מערכי שיעור במבנה נתונים בשפת ג'אווה , כתבה: אביטל EVI גרינולד

תוכן עניינים

מעבדה / דף	נושא	במה עוסק	עמודים
1	מחלקות ועצמים	הפניות ופעולה בונה מעתיקה, מערך של עצמים	2-4
2	מחלקות ועצמים	יחס הכלה בין מחלקות, עצם מורכב	5-6
3	מחלקות ועצמים	עצם מורכב עם תכונה מערך של עצמים	7-8
4	רקורסיה	פעולות של הדפסות רקורסיביות	9-11
5	רקורסיה	פעולות רקורסיביות המחזירות ערך	12-15
6	רשימה	מעקב ואיתור שגיאות	16-18

רב השיעורים שאני מעבירה לתלמידים הם בצורת מעבדות ולכן מה שמצורף לכאן הן אוסף של מעבדות בנושאים שונים ליחידה – מבנה נתונים מבוסס על שפת ג'אווה.
על פי רב, בכל מעבדה יש משהו חדש ושימוש בדברים שלמדו קודם.
צורת ההוראה היא ספירלית - מעגלית וכך התלמיד מטמיע את החומר טוב יותר.

<u>המלצה</u>:
השאירו פעולות נוספות שלא יופיעו בדפי המעבדה ויישארו לצורך בחינה.

למורה

מעבדה מספר 1: פעולה בונה מעתיקה והפניות

השיעור מבוסס על מעבדה Point מספר עצוב תכנה בשפת ג'אווה של האוניברסיטה העברית בירושלים.
פרק 3 , דך עבודה מספר 1

ידע קודם:
- ✓ בניית מחלקות ויצירת מופעי עצם ממחלקות.
- ✓ תרשים עצמים
- ✓ מערך של עצמים
- ✓ מיונים

מטרת המעבדה:
- הבחנה בין השמה של הפניות ליצירת העתק של העצם, שימוש בפעולה בונה מעתיקה
- השוואת עצמים
- הכרה עם המושג **העמסת פעולות overloading**
- בנית מערך של עצמים.
- זימון פעולות שעצם אחד מפעיל על עצם אחר במעבר על אברי מערך העצמים.
- מיון מערך.

לאחר שהתלמידים יצרו את מחלקת נקודה במישור Point , התלמידים מרחיבים את המחלקה ומוסיפים, פעולה בונה מעתיקה.

תשומת לב ל-
- מה גודל מערך המרחקים, מערך distances ?
- יצירת מרחק בין שתי נקודות סמוכות בעזרת זימון פעולת distance של עצם מטיפוס Point ולא שיכפול קוד
- במעבר על לולאת מערך הנקודות וטיפול בזוגות סמוכים, יש לשים לב לא לחרוג מגבולות מערך.
- ניתן להפעיל נקודה במקום index, index_1 או index,index-1

המלצות
- ניתן לתת דף מעבדה זה לעבודה עצמית ולבקש מהתלמידים שיסיקו מסקנות.
- חשוב שהמורה יסכם את הנקודות החשובות המתקבלות ממעבדה זו
- כדאי ורצוי לתת בוחן קצרצר שבודק שאכן התלמידים הבינו את הנלמד ממעבדה זו

מעבדה 1: העמסת פעולות, פעולה בונה מעתיקה, מערך של עצמים

מטרת התרגיל:
חידוד נושא הפניות, צורך בפעולת euqals ופעולה בונה מעתיקה, העמסת פעולות.

(1) צור פרויקט **Points** .

(2) העתק אליו את מחלקה **Point** שיצרת בעבר.
דף העבודה נמצא בפרק 3 בספר עצוב תכנה: דף עבודה 1 עמוד 65-67

(3) צור מחלקה **TestPoint** אשר מכילה את ההוראות הבאות:

```
Point p1 = new Point(1.5,5);
Point p2 = new Point(1.5,5);
Point p3 = p1;
System.out.println("p1==p2 --> " + (p1==p2));
System.out.println("p1==p3 --> " + (p1==p3));
```

א- לפני שאתה מריץ את התוכנית , שער וכתוב מה יהיה הפלט. לווה את הפלט בתרשים עצמים.
ב- הרץ והשווה למה שרשמת . האם לזה ציפית?
אם לא, נסה להבין מה ההבדל בין הוראות (2) ו (3)

(4) חזור למחלקה Point והוסף את הפעולה: `public boolean equals(Point p2)`
אשר מחזירה 'אמת' אם נקודה p2 נמצאת באותן קורדינטות כמו הנקודה המפעילה את הפעולה , 'שקר' – אחרת.

