

תוכנה 1

תרגיל מספר 11

הנחיות כלליות:

- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
 - הגשת התרגיל תעשה במערכת ה VirtualTAU בלבד (<http://virtual2002.tau.ac.il/>).
 - יש להגיש קובץ zip יחיד הנושא את שם המשתמש (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip) קובץ ה zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
 - ב. קבצי ה java של התוכניות אותם התבקשתם לממש.
 - ג. קובץ טקסט עם העתק של כל קבצי ה java
 - ד. קובץ טקסט בשם answers עם התשובות לשאלות
-

חלק 1: Address Book GUI

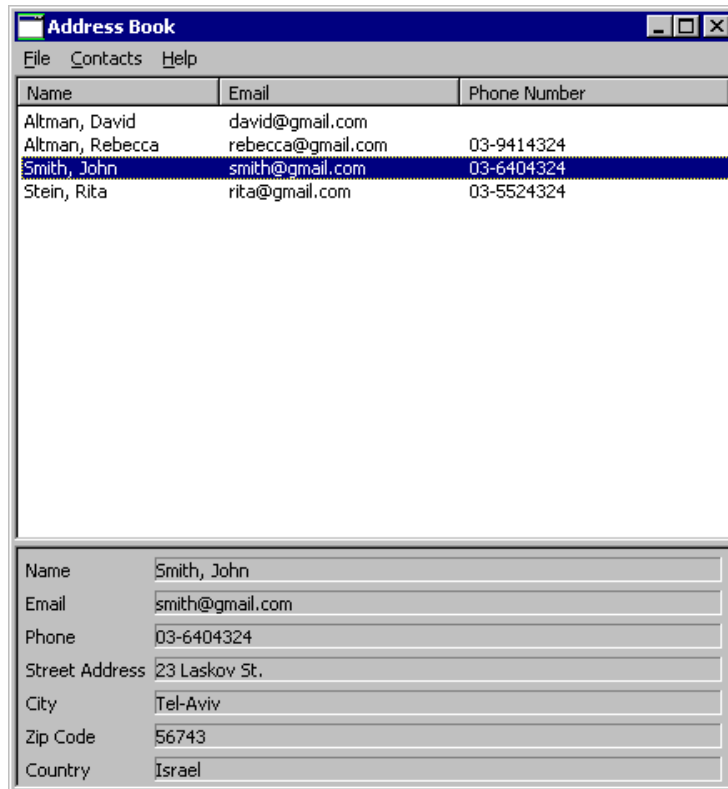
In assignment 9 we began implementing an address book system. In this assignment we are going to use the Standard Widget Toolkit (SWT) to implement a graphical user interface (GUI) for your address book. You should be able to use the classes from assignment 9 without modifications, only replacing the 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 address 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
- You can use the same instructions from Assignment05 for installing SWT for your project (HyperSudoku Question).

חלק 2: קלט / פלט

כתבו תכנית בשם SimpleShell הקוראת כקלט פקודות מהמשתמש ומבצעת אותן. התכנית תקרא בכל פעם פקודה מהמשתמש, תבצע אותה ותקרא את הפקודה הבאה וכן הלאה.

רשימת הפקודות:

הסבר	פקודה
<p>הדפסת פרטי קובץ / ספרייה. path הינו נתיב אל הקובץ / הספרייה. פלט:</p> <ul style="list-style-type: none"> אם path הינו נתיב לקובץ קיים יודפסו פרטי הקובץ בפורמט: <file name> File <read/write permission> <file size in bytes> אם path הינו נתיב לספרייה קיימת יודפסו פרטי הספרייה בפורמט: <dir name> Dir <read/write permission> <dir size in bytes> גודל מחיצה הוא סכום גדלי כל הקבצים שנמצאים בה (באופן ישיר תחתיה ובתתי ספריות). מחיצה שאינה מכילה כלל קבצים גודלה 0. בנוסף יודפסו פרטי הקבצים והמחיצות הנמצאים ישירות תחת מחיצה זו. 	<p>ls <path></p>
<p>העתקה של קובץ / ספרייה. path1 הינו נתיב לקובץ / ספרייה קיים/ת. path2 הינו נתיב לקובץ / ספרייה שאינו/ה קיים/ת.</p> <p>אם <path1> הינו נתיב לקובץ ייווצר קובץ זהה לו בנתיב המצוין ע"י <path2>. אם <path1> הינו נתיב לספרייה תיווצר ספרייה חדשה בנתיב המצוין ע"י <path2> ותכולתה של הספרייה המקורית תועתק לגוף הספרייה החדשה באופן רקורסיבי (אם הספרייה מכילה ספרייה אחרת, ספרייה זו תועתק אף היא על כל תכולתה).</p>	<p>cp <path1> <path2></p>
<p>יצירה של קובץ טקסט ע"י שרשרת של מספר קבצי Java.</p> <p>dir – הספרייה שבה נמצאים קבצי הג'אווה שנרצה לשרשר. הקבצים יכולים להימצא ישירות תחת ספרייה זו וגם בתתי-ספריות. file – פרמטר אופציונאלי, שם קובץ הטקסט אליו יש לשרשר את הפלט. במידה ולא צוין קובץ כזה הפלט יופנה ל standard output. עבור כל קובץ משורשר יודפס שם הקובץ (הנתיב המלא) לאחרי שורה רווח, תוכנו של הקובץ ושוב שורה רווח. בין קבצים תודפס שורה מפרידה המורכבת מרצף של 70 תווי '!'.</p>	<p>jcat <dir> [file]</p>
<p>יודפס bye והתוכנית תסיים את ריצתה.</p>	<p>Exit</p>

בכל שורת קלט תופיע פקודה בודדת.

עבור כל פקודה לא חוקית תודפס הודעה למשתמש המפרטת את רשימת הפקודות החוקיות והמבנה שלהן (אילו פרמטרים מקבלות).

במידה ומתרחשת שגיאה כלשהי במהלך הרצת פקודה (חוקית), כגון ארגומנט לא חוקי יש להדפיס הודעת שגיאה למשתמש, ידידותית ככל האפשר.

לדוגמה, אם בפקודה ls ה-path מצביע לקובץ שאינו קיים תודפס הודעת שגיאה כדוגמת

```
ls: cannot access <path>: no such file or directory
```

כאשר במקום <path> יופיע שם הקובץ אותו לא הצלחתם למצוא.

להלן דוגמא להרצת התוכנית SimpleShell:

```
Enter command: ls .
```

```
.                Dir          rw      8703
-----
.classpath       File         rw      226
.project         File         rw      379
ex8              Dir          rw      8098
```

```
Enter command: ls d:\temp\test
```

```
test            Dir          rw      10314330
-----
f1              File         rw      5148358
f128798798798172983 File        rw      17384
f2              File         rw      115
myDir           Dir          rw      5148473
```

```
Enter command: cp d:\temp\test .\test2
```

```
Enter command: ls .
```

```
.                Dir          rw      10323033
-----
.classpath       File         rw      226
.project         File         rw      379
ex8              Dir          rw      8098
test2            Dir          rw      10314330
```

```
Enter command: exit
```

```
Bye
```

הערות:

- שימו לב לדמיון הרב בין הפקודות cp ו-jcat. הימנעו ככל האפשר משכפול קוד במימוש של הפקודות הנ"ל.
- כדי להדפיס את הפלט בטבלה מסודרת השתמשו ב 't' (tab) כדי לייצר עמודות.
- על התוכנית להשתמש ב- File.pathSeparator על מנת להתאים למערכות הפעלה שונות.
- בשביל הרשאות כתיבה/קריאה (rw) השתמשו במתודות המובנות ב- File (canRead, canWrite), ויש להציג הרשאות אלו באופן המוצג בדוגמא הנ"ל.