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
  ד. קובץ טקסט בשם answers עם התשובות לשאלות

חלק א (30%):

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 הנשק בעזרת ייצוג פנימי של int.

לכל אחת מהמחלקות יהיה בנאי המתאים לייצוג הפנימי שלה וכמובן כל אחת מהן מממשת את המנשק.

```java
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);

	/**
	 * 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.
	 * Here, we regard IP addresses as four digit numbers in
	 * radix 256 (    256 בסיס  )
	 * @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 – 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: אין להמיר את הייצוג הפנימי לייצוג אחר לצורך מימוש פעולה (רק לצורך פלט). לצורך מימוש ייצוג מסוים אין להיעזר במחלקות המממשות ייצוג שונה (למשל אין 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) {
		...
	}
}
```

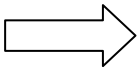
כל אחת מהמתודות הסטטיות יוצרת אובייקט מטיפוס IPAddress, כשהאובייקט הקונקרטי נקבע על סמך טיפוס הקלט.

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 שממשות 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 (חלקית) 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 = ...;                  // 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));
        System.out.println(ip3.isPrivateNetwork());
    }
}
```

חלק ב (10%):

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 **טרם** לכתיבת 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)
10. 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 (15%):

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
}
```

חלק ד (45%):

מבוא

בחלק זה נרצה לממש מערכת המלצה לסרטים\ספרים. הרעיון הוא לקבל מהמשתמש רשימת ספרים שהוא קרא ולהמליץ לו על הספר שהכי יתאים לו שהוא עדיין לא קרא. מערכות כאלה נמצאות בשימוש על ידי שירותים מוכרים כמו Amazon או Netflix, ועד לאחרונה Netflix ערכו תחרות שנתית עם פרס כספי של 1 מיליון דולר לצוות שיפתח אלגוריתם יותר טוב ב-10% מהאלגוריתם שלהם (למידע נוסף: http://en.wikipedia.org/wiki/Netflix_Prize).
בתרגיל זה אנחנו נבחר במערכת להמלצת ספרים, אך תוכלו באופן דומה לממש מערכת לסרטים או כל מוצר צריכה אחר.

כיצד ממליצים על ספר ?

גישה נאיבית:
נניח שנתון לנו משתמש שקוראים לו יניב, כיצד נמצא ספר שהוא יאהב לקרוא ? הדרך הכי פשוטה זה לתת את המלצה מבלי תלות במשתמש עצמו. פשוט נחשב את ממוצא הדירוגים עבור כל ספר במערכת ונמליץ על 5 הספרים בעלי הממוצע הכי גבוה שיניב עדיין לא קרא. המידע היחיד שיחודי ליניב בגישה הזו הוא האם הוא קרא את הספר כבר או לא.

גישה יותר מתוחכמת:
נוכל לספק המלצה טובה יותר אם נסתכל על הספרים שיניב כבר דירג בעבר ולהשוות אותם לדירוגים של משתמשים אחרים במערכת. נניח ידידה שלכם קראה ספר ששניכם קראתם ואהבתם אותו. אז בפעם הבאה שהיא תמליץ לכם על ספר שהיא קראה ואתם לא, כנראה שגם תרצו לקרוא אותו.
לעומת זאת, אם אתם בדרך כלל לא מסכימים לגבי ספרים אז כנראה שלא תרצו לקרוא ספר שהיא תמליץ עליו.

תוכנית יכולה לחשב את מידת ההתאמה של שני משתמשים באופן הבא: נסתכל על דירוג הספרים של משתמש כווקטור ונחשב את ההתאמה בין שני משתמשים על ידי מכפלה סקלרית בין וקטורי הדירוג שלהם. להזכירכם עבור שני הוקטורים הבאים:

$$\vec{\alpha} = (a_1, ..., a_n)$$
$$\vec{\beta} = (b_1, ..., b_n)$$

המכפלה הסקלרית היא:

$$\langle \alpha, \beta \rangle = a_1 b_1 + ... + a_n b_n$$

לדוגמא (טבלת הדירוג בעמוד הבא):

נניח שבמערכת שלנו יש 3 ספרים וניב דירג אותם [5,3,-5], מיכל דירגה אותם [1,5,-3], לירון דירגה אותם [5,-3,5], ואורן דירג אותם [1,3,0].

