

תוכנה 1

תרגיל מספר 7

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

חלק א' (40%):

בחלק זה נשתמש בשרשראות מרקוב על מנת לכתוב תוכנית שהופכת קובץ טקסט לסיפור מצחיק.



התוכנית מורכבת מ- 2 שלבים :

1. בניית מילון בעזרת זוגות מילים עוקבות בטקסט.
2. הדפסת סיפור אקראי על פי המילון שבנינו.

שלב 1 - בניית המילון:

נעבור על קובץ טקסט נתון ועבור כל צמד מילים עוקבות נשמור אותן במבנה WordTuple כאשר המילה הראשונה בצמד תהיה ה- prevWord והמילה השנייה תהיה currWord.

עבור כל WordTuple נשמור את רשימת המילים שהופיעו לאחר צמד המילים בטקסט המקורי. לדוגמא: אם הטקסט המקורי הוא roses are red and roses are blue, עבור הצמד [roses,are] נחזיק מערך שיכיל את ה- WordTuple של הצמד [are,red] ואת ה- WordTuple של [are,blue]. עבור הצמד [and, roses], לעומת זאת, יהיה רק WordTuple של הצמד [roses, are]. בנוסף עבור כל WordTuple נשמור גם מצביע ממערך שנקרא לו מילון (dict).

עליכם להשלים את מימוש את המחלקה WordTuple שמייצגת צמד מילים, ואת המתודה buildDict במחלקה Mimic שמקבלת שם של קובץ טקסט ובונה מילון של WordTuples. שימו לב: להערות התייעוד הקיימות ניתן להוסיף תנאים בחוזה (על תקינות הקלט, וכו').

נניח שסדר המילים בטקסט המקורי הוא $w_1, w_2, w_3, w_4, \dots$. על מנת לבנות את המילון נבצע:

- i. נעבור על כל המילים בטקסט.
 - ii. עבור כל זוג מילים עוקבות $[w_{i-1}, w_i]$ נבדוק האם הוא כבר קיים במילון. במידה ולא, ניצור עצם חדש מסוג WordTuple עבורו $prevWord == w_{i-1}$ ו- $currWord == w_i$. נוסיף למילון את העצם.
 - iii. נחפש את המילה w_i ברשימת המילים העוקבות של ה-WordTuple הקודם¹, $[w_{i-2}, w_{i-1}]$ (היא תופיע במערך כחלק מצמד $[w_{i-1}, w_i]$). אם מצאנו אז נמשיך הלאה. אם לא מצאנו אותה, נוסיף את ה- WordTuple $[w_{i-1}, w_i]$ שכבר יצרנו ומצאנו במילון בשלב ii למערך המילים העוקבות של הצמד $[w_{i-2}, w_{i-1}]$. במידה ומערך המילים העוקבות מלא, לא נוסיף לתוכו איברים נוספים*.
 - iv. נמשיך לצמד המילים $[w_i, w_{i+1}]$, וכך הלאה, עד לסוף הטקסט.
- שימו לב, שאתם מקבלים מתודות עזר מוכנות כגון קריאה מקובץ. ניתן להגדיר מתודות עזר נוספות.
- * ניתן להניח כי גודל המערך following WordTuples במחלקה WordTuple יהיה מוגבל ל-20 תאים, אך את המערך dict יש להקצות בצורה שיוכל להכיל את כל זוגות המילים לפי הטקסט הנתון. את כל המילים יש להפוך ל-lowercase.

¹ למילים w_1, w_2 לא קיים צמד קודם, כמובן. צריך לבדוק רק החל מ- w_3 .



שלב 2 - הדפסת סיפור:

נתחיל מצמד מילים התחלתי ונבחר אקראית מילה שתמשיך אותו מתוך רשימת המילים העוקבות לצמד זה בטקסט המקורי, נמשיך באופן דומה עבור הצמד החדש שנוצר, וכך הלאה.

לדוגמא, אם הטקסט המקורי הוא roses are red and roses are blue :

i. נתחיל בהדפסת המילה הראשונה בצמד שקיבלנו כארגומנט. נניח שקיבלנו את roses are, אז נדפיס roses.

ii. נמצא במילון את ה- WordTuple המתאים לזוג מילות ההתחלה שקיבלנו בתור ארגומנט.

iii. נדפיס את המילה ה"נוכחית" בצמד – עבור הצמד [roses,are] נדפיס את are.

iv. בעזרת רשימת המילים העוקבות, נבחר אקראית אחת מהן – למשל, אם המילים העוקבות האפשריות הן blue, red, נבחר, למשל, blue. המתודה לבחירת מילה אקראית כבר ממומשת במחלקה.

v. נקבל צמד מילים חדש המורכב מהמילה שהיתה ה"נוכחית" בצמד הקודם והמילה שבחרנו אקראית - עפ"י הדוגמא: נקבל את הצמד החדש [are,blue].

vi. נחזור לצעד iii עבור הצמד החדש ונבצע שוב את השלבים, עד שנדפיס את כמות המילים הדרושה.

vii. במידה ונתקעים (לא נמצאה מילה עוקבת, או שצמד המילים ההתחלתיות לא מופיע במילון), התכנית תסתיים.

