תוכנה 1
תרגיל מספר 10

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 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.
להלן הגדרת הבנאים:

```
public SalariedEmployee(String firstName, String lastName, int id, double weeklySalary)
public HourlyEmployee(String firstName, String lastName, int id, double hourlyWage, int hours)
public CommissionEmployee(String firstName, String lastName, int id, double commissionRate, double
totalSales)
public BasePlusCommissionEmployee(String firstName, String lastName, int id, double
commissionRate, double totalSales, double baseWeeklySalary)
```

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 טיפוסי 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	*first-name last-name* **ID:** *identifier*
SalariedEmployee	**salaried employee:** *first-name last-name* **ID:** *identifier* **weekly salary: $***weekly-salary*
HourlyEmployee	**hourly employee:** *first-name last-name* **ID:** *identifier* **hourly wage: $***wage***; hours worked:** *hours*
CommissionEmployee	**commission employee:** *first-name last-name* **ID:** *identifier* **gross sales:** *sales***;** **commission rate:** *rate*
BasePlusCommissionEmployee	**base salaried commission employee:** *first-name last-name* **ID:** *identifier* **gross sales:** *sales***;** **commission rate:** *rate***;** **base salary: $***salary*

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: 1111
weekly salary: $800.00
earned $800.00

hourly employee: Karen Price
ID: 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%.

עליכם להשתמש במתודה הבאה עבור יצירת העובדים השונים:

```
private static void createEmployees(List<Employee> employees)
```

על המתודה לקבל רשימה ריקה כארגומנט ולהכניס לתוכה את סוגי העובדים השונים.
עליכם להשתמש ברשימה זו בתוכנית הבדיקה שלכם.

שאלות ותשובות:

1. נאמר להשתמש ב -String.format . למה ? לדאוג שיהיו 2 ספרות אחרי הנקודה ? מתי בדיוק נצטרך להשתמש בה ? במידה וכן אפשר להשתמש גם ב -DecimalFormat?

תשובה:

אין חובה להשתמש ב- String.Format אך מומלץ להשתמש בו משתי סיבות:

א. זה מאוד מפשט את הקוד ומאפשר בקלות להדפיס את הפורמט הנדרש.
ב. מומלץ להכיר את השימוש בו וזאת הזדמנות טובה לכך.

2. בסעיף ד' נאמר שמקבלים רשימה של עובדים. האם הכוונה למערך או ל –List ?

תשובה:

הכוונה לרשימה List. בנוסף רצוי (לא חובה) לממש מתודה נוספת שתעבוד על מערך (לא מוגדר בגודל) ולהוסיף אותה בעזרת העמסת שירותים.

חלק ב׳: ספר כתובות

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`'.

<u>שאלות ותשובות</u> :

1. הבנתי מהדוגמאות ששדה הכתובת לא חובה, וראיתי מהדוגמא מה קורה כאשר אין רחוב לדוגמא, אבל מה קורה אם אין מיקוד? או אין עיר? האם עליי לבדוק כל אחד מהם או שאפשר להניח שנגיד עיר ומיקוד באים ביחד ואין מצב שאחד יהיה בלי השני?

תשובה: אם זה לא מפורט בתרגיל, אתם יכולים להניח מה שנראה לכם הכי הגיוני ופשוט לציין למה הנחתם זאת (להוסיף הסבר קצר).

2. כאשר אני ממש את הפונקציה search שמחזירה Iterable של Contact, האם עליי לשכפל כל Contact כזה שתהיה הפרדה בין מבני הנתונים?

תשובה: זאת החלטה שלכם. שימו לב, בג'אווה נהוג שמתודה לא תחזיר הפניה לאובייקט עצמו (שהתקבל כארגומנט) אלא להעתק של האובייקט, אלא אם נדרש ספציפית שתוחזר הפנייה לאובייקט עצמו.

3. בחלק השני כאשר יש קלטים לא תקינים, אני מדפיס שגיאה למסך. האם זה בסדר או שיש איזושהי דרך מסוימת שיש להחזיר שגיאות?

תשובה: אם לא מוגדר בתרגיל, אתם רשאים להחליט איך עליך לדווח על שגיאות. עליכם לקחת בחשבון את סוג המנשק שלכם עם המשתמש בתוכנית שאתם כותבים. למשל אם בחלק השני אתם כותבים תוכנית אינטראקטיבית (כלומר שהמשתמש מכניס קלט ומקבל פלט בחזרה), הגיוני שהודעת השגיאה תופיע גם בצורה זו על מנת שהמשתמש יבין שהוא ביצע פעולה בצורה לא חוקית. לעומת זאת, אם הייתם נדרשים לכתוב מנשק משתמש גרפי אזי היה יותר נכון להקפיץ חלון עם הודעת שגיאה למשתמש.

