

תוכנה 1

תרגיל מספר 6

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. (אין להגיש קבצי class).
 - ג. קובץ טקסט (פורמט txt או doc) עם העתק של כל קבצי ה java
 - ד. קובץ טקסט (פורמט txt או doc) בשם answers עם התשובות לשאלות

חלק א:

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 כל חלק הוא מספר שלם בין 0 ל-255. נממש שלושה ייצוגים שונים עבור המנשק IPAddress. המימוש הראשון עושה שימוש במחרוזות, השני במערך של מספרים ואילו השלישי משתמש ב int (הסבר מפורט בהמשך).

- א. כיתבו **שלוש** מחלקות שונות המממשות את המנשק IPAddress המוגדר למטה:
1. מחלקה בשם IPAddressString. מחלקה זו מממשת את המנשק בעזרת ייצוג פנימי של String.
 2. מחלקה בשם IPAddressShortArray. מחלקה זו מממשת את המנשק בעזרת ייצוג פנימי של מערך בגודל 4 של short. כל תא במערך יחזיק מספר בתחום 0..255.
 3. 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.

```

package il.ac.tau.cs.sw1.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 one 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);

    /**
     * Returns whether this address is a private network address.
     * There are three ranges of addresses reserved for 'private
     * networks' 10.0.0.0 - 10.255.255.255, 172.16.0.0 -
     * 172.31.255.255 and 192.168.0.0 - 192.168.255.255.
     * @return true if this address is in one of the private
     *         network address ranges, false otherwise.
     */
    public boolean isPrivateNetwork();

    /**
     * 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 – מערך בגודל 4 בו 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. את ארבעת חלקי הכתובת, לפיכך, ניתן לייצג באמצעות 32 ביטים שזהו בדיוק גודלו של int .
נשתמש ב int לא כמספר אלא כרצף של 32 ביטים (גודל שח int). את הפעולות הדרושות נבצע בעזרת פעולות על ביטים (&, <<, >> וכדומה) ולא בעזרת פעולות אריתמטיות.
לדוגמא, הכתובת 127.0.0.1 תיוצג ע"י רצף הביטים 01111111000000000000000000000001 .
החלק הראשון מיוצג 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.
- הערה : אין להמיר את הייצוג הפנימי לייצוג אחר לצורך מימוש פעולה (רק לצורך פלט). לצורך מימוש ייצוג מסוים אין להיעזר במחלקות המממשות ייצוג שונה.

בנוסף, ממשו את המחלקה IPAddressFactory המגדירה את המתודות הבאות :

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

public class IPAddressFactory {

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

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

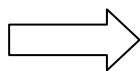
    public static IPAddress createAddress(int ip) {
        ...
    }
}
```

כל אחת מהמתודות הסטטיות יוצרת אובייקט ט 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 ממשק .

דוגמה לפעולת mask :

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 (חלקית) המדגימה את השימוש במחלקה IPAddressFactory ובמנשק.

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

public class TestIpAddressInt {

    public static void main(String[] args) {
        int address1 = ...; // 192.168.0.1
        int address2 = ...; // 127.0.0.1
        int private1 = ...; // 10.1.255.1
        short[] address3 = new short[] {127, 0, 0 , 1};

        IPAddress ip1 = IPAddressFactory.createAddress(address1);
        IPAddress ip2 = IPAddressFactory.createAddress(address2);
        IPAddress ip3 = IPAddressFactory.createAddress(private1);
        IPAddress ip4 = IPAddressFactory.createAddress(ip1.toString());
        IPAddress ip5 = IPAddressFactory.createAddress(address3);

        // 192.168.0.1, 127.0.0.1, 10.1.255.1, 192.168.0.1, 127.0.0.1
        System.out.println(ip1 + ", " + ip2 + ", " + ip3 + ", " +
            ip4 + ", " + ip5);
        System.out.println(ip1.equals(ip4)); // true
        System.out.println(ip5.equals(ip2)); // true
        System.out.println(ip1.equals(ip2)); // false

        System.out.println(ip1.mask(ip2)); // 64.0.0.1

        System.out.println(ip3.isPrivateNetwork()); // true
        System.out.println(ip2.isPrivateNetwork()); // false
    }
}
```

חלק ב:

נדון במחלקות שמייצגות מספרים רציונאליים, כגון $1/3$ או $99/100$. תזכורת מתמטית קצרה: כדי לצמצם שברים צריך לחלק את המונה והמכנה בגורם המשותף הגדול ביותר (GCD באנגלית). למשל, הגורם המשותף הגדול ביותר של 42 ו-56 הוא 14, ולכן:

$$\frac{42}{56} = \frac{3 \cdot 14}{4 \cdot 14} = \frac{3}{4}$$

כדי לחבר או לחסר שברים, צריך למצוא את המכנה המשותף (LCM באנגלית). למשל, המכנה המשותף של 21 ו-6 הוא 42, ולכן:

$$\frac{2}{21} + \frac{1}{6} = \frac{4}{42} + \frac{7}{42} = \frac{11}{42}$$

א. בין המתכנתים ביל ואומה התפתח ויכוח האם **טורם** לכתובת המחלקה המייצגת מספרים רציונאליים יש לכתוב **מנשק** המתאר את הפונקציונאליות של המחלקה. ביל טען כי המחלקה פשוטה דיה וכי אין הצדקה לתאר את אותו הדבר גם ע"י מנשק וגם ע"י מחלקה. ציינו 3 טיעונים **נגדיים** המצדדים בשימוש במנשק. נמקו או הדגימו בקצרה את טיעוניכם.

ב. לאחר שביל השתכנע בנחיצות המנשק, החל דיון בינו ובין אומה על נחיצות השרותים הבאים. עבור כל אחד מהם ציינו האם יש לו מקום במנשק, במחלקה, בשניהם או באף אחד מהם. נמקו בקצרה:

1. add - לחיבור שני מספרים רציונליים
 2. subtract - לחיסור שני מספרים רציונליים
 3. multiply - להכפלת שני מספרים רציונליים
 4. divide - לחלוקת שני מספרים רציונליים
 5. equals - להשוואת שני מספרים רציונליים
 6. gcd - לחישוב הגורם המשותף הגדול ביותר של שני שלמים
 7. lcm - לחישוב מכנה משותף של שני שלמים
 8. normalize - להבאת שבר פשוט למצב מצומצם
 9. toString - לייצוג המספר כמחרוזת של שבר פשוט (למשל לצורכי הדפסה)
 01. toDecimalString - לייצוג המספר כמחרוזת של שבר עשרוני (למשל לצורכי הדפסה)
- ג. מה לגבי השרותים: `getNumerator` ו-`getDenominator` להחזרת המונה והמכנה של השבר בהתאמה? ביל ואומה הסכימו כי הדבר תלוי בהקשרי השימוש של הטיפוס החדש. ציינו באילו הקשרים יש הצדקה להכללת השרותים במנשק ובאילו אין. נמקו או הדגימו את תשובתכם.

חלק ג:

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
}
```