עליכם לממש את המתודה printMimic המקבלת מילון (פלט משלב 1), מילה ראשונה ושנייה להתחיל איתן, ואת מספר המילים שעלינו להדפיס. המתודה תדפיס בהתאם לאלגוריתם הני"ל. בנוסף עליכם לממש את המתודה main אשר תקבל ארגומנטים מהמשתמש, תיצור מילון ותדפיס את הטקסט החדש לפי המילון. התוכנית תופעל בצורה הבאה:

```
Mimic <filename> <startWord1> <startWord2> <wordCount>
```

הערות:

באתר הקורס תוכלו להוריד את הקבצים הדרושים על מנת להתחיל לעבוד, ובנוסף 2 קבצי טקסט בשביל בדיקות. הקבצים הם:

WordTuple - מחלקה עבור צמד מילים. השלימו את המתודות הריקות.

Mimic - המחלקה הראשית שבונה את המילון ומדפיסה סיפור חדש המבוסס על המילון.

alice.txt, small.txt - קבצי טקסט לצרכי בדיקה.

חלק ב' / (40%) :

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 שנתן לראות, מורכבת מארבעה חלקים ערכו של כל חלק הוא מספר שלם בין 0 ל-255.

נממש את המנשק בעזרת שלושה ייצוגים שונים: הראשון עושה שימוש במחרוזות, השני במערך של מספרים ואילו השלישי משתמש ב int (הסבר מפורט בהמשך).

א. כתבו **שלוש** מחלקות שונות המממשות את המנשק IPAddress המוגדר למטה :

1. כתבו מחלקה בשם IPAddressString. מחלקה זו מממשת את המנשק בעזרת ייצוג פנימי של String.

2. כתבו מחלקה בשם IPAddressShort. מחלקה זו מממשת את המנשק בעזרת ייצוג פנימי של מערך בגודל 4 של short. כל תא במערך יחזיק מספר בתחום 0..255.

3. כתבו מחלקה בשם IPAddressInt. מחלקה זו מממשת את המנשק בעזרת ייצוג פנימי של int.

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).

```

Package ip;

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);

    /**
     * Mask the current address with the given one. The masking
     * operation is a bitwise 'and' operation on all four parts of
     * the address.
     * @param mask
     *         the IP address with which to mask
     * @return A new IP address representing the result of the mask
     *         operation. The current address is not modified.
     */
    public IPAddress mask(IPAddress mask);
}

```

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 הכתובת הוא מספר שלם בתחום 0-255 (כולל), לפיכך ניתן לייצג אותו בעזרת 8 ביטים. לפיכך, את ארבעת חלקי הכתובת ניתן לייצג באמצעות 32 ביטים שזהו בדיוק גודלו של int.
- נשתמש ב int לא כמספר אלא כרצף של 32 ביטים את הפעולות הדרושות נבצע לא בעזרת פעולות אריתמטיות כי אם בעזרת פעולות על ביטים (&, <<, >> וכדומה) לדוגמא, הכתובת 127.0.0.1 תיוצג ע"י רצף הביטים 01111111000000000000000000000001. החלק הראשון ייצוג ע"י הביטים במקומות 0-7 (משמאל לימין), החלק השני ע"י הביטים 8-15, השלישי ע"י 16-23 והרביעי "י" ביטים 24-31.

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 :

```
Package ip;

public class IPAddressFactory {

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

    public static IPAddress createAddress(short[] ip) {
        ...
    }

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

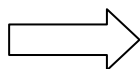
כל אחת מהמתודות הסטטיות יוצרת אובייקט מטיפוס IPAddress, כשהאובייקט הקונקרטי נקבע על סמך טיפוס הקלט.

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.

דוגמה לפעולת mask :

'and' (&) truth table

Bit1/bit2	0	1
0	0	0
1	0	1



Mask example: if we mask 192.168.0.1 with the mask 127.0.0.1 then the result in binary is:
01000000.00000000.00000000.00000001
➔ IP address 64.0.0.1

מצורפת תכנית המדגימה את השימוש במחלקה IPAddressFactory ובמנשק.