חלק ג': Sorted Set Iterator

בחלק זה הנכם מתבקשים לממש מחלקות שונות המגדירות פעולות בינאריות על קבוצות, כדוגמת פעולות האיחוד, חיתוך, הפרש סימטרי וכדומה. לשם כך הגדרנו שני מנשקים - SortedSet ו- Iterator –המגדירים קבוצות מסודרות ואיטרטור על הקבוצות. שימו לב, אלו **אינם** המנשקים הנושאים שם זהה הקיימים ב JDK. ראו את התיעוד המלא של המנשקים בקוד המופיע באתר.

```
package il.ac.tau.cs.sw1.sortedset;

/**
 * A Set that provides a total ordering on its elements.
 * All elements must implement the Comparable interface.
 */
public interface SortedSet<T extends Comparable<T>> {

	/**
	 * Return true if this set contains the specified element.
	 */
	public boolean contains(T element);

	/**
	 * Return the maximal (highest) element in this set.
	 */
	public T getMaximum();

	/**
	 * Return the minimal (lowest) element in this set.
	 */
	public T getMinimum();

	/**
	 * Return true if this set contains no elements.
	 */
	public boolean isEmpty();

	/**
	 * Return an iterator over the elements of this set.
	 * The iterator traverses the elements in an ascending
	 * order.
	 */
	public Iterator<T> iterator();

	/**
	 * Return the number of elements in this set (its
	 * cardinality).
	 */
	public int size();

}
```

```java
package il.ac.tau.cs.sw1.sortedset;

/**
 * An iterator over a collection of elements.
 */
public interface Iterator<T> {
	/**
	 * Advances the iterator to the next element.
	 */
	void advance();

	/**
	 * Returns true if the iterator has reached the end of
	 * the collections.
	 */
	boolean atEnd();

	/**
	 * Returns the current element in the iteration.
	 */
	T get();
}
```

האיטרטור הנתון פועל באופן מעט שונה מזה המוכר לנו ב Java, במקום לשאול האם קיים עוד איבר באוסף נברר האם הגענו לסוף. כמו כן, הפונקציה advance המקדמת את האיטרטור **אינה** מחזירה ערך. כדי לקבל את האיבר עליו מצביע האיטרטור נשתמש בפונקציה get, שימוש בפונקציה זו אינו מקדם את האיטרטור. נתון קטע קוד קצר המדגים שימוש באיטרטור החדש.

```java
for (Iterator<T> iter = ...; !iter.atEnd(); iter.advance()) {
    T element = iter.get();
}
```

<u>פעולות על קבוצות:</u>

הינכם נדרשים לממש את ארבע הפעולות הבינאריות של איחוד, חיתוך הפרש והפרש סימטרי כפי שיוגדרו להלן. תוצאת כל הפעולות היא בעצמה קבוצה ממוינת.

- **איחוד** – בהינתן שתי קבוצות A ו-B קבוצת האיחוד היא

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

- **חיתוך** - בהינתן שתי קבוצות A ו-B קבוצת החיתוך היא

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

- **הפרש**- בהינתן שתי קבוצות A ו-B ההפרש של A מ-B הוא קבוצת כל האברים שנמצאים ב-A ולא נמצאים ב-B

$$A \backslash B = \{x | x \in A \wedge x \notin B\}$$

- **הפרש סימטרי**- בהינתן שתי קבוצות A ו-B ההפרש הסימטרי הוא האיחוד של קבוצות ההפרשים (A מ-B ו-B מ-A)

$$A \ominus B = (A \backslash B) \cup (B \backslash A)$$

<u>דרישות המימוש:</u>

המחלקות השונות שהנכם נדרשים לממש ייצגו את תוצאת הפעולות השונות על קבוצות הקלט. קבוצת התוצאה, כמו גם קבוצות הקלט עצמן אינן ניתנות לשינוי לאחר שנוצרו (immutable).

לא ניתן לאחסן את הערך null בקבוצות הנ"ל. ניסיון להוסיף ערך זה לקבוצה (בעת יצירתה) יגרום לזריקת חריג מטיפוס IllegalArgumentException.

את המחלקות הינכם מתבקשים לממש בשני אופנים שונים, גישה "עצלה" (lazy) וגישה "להוטה" (eager).

גישה "להוטה" – eager:
בגישה זו, בעת יצירת אובייקט של אחת המחלקות המממשות פעולה בינארית על קבוצות (איחוד, חיתוך הפרש או הפרש סימטרי) נחשב את תוצאת הפעולה באופן מיידי ונשמור אותה. מכאן ואילך כל השירותים המוגדרים במנשק יפעלו על אובייקט התוצאה (עליכם לבחור כיצד לייצג אותו).

גישה ב' – lazy:
בגישה זו לא נבצע חישוב בטרם נדרש לו, למשל לא נחשב את איבר הבא בתוצאה בטרם נצרך לקדם את האיטרטור, ובכל מקרה בשום שלב לא נחשב את תוצאת הפעולה בכללותה. בעת יצירת האובייקט נשמור הפניות לשתי קבוצות הקלט (מובטח לנו שהן לא ישתנו בהמשך בזכות ה-immutability) אך לא נבצע כל חישוב. בכל קריאה לשירות נחשב בעזרת האיטרטורים של קבוצות הקלט את המינימום הנדרש לשירות.
עבור השירותים size, getMinimum ו-getMaximum נחשב את תוצאתם בפעם הראשונה שנדרש לכך ונשמור תוצאה זו כדי שלא נאלץ לחזור על החישוב בפעמים הבאות שנדרש לכך (memorization). מכיוון שהקבוצות אינן ניתנות לשינוי מובטח לנו שתוצאת החישוב לא תשתנה. בכל אחת מהמחלקות נממש איטרטור חכם שכאשר נקדם אותו הוא ייתן את האשליה שהוא מתקדם על קבוצת התוצאה ולא על קבוצות הקלט. את האיטרטורים יש לממש כמחלקות פנימיות של מחלקות הקבוצה השונות. שימו לב שאם יש שכפול קוד בין האיטרטורים עליכם להעבירו למחלקה משותפת.

הינכם נדרשים לממש את המחלקות הבאות (ראו שלד פתרון באתר):

א. **UnionSortedSet** – מחלקה זו מייצגת את תוצר פעולת האיחוד בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים באיחוד של A ו-B.

ב. **IntersectionSortedSet** – מחלקה זו מייצגת את תוצר פעולת החיתוך בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים בחיתוך של A ו-B.

ג. **DifferenceSortedSet**– מחלקה זו מייצגת את תוצר פעולת ההפרד בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים בהפרש של A ו-B (A\B).

ד. **SymetricDifferenceSortedSet** – מחלקה זו מייצגת את תוצר פעולת ההפרש הסימטרי בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים שנמצאים בהפרש הסימטרי של A ו-B

בנוסף למחלקות הקונקרטיות הללו הינכם נדרשים לממש את המחלקה האבסטרקטית BinaryOpSortedSet. מחלקה מופשטת זו מייצגת קבוצה ממוינת שהיא תוצר של פעולה בינארית על

קבוצות ממוינות. יש לממש במחלקה זו פונקציונאליות רבה ככל שניתן (כל הפונקציות שלא קשורות במימוש הפעולה הבינארית הספציפית). השירותים אותם הינכם נדרשים לממש **באופן מלא** במחלקה זו הינם:

- `final public int size();`
- `final public boolean isEmpty();`
- `final public T getMinimum();`
- `final public T getMaximum();`

ייתכנו כמובן שירותים ושדות נוספים במחלקה זו. בנוסף, במידה ותתקלו בצורך לשתף קוד בין מחלקות האיטרטורים השונות המקום הטבעי הוא במחלקה פנימית אבסטרקטית במחלקה BinaryOpSortedSet.
כפי שצוין לעיל הנכם נדרשים לספק שני מימושים שונים למחלקות אלו. מימושים אלו יהיו בלתי תלויים זה בזה **ואין לשתף בינהם קוד**, גם לא דרך מחלקות אבסטרקטיות משותפות.

<u>הערות:</u>

את הפרוייקט sortedset תמצאו ב-resources.zip באתר. ייבאו אותו לאקליפס (File->Import…).

בקבצים שסיפקנו לכם תמצאו את המחלקה SimpleSortedSet המממשת את המנשק SortedSet. מחלקה זו תשמש לכם כדוגמה בלבד ולצרכי בדיקה של הקוד, אין לעשות בה שימוש במימוש שלכם. כלומר כדי לבדוק שהפעולות הבינאריות שמימשתם נכונות תוכלו להשתמש במחלקה זו, אך אין לעשות בה שימוש במימוש הפנימי של המחלקות השונות.

יש להגיש את קבצי ה-Java כפי שצורפו לפרוייקט: שתי ספריות lazy ו-eager שמכילות את קבצי הקוד