

# תוכנה 1 – סתיו תשע"ג

## תרגיל מספר 7

### הנחיות כלליות:

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

- הגשת התרגיל תעשה במערכת ה moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer\_hw7.zip). קובץ ה-zip יכיל:
  - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
  - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש.
  - ג. קובץ טקסט אחד עם העתק של כל קבצי ה-java.

### חלק א' (30 נק')

בחלק זה עליכם לממש שתי מתודות עבור מבנה מקושר נתון.

בשני המקרים, עליכם לבצע מעבר יחיד על הרשימה. לא ניתן להעתיק את תוכן הרשימה למבנה נתונים מטיפוס אחר (למשל מערך). ניתן להניח שהרשימה לא ריקה (מכילה תא אחד לפחות). ה-next של התא האחרון ברשימה הוא null. השתמשו 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;
    }
}
```

1. עליכם לממש את המתודה הבאה, אשר מחזירה את התא האמצעי ברשימה. אם מס' התאים N הוא אי זוגי, נחזיר את התא ה- $(N+1)/2$  (למשל, עבור רשימה באורך 3 נחזיר את התא השני). עבור רשימה באורך N זוגי נחזיר את התא ה- $N/2$  (למשל, עבור רשימה באורך 4 נחזיר את התא השני).

```
public static LinkedListNode getMiddleNode(LinkedListNode head)
```

מצאו את התא האמצעי תוך שימוש במשתני עזר בודדים. אל תצרו עותק של הרשימה משום טיפוס.

2. ממשו את המתודה הבאה, אשר מבצעת מעין "טריפת קלפים" בחוליות הרשימה. טריפת הקלפים מתבצעת כך: התא השני מצורף לאחור התא הראשון; התא השלישי מצורף לפני התא הראשון; התא הרביעי מצורף אחרי התא השני וכך הלאה. כלומר, לסירוגין מוסיפים חוליה לוסף ולתחילת הרשימה.

