To the 1

Problem 9

General Instructions:

1. Read carefully the submission guidelines found on the course site (http://virtual2002.tau.ac.il/).
2. Submit your problem through the VirtualTAU system (http://virtual2002.tau.ac.il/).
3. Submit a single .zip file containing your user name (for example, for user zvainer, submit the file zvainer.zip).
4. The .zip file should contain:
   a. A text file named details.txt containing your name and ID number.
   b. The java files of the programs you submitted.
   c. A text file containing backups of all .java files.
   d. A text file named answers containing your answers for part A.

Part A: Project Submission

In the following exercises, “to compute” means to calculate or perform a calculation.

Details:

1. Compute the output of the following code:

```java
SalariedEmployee e1 = new SalariedEmployee("John", "Lennon", 1, 1000);
HourlyEmployee e3 = new HourlyEmployee("George", "Harrison", 3, 10, 30);
CommissionEmployee e5 = new CommissionEmployee("Ringo", "Starr", 5, 0.1d, 200);
BasePlusCommissionEmployee e7 = new BasePlusCommissionEmployee("Paul", "McCartney", 7, 0.1d, 100, 500);
```

2. Compute the output of the following code:

```java
Employee e = new Employee("John", "Lennon", 1, 1000);
System.out.println(e.toString());
```

3. Compute the output of the following code:

```java
Employee e = new Employee("John", "Lennon", 1, 1000);
System.out.println(e.getEarnings());
```

4. Compute the output of the following code:

```java
Employee e = new Employee("John", "Lennon", 1, 1000);
System.out.println(e.toString());
```

5. Compute the output of the following code:

```java
Employee e = new Employee("John", "Lennon", 1, 1000);
System.out.println(e.getEarnings());
```

6. Compute the output of the following code:

```java
Employee e = new Employee("John", "Lennon", 1, 1000);
System.out.println(e.toString());
```

7. Compute the output of the following code:

```java
Employee e = new Employee("John", "Lennon", 1, 1000);
System.out.println(e.getEarnings());
```
יש לממש את המתודה getEarnings לפי ההסבר על סוגי טיפוסי העובדים השונים. בווסף, הטבלה הבאה מתארת את פרטיהם המוחזנים.getMethod מתודותיה של 클래스, המוחזשות זו קבוצת לשתי משכורות תועורת ומיה時点で מחזירה את הפרטים פודרט—theyカフェ על רוחות (String.format(.

<table>
<thead>
<tr>
<th>Class</th>
<th>toString</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>first-name last-name</td>
</tr>
<tr>
<td></td>
<td>ID: identifier</td>
</tr>
<tr>
<td>SalariedEmployee</td>
<td>salaried employee: first-name last-name</td>
</tr>
<tr>
<td></td>
<td>ID: identifier</td>
</tr>
<tr>
<td></td>
<td>weekly salary: weekly-salary</td>
</tr>
<tr>
<td>HourlyEmployee</td>
<td>hourly employee: first-name last-name</td>
</tr>
<tr>
<td></td>
<td>ID: identifier</td>
</tr>
<tr>
<td></td>
<td>hourly wage: wage; hours worked: hours</td>
</tr>
<tr>
<td>CommissionEmployee</td>
<td>commission employee: first-name last-name</td>
</tr>
<tr>
<td></td>
<td>ID: identifier</td>
</tr>
<tr>
<td></td>
<td>gross sales: sales; commission rate: rate</td>
</tr>
<tr>
<td>BasePlusCommissionEmployee</td>
<td>base salaried commission employee: first-name last-name</td>
</tr>
<tr>
<td></td>
<td>ID: identifier</td>
</tr>
<tr>
<td></td>
<td>gross sales: sales; commission rate: rate</td>
</tr>
<tr>
<td></td>
<td>base salary: salary</td>
</tr>
</tbody>
</table>

BasePlusCommissionEmployee ממית את המחלקה משכורת של arbe hand מחלבות זה המרוכב מתוצרת בelta קבアウト עם סממע שיכורה. כמי במחלקה הקורמות גם את המפרטים.getenvים את המשכירות העבירי בנהאי כל ניצי היח.

齑 במקורה של השינתיו ייבושה על просмотрشفיל כרד.

<table>
<thead>
<tr>
<th>Class</th>
<th>toString</th>
</tr>
</thead>
<tbody>
<tr>
<td>first-name last-name</td>
<td></td>
</tr>
<tr>
<td>ID: identifier</td>
<td></td>
</tr>
<tr>
<td>weekly salary: weekly-salary</td>
<td></td>
</tr>
<tr>
<td>hourly wage: wage; hours worked: hours</td>
<td></td>
</tr>
<tr>
<td>commission employee: first-name last-name</td>
<td></td>
</tr>
<tr>
<td>ID: identifier</td>
<td></td>
</tr>
<tr>
<td>gross sales: sales; commission rate: rate</td>
<td></td>
</tr>
<tr>
<td>base salary: salary</td>
<td></td>
</tr>
</tbody>
</table>


 smokey 1 - ביפוסרצbrities ערך העובדים לפי שינה. smokey 2 - את המחלבות הלוחות ולעובר את המשכורות לעובדים המתחדשים לפי שינה ב 10%. smokey 3 - כמות arbe hand משכורות קובע בין שהאלעה לש 5% לרוב סייך. smokey 4 - את המחלבות הלוחות ולעובר את המשכורות לעובדים שמטؤمنות כי כי רכיבים מחל בוחת. arbe hand, מים arbear שווין עודמה על.travel 5% לאזר arbear שווין arbear עודמה על.travel
This is the first part of an address book application assignment. In this part of the assignment you will write a class for maintaining an 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. This assignment consists of two parts. In the first part you will implement the core address book functionality (adding/removing contacts etc.). The second part will provide a console based interface for the address book you implemented in part 1. 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 do not require changing the underlying model and it enables us to maintain one model for several different views.

Part 1 – The Model

In this part you will implement the core address book functionality. We provide you the IAddressBook interface as well as the Contact and Address classes. You are required to implement the class AddressBook which implements the IAddressBook interface.

The class AddressBook is a collection of contacts easily accessible by a contact's name, and sorted in alphabetical order (case insensitive).

Remarks:
- Names equality is case insensitive. For example, “Doe, John” is equal to “doe, john”. However, while operations are performed in a case insensitive manner you should display names in their original form as provided by the user.
- The required methods are under specified, e.g. what should happen if you call update() for a contact that doesn’t exists? You should decide how to handle such cases and specify the method contract accordingly.
- Use the standard collections framework (sets, maps, list etc.) for the underlying structure of the class.
- 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 – Textual View

In this part you will implement a simple textual user interface for the address book. You will implement the class TextualAddressBookView.

The class defines the public method show(), this is the entry point to the viewer. Calling the show method will put the view in an interactive mode. In this mode the view reads a command from the user and executes it until the command `exit' is encountered.

The class Main (also supplied) creates a new view and calls the show method on it.

Hint: you need to use "System.in" which we saw at recitation 05.

The application will support the following commands:

- c
  create a new address book

- a <name>;<email>;<telephone>;<street>;<city>;<zip>;<country>;  
  add a new contact to the address book. Note that all fields must be provided on the command line. While the name field is mandatory other fields are optional fields and may have the empty string as their value.

- p <name prefix>  
  print to the standard output all contacts for which the name field starts with the given prefix (case insensitive)

- x  
  print all the contacts in the address book

- d <name>  
  delete the specified contact

- u <name>;<email>;<telephone>;<street>;<city>;<zip>;<country>;  
  update an existing contact with the new information.

- e  
  exit

Here is an example session:

> c
> 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
Name: Altman, David
Email: david@gmail.com

Name: Altman, Rebecca
Email: rebecca@gmail.com
Tel: 03-9414324
Address: Tel-Aviv, 42732
Israel
> p Altman, Rebecca
Name: Altman, Rebecca
Email: rebecca@gmail.com
Tel: 03-9414324
Address: Tel-Aviv, 42732
Israel

> x
Name: Altman, David
Email: david@gmail.com

Name: Altman, Rebecca
Email: rebecca@gmail.com
Tel: 03-9414324
Address: Tel-Aviv, 42732
Israel

Name: Smith, John
Email: smith@gmail.com
Tel: 03-6404324
Address: 23 Laskov St.
Tel-Aviv, 56743
Israel

Name: Stein, Rita
Email: rita@gmail.com
Tel: 03-5524324

> d Stein, Rita
> u Altman, David; david@gmail.com; 03-9414324;;;
> e

Note that the first line should be 'c'.