

תוכנה 1

תרגיל מספר 8

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 יכול:
 - 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 (15%):

בחלק זה עליכם לממש מתודה המחזירה מצביע (reference) אל התא (LinkedListNode) ה- n לפני אחרון ברשימה מקושרת. עליכם לבצע מעבר יחיד על הרשימה, ואין ליצור העתק של הרשימה (או העתקת תוכן הרשימה לתוך מערך). עליכם להשתמש במחלקה הבאה המגדירה תא יחיד ברשימה מקושרת:

```
public class LinkedListNode {
    private int value;
    private LinkedListNode next;

    public LinkedListNode(int value){
        this.value = value;
    }
    public int getValue() {
        return value;
    }
    public void setValue(int value) {
        this.value = value;
    }
    public LinkedListNode getNext() {
        return next;
    }
    public void setNext(LinkedListNode next) {
        this.next = next;
    }
}
```

עליכם לממש את המתודה הבאה:

```
public static LinkedListNode nthToLast(LinkedListNode head, int n) {}
```

שימו לב:

- אין להניח שאורך הרשימה הוא גדול מ- n או כי המצביע לראש הרשימה שונה מ- null. במידה ולא ניתן להחזיר את האיבר ה- n לפני אחרון יש להחזיר null.
- במידה ו- n==0 אזי יש להחזיר את האיבר האחרון ברשימה.
- מותר להשתמש במשתני עזר בודדים אך אין ליצור n משתני עזר או מבני נתונים בגודל n.

חלק ב (45%):

מבוא

בחלק זה נרצה לממש מערכת המלצה לסרטים וספרים. הרעיון הוא לקבל מהמשתמש רשימת ספרים שהוא קרא ולהמליץ לו על הספר שהכי יתאים לו, מבין הספרים 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 היא לתת את ההמלצה מבלי תלות במשתמש עצמו. פשוט נחשב את הממוצע הדירוגים עבור כל ספר במערכת ונמליץ על 5 הספרים בעלי הממוצע הכי גבוה שיניב עדיין לא קרא. המידע היחיד שייחודי 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 שידידה שלכם קראה ואהבה מסי ספרים שגם אתם קראתם ואהבתם. בפעם הבאה שהיא תמליץ לכם על ספר שהיא קראה ואתם לא, כנראה שתמצאו לקרוא אותו. לעומת זאת, אם אתם בדרך כלל לא מסכימים לגבי ספרים אז כנראה שלא תמצאו לקרוא ספר שהיא ממליצה עליו.

תוכנית יכולה לחשב את מידת ההתאמה בין 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:

$$\vec{\alpha} = (a_1, \dots, a_n)$$

$$\vec{\beta} = (b_1, \dots, b_n)$$

המכפלה הסקלרית היא:

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

לדוגמה (טבלת הדירוג בעמוד הבא):

נניח שבמערכת שלנו יש 3 ספרים ויניב 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: $(5 \times 1) + (3 \times 5) + (-5 \times -3) = 5 + 15 + 15 = 35$

ורמת ההתאמה של יניב ולירון היא: $(5 \times 5) + (3 \times -3) + (-5 \times 5) = 25 - 9 - 25 = -9$

ורמת ההתאמה של יניב ואורן היא: $(5 \times 1) + (3 \times 3) + (-5 \times 0) = 5 + 9 + 0 = 14$

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.

שלב א' - נממש רשימה מקושרת

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 מופעל 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' - נוסף 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. למשל יש לבדוק שהקובץ קיים לפני שמנסים לגשת אליו (בעזרת מתודות של המחלקה File),
ואם הקובץ לא קיים יש לספק הודעת שגיאה. אין צורך להתייחס לחריגים (exceptions) אחרים
הקשורים בקריאה מקובץ. יש להתייחס למקרה שבו המשתמש מכניס מספר שונה של ארגומנטים
מהדרוש.

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 יהיו 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
BookRecommend books.txt ratings.txt 0
הבא:
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 אלו הן רק חלק מהבדיקות שנשתמש בהן 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. יורדו נקודות על פתרון שאינו תואם למנשק. שימו לב שאתם רשאים להגדיר מתודות עזר כרצונכם.
- אין צורך להגיש את המימוש עבור רשימה מקושרת (מסעיף א') אך יש להגיש את הרשימה המקושרת RatingLinkedList.

חלק ג (40%): חפשו אותי

בחלק זה אתם נדרשים לממש מנוע חיפוש פשוט. חלקים מהקוד כבר נכתבו עבורכם והם זמינים להורדה באתר הקורס.

מנוע החיפוש שלנו יטפל במספר מצומצם של דפי HTML אותם הוא יקרא מהרשת. דפים אלו קבועים מראש. תוכלו למצוא את הרשימה במחלקה `SearchEngine`. אם תרצו תוכלו להוסיף או לשנות את הדפים בהם אתם משתמשים.

לאחר שהורדנו דף HTML מהרשת נתייחס רק לחלק הטקסט שבדף ונפרק חלק זה למילים בודדות. קוד זה כבר מומש עבורכם במחלקה `HTMLTokenizer`. בנוסף המחלקה `SearchEngine` שבה הקוד המפעיל את המערכת ומתקשר עם המשתמש קיימת אף היא.

מה עליכם לעשות:

נרצה ליצור אינדקס של כל המילים שהופיעו בכל הדפים שהורדנו מהרשת. קיומו של אינדקס זה יאפשר לנו מאוחר יותר לבצע חיפושים עבור מילה מסוימת. עליכם לממש מחלקה בשם `MyWordIndex` המממשת את המנשק `WordIndex`. מחלקה זו שומרת את אינדקס המילים, מאפשרת הוספת מילים לאינדקס וחיפוש בו.

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

import java.util.Collection;
import java.util.List;

public interface WordIndex {

    /**
     * Add the words originating in the specified URL.
     *
     * @param words
     *         - collection of words to add to the index
     * @param strURL
     *         - the location of the page containing the words
     */
    void index(Collection<String> words, String strURL);

    /**
     * Search for a given word in the index
     *
     * @param word
     *         - the word to search
     * @return A list of pages containing the word. The pages are
     *         ordered according to the relative importance of
     *         the word within them.
     */
    List<String> search(String word);
}
```


המתודות השונות:

• המתודה index

מתודה זו אחראית על אכלוס מבנה הנתונים שלכם. המתודה מקבלת אוסף של מילים (ייתכנו חזרות) ואת כתובת האינטרנט של הדף מהן הגיעו. עליכם לבחור את מבני הנתונים שבהם תשתמשו ולדאוג לשמור על הקשרים הבאים: לכל מילה באילו דפים היא מופיעה וכמה פעמים בכל דף, לכל דף כמה מילים בסי"כ מופיעות בו (עם חזרות). רשימת המילים בקלט היא כפי שהופיעה בדף המקורי, אולם יש לשמור גרסת lowercase של המילים.

• המתודה search

מקבלת מילה לחיפוש ומחזירה רשימה ממוינת של כתובות אינטרנט בהן המילה מופיעה. נמיון את הרשימה כך שכלל שהמשקל היחסי של מילה בדף גבוה יותר כך הכתובת תופיע במקום גבוה יותר ברשימה.

המשקל היחסי של מילה בדף יהיה מספר המופעים של המילה בדף חלקי מספר המילים הכולל באותו דף.

הדרכה:

השתמשו במחלקות מ Java Collections, בפרט המנשק Map והמחלקה HashMap יכולים להועיל.

את מיון הרשימה של כתובות האינטרנט בצעו בעזרת הפונקציה Collections.sort(...). שימו לב שה"מיון הטבעי" של הכתובות (String) הוא מיון לקסיקוגרפי ולא כפי שמתבקש בשאלה. כדי לשנות את שיטת המיון עליכם לכתוב מחלקה המממשת את המנשק Comparator. שימו לב שתוצאות ההשוואה תלויות במלת החיפוש הנוכחית.

הערה: המילים המתקבלות מדף ה-HTML אינן "נקיות" (כוללות סימני פיסוק וכדומה). אין צורך לבצע כל פעולה על מילים אלא להשתמש בהן כמו שהן.

כדי לבדוק את התכנית שלכם כיתבו מחלקה ובה פונקציית main המייצרת אובייקט חדש מטיפוס SearchEngine ומעבירה לו בבנאי מופע של המחלקה MyWordIndex שמימשתם. לאחר יצירת האובייקט ייקרא השירות run. לדוגמא:

```
public class Main {
    public static void main(String[] args) {
        SimpleSearchEngine searchEngine =
            new SimpleSearchEngine(new MyWordIndex());
        searchEngine.run();
    }
}
```

דוגמאות:

עבור רשימת הכתובות המופיעה בקובץ ה Main (המוכן מראש, ראו בהמשך) ומילת החיפוש "java" יתקבל הפלט:

```
> java
1. http://www.java.com/en/
2. http://en.wikipedia.org/wiki/Java_(programming_language)
3. http://en.wikipedia.org/wiki/Java
4. http://www.gutenberg.org/files/27152/27152-h/27152-h.htm
```

הערות כלליות :

בשני חלקי התרגיל חלק מהקוד כבר נכתב בעבורכם ועליכם לספק מימוש לחלקים החסרים. תוכלו לייבא את הקוד הקיים ע"י שתורידו את קבצי הזיפ המופיעים באתר אליכם למחשב. כעת תוכלו לייבא את הפרויקט בעזרת תפריט הייבוא (File->import) של Eclipse.

בחרו בייבוא של פרויקט קיים (Existing Projects into Workspace) לאחר מכן הוסיפו את קובץ הזיפ בעזרת Select Archive File ובחירת הקובץ המתאים.

בהצלחה !