```
public static LinkedListNode cardShuffle(LinkedListNode head)
```

את תוצאת "טריפת הקלפים" יש ליצור כרשימה חדשה, כך שעבור כל `LinkedListNode` ברשימה המקורית נוצר `LinkedListNode` בעל אותו הערך ברשימה החדשה. ערך ההחזרה של המתודה הוא החוליה הראשונה ברשימה החדשה.

לדוגמא, אם הרשימה ההתחלתית הייתה:

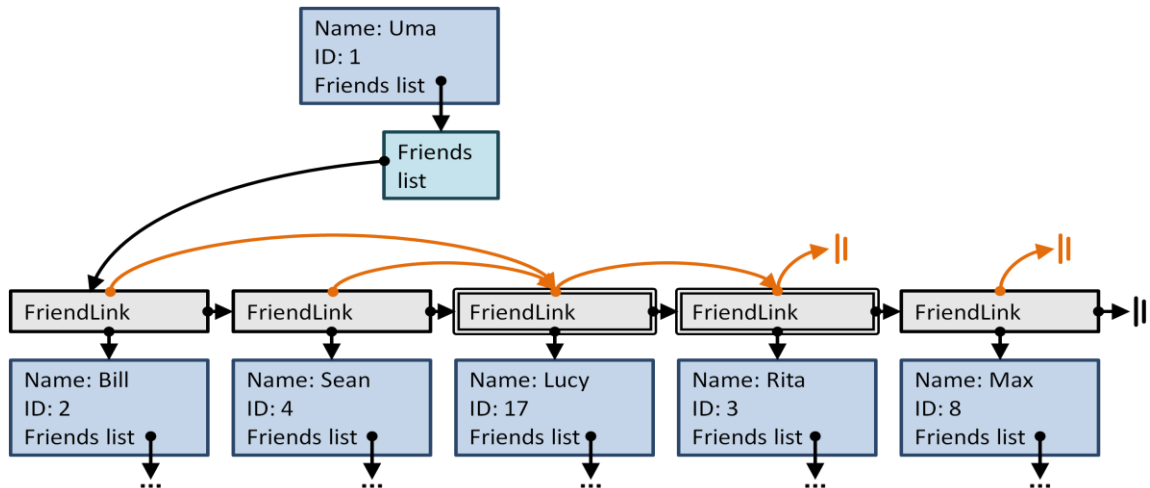
1 2 3 4 5 6 7 8 9

לאחר טריפת הקלפים, הרשימה תהיה

9 7 5 3 1 2 4 6 8

## חלק ב' (70 נק')

בחלק זה של התרגיל נכתוב מחלקות לניהול רשימת חברים ברשת חברתית. ברשת שלנו, לכל משתמש יש שם, מזהה ייחודי, וחברים המופיעים ברשימה מקושרת לפי סדר הוספתם כחברים. ניתן להגדיר חלק מן החברים ברשימה כ"חברים קרובים". מופעי המחלקה `SocialNetworkUser` ייצגו משתמשים ברשת החברתית. לכל משתמש תהיה רשימת חברים מטיפוס `FriendsList`. הרשימה המקושרת הזו תורכב מאיברים מטיפוס `FriendLink`, וכל איבר כזה יכיל מצביע ל- `SocialNetworkUser` הרלוונטי. להלן תרשים סכמטי המתאר חלק מן הקשרים בין המחלקות. קשרים אלה יוסברו בהמשך. שימו לב: לא כל המצביעים הדרושים מתוארים בתרשים, וכנראה שתצרו להגדיר מצביעים נוספים.



בצעו את המשימות הבאות. אתם רשאים להוסיף חוזים, שדות, מתודות, ומחלקות עזר לפי הצורך, אך שמרו על שמות מחלקות וחתימות של מתודות נתונות, ואל תשתמשו באוספים קיימים מהספרייה הסטנדרטית - עליכם לממש את המבנה המקושר בעצמכם.

1. כתבו את המחלקה FriendLink. מחלקה זו תכיל מצביע לעצם מטיפוס SocialNetworkUser, וְשני מצביעים נוספים: מצביע לחבר הבא ברשימה, ומצביע לחבר הקרוב הבא ברשימה. אובייקטים מהמחלקה FriendLink תמיד יצביעו לחבר הקרוב הבא, גם אם הם עצמם אינם חברים קרובים. אם אין חבר לחבר קרוב בהמשך הרשימה, המצביע המתאים יפנה ל-null. כאשר החבר הבא ברשימה הוא חבר קרוב, שני המצביעים יפנו לאותו החבר. **שימו לב** - המחלקה FriendLink לא תכיל שדה שמציין אם החבר קרוב או לא.

התרשים למעלה מתאר את רשימת חמשת החברים של אומה (לכל משתמש יש רשימת חברים משלו). לואי וריטה הן החברות הקרובות היחידות של אומה (חוליות ה-FriendLink המתאימות להן בתרשים מסומנות בקו מתאר כפול). כל FriendLink מכיל מצביע לזה שאחריו ברשימה (חץ שחור), אך גם לחבר הקרוב הבא (חץ כתום). למשל, במקרה של ביל ושון, החברה הקרובה הבאה ברשימה היא לואי, ואחרי ריטה ומקס אין חברה קרובה ועל כן מצביע החבר הקרוב שלהם שווה ל-null.

2. כתבו את המחלקה FriendsList. מחלקה זו תייצג רשימה מקושרת של עצמים מסוג FriendLink, ותאפשר גישה להתחלת ולסוף הרשימה. חישוב מה יהיה הייצוג הפנימי של מחלקה זו (בתרשים הסכמטי מתואר מצביע לתחילת הרשימה, אך אתם יכולים להשתמש בשדות נוספים). בהוספת חבר הרשימה תדאג **לא ליצור כפילויות**, וכן **שחברות היא הדדית** - כאשר ביל מתווסף לרשימת החברים של אומה, אומה צריכה להתווסף לרשימת החברים של ביל; אם אומה חברה קרובה של ביל אז ביל חבר קרוב של אומה.

להלן רשימת המתודות שעליכם לממש במחלקה זו.

```
/**
 * Returns the user that this friends list belongs to
 */
public SocialNetworkUser getUser()

/**
 * Returns the friend that was added first to the friends list
 */
public SocialNetworkUser getOldestFriend()

/**
 * Returns the oldest friend that is also a close one
 * (this friend didn't necessarily become close first)
 */
public SocialNetworkUser getOldestCloseFriend()

/**
 * Returns the number of friends in the list
 */
public int getNumFriends()

/**
 * Returns the number of close friends in the list
 */
public int getNumCloseFriends()
```

```

/**
 * @pre user != null
 * @pre user != getUser()
 *
 * If user is already a friend, nothing happens.
 * Otherwise, the user is added as a friend to the end of the list.
 * Add this.getUser() to the friends list of user
 * Make sure you don't create an endless loop!
 */
public void addFriend(SocialNetworkUser user)

/**
 * If user is not in the friends list, nothing happens.
 * Otherwise, find user in the friends list, make him a close friend and
 * update all the relevant fields and pointers of other friends in the
 * list accordingly.
 * Make this.getUser() a close friend in the list of user.
 */
public void upgradeToCloseFriend(SocialNetworkUser user)

```

3. הוסיפו ל- FriendsList את המתודה הבאה, שמאפשרת לחפש חברים בדרגות קרבה שונות. חברים שמופיעים ברשימה הם מדרגה 0. חברים של חברים הם מדרגה 1, וכך הלאה. הפרמטר onlyClose מאפשר לחפש רק בקרב חברים קרובים, חברים קרובים של חברים קרובים וכך הלאה.

```

/**
 * @pre user != null
 * @pre degree >=0
 *
 * Returns true iff user is a friend up to the given degree,
 * where friends in the list are of degree 0,
 * friend of friends are of degree 1 and so on.
 * If onlyClose==true, search only among close friends.
 */
public boolean isAFriendUpToDegree(SocialNetworkUser user, int degree,
                                   boolean onlyClose)

```

נניח שהפעם לאומה יש שני חברים, ביל (חבר רגיל) ושון (חבר קרוב), ושאלה החברויות היחידות ברשת. אזי אלו החברים של שון מדרגות שונות:

דרגת קרבה	חברים	חברים קרובים
0 (חברים)	אומה	אומה
1 (חברים של חברים)	שון, ביל	שון
2 (חברים של חברים של חברים)	אומה	אומה

שימו לב שבפרט, כל אחד הוא חבר מדרגה 1 של עצמו אמ"מ יש לו חבר אחד לפחות, וחבר קרוב מדרגה 1 של עצמו אמ"מ יש לו חבר קרוב אחד לפחות.

המתודה תחזיר true עבור חבר X ודרגה Y אם X הוא חבר מדרגת קרבה בין 0 ל-Y. בדוגמא שלנו:

```

seansFriendsList.isAFriendUpToDegree(uma, 0, true) //true (Uma is a close
friend of degree 0)
seansFriendsList.isAFriendUpToDegree(uma, 1, false) //true (Uma is a friend
of degree 0 < 1)
billsFriendsList.isAFriendUpToDegree(bill, 1, true) //false (Bill has no
close friends)

```

4. כתבו את המחלקה SocialNetworkUser. עליה להכיל את המתודות הבאות.

```
/**
 * @pre name!=null
 * @pre name.length() > 0
 * @post getName() == name
 * @post getId() returns an ID unique for this user
 * @post getFriendsList() != null
 * @post getFriendsList().getNumFriends() == 0
 */
public SocialNetworkUser(String name)

public String getName()

public int getId()

public FriendsList getFriendsList()

/**
 * Returns the user details in the following format:
 * "Name: <name>, ID: <ID>, #friends: <num_friends> (<num_close_friends>
 *   close)"
 * For example,
 * Name: Uma, ID: 1, #friends: 2 (1 close)
 */
public String toString()
```

5. הוסיפו ל- FriendsList מתודה המאפשרת הדפסה של החברים ברשימה עם פרטיהם.

```
/**
 * Prints the friends list to System.out.
 * - denotes a "regular" friend and + denotes a close friend
 * Example for the format:
 * #friends: 2 (1 close)
 * - Name: Bill, ID: 2, #friends: 1 (0 close)
 * + Name: Sean, ID: 4, #friends: 1 (1 close)
 */
public void printFriends()
```

6. כתבו תכנית בדיקה בשם SocialNetworkTest. פונקציה ה-main של התכנית תיצור תחילה את המשתמשים אומה, ביל ושון. לאחר מכן היא תוסיף את ביל ואז את שון כחברים של אומה. אז, שון יהפוך להיות חבר קרוב של אומה. אחרי כל שלב, הדפיסו את התוצאה. היא יכולה להיראות למשל כך:

```
>>>>>> Start
Name: Uma, ID: 1, #friends: 0 (0 close)
#friends: 0 (0 close)

Name: Bill, ID: 2, #friends: 0 (0 close)
#friends: 0 (0 close)

Name: Sean, ID: 3, #friends: 0 (0 close)
#friends: 0 (0 close)

>>>>>> After adding friends
Name: Uma, ID: 1, #friends: 2 (0 close)
#friends: 2 (0 close)
- Name: Bill, ID: 2, #friends: 1 (0 close)
- Name: Sean, ID: 3, #friends: 1 (0 close)
```

```
Name: Bill, ID: 2, #friends: 1 (0 close)
#friends: 1 (0 close)
- Name: Uma, ID: 1, #friends: 2 (0 close)

Name: Sean, ID: 3, #friends: 1 (0 close)
#friends: 1 (0 close)
- Name: Uma, ID: 1, #friends: 2 (0 close)

>>>>>> After upgrade
Name: Uma, ID: 1, #friends: 2 (1 close)
#friends: 2 (1 close)
- Name: Bill, ID: 2, #friends: 1 (0 close)
+ Name: Sean, ID: 3, #friends: 1 (1 close)

Name: Bill, ID: 2, #friends: 1 (0 close)
#friends: 1 (0 close)
- Name: Uma, ID: 1, #friends: 2 (1 close)

Name: Sean, ID: 3, #friends: 1 (1 close)
#friends: 1 (1 close)
+ Name: Uma, ID: 1, #friends: 2 (1 close)
```

חשבו בעצמכם על בדיקות נוספות (של מתודות נוספות ושל מקרי קצה). יש להגיש את המחלקות SocialNetworkTest, SocialNetworkUser, FriendsList, FriendLink ומחלקות עזר אחרות בהן השתמשתם.