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

גרסת ג'אווה

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

צור פרויקט **MyList** .
בנה את המחלקות: **UtilListInt**
TestUtilListInt

המחלקה UtilListInt תכיל את הפעולות המפורטות להלן,
תוכנית הבדיקה TestUtilListInt תזמן אותן ותציג פלטים ברורים.

רשימה מקושרת או בקיצור **רשימה** הוא מבנה דינמי הבנוי מאפס חוליות או יותר.
רשימה ריקה היא רשימה ללא חוליות , כלומר ייצוג שלה יהיה: Node<T> list = null;
חוליה , **Node**<T> , הוא טיפוס המכיל ערך והפנייה לחוליה הבאה או null .

להלן ממשק המחלקה **Node**<T> :

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

הנחיות והערות:

(1) ביצירת המחלקות יש להשתמש בטיפוס Node<T> , לשם כך ייבא את המחלקה מחבילת התוכנות unit4
import unit4.collectionsLib.Node;

(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.
לפעולות במימוש רקורסיבי הוסף את האות R לשם הפעולה.
למשל פעולה בשם **sum** תקבל את השם **sumR**.

(4) לכל אחת מהפעולות, רשום 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, nextPos, prevPos וכדומה.

(א) פעולות בנייה		
1	פעולה המקבלת מספר טבעי n ומחזירה רשימה של מספרים טבעיים בסדר עולה מ 1 עד n כולל.	**public static** Node<Integer> **create1_N**(**int** n)
2	פעולה המקבלת הפנייה לחוליה ראשונה ברשימה של מספרים שלמים ומדפיסה אותה כך: n1 → n2 → … → null	**public static void print** (Node<Integer> list)
3	פעולה היוצרת רשימה עם n מספרים אקראיים בין low ל up כולל כאשר n מספר טבעי ומחזירה הפנייה לחוליה הראשונה ברשימה.	**public static** Node<Integer> **buildRandom** (**int** n, **int** low, **int** up)
4	פעולה הבונה רשימה של מספרים שנקלטים מהמשתמש. סוף קלט 99. הפעולה מחזירה הפנייה לחוליה ראשונה ברשימה, null אם היא ריקה.	**public static** Node<Integer> **create**()

(ב) פעולות המחזירות ערך מספרי		
5	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את אורכה, כלומר את מספר האיברים ברשימה. (*)	**public static int count** (Node<Integer> list)
6	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את סכום ערכיה. (*)	**public static int sum** (Node<Integer> list)
7	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את ממוצע ערכיה. 0 - אם הרשימה ריקה.	**public static double average** (Node<Integer> list)
8	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את מכפלת ערכיה. הנחה: הרשימה אינה ריקה. (*)	**public static int product** (Node<Integer> list)
9	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את הערך המקסימלי . הנחה: הרשימה אינה ריקה. (*)	**public static int max** (Node<Integer> list)
10	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את הערך הכי קטן . הנחה: הרשימה אינה ריקה. (*)	**public static int min** (Node<Integer> list)
11	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את סכום הערכים הזוגיים. (*)	**public static int sumEven** (Node<Integer> list)
12	פעולה המקבלת רשימה של מספרים שלמים ומחזירה את סכום הערכים במקומות האי-זוגיים.	**public static int sumOddPlaces** (Node<Integer> list)
13	פעולה המקבלת רשימה של מספרים שלמים ומספר שלם נוסף x , ומחזירה את מספר הפעמים שהמספר x מופיע ברשימה. (*)	**public static int countX** (Node<Integer> list, **int** x)

