תוכנה 1

תרגיל מספר 7

מנשקים Interfaces

הנחיות כלליות:
קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

- הגשת התרגיל תעשה במערכת ה-moodle בלבד (http://moodle.tau.ac.il/).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש aviv יקרא הקובץ aviv_hw7.zip). קובץ ה-zip יכיל:
  א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
  ב. קבצי ה- java של התוכניות אותם התבקשתם לממש, כולל תיקיות החבילה.

הגשת מחלקה עם חבילות: יש לכווץ בתוך קובץ ה zip-שאתם מגישים את כל היררכיית התיקיות מתחת ל src-באקליפס. למשל, כדי להגיש את המחלקה sw1.pac.MyClass העתיקו את התיקיה sw1 שמתחת ל src-כולל כל מה שבתוכה לתוך קובץ ה-zip.

חלק א' (50 נק')

בחלק זה נתרגל כתיבת מחלקות הממשות מנשק נתון.

נתונים המנשקים `Predictor`,`PredicatesSet`, והמחלקות: StartWith, Predicates, Big, Even,

עליכם להשלים את המחלקות על פי המתודות המתוארות במנשק אותו הן מממשות

המחלקות Big,Even יבדקו האם הפרדיקט מתקיים על שדה הגיל של כל Person, והפרדיקט StartWith יבדוק האם הפרדיקט מתקיים על שדה השם. הפרדיקטורים Big ו-StartWith מקבלים ארגומנט נוסף, מספר ותו בהתאמה אשר אליו הם משווים את השדה המתאים. תיאור הפעולות מופיע בתיעוד הקוד.

- כל המחלקות בחלק זה ימומשו כחלק מחבילה בשם il.ac.tau.cs.software1.predicate
- על כל מחלקה להכיל שדות פרטיים אליהם ניתן לגשת עם מתודות getter ו- setter ציבוריות.
- שימו לב כי כאשר משתשמשים ב-System.in, יש לפתוח את הזרם ולסגור אותו פעם אחת בלבד במהלך התכנית כולה.

1. עתה נעבור לכתיבת התוכנית (המחלקה) **Main** שתעשה שימוש בטיפוסים שיצרתם בסעיף הקודם. התוכנית תקבל מהמשתמש (דרך ה- `System.in`) סדרה של Person ותפעיל על הרשימה פרידקטורים לבחירתו.

נתחיל בכתיבת המתודה `getPersonFromUser()`

המתודה תדפיס למסך תפריט שיאפשר למשתמש להגדיר כמה אנשים הוא מעוניין להכניס ותקלוט את הנתונים המתאימים דרך המקלדת, לבסוף תחזיר רשימה של אנשים על פי הקלט שנקלט מן המשתמש, על פי סדר ההכנסה.

דוגמא לתוכן חלון ה-Console בגמר הרצת המתודה (קלט המשתמש מופיע בירוק):

```
Please choose number of persons (or X for exit):
A
Unknown command. Please try again.
Please choose number of persons (or X for exit):
2
Please enter age: 13
Please enter name: Avi
Please enter age: 22
Please enter name: Yael
Your list is: [(Avi,13),(Yael,22)]
```

<u>עליכם לממש את המתודה על פי דוגמת הפלט המופיע לעיל.</u>

<u>הערות:</u>

- ריצת תכנית תסתיים אם המשתמש בחר באפשרות היציאה בתפריט (הקיש על 'X').
- ניתן להניח שהמשתמש מכניס קלט חוקי (מחרוזות או מספרים שלמים כנדרש), אך אם הוקש בהוראה הראשונה מספר לא חוקי או תו שונה מ-X, יש להדפיס למסך " Unknown command. Please try again." ולהדפיס מחדש את התפריט.
- קלט חוקי עבור מספר אנשים הינו מספר חיובי ושלם גדול מ-0
- המתודה לבסוף תדפיס את הרשימה שהתקבלה על פי הפורמט שהוצג למעלה.

2. כעת נעבור למימוש המתודה `getPredictorFromUsers()`.
המתודה תציג תפריט למשתמש אשר תבקש ממנו לבחור איזה פרדיקטור ברצונו להפעיל על רשימת האנשים שהוזנה – ובמידה והפרדיקטור דורש ארגומנט נוסף (כמו למשל startWith) תבקש את הארגומנט המתאים ליצירתו. המתודה לבסוף תחזיר את ה-predictor הרצוי.

