressBookViewer` which uses the view.

The `TextualAddressBookViewer` application will run in two modes: a) an interactive mode and b) a batch mode.
In interactive mode the user will enter commands and the application will execute them. In batch mode a file of command is supllied to the application, each line contains a single command. The application will read the file one line at a time and will execute the command. The file name is supplied to the application on the command line. If a file name is not given the application will run in interactive mode. If a file is given it will run in batch mode.

The list of possible commands is as follows:

- **`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
- **`r <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
- **`e`** – exit application

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
e
```

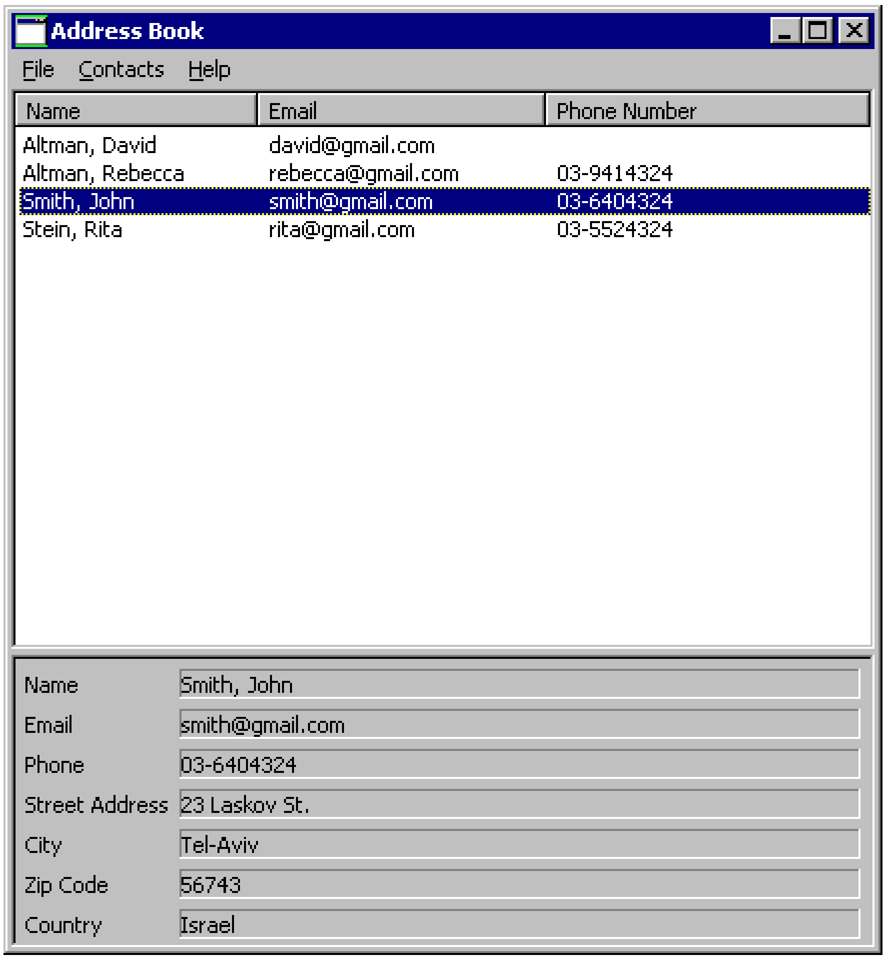
Note that the first line should be "`n`" or "`l <filename>`".

Part 3 – Graphical User Interface

In this part of the assignment you will use the Standard Widget Toolkit (SWT) to implement a graphical user interface (GUI) for your address book. You should be able to use the address book from part 1 without modifications, only replacing the textual interface for a GUI one.

The GUI will be implemented in one class named `GUIAddressBookViewer`. A template for this class can be found on the course web-site. If you run the application (instructions for installing and running an SWT-based application are below), you can see that the GUI is a window consisting of three parts (see the figure below for a snapshot):

- a menu bar
- a table for showing all the contacts (should be ordered by name alphabetically)
- a form for showing the full details of the currently selected (in the table) contact



Name	Email	Phone Number
Altman, David	david@gmail.com	
Altman, Rebecca	rebecca@gmail.com	03-9414324
Smith, John	smith@gmail.com	03-6404324
Stein, Rita	rita@gmail.com	03-5524324

Name	Smith, John
Email	smith@gmail.com
Phone	03-6404324
Street Address	23 Laskov St.
City	Tel-Aviv
Zip Code	56743
Country	Israel

Your assignment is to support all the options of the menu bar, that is:

- File→New Address Book: Create a new, blank, address book
- File→Open: Load an address book from a file, using the standard file open dialog
- File→Save: Save an address book. If it was opened from a file, save it to the same file. Otherwise, request a filename in a standard file save dialog

- File→Exit: Exit the application
- Contacts→New: Open a dialog asking for the details of the new contact; create a new contact accordingly.
- Contacts→Edit: Edit the details of the currently selected (in the table) contact; use the same dialog box as the one for creating a new contact.
- Contacts→Delete: Delete the currently selected contact from the address book.

Notes:

- Before performing a File→New/Open/Exit operation, you should check that there are no unsaved changes to the current working address book, and if this is not the case, show a message box asking the user to confirm or allowing him to save the current address book before exiting.
- The table should always be consistent with the address book's contents.
- Decide which exceptions should be handled and how (e.g. showing a message box telling the user about the problem). Any reasonable decision will be accepted. Application crashes are not reasonable.
- Fill in the table with the contacts of the address book, in alphabetic name order (as in Part I), including keeping it synchronized with the address book.
- On highlight (selection) of a table row, fill in the full details in the form below it. On double click (default selection) perform the Contact->Edit functionality.
- The `setHeaderVisible` method of the `Table` class is buggy under Linux. As a result, on Linux the column names of the table might be invisible (see https://bugs.eclipse.org/bugs/).

General Advice:

GUI programming is often done using existing code and altering it to suit your needs. The skeleton that is provided to you contains much of the functionality you need in order to implement the address book.

Configuring Eclipse to use the SWT Libraries and Running an SWT-based Application:

- Download the .zip of SWT 3.4.
  For windows visit:
  http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.4-200806172000/swt-3.4-win32-win32-x86.zip

- For Linux visit:
  http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.4-200806172000/swt-3.4-gtk-linux-x86.zip

- For OS X visit:
  http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.4-200806172000/swt-3.4-carbon-macosx.zip

- Follow the instructions in http://www.eclipse.org/swt/eclipse.php

Note: To open the properties dialog of your Java project, right click on you project in the package explorer view.

הוראות הגשה:

קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
הגשת התרגיל תעשה ע"י המערכת VirtualTAU (http://virtual.tau.ac.il). הוראות שימוש במערכת ניתן למצוא ב- http://virtual2002.tau.ac.il/upload/misc/main1.html
הגשת התרגיל תתבצע ע"י יצירת קובץ zip שנושא את שם המשתמש. לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip.
קובץ ה zip יכיל:

- קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
- קבצי ה-java. של התכניות שהתבקשתם לכתוב.
- קובץ טקסט עם העתק של כל קבצי ה Java.