(ג) פעולות המחזירות ערך בוליאני		
14	הפעולה מקבלת רשימה של מספרים שלמים ומחזירה 'אמת' אם היא ממוינת מסודרת בסדר עולה,' שקר' – אחרת. עבור רשימה ריקה תחזיר 'אמת'. (*)	**public static boolean isUp** (Node<Integer> list)
15	הפעולה מקבלת רשימה של מספרים שלמים ומספר שלם נוסף x, ומחזירה 'אמת' אם המספר x נמצא ברשימה, 'שקר' – אחרת. (*)	**public static boolean isExist**(Node<Integer> list, **int** x)
16	הפעולה מקבלת רשימה של מספרים שלמים ומחזירה 'אמת' אם היא מהווה סדרה חשבונית,' שקר' – אחרת. עבור רשימה ריקה תחזיר 'אמת'. (*) **סדרה חשבונית** היא סדרת מספרים שההפרש בין כל שני ערכים סמוכים זהה.	**public static boolean isArithmetic** (Node<Integer> list)

(ד) פעולות המחזירות הפנייה למקום ברשימה		
17	פעולה המקבלת רשימה של מספרים שלמים לא ריקה ומספר צעדים step ומחזירה את ההפניה לחוליה במקום step. אם step גדול ממספר החוליות, תחזיר null . (*)	**public static** Node<Integer> **goto** (Node<Integer> list, **int** step)
18	פעולה המקבלת רשימה של מספרים שלמים לא ריקה ומחזירה את ההפניה לחוליה אחרונה, או null אם הרשימה ריקה (*)	**public static** Node<Integer> **gotoLast** (Node<Integer> list)
19	פעולה המקבלת רשימה של מספרים שלמים ומספר שלם נוסף x ומחזירה הפניה למקום ההופעה הראשון של x מתחילת הרשימה, או null – אם לא נמצא. (*)	**public static** Node<Integer> **firstPosX**(Node<Integer> list, **int** x)
20	פעולה המקבלת רשימה של מספרים שלמים ומחזירה הפנייה למיקום הערך הקטן ביותר ברשימה. הנחות: הרשימה אינה ריקה. כל הערכים שונים זה מזה.	**public static** Node<Integer> **posMin** (Node<Integer> list)

הערות: (1) בכל הפעולות הבאות נחזיר הפנייה לחוליה ראשונה ברשימה חדשה אשר נוצרה מרשימה קיימת או null .
באומרנו: מחזיר רשימה, הכוונה מחזיר הפניה לחוליה הראשונה ברשימה או null .
(2) בכל הפעולות הבאות , הנח pos או prev הם מקומות ברשימה או null.

(ה) שינוי רשימה קיימת, בפועל מחזיר רשימה חדשה		
21	פעולה המקבלת רשימה, מקום ברשימה pos וערך x ומחזירה רשימה חדשה כאשר הערך x נוסף מקום אחד אחרי pos . אם pos=null , תוחזר רשימה בה x התווסף לפני חוליה ראשונה. יעילות הפעולה היא O(1)	**public static** Node<Integer> **insert**(Node<Integer> list, Node<Integer> pos, **int** x)
22	פעולה המקבלת רשימה, מספר צעדים step וערך x ומחזירה רשימה חדשה בה הערך x נוסף אחרי step צעדים מתחילת הרשימה. אם step=0 , x יתווסף לפני חוליה ראשונה. אם step גדול מאורך הרשימה, x יתווסף בסוף הרשימה.	**public static** Node<Integer> **insert**(Node<Integer> list, **int** step, **int** x)
23	פעולה המקבלת רשימה ומקום ברשימה prev ומחזירה רשימה חדשה ללא החוליה העוקבת ל prev. אם prev=null , תוחזר רשימה ללא החוליה הראשונה. יעילות הפעולה היא O(1)	**public static** Node<Integer> **delete**(Node<Integer> list, Node<Integer> prev)
24	פעולה המקבלת רשימה ומספר צעדים step מתחילת הרשימה ומחזירה רשימה חדשה ללא הערך הנמצא במיקום step .	**public static** Node<Integer> **delete**(Node<Integer> list, **int** step)

(ו) פעולות על רשימות ממוינות		
25	פעולה המקבלת רשימה ממוינת בסדר עולה ומספר שלם x ומחזירה רשימה חדשה הכוללת את הערך x כך שהיא שומרת על המיון שלה.	**public static** Node<Integer> **insertSort**(Node<Integer> list, **int** x)
26	פעולה המקבלת שתי רשימות ממוינות בסדר עולה. בכל רשימה הערכים שונים זה מזה. הפעולה מחזירה רשימה חדשה שהיא מיזוג של שתי הרשימות. כל ערך מופיע רק פעם אחת לכל היותר. הרשימה החדשה מכילה את הערכים משתי הרשימות בסדר עולה. מותר לשנות את הרשימות המקוריות. המיזוג יתבצע ביעילות ליניארית.	**public static** Node<Integer> **merge**(Node<Integer> list1, Node<Integer> list1)
27	פעולה המקבלת שתי רשימות ממוינות בסדר עולה. בכל רשימה הערכים שונים זה מזה. הפעולה מחזירה רשימה חדשה שהיא חיתוך של שתי הרשימות. כלומר: הרשימה החדשה תכיל את הערכים הנמצאים בשתי הרשימות והיא בסדר עולה. מותר לשנות את הרשימות המקוריות. החיתוך יתבצע ביעילות ליניארית.	**public static** Node<Integer> **cut**(Node<Integer> list1, Node<Integer> list1)

הערה: בפעולות הבאות הכוונה לערכים שבחוליות ולא הפניות לחוליות עצמן.

(ז) פעולות על שתי רשימות		
28	פעולה המקבלת שתי רשימות list1, list2 , ומחזירה 'אמת' אם כל איברי list2 נמצאים בסדר כלשהו ב-list1, 'שקר'-אחרת.	**public static boolean isAllExist** (Node<Integer> list1, Node<Integer> list2
29	פעולה המקבלת שתי רשימות list1, list2 ומחזירה 'אמת' אם שתי הרשימות מכילות אותן ערכים ללא חשיבות לסדר, 'שקר'-אחרת.	**public static boolean isSameValues** (Node<Integer> list1, Node<Integer> list2)
30	פעולה המקבלת שתי רשימות list1, list2 ומחזירה 'אמת' אם לשתי הרשימות אין ערכים משותפים, 'שקר'-אחרת.	**public static boolean isAllDifferent** (Node<Integer> list1, Node<Integer> list2)
31	פעולה המקבלת שתי רשימות list1,list2 ומחזירה 'אמת' אם הרשימה list2 מוכלת בשלמותה ברשימה list1 החל מראשית הרשימה list1, 'שקר'-אחרת.	**public static boolean isIncludeStart** (Node<Integer> list1, Node<Integer> list2)
32	פעולה המקבלת שתי רשימות list1,list2 ומחזירה 'אמת' אם רשימה list2 מוכלת בשלמותה ברשימה list1, 'שקר'-אחרת.	**public static boolean isInclude** (Node<Integer> list1, Node<Integer> list2)

(ח) פעולות מיון רשימות		
33	פעולה הבונה רשימה ממוינת בסדר עולה ממספרים שנקלטו מהמשתמש. סוף קלט 99. הפעולה מחזירה הפנייה לחוליה הראשונה ברשימה, null אם היא ריקה.	**public static** Node<Integer> **createSorted**()
34	פעולה המקבלת רשימה של מספרים בסדר כלשהו ומחזירה רשימה ממוינת בסדר עולה בשיטת **מיון-הכנסה insertion-sort**	**public static** Node<Integer> **insertionSort** (Node<Integer> list)
35	פעולה המקבלת רשימה של מספרים בסדר כלשהו ומחזירה רשימה ממוינת בסדר עולה בשיטת **מיון-בחירה selection-sort**	**public static void selectionSort** (Node<Integer> list)

הנחיה והסבר:
ב**מיון-הכנסה** יוצרים רשימה ריקה ומכניסים אליה את הערכים מהרשימה תוך שמירה על סדר עולה.
בפעולה זו נעשה שימוש בפעולה **הכנס-לרשימה-ממוינת**.
ב**מיון-בחירה** עוברים על הרשימה המקורית ומוצאים את הערך הקטן ביותר ומחליפים אותו עם הערך הנוכחי.