דוגמא לתוכן חלון ה-Console בגמר הרצת המתודה (קלט המשתמש מופיע בירוק):

עליכם לממש את המתודה על פי דוגמת הפלט הבאה:

```
Please choose Predictor:
E – Even
B – Big
S – StartWith
K
Unknown command. Please try again.
Please choose Predictor:
E – Even
B – Big
S – StartWith
B
Please enter a number to compare with:
14
```

הערות:

- ניתן להניח שהמשתמש מכניס קלט חוקי (תו או מספרים שלמים כנדרש), אך אם הוקש בהוראה הראשונה תו לא חוקי, יש להדפיס למסך "Unknown command. Please try again." ולהדפיס מחדש את התפריט.
- קלט חוקי עבור גיל הינו מספר חיובי ושלם גדול מ-0
- קלט חוקי עבור StartWith הינו תו באנגלית (a-z,A-Z)
- במידה והמשתמש בחר באופציה S, יש להציג ולקלוט ממנו תו באופן הבא:
  Please enter a char to compare:

3. כעת נעבור למימוש המתודה `apply()`.

המתודה תציג תפריט למשתמש אשר תשאל אותו האם ברצונו להפעיל את הפרדיקטור על הרשימה. במידה וכן, המתודה תפעיל את הפרדיקטור שהוכנס, תדפיס את הרשימה שהתקבלה ותשאל אותו האם ברצונו להכניס פרידקטור נוסף – במידה וכן, הלולאה תמשיך עד אשר יבחר המשתמש להכניס רשימה חדשה.

דוגמא לתוכן חלון ה-Console בגמר הרצת המתודה (קלט המשתמש מופיע בירוק):

עליכם לממש את המתודה על פי דוגמת הפלט הבאה:

```
What would you like to do:
R – Remove
E – Retrain
C – Collect
F – Find
N – Insert a new list
K
Unknown command. Please try again.
What would you like to do:
R – Remove
E – Retrain
C – Collect
F - Find
N – Insert a new list
R
The result is: [(Avi,13)]
What would you like to do:
R – Remove
E – Retrain
C – Collect
F - Find
N – Insert a new list
N
Please choose number of persons (or X for exit):
X
```

הערות:

- אם הוקש בהוראה הראשונה תו לא חוקי, יש להדפיס למסך " Unknown command. Please try again." ולהדפיס מחדש את התפריט.
- אם הפעלת הפרדיקטור לא שינתה את הרשימה, יש להדפיס את הרשימה כפי שהייתה ללא שינוי
- במידה והמשתמש בחר להכניס רשימה חדשה – התכנית תמשיך לרוץ כפי שתואר בסעיף 1.
- תיאור המתודות הנ"ל מופיע בתיעוד הקוד.

4. כעת נעבור למימוש מתודת ()run. המתודה למעשה תפעיל את התכנית, עד אשר המשתמש בוחר לצאת מן התכנית (כלומר בשלב הכנסת הרשימה, בוחר בתו 'X').
המתודה ראשית מבקשת מהמשתמש להכניס רשימה של אנשים (על ידי קריאה למתודה `getPersonFromUser()`), לאחר מכן מבקשת לבחור את הפרדיקטור הרצוי (על ידי קריאה למתודה `getPredictorFromUsers()`), שואלת את המשתמש באיזה פעולה הוא בוחר (על ידי קריאה למתודה ()apply) וחוזר חלילה. ריצת התכנית תסתיים רק כאשר המשתמש בוחר לצאת – וזאת ניתן לעשות רק מהתפריט בו המשתמש מכניס רשימה חדשה.

ניתן לכתוב פונקציית ()main אשר בודקת את ריצת התכנית שלכם בתוך המחלקה Main. כדאי וצריך לבדוק את ריצת התכנית שלכם על מספר דוגמאות.
בחלק זה עליכם להגיש את כל המחלקות והמנשקים תחת החבילה il.ac.tau.cs.software1.predicate.

