חומרים שהוכנו על-ידי משתתפי קורס מורים מובילים תשע''ה

ניתן להשתמש בחומרים לצורך הוראה בלבד.

לא ניתן לפרסם את החומרים או לעשות בהם כל שימוש מסחרי

ללא קבלת אישור מראש מצוות הפיתוח

כתיבה ועריכה:

אביטל גרינולד, דורית כהן, משה שטיינר

מחלקת שירות הכוללת אוסף פעולות שימושיות
המטפלות ברשימה מקושרת של מספרים שלמים

מדריך למורה

מתאים לשתי שפות התכנות: **Java** ו- **C#**

מטרת המשימה: בניית מחלקת שירות המכילה אוסף פעולות שימושיות על רשימה של מספרים שלמים.

מבנה המסמך:

- מטרות ודגשים
- הוראות לתלמיד
- ממשק המחלקה חוליה Node<T>
- הנחיות עבודה

מטרות ודגשים:

- הכרות עם מבנה נתונים דינמי **רשימה.**

- לחדד את המושג של הפנייה ועצם המכיל תכונה שהיא הפנייה לאותו העצם, שבעזרתו ניתן לבנות את המבנה רשימה.

- הפנייה לחוליה מקבלת משמעויות שונות בהקשרים שונים:
  - רשימה, כאשר הכוונה הפנייה לחוליה הראשונה ברשימה
  - חוליה בודדת להוספה או מחיקה
  - מיקום ברשימה

- אין חוליה ריקה אבל יש רשימה ריקה: זוהי רשימה ללא חוליות. בהפניה אליה נציב את הערך null .

- יש לתת את הדעת לכך שבפעולות היוצרות שינוי על הרשימה נכתוב פעולות המחזירות הפנייה לרשימה חדשה ולא void . בניגוד למה שהכרנו במערכים וזאת עקב בעייתיות של טיפול בחוליה ראשונה ברשימה או ברשימה ריקה.

- לשם פשטות, החלטנו, באופן שרירותי, לבנות מחלקת שירות לטיפול ברשימה של שלמים. בסיכום המעבדה מומלץ לערוך דיון עם התלמידים, המברר אילו פעולות ניתן ליישם על חוליה גנרית, ומהם השינויים הדרושים לכך.

ממשק המחלקה : **Node**<T>

יעילות	כותרת הפעולה - Java	כותרת הפעולה – #C	תיאור הפעולה
O(1)	public **Node**(T x)	public **Node**(T x)	יוצרת חוליה בעלת ערך x והפנייה ל null
O(1)	public **Node**(T x, Node<T> next)	public **Node**(T x, Node<T> next)	יוצרת חוליה בעלת ערך x והפנייה לחוליה next
O(1)	public T **getValue**()	public T **GetValue**()	מחזירה את ערך החוליה
O(1)	public Node<T> **getNext**()	public T Node<T> **GetNext**()	מחזירה את ההפניה לחוליה העוקבת או null
O(1)	public void **setValue** (T x)	public void **SetValue** (T x)	משנה את ערך החוליה ל x
O(1)	public void **setNext** (Node<T> next)	public void **SetNext** (Node<T> next)	משנה את ההפניה לחוליה הבאה ל next
O(1)	public boolean **hasNext**()	public bool **HasNext**()	מחזירה 'אמת' אם יש חוליה עוקבת, 'שקר'-אחרת.
O(1)	public String **toString**()	public String **ToString**()	מחזירה מחרוזת המתארת את ערך החוליה

<u>הנחיות והערות לתלמיד</u>

(1) ביצירת המחלקות יש להשתמש בטיפוס Node<T>

Java לשם כך ייבא את המחלקה מחבילת התוכנות unit4

```
import unit4.collectionsLib.Node;
```

C# לשם כך הוסף את ספרית unit4.dll לרשימת ה-reference,
בנוסף, יש להוסיף התייחסות לספריה בראש הקובץ: using Unit4.CollectionsLib;
זכרו למחוק את כל משפטי ה-using האחרים למעט using System;

(2) הוסף בכל פעם מימוש של פעולה אחת ובדוק אותה. לאחר שבדקת שהיא עובדת - עבור לפעולה הבאה וכך הלאה.

(3) ממש את הפעולות שמסומנות ב (*) הן באופן איטרטיבי (בלולאה) והן באופן רקורסיבי.
הוסף לשם הפעולה R, למשל:
Java הפעולה האיטרטיבית sum תיקרא sumR בצורתה הרקורסיבית.
C# הפעולה האיטרטיבית Sum תיקרא SumR בצורתה הרקורסיבית.