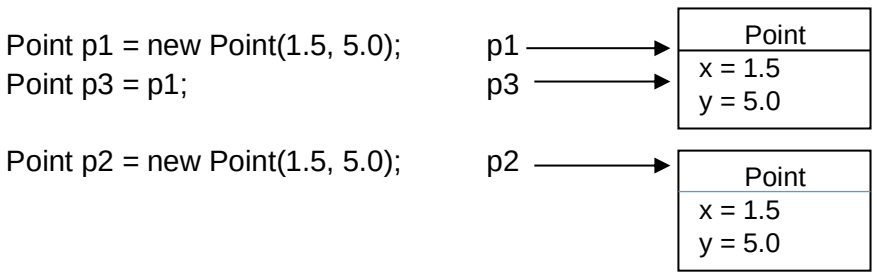
(5) חזור לתוכנית הראשית והוסף את הבדיקות הבאות:

```
System.out.println("check the method equals");
System.out.println("p1.euqlas(p2) --> " + p1.equals(p2));
System.out.println("p1.euqlas(p3) --> " + p1.equals(p3));
```

(6) כמה נקודות (עצם מטיפוס Point) נוצרו בתוכנית הראשית? לווה תשובתך בתרשים עצמים.

(7) א- מה יקרה לדעתך אחרי שנוסיף את הפעולה: p3.setX(2.0);
ב- שרטט את תרשים העצמים ורשום מה יוצג כפלט עבור הדפסת ערכי הנקודות p1,p2,p3 .
ג- הוסף הוראות בדיקה מתאימות לתוכנית הראשית וודא אם צדקת או לא.

שני משתנים מטיפוס מסוים **שווים** אם הם מפנים לאותו עצם.
בדוגמה שלנו: Point p3 = = p1 . למעשה יש כאן עצם אחד .
שינוי ערך של אחת התכונות במשתנה אחד, בהכרח יביא לשינוי אותה התכונה במשתנה השני.
ברגע שיצרנו משתנה מטיפוס מסוים בעזרת הפעולה **new** , ייצרנו הפנייה חדשה לעצם מהטיפוס.
בדוגמה שלנו: Point p2 = new Point(p1.getX(), p1.getY()); . נוצרו שני עצמים עם ערכים זהים.



פעולה בונה מעתיקה - Copy Constructor

הסברים ב פרק 3 בספר עצוב תכנה, עמוד 50

(8) הוסף פעולה **בונה מעתיקה** למחלקה **Point** , פעולה כזו מקבלת כפרמטר הפנייה לעצם מטיפוס Point ומעתיקה את התכונות שלו לנקודה החדשה.
כלומר יוצרת נעתק של הנקודה, כלומר: הפנייה לעצם חדש מטיפוס Point עם אותן הערכים.

```
public Point(Point p1)
{
  this.x = p1.x;
  this.y = p1.y;
}
```

(9) הוסף פעולה בונה ללא פרמטרים אשר יוצרת נקודה בראשית הצירים.

למנגנון שמאפשר ליצור פעולות עם אותו שם ופרמטרים שונים קוראים **העמסה** = **Overloading**

מערך של עצמים מטיפוס Point

(10) צרו מחלקה ArrayPoints המכילה פעולה ראשית.
הפעולה הראשית תבצע:

א- תקלוט מהמשתמש את מספר הנקודות, n .
ב- תיצור מערך בגודל n מטיפוס Point בשם arrPoints
ג- לכל נקודה תקלוט את ערכי x,y של הנקודה, תיצור מהן נקודה ותציב במערך הנקודות.
ד- תדפיס את אברי מערך הנקודות.
ה- תיצור מערך נוסף distances מטיפוס ממשי שיכיל את מרחקי הנקודות הסמוכות.
לשם יצירת מערך.
ו- תציג כפלט את מערך המרחקים.
ז- תמיין את מערך distances מקטן לגדול.
ח- תציג שוב כפלט את מערך המרחקים הממוין.