דוגמא לריצת תוכנית מלאה:

```
Please choose number of persons (or X for exit):
2
Please enter age: 13
Please enter name: Avi
Please enter age: 22
Please enter name: Yael
Your list is: [(Avi,13),(Yael,22)]
Please choose Predictor:
E – Even
B – Big
S – StartWith
S
Please enter a char to compare with:
B
What would you like to do:
R – Remove
E – Retrain
C – Collect
F - Find
N – Insert a new list
E
The result is: []
What would you like to do:
R – Remove
E – Retrain
C – Collect
F - Find
N – Insert a new list
```

```
F
The result is: -1
What would you like to do:
R – Remove
E – Retrain
C – Collect
F - Find
N – Insert a new list
N
Please choose number of persons (or X for exit):
X
```

חלק ב' (50 נק') – כתובת IP

המנשק IPAddress המופיע למטה מייצג כתובת של Internet Protocol (IP). דוגמאות לכתובות IP הן:

127.0.0.1
192.168.1.10

כתובת IP, כפי שניתן לראות, מורכבת **מארבעה** חלקים. ערכו של כל אחד מהחלקים הוא מספר שלם בין 0 ל-255. בסעיף זה נממש את המנשק IPAddress על ידי שלושה ייצוגים שונים: הראשון עושה שימוש במחרוזות, השני במערך של מספרים מסוג short ואילו השלישי משתמש ב-int יחיד (הסבר מפורט בהמשך).

א. כתבו **שלוש** מחלקות שונות המממשות את המנשק IPAddress המוגדר למטה:
1. מחלקה בשם IPAddressString, המממשת את המנשק בעזרת ייצוג פנימי של String.
2. מחלקה בשם IPAddressShort, המממשת את המנשק בעזרת ייצוג פנימי של מערך בגודל 4 של short. כל תא במערך יחזיק מספר בתחום 0..255.
3. מחלקה בשם IPAddressInt, ה מממשת את המנשק בעזרת int יחיד.

לכל אחת מהמחלקות יהיה בנאי המתאים לייצוג הפנימי שלה וכמובן כל אחת מהן מממשת את המנשק. ניתן להניח בחוזה שהבנאים מקבלים קלט תקין ליצירת כתובת IP (בהתאם לייצוג הפנימי של כל מחלקה).

```
public interface IPAddress {

	/**
	 * Returns a string representation of the IP address, e.g.
	 * "192.168.0.1"
	 */
	public String toString();

	/**
	 * Compares this IPAddress to the specified object
	 * @param other
	 *            the IPAddress to compare the current against
```

```
     * @return true if both IPAddress objects represent the same
     * IP address, false otherwise.
     */
    public boolean equals(IPAddress other);

    /**
     * Returns one of the four parts of the IP address. The parts
     * are indexed from left to right. For example, in the IP
     * address 192.168.0.1 part 0 is 192, part 1 is 168,
     * part 2 is 0 and part 3 is 1.
     * (Each part is called an octet as its representation
     * requires 8 bits.)
     * @param index
     *            The index of the IP address part (0, 1, 2 or 3)
     * @return The value of the specified part.
     */
    public int getOctet(int index);




    /**
    * There are four classes of private networks
    * (http://en.wikipedia.org/wiki/IPv4#Private_networks)
    * 10.0.0.0 - 10.255.255.255
    * 172.16.0.0 - 172.31.255.255
    * 192.168.0.0 - 192.168.255.255
    * 169.254.0.0 - 169.254.255.255
    *
    * This query returns true if this object is a private network
    address
    */
    public boolean isPrivateNetwork();

}
```

להלן פירוט לגבי אופן מימוש הייצוגים השונים:

1. מחרוזת – יש להשתמש במחרוזת יחידה לצורך ייצוג כתובת ה IP . כל הפעולות יבוצעו בעזרת מחרוזת זו.
2. מערך – כל אחד מחלקי הכתובת (מספר שלם 0-255) יוחזק בתא במערך.
3. int – נשים לב שכל אחד מחלקי כתובת ה-IP הוא מספר שלם בתחום 0-255 (כולל), לפיכך ניתן לייצג אותו בעזרת 8 ביטים. על כן, את ארבעת חלקי הכתובת ניתן לייצג באמצעות 32 ביטים (4 בתים) וזהו בדיוק גודלו של int.

לפיכך, נשתמש ב-int לא כמספר, אלא כרצף בינארי של 32 ביטים. לדוגמא, הכתובת 127.0.0.1 תיוצג ע"י רצף הביטים 00000001**00000000**00000000**01111111**.

החלק הראשון ייוצג ע"י הביטים במקומות 0-7 (משמאל לימין), החלק השני ע"י הביטים 8-15, השלישי ע"י 16-23 והרביעי ע"י ביטים 24-31.

127 -> 01111111 (ביטים 0-7 משמאל לימין)
0 -> 00000000 (ביטים 8-15)
0 -> 00000000 (ביטים 16-23)
1 -> 00000001 (ביטים 24-31)

<u>הנחיה:</u> על מנת להשתמש בייצוג זה, עליכם לדעת איך לחלץ את ערכו של כל בית (8 ביט) מהרצף הבינארי באורך 4 הבתים (32 ביטים) שמרכיב את ה-int. נציע 2 דרכים אפשריות:

1. שימוש באופרטורים על ביטים (<<,>>,~,&,|) להזזה או למיסוך של הביטים ברצף הבינארי כך שיאופסו כל הביטים מלבד אלו השייכים לבית הרצוי.
2. שימוש במחלקה ByteBuffer
   - צרו אובייקט בגודל 4 בתים ע"י ByteBuffer.allocate(4)
   - היעזרו במתודות put(i)/get(i) להצבה ולחילוץ של בית במיקום i.
   - שימו לב שהמתודה get(i) תחזיר את הבית באינדקס i באובייקט ה-ByteBuffer שיצרתם כמשתנה מטיפוס byte עם טווח ערכים של (-128..+127).

   כדי לבצע המרה של בית b מטיפוס byte למשתנה מטיפוס int עם טווח הערכים 0..255 לו אנו זקוקים, ניתן להשתמש בטריק הבא:

   int i = (int) (b & 0xFF);

הערה חשובה: עליכם לממש את המתודות באופן שונה בכל מחלקה בהתאם לייצוג הפנימי. אין להמיר את הייצוג הפנימי לייצוג אחר לצורך מימוש פעולה (רק לצורך פלט). המחלקות השונות לא "ייעזרו" זו בזו (למשל, אסור להשתמש ב- IPAddressString על מנת לממש את IPAddressInt).

בנוסף, ממשו את המחלקה IPAddressFactory המגדירה את המתודות הבאות:

```
public class IPAddressFactory {

      public static IPAddress createAddress(String ip) {
            ...
      }

      public static IPAddress createAddress(short[] ip) {
```

```
        ...
    }

    public static IPAddress createAddress(int ip) {
        ...
    }
}
```

כל אחת מהמתודות הסטטיות יוצרת אובייקט מטיפוס IPAddress, כשהאובייקט הקונקרטי נקבע על סמך טיפוס הקלט.

הערה: מחלקה שתפקידה היחיד הוא יצור אובייקטים של מחלקות אחרות נקראת *factory class*. מחלקות אלו מסתירות את פרטי יצור האובייקטים מלקוחות של אובייקטים אלו. השימוש בטכניקה זו נועד להסתיר את המחלקות הקונקרטיות שמממשות מנשק.

להלן תכנית המדגימה את השימוש במחלקה IPAddressFactory ובמנשק.

```
public class TestIPAddress {

    public static void main(String[] args) {
        int address1 = -1062731775; // 192.168.0.1
        short[] address2 = { 10, 1, 255, 1 }; // 10.1.255.1

        IPAddress ip1 = IPAddressFactory.createAddress(address1);
        IPAddress ip2 = IPAddressFactory.createAddress(address2);
        IPAddress ip3 = IPAddressFactory.createAddress("127.0.0.1");

        for (int i = 0; i < 4; i++) {
            System.out.println(ip1.getOctet(i));
        }

        System.out.println("equals: " + ip1.equals(ip2));
    }
}
```

מצורפת תכנית בשם **TestIPAddress** המדגימה את השימוש במחלקה IPAddressFactory ובמנשק.

בחלק זה עליכם להגיש את כל הקבצים המצורפים בתיקייה ip. הקובץ TestIPAddress, נועד לבדיקה עצמית בלבד, אותו אתם יכולים להרחיב ולשנות ולבדוק את עצמכם (ניתן להגיש גם אותו, אך הוא לא יבדק).

בהצלחה!