Assignment No. 11
Address Book

In this assignment you are required to write an address book application. An address book is used for storing contact information sorted in alphabetical order of people's names.
This assignment consists of two 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 Graphical User Interface (GUI) for the address book.
The two 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 while the model remains fairly constant. Model-View separation ensures us that changing the view does not require changing the underlying model and it enables us to maintain one model for several different views.

Part 1 – The Model

This is the core implementation of the address book. We supply you with a skeleton implementation were large portion of the code had already been implemented for you.
Four classes comprise the core functionality of the address book:

- **`Address`** – Represents a location (street, city, country and zip code)
- **`Contact`** – Represents a person's contact information (name, email, phone and address)
- **`AddressBook`** – A sorted collection of contacts
- **`AddressBookUtils`** – Utility functions for loading and saving an address book

The **`Address`** and **`Contact`** class have been fully implemented for you whereas the other two classes are missing parts of their implementation which you will need to provide.

The **`AddressBook`** class is a sorted collection of contacts. The contacts should be easily accessible by a person's name and are sorted alphabetically in a **case insensitive** manner. The class provides methods for adding and deleting contacts, searching for contacts, etc. (see the code).

The class **`AddressBookUtils`** provides utility functions for saving and loading the content of the entire address book. It implements the following functions:

- **`static void saveAddressBook(AddressBook ab, String filename)`** – save the entire address book to the given file (using serialization).
- **`static AddressBook loadAddressBook(String filename)`** – load an address book from the given file (using serialization)

Notes:

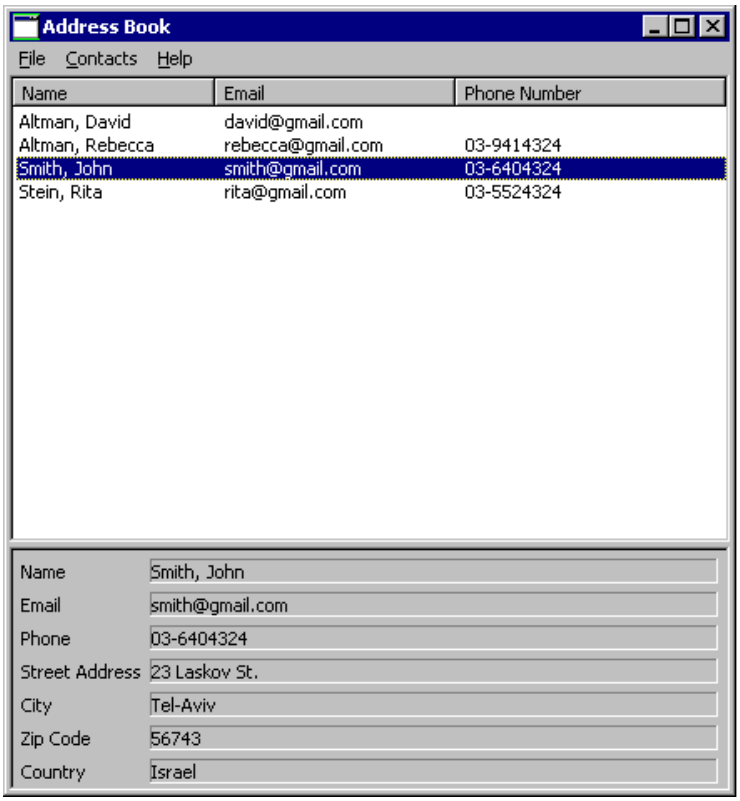
- Names equality is **case insensitive**. For example, “Doe, John” is equal to “doe, john”. However, the address book should retain the names in their original form as provided by the user.
- Add exceptions and throws clauses where appropriate.
- Use the standard collections framework (sets, maps, list etc.) for the underlying representation.
- 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 fields may be empty at some times.

Part 2 – 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 classes from part 1 without modifications, only replacing the textual interface for a GUI one.

The GUI will be implemented in one class named `AddressBookView`. A template for this class can be found on the course website. In the published zip file you will also find the class AddressBookApplication serving as your application entry point. If you run the application (see instructions below), you will see that the GUI is a window consisting of three parts:

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



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 is acceptable (Application crashes are not reasonable).
- Fill in the table with the contacts of the address book, in alphabetic name order keeping it synchronized with the underlying address book.
- On highlight (selection) of a table row, fill in the details in the form below it. On double click (default selection) open a dialog where the user can edit the contact's details.
- 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/).
- Write methods to implement the various actions. Have your listeners call these methods, rather than implementing the full event handling capability within the listener's body.
- The AddressBook.jar file (published on our web site) is an implementation of the address book functionality. You can play with it (double-click to execute) to better understand what is expected of you.

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.

Configure Eclipse to use the SWT Library and Run an SWT-based Application:

- Download the stable SWT release at http://www.eclipse.org/swt/
- Read the article "Developing SWT applications using Eclipse" and follow the instructions

<u>הוראות הגשה:</u>

קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
הגשת התרגיל תעשה ע"י במערכת ה VirtualTAU בלבד.
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.