רמת ההתאמה בין יניב למיכל היא: (5x1)+(3x5)+(-5x-3)=5+15+15=35,

ורמת ההתאמה של יניב ולירון היא: (5x5)+(3x-3)+(-5x5)=25-9-25=-9,

ורמת ההתאמה של יניב ואורן היא: (5x1)+(3x3)+(-5x0)=5+9+0=14.

נשים לב, שאם שני משתמשים אהבו ספר כלשהו (נתנו לו דירוג גבוה) או לא אהבו ספר כלשהו (נתנו לו דירוג נמוך) זה מגדיל את רמת ההתאמה שלהם.

ברגע שחישבנו את רמת ההתאמה בין יניב לכל משתמש אחר, ניתן לזהות את המשתמש שדירוג הספרים שלו הכי דומה לזה של יניב. במקרה שלנו זאת תהיה מיכל, ולכן נמליץ ליניב את הספרים בעלי הדירוג הגבוה מהרשימה של מיכל שיניב עדיין לא דירג.

<u>שלב א' - נממש רשימה מקושרת</u>

נגדיר מחלקה Element בעלת 2 שדות:

```
public class Element {

	private String value;
	private Element next;
```

כאשר השדה value הוא תוכן האיבר הנוכחי (מטיפוס מחרוזת), והשדה next יצביע לאיבר הבא (מטיפוס Element).

בעזרת Element נוכל להגדיר רשימה מקושרת של איברים מסוג Element כאשר כל איבר מצביע לאיבר הבא ונוכל לטייל בין האיברים בעזרת השדה next.

עליכם לממש את המחלקה Element ואת המחלקה LinkedList שתממש את המנשק הבא המייצג רשימה מקושרת:

```
public interface ILinkedList {
	/**
	 * Returns the number of elements in this list.
	 *
	 * @return the number of elements in this list
	 */
	public int size();

	/**
	 * Appends the specified element to the end of this list.
	 *
	 * @param e - element to be appended to this list
	 */
	public void add(String e);

	/**
	 * Returns the element at the specified position in this list.
	 *
	 * @pre (0<=index<size())
	 * @param index - index of the element to return
	 * @return the element at the specified position in this list
	 */
	public String get(int index);
}
```

רמז: החזיקו מצביע לאיבר הראשון והאחרון (מטיפוס Element) במחלקה LinkedList.

שלב ב' - נגדיר את המחלקות

נגדיר את 3 המרכיבים העיקריים במערכת שלנו כמחלקות באופן הבא:

המחלקה **Book** תייצג ספר, המחלקה **Member** תייצג משתמש במערכת, והמחלקה **Rating** תייצג דירוג של משתמש בודד עבור ספר ספציפי. באתר הקורס תוכלו למצוא מנשקים עבור כל אחת מהמחלקות וקובץ מחלקה שלדי המממש את המנשק. עליכם לממש את המתודות החסרות ולהוסיף שדות מופע\מחלקה במקרה הצורך.

מספר דגשים:

- לכל משתמש במערכת יש מזהה יחודי id, אתם רשאים לקבוע אותם כרצונכם.
- לכל ספר במערכת יש מזהה יחודי id, אשר תואם למספר השורה בקובץ הקלט (למשל בקובץ books.txt).
- עליכם להשתמש ברשימה המקושרת משלב א' על מנת לממש את רשימת הדירוג של כל משתמש. לצורך כך תיצרו מחלקה חדשה RatingLinkedList, העתיקו את המימוש משלב א', ושנו את הטיפוס של value לטיפוס מסוג IRating. שימו לב, יש לשנות את הטיפוס בכל המתודות של המחלקה.
- אנחנו מספקים לכם 2 קבצי קלט עם נתונים אמיתיים:
  a. books.txt המכיל את רשימת הספרים שתופיע במערכת.
  b. ratings.txt המכיל את שמות המשתמשים ואת הדירוג של כל אחד על הספרים במערכת.
- כל משתמש רשאי לדרג את הספר לפי הטבלה הבאה:

Rating	Meaning
-5	Hated it!
-3	Didn't like it
0	Haven't read it
1	ok - neither hot nor cold about it
3	Liked it!
5	Really liked it!

- המחלקה Book והמחלקה Member מספקות מתודות סטטיות שתפקידן לקרוא את הקובץ המתאים ולהחזיר מבנה נתונים. למשל עבור המחלקה Book המתודה תקרא את הקובץ books.txt ותחזיר מערך של כל הספרים במערכת (הסבר מפורט יותר לגבי קריאה מקובץ ניתן למצוא בשלב ה').
- תיאור מלא של כל מתודה מופיע בחוזה המחלקה (כמתואר במנשק של כל מחלקה).
- אתם רשאים להוסיף כל מתודת עזר שתצטרכו, אך שימו לב שאין לשנות את החוזה המוגדר במנשק.

שלב ג' - נממש את הגישה הנאיבית להמלצת ספר

ממשו את המתודה הממליצה למשתמש ספר על בסיס הגישה הנאיבית.

```
	/**
* Returns the book that is recommended to the user m1, which has the highest rating in the system
and the user m1 hasn't rated yet.
* If no book recommendation can be found in the system (either the user has already rated all
books, or nobody rated certain books)
* the method will return null.
*
* @param books
* @param m1
* @return the book that is recommended to the user m1, which has the highest rating in the system
and the user m1 hasn't rated yet.
*/
public static IBook getRecommendationByAverageRating(IMember[] members,IBook[] books, IMember m1)
```

שלב ד' - נממש את הגישה המשופרת להמלצת ספר

ממשו את המתודה הממליצה למשתמש ספר על בסיס הגישה המשופרת.

```
/**
* Returns the book that is recommended to the user m1 based on the similarity of his ratings to
the ratings of other members.
* If no book recommendation can be found in the system (either the user has already rated all
books, or nobody rated certain books)
* the method will return null.
*
* @param members - array of all members
* @param books   - array of all the books
* @param m1 - the member for whom we need to find a recommendation
* @return the book that is recommended to the user m1 based on the similarity of his ratings to
the ratings of other members.
*/
public static IBook getRecommendationByCollaborativeFiltering(IMember[] members, IBook[] books,
IMember m1)
```

שלב ה' - נוסיף קלט מהמשתמש והדפסה למסך

עליכם לממש את המתודה main שתקבל קלט מהמשתמש ותמליץ לו על ספר ב-2 הגישות הנ"ל.

פורמט הקלט הוא:

Usage: BookRecommend <books filename> <ratings filename> <member ID number>

הארגומנטים הם:

<books filename> - שם הקובץ המכיל מידע אודות הספרים. על כל שורה בקובץ להכיל מידע לגבי ספר בודד, כאשר המבנה הוא <book title>,<author>.

<ratings filename> - שם הקובץ המכיל מידע אודות המשתמשים ומערך הדירוג של כל אחד מהם. עבור כל משתמש מוקצות 2 שורות בקובץ: בשורה הראשונה מופיע שם המשתמש, ובשורה השנייה מופיע מערך דירוג הספרים. כאשר מתבצעת קריאה מהקובץ ונבנה מבנה הנתונים, יש לשמור עבור כל משתמש את רשימת הדירוג שלו. כלומר יש לעבור על המערך ועבור כל דירוג של ספר על ידי משתמש מסוים, יש ליצור אובייקט חדש מטיפוס Rating ולהכניס אותו לרשימת הדירוג של אותו משתמש במערכת. במידה ומשתמש סימן שהוא לא קרא את הספר (כלומר הדירוג הוא 0) יש להתעלם ולא להוסיף דירוג זה למערכת.

<member ID number> - מספר מזהה של משתמש במערכת שעבורו נרצה למצוא המלצה לספר. למשל מזהה 0 מתאים למשתמש הראשון במערך המשתמשים.

יש לבצע בדיקת קלט על הארגומנטים שהמשתמש מכניס ולספק הודעת שגיאה מתאימה במקרה הצורך.

שלב ו' - נשפר את האלגוריתם

נשים לב שהאלגוריתם עדיין לא מושלם, כי הוא לא מסוגל לתת המלצה ב- 2 המקרים הבאים:

1. המשתמש שעבורו רוצים לקבל המלצה כבר דירג את כל הספרים במערכת. במקרה זה יש להדפיס הודעה מתאימה למסך.
2. המשתמש שעבורו רוצים לקבל המלצה דירג את כל הספרים במערכת חוץ מאשר ספרים שאף אחד אחר גם לא דירג אותם. במקרה זה, יש לבחור אקראית ספר אחד מתוך הספרים שהמשתמש לא קרא ולהמליץ עליו.ממשו את המתודה הבאה שמוצאת ספר אקראי במערכת שהמשתמש לא דירג:

```
/**
* Returns a random book which the given user hasn't rated yet.
* @pre !member.hasReadAllBooksInSystem()
* @param books
* @param member
* @return a random book which the given user hasn't rated yet.
*/
public static IBook getRandomUnratedBook(IBook[] books, IMember member)
```

רשות (לא להגשה)

נסו לשפר את האלגוריתם כרצונכם על מנת להגיע לתוצאות טובות יותר. רעיונות אפשריים:

- אנחנו חיפשנו משתמש בודד בעל רמת ההתאמה הכי גבוהה ובחרנו את ההמלצה על בסיס רשימת הדירוג שלו, אך ניתן להשתמש בכמה משתמשים (למשל 3) שהם בעלי ההתאמה הכי גבוהה ולבחור את ההמלצה על בסיס שילוב הרשימות שלהם. במקרה זה יש להחליט איזה משקל ניתן לכל אחד מהם ולכל המלצה שלהם.
- ניתן להוסיף קטגוריות לספרים (למשל מדע בדיוני). ניתן להשתמש בקטגוריות וליצור וקטור המייצג עד כמה המשתמש מעדיף לקרוא קטגוריות מסוימות ולעשות חישוב דומה על מנת למצוא את רמת ההתאמה של ספר למשתמש על בסיס סוג הספר ולא דירוגים.

שאלות ותשובות:

- האם ניתן להניח שמספר הספרים או מספר המשתמשים במערכת קבוע? לא. אך אתם רשאים להניח שלא יהיה יותר מ 1000 משתמשים או ספרים במערכת.
- מהו פורמט קבצי הקלט? הפורמט מוגדר בסעיף ה'. אתם יכולים לראות לדוגמא את הפורמט בקבצים שסיפקנו לכם.
- מהו פורמט הודעת הפלט של התוכנית? דוגמת הרצה על הקבצים שקיבלתם - עבור הקלט הבא: BookRecommend books.txt ratings.txt 0

נקבל את הפלט הבא:

```
Book recommendation for member 0
by average rating: Garth Nix,Sabriel
by collaborative filtering: Daniel Keyes,Flowers For Algernon
```

- מה הפלט שצריך להיות במקרה והמשתמש כבר דירג את כל הספרים במערכת?

```
Member 0 has read all books in our system
```

- היכן בתוכנית יש לממש את שלב ו'? עליכם לממש תחילה את המתודה `getRandomUnratedBook`. לאחר מכן עליכם להוסיף בדיקה ב- `main` שתבדוק אם אחת הגישות הקודמות לא הניבה המלצה לספר, אזי יש לפעול עפ"י המתואר בשלב ו'. הפלט במקרה זה צריך להיות כמו מקודם.
- מה קורה כאשר המשתמש הכי מתאים לי קרא בדיוק את אותם ספרים כמוני? במקרה זה, הגישה הנאיבית תמצא לנו המלצה אך הגישה המשופרת לא תוכל למצוא המלצה מכיוון שקראנו בדיוק את אותם ספרים. במקרה זה עבור הגישה המשופרת, תבחרו ספר אקראית מתוך רשימת הספרים שעדיין המשתמש לא קרא.

הערות כלליות:

- כחלק מהקבצים לתרגיל, מצורפת חבילה test שבה ניתן למצוא מחלקות בדיקה.
- שימו לב כי <u>אלו הן רק חלק מהבדיקות</u> שנשתמש על מנת לבדוק את התרגיל שלכם.
- תוכלו להיעזר בבדיקות בשביל דוגמאות שימוש במתודות מסוימות במקרה הצורך.
- הפתרון שלכם חייב להיות לפי החוזה המוגדר במנשקים הנתונים - אין לשנות חוזה או חתימה של מתודה. פתרון שלא תואם למנשק יורדו לו נקודות. אך שימו לב שאתם רשאים להגדיר מתודות עזר כרצונכם.

בהצלחה !