למורה

מעבדה מספר 2: עצם מורכב

השיעור מבוסס על מעבדה Point מספר עצוב תכנה בשפת ג'אווה של האוניברסיטה העברית בירושלים.
פרק 3 , דך עבודה מספר 1

ידע קודם:
- ✓ יצירת מחלקה
- ✓ מערך של עצמים
- ✓ פעולה בונה מעתיקה

מטרת המעבדה:
- יחס הכלה בין מחלקות / טיפוס מורכב
- תרשים עצמים ליחס הכלה
- שימוש בהעמסת פעולות בפעולה הבונה
- גילוי הצורך בזימון פעולה בונה מעתיקה בפעולה הבונה של המחלקה.
- גילוי הצורך בזימון פעולה בונה מעתיקה בפעולה get לתכונה שהיא עצם.

תשומת לב ל-
- תכונות המחלקה Segment הם שתי נקודות קצות המקטע, ללא התחשבות בכך שיש פרמטרים שונים בפעולה הבונה הראשונה
- תלמידים נוטים להוסיף תכונות למחלקה על פי הפרמטרים של הפעולה הבונה וזה ממש לא נכון נחוץ, מיותר ולא נכון ברב המקרים
- בפעולה הבונה שמקבלת כפרמטרים שתי נקודות, יש לבצע העתקים של הנקודות בעזרת זימון פעולה בונה מעתיקה
- בפעולה length , יש להשתמש בפעולה distance של Point ולא לחשב זאת מחדש. אי שיכפול קוד.
- הפעולה getPoint1 מחזירה הפנייה לנקודה, דונו עם התלמידים האם רוצים שיחזיר העתק של הנקודה או את הנקודה עצמה ועל פי זה שיממשו את הפעולה. (הדיון יהיה לאחר ביצוע המעבדה)

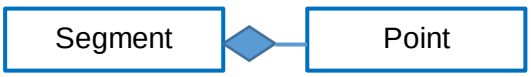
המלצות
- חשוב להרגיל את התלמידים לעבוד במקביל על שתי המחלקות , מחלקת שממנה יוצרים עצם ומחלקת הבדיקה. כל פעולה שמוסיפים במחלקה, מיד לבדוק במחלקת הבדיקה.
- בבניית מחלקה שממנה יוצרים עצמים, אני נוהגת להתחיל מפועלה בונה, פעולת toString ומיד לזמן אותן במחלקת הבדיקה. רק אח"כ מוסיפה את שאר הפעולות.
- חשוב להרגיל לתעד כל פעולה
- הקפידו שהתלמידים ירשמו פלטים משמעותיים בתכנית הבדיקה כך שניתן יהיה להבין מה התכנית בדקה.
- בכל מקום שניתן דרשו מהתלמיד שישתמש בפעולות של המחלקה ולא יחשב מחדש בתכנית הראשית את מה שמבקשים לבדוק. לדוגמה , ראה סעיף ו' בדף העבודה.

מעבדה 2: טיפוסים מורכב – מקטע Segment

מטרת התרגיל: היכרות עם טיפוס מורכב

המחלקה **Segment** מגדירה קטע של ישר המאופיין ע"י 2 נקודות הקצה שלו.

לפניך תרשים היררכית המחלקות.



לפניך טבלת פעולות המחלקה:

תיאור הפעולה	חתימת (כותרת) הפעולה
פעולה הבונה מקטע שנוצר מנקודות (x1,y1) ו- (x2,y2)	`public Segment(double x1, double y1, double x2, double y2)`
פעולה הבונה מקטע מנקודות p1 ו- p2	`public Segment(Point p1, Point p2)`
פעולה המחזירה אורך המקטע	`public double length()`
פעולה המחזירה את שיפוע ההקטע. במידה ואין שיפוע – מחזיר (-99999)	`public double slope()`
פעולה המחזירה מחרוזת המתארת את המקטע בצורה הבאה: [(x1,y1) – (x2,y2)]	`public String toString()`

(א) יש לכתוב ייצוג למחלקה Segment על פי הכתוב בשורה ראשונה. כלומר, לכתוב כותרת ותכונות.