(4) לכל אחת מהפעולות, נתח את היעילות. נמק תשובתך.

(5) באומרנו "רשימה" , הכוונה להפניה לחוליה ראשונה ברשימה או null אם היא ריקה.

(6) הקפד לתת שמות שמרמזים על תפקיד ההפניה לעצם מטיפוס Node<T> .
- אם משמעותו רשימה , קבע שם כגון: lst, list וכדומה.
- אם משמעותו חוליה בודדת, קבע שם כגון: node .
- אם משמעותו מקום ברשימה, קבע שם כגון: position, pos, p וכדומה.

<u>הנחיות והערות למורה</u>

(1) אוסף הפעולות המופיעות במסמך לתלמיד מדורגות מהקל לקשה.

(2) הפעולות 13-1 הן בסיסיות ומומלץ שכל תלמיד יממש אותן.

(3) מומלץ לוודא שהתלמידים מגוונים את מימוש הפעולות, חלק באופן איטרטיבי וחלק באופן רקורסיבי.

(4) אוסף פעולות זה הוא בגדר המלצה. כל מורה יכול לבחור כראות עיניו אילו פעולות לתת למעבדה בכתה, איזה לשיעורי בית ואיזה לבחנים או מבחנים.

(5) הפעולות חולקו לקבוצות על פי סוגים שונים.

(6) מומלץ שהתלמיד יממש לפחות פעולה אחת מכל קבוצה.

(7) לגבי פעולות מורכבות יותר, מומלץ לערוך דיון עם התלמידים על אופן המימוש ולהציג פתרונות שונים בפני הכתה. חשוב לדון ביעילות הפעולות על פי המימוש הנבחר.

(8) יש לשים לב למימוש פשוט של פעולה תוך זימון פעולה אחרת מאוסף הפעולות.
לדוגמה בשפת Java: מימוש הפעולה isSameValues ישתמש בפעולה isAllExist .
בשפת C# : מימוש הפעולה IsSameValues ישתמש בפעולה IsAllExist .

(9) במימוש רקורסיבי, ניתן לזמן פעולת עזר רקורסיבית עם פרמטרים נוספים לפתרון פשוט וקל.

(10) באוסף הפעולות קיימים שני סוגים של מיון: מיון-הכנסה ומיון-בחירה.
מיון-הכנסה משתמש בפעולה הכנס-לרשימה-ממוינת ולכן הפעולה תחזיר הפנייה לרשימה חדשה.
לעומת זאת, במיון-בחירה יש למצוא בכל שלב את המיקום של הערך הקטן ביותר מהמיקום הנוכחי ועד לסוף הרשימה ולבצע החלפה. במקרה זה הפעולה לא תחזיר רשימה חדשה אלא תשנה את הקיימת. כלומר, הפעולה תוגדר כ- void .

<u>התייחסות למימוש חלק מהפעולות</u>

פעולה 12: סכום ערכים במקומות אי-זוגיים.
יש לשים לב שלא יבצעו פעמיים קידום על הרשימה מבלי לברר קודם שניתן.

פעולות 27+26: מיזוג וחיתוך רשימות ממוינות
הציגו בפני התלמידים שתי אפשרויות:
<u>האחת</u>, שיוצרת רשימה חדשה ויוצרת חוליות חדשות עם ערכים מהרשימות המקוריות תוך שמירה על סדר ממוין.
<u>השנייה</u>, מחזירה הפנייה לרשימת המיזוג או רשימת החיתוך תוך שימוש בחוליות הקיימות של שתי הרשימות.

פעולה 28: האם כל ערכי רשימה 2 נמצאים ברשימה 1.
בשפת Java : שימוש בפעולה isExist
בשפת C# : שימוש בפעולה IsExist

פעולה 29: האם הערכים בשתי הרשימות זהים , לא חשוב הסדר.
בשפת Java: שימוש בפעולה isAllExists
בשפת C# : שימוש בפעולה IsAllExists

פעולה 30: האם אין ערכים משותפים בשתי הרשימות
בשפת Java : שימוש ב isExist
בשפת C# : שימוש בפעולה IsExist

שאלה לדיון: איך היית משנה את מימוש הפעולה אם שתי הרשימות היו ממוינות בסדר עולה?
תשובה: שימוש בפועלה cut העובדת על רשימות ממוינות.

פעולה 32: האם רשימה 2 מוכלת ברשימה 1
הפעולה משתמשת בפעולה 30: האם רשימה 2 מוכלת ברשימה 1 מראשיתה.