

משימה #6 בקורס תוכנה 1

הוראות הגשה:

1. קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
2. הגשת התרגיל תעשה ע"י המערכת VirtualTAU (<http://virtual.tau.ac.il>).
3. הגשת התרגיל תתבצע ע"י יצירת קובץ zip שנושא 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 התכניות שהתבקשתם לכתוב.
 - ג. קובץ טקסט עם העתק של כל קבצי ה Java.
 - ד. קובץ טקסט עם פתרונות לשאלות בהן לא נדרשתם לכתוב קוד.

חלק א:

המנשק IPAddress 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 מספרים, כל אחד בין 0 ל-255. ניתן להסתכל על כל אחד מארבעת המספרים כמספר בן 8 ביטים (8 ספרות בינאריות). לדוגמה המספר 127 בבסיס בינארי הינו 01111111, המספר 192 בבסיס בינארי הינו 11000000. כתובת ה-IP 127.0.0.1 בייצוג בינארי תיראה כרצף של 32 ביטים (שימו לב שרצף של 32 ביטים זהו בעצם int):
01111111 00000000 00000000 00000001

לפרטים נוספים על בסיס בינארי והמרות ממנו ואליו ניתן לקרוא כאן:
http://en.wikipedia.org/wiki/Binary_numerical_system
לפרטים נוספים לגבי פעולות על ביטים ניתן לפנות לתרגול מס' 1.

- א. כתבו שלשה מימושים שונים למשק IPAddress המופיע 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 הנשק 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 הנשק בעזרת ייצוג פנימי של int. להזכירכם, ב int יש 32 ביטים כך שהוא מספיק כדי להכיל כתובת IP.
- לכל אחת מהמחלקות יהיה בנאי המתאים לייצוג הפנימי שלה וכמובן כל אחת מהן מממשת את המנשק.

```

package il.ac.tau.cs.software1.ex5;

public interface IIPAddress {

    /**
     * Returns a string representation of the IP address,
     * for example "192.168.0.1"
     */
    public String toString();

    /**
     * @param ip IP address to which we compare this address
     * @return true if both objects represent the same address
     */
    public boolean equals(IIPAddress ip);

    /**
     * Returns one of the four parts of the IP address
     *
     * @param index legal vales are [0..3], for example
     *     192.168.0.1 would return 192 for index 0, 168 for index 1,
     *     0 for index 2 and 1 for index 3.
     * @return the desired part of the IP address (one of four parts)
     *     [0..255]
     */
    public short 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();

    /**
     * This command receives as a parameter an IP address
     * and uses it to mask the current address
     * i.e. perform a bitwise 'and' operator on all four parts.
     * A bitwise 'and' operator is represented in Java as the
operator
     * '&', given two numbers, it compares them bit by bit, and
applies
     * the and operator to each bit pair. The result is a number of
the
     * same length of bits with the results of the operations. Look
at
     * the 'and' truth table on the side for a reminder on how and
     * works and examples
     *
     * @param mask the IP address with which to mask
     * @return the mask result
     */
    public IIPAddress mask(IIPAddress mask);
}

```

ממשו את המחלקה IPAddressFactory שמגדירה 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 טיפוס 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 number 64.0.0.1

חלק ב:

נדון במחלקות שמייצגות מספרים רציונליים, 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 **טרם** לכתוב המחלקה המייצגת מספרים רציונליים 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 {

    public int units();           מספר נקודות (שעות) - (שלם בין 1 ל 5)

    public int level();         רמת הקורס (שלם בין 1 ל 3)

    public int numOfStudents(); מספר הסטודנטים הרשומים כרגע לקורס

    public int maxNumStudents(); המספר המירבי המותר של סטודנטים רשומים לקורס

    public boolean registered(Student s);
    החזר true אם ורק אם הסטודנט s רשום לקורס.

    public void register(Student s);
    רשום את הסטודנט s לקורס. הפעולה חוקית רק אם s לא רשום לקורס, והקורס לא
    מלא

    public void drop(Student s);
    בטל את הרישום של s לקורס. הפעולה חוקית רק אם s רשום לקורס
}

```

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 יהיה:

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 בשאילותות). 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 בשאילותות). 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.
- הגדירו את משתמר הייצוג (representation invariant) של המחלקה `SimpleCourse`.
 - השלימו את הקוד של 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  
}
```