(ב) יש לממש את המחלקה Segment . חובה להקפיד על תיעוד מלא. לכל פעולה מה מקבלת ומה מבצעת או מחזירה.

(ג) יש לבנות מחלקה TestSegment המכילה פעולה ראשית ובה יצירת שני מקטעים האחד על-פי שיעורים x1,y1,x2,y2 והשנייה על פי שתי נקודות.

(ד) יש לשנות את אחת הנקודות שיצרו את המקטע השני ולבדוק האם הוא השתנה או לא.
שאלה: האם מעוניינים שהמקטע יהיה תלוי בנקודות שיצרו אותו?
תשובה: סביר להניח שלא.
לכן אם המקטע השתנה יש לתקן את הפעולה שבנתה את המקטע על-פי 2 נקודות.
רמז – השתמש בפעולה בונה מעתיקה עבור יצירת המקטע.

(ה) יש להוסיף בדיקות גם לשאר פעולות המחלקה.

(ו) ברצוננו לבדוק האם המקטע מהווה קטע מישר עולה, ישר יורד, ישר מקביל לציר x או ישר מקביל לציר y . יש להוסיף קטע קוד או פעולה במחלקת הבדיקה אשר תקבל ישר ותציג כפלט הודעה מתאימה לסוג המקטע.

(ז) הוסף פעולה get Point1 אשר מחזירה את הנקודה p1 .

(ח) זמן את ההוראות הבאות בפעולה הראשית ובדוק את הפלט.

(ט) מה קרה למקטע seg1 ? האם הוא השתנה?

```
Point p3 = seg1.getPoint1();
System.out.println(p3);
p3.setY(7);
System.out.println(p3);
System.out.println(seg1);
```

(י) במידה והמקטע seg1 השתנה, יש לתקן את הפעולה getPoint1 כך ששינוי הנקודה p3 לא ישפיע על שינוי המקטע.
רמז – השתמש בפעולה בונה מעתיקה כדי לקבל העתק את הנקודה ולא הפנייה לנקודה עצמה.

למורה

מעבדה מספר 3: בניית מחלקה עם תכונה מערך של עצמים.

השיעור מבוסס על מעבדה Point מספר עצוב תכנה בשפת ג'אווה של האוניברסיטה העברית בירושלים.
פרק 3 , דך עבודה מספר 1

ידע קודם:

- ✓ יצירת מחלקה
- ✓ מערך של עצמים
- ✓ פעולה בונה מעתיקה

מטרת המעבדה:

- טיפול במחלקה עם תכונה שהיא מערך של עצמים
- בניית פעולה בונה כך שתאתחל את מערך העצמים
- הבחנה בין הפנייה למערך ליצירת מערך שמכיל העתק הנקודות

תשומת לב ל-

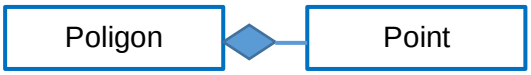
- פעולה בונה לפוליגון, מאתחלת את מערך הנקודות – תכונת הפוליגון בגודל אורך המערך המועבר כפרמטר ומעתיקה את כל הנקודות למערך התכונה
- בפעולות של צלע קצרה ביותר והיקף , יש לקחת בחשבון גם את הצלע ש"סוגרת" את המצולע, כלומר הצלע המתקבלת מנקודה אחרונה במערך לנקודה ראשונה.
- יש לזמן את פעולת distance של מחלקת Point על הנקודות שבמערך.

מעבדה 3: טיפוס מורכב המכיל כתכונה מערך של עצמים

מטרת התרגיל: היכרות עם טיפוס מורכב שהתכונה שלו מערך של עצמים

המחלקה **Poligