

תוכנה 1

תרגיל מספר 9

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 יכול:
 - 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 א': מערכת תשלומים

בחברת המחשבים "תוכנה להמונים" העובדים מקבלים שכר על בסיס שבועי. ישנם ארבעה סוגי עובדים:

1. עובדים המקבלים משכורת שבועית ללא קשר למספר השעות שעבדו (SalariedEmployee)
2. עובדים המקבלים שכר לפי שעה (HourlyEmployee). עובדים אלו מקבלים שכר לכל שעת עבודה וכן תשלום שעות נוספות לכל שעת עבודה מעבר ל-40 שעות שבועיות. שכר שעה נוספת הוא פי 1.5 משכר של שעה רגילה.
3. עובדי המקבלים עמלת מכירות (CommissonEmployee). עובדים אלו מקבלים שכר כאחוז מהמכירות שלהם.
4. עובדים שמשכורתם מורכבת משכר בסיס ועמלות (BasePlusCommissionEmployee)

משימה

- א. ממשו את המחלקה האבסטרקטית Employee. לכל עובד יש שם פרטי, שם משפחה ומספר מזהה. המועברים כערכים בבנאי ואינם ניתנים לשינוי לאחר מכן. בנוסף, המחלקה מגדירה את השירותים getEarnings() ו- toString(). הראשון מחזיר את המשכורת השבועית של העובד (מספר ממשי), ואילו השני דורס את השירות מהמחלקה Object ומחזיר מחרוזת המתארת את האובייקט (ראו פירוט בהמשך). לנוחותכם תוכלו למצוא את שלד המחלקה באתר הקורס.
- ב. ממשו את המחלקות SalariedEmployee, HourlyEmployee ו-CommissionEmployee. השתמשו בירושה כדי לא לשכפל קוד. הפרמטרים הקובעים את המשכורת יועברו בבנאי של המחלקה וכן ניתן יהיה לשנותם בהמשך, במידת הצורך.
דוגמא:

```
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);
```

יש לממש את המתודה `getEarnings` לפי ההסבר על סוגי טיפוסים העובדים השונים. בנוסף, הטבלה הבאה מתארת את פורמט המחרוזת המוחזרת מהמתודה `toString`. שימו לב, הטקסט המודגש הוא קבוע. יש להקפיד שהמחרוזת המוחזרת מהשירות תהיה תואמת לפורמט תוך הקפדה על רווחים ושורות. (היעזרו במתודה `String.format`)

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

ג. ממשו את המחלקה `BasePlusCommissionEmployee`. משכורתו של עובד מטיפוס זה מורכבת ממשכורת בסיס קבועה וגם מעמלת מכירות. כמו במחלקות הקודמות גם כאן הפרמטרים הקובעים את המשכורת יועברו בבנאי וכן ניתן יהיה לשנותם בהמשך. גם במקרה זה השתמשו בירושה על מנת למנוע שכפול קוד.

ד. ממשו את הפונקציה `printPayrollReport` שבמחלקה `PayrollSystemTest`. הפונקציה מקבלת רשימה של עובדים ועבור כל אחד מהם מדפיסה את פרטיו וכן את משכורתו. דוגמת פלט:

```
salaried employee: John Smith
ID: 111-11-1111
weekly salary: $800.00
earned $800.00
```

```
hourly employee: Karen Price
ID: 222-22-2222
hourly wage: $16.75; hours worked: 40.00
earned $670.00
```

ה. ממשו תכנית בדיקה קטנה במחלקה `PayrollSystemTest`. התכנית תגדיר לפחות שני עובדים מכל סוג ותדפיס את דו"ח המשכורות עבור ארבעה שבועות:
שבוע 1 - יודפס הדו"ח עבור העובדים כפי שהוגדרו.
שבוע 2 - הנהלת החברה החליטה להעלות את המשכורת לעובדים המשתכרים לפי שעה ב 10%.
שבוע 3 - כל העובדים שיש להם רכיב משכורת קבוע קיבלו העלאה של 5% לרכיב שכר זה.
שבוע 4 - הוחלט להעלות את העמלות המשולמות לעובדים שבמשכורתם קיים רכיב עמלה באחוז. לדוגמה, אם טרם השינוי העמלה עמדה על 5% לאחר השינוי היא תעמוד על 6%.

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 exist? 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'.