תוכנה 1
תרגיל מס' 11

חלק 1: Enums

בתרגיל 9 נתבקשתם לממש את המחלקה MyCar בכמה אופנים שונים. כעת שכתבו מחלקה זו ע"י הגדרתה כ-Enum.

תזכורת: המחלקה MyCar

```
public class MyCar {
    public static final int LAMBORGHINI    = 2;
    public static final int SUSITA         = 4;
    public static final int RENAULT        = 5;
    private int type;
    public MyCar(int type) {
        this.type = type;
    }
    public String getManufacturerName() {
        switch (type) {
            case LAMBORGHINI:
                return "LAMBORGHINI";
            case RENAULT:
                return "RENAULT";
            case SUSITA:
                return "SUSITA";
            default:
                return "Manufacturer Unknown";
        }
    }

    public int getNumOfDoors() {
        switch (type) {
            case LAMBORGHINI:
                return LAMBORGHINI;
            case RENAULT:
                return RENAULT;
            case SUSITA:
                return SUSITA;
            default:
                return -1;
        }
    }
}
```

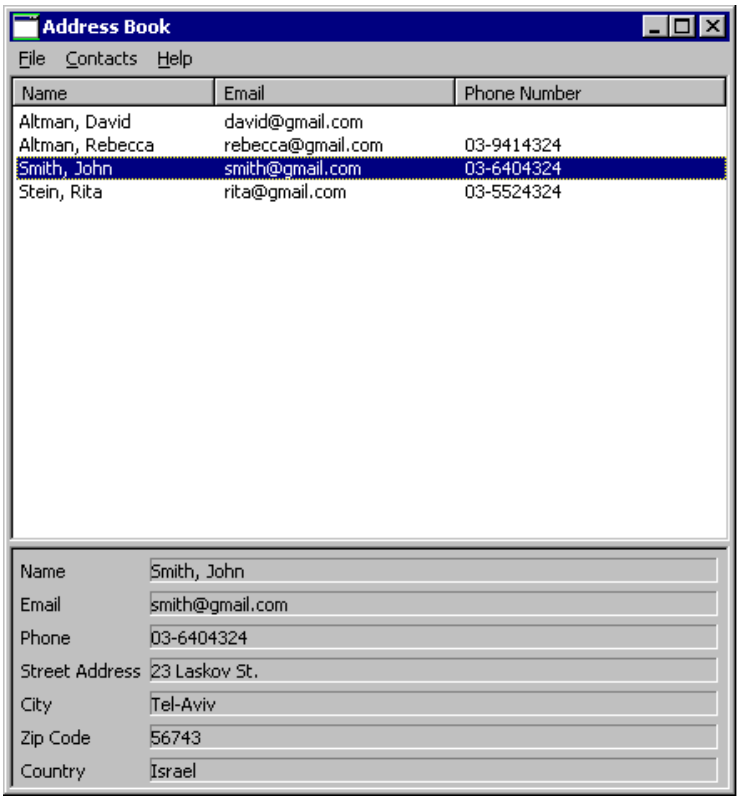
חלק 2: GUI

In assignment 8 we began implementing an address book system. In this assignment we are going to 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 assignment 8 without modifications, only replacing the textual interface for a GUI one.

The GUI will be implemented in a class named `SWTAddressBookView`. You can find a template for this class in the zip file, published on the course' web site.
In addition to that class you are also required to implement the two static functions in the `AddressBookUtilsClass`. (**hint**: notice that the `IAdreessBook` interface extends Serializable)

The zip file also contains the class `AddressBookApplication`, this class will serve as your entry point. Running the application (see instructions below), you will notice 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



Obviously, the first time you will run the application the table and form will be empty.

What you need to do:

Support all the options of the menu bar, that is:

- **File→New Address Book**: Create a new, empty, 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.

Additional requirements:

- Before performing a File→New/Open/Exit operation, the application should check that there are no unsaved changes to the current working address book. If there are use a message box to ask the user whether she would like to save those changes prior to exiting. If the user chooses to save the adders book follow the guidelines for the **Save** operation.
- The table view should always be consistent with the underlying address book's contents.
- Your application should never crash. The only way to terminate the application should be by choosing to do so. 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 (crashes are not reasonable).
- Fill in the table with the contacts of the address book, sorted by name in alphabetic 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.
- 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 (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. We provided TODO markers throughout the code to ease the implementation. Look for those marker for guidance.

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

הוראות הגשה:

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.