```
package tests;

import ip.IPAddress;
import ip.IPAddressFactory;

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

חלק ג' (20%):

נתונים המנשקים הבאים המתארים קורס וסטודנט במערכת מרשם קורסים:

```
public interface Student {  
  
    הרשם לקורס c . הפעולה חוקית אם הסטודנט לא רשום לקורס c , הקורס לא  
    מלא ומספר הנקודות הכולל של הסטודנט לאחר הרישום יהיה לכל היותר 10.  
    public void register(Course c);  
  
    בטל את הרישום לקורס c . הפעולה חוקית אם הסטודנט רשום לקורס c.  
    public void drop(Course c);  
  
    מספר הנקודות הכולל של הקורסים שהסטודנט רשום להם  
    public int totalUnits();  
  
    שם הסטודנט  
    public String name();  
}
```

```
public interface Course {  
  
    מספר נקודות (שעות) - (שלם בין 1 ל 5)  
    public int units();  
  
    רמת הקורס (שלם בין 1 ל 3)  
    public int level();  
  
    מספר הסטודנטים הרשומים כרגע לקורס  
    public int numOfStudents();  
  
    המספר המרבי המותר של סטודנטים רשומים לקורס  
    public int maxNumStudents();  
  
    החזר true אם ורק אם הסטודנט s רשום לקורס.  
    public boolean registered(Student s);  
  
    רשום את הסטודנט s לקורס. הפעולה חוקית רק אם s לא רשום לקורס, והקורס  
    לא מלא  
    public void register(Student s);  
  
    בטל את הרישום של s לקורס. הפעולה חוקית רק אם s רשום לקורס  
    public void drop(Student s);  
}
```


להלן דוגמת שימוש במנשקים :

```
Course c1 = new ... ; // a course of 3 units, and max number of
                        // students 40
Course c2 = new ... ; // a course with 4 units, and max number of
                        // students 40

Student s1 = new ... ;
Student s2 = new ... ;

System.out.println(s1.totalUnits());

s1.register(c1);
s2.register(c1);
s1.register(c2);

System.out.println(s1.totalUnits());
s1.drop(c1);
System.out.println(s1.totalUnits());
```

הפלט של סדרת הפעולות הוא :

0
7
4

- א. כתבו את החוזה של המנשק Course : לכל שרות כתבו תנאי קדם (precondition) ותנאי אחר (postcondition) באופן המקובל (ביטויים בוליאניים שיכולים להשתמש בשאילתות). במידת הצורך, הוסיפו במילים תנאים שלא ניתנים לביטוי בצורה הרגילה. יש להוסיף את הערות החוזה לקובץ העזר Course.java המצורף לתרגיל.
- ב. כתבו את החוזה של המנשק Student : לכל שרות כתבו תנאי קדם (precondition) ותנאי אחר (postcondition) באופן המקובל (ביטויים בוליאניים שיכולים להשתמש בשאילתות). במידת הצורך, הוסיפו במילים תנאים שלא ניתנים לביטוי בצורה הרגילה. יש להוסיף את הערות החוזה לקובץ העזר Student.java המצורף לתרגיל.
- ג. המחלקה SimpleCourse אמורה לממש את המנשק Course בצורה פשוטה, תוך שימוש במערך לייצוג הסטודנטים הרשומים לקורס. נתון חלק מהקוד – כל השדות, הבנאי, ומימוש של שרות אחד.
1. הגדירו את משתמר הייצוג (representation invariant) של המחלקה SimpleCourse
 2. השלימו את קוד המחלקה SimpleCourse.

```

public class SimpleCourse implements Course {

    private int maxNumStudents;
    private int top;
    private int units;
    private int level;
    private Student[] students;

    public SimpleCourse(int maxNumStudents, int units, int level) {
        this.maxNumStudents = maxNumStudents;
        students = new Student [maxNumStudents];
        this.units = units;
        this.level = level;
        top = -1;
    }

    public void register(Student s) {
        students[++top] = s;
    }

    // more code omitted
}

```

הערות: ייתכן שלחלק מהשירותים אין תנאי קדם וואו תנאי אחר. השתדלו להשתמש בחוזה במתודות של המנשק במידת האפשר (כפי שבתרגול 5 השתמשנו במתודות ציבוריות בתנאי הקדם והאחר). עם זאת, היזהרו לא ליצור חוזה "רקורסיבי" (למשל, להגדיר את תנאי האחר של name() בתור name(@post \$ret==name())). חישבו היטב על המחלקות המממשות את המנשקים, ומהי האחריות של כל מחלקה. למשל, מי מנהל את הרישום לקורסים, את מס' הנקודות שיש לכל סטודנט וכו'.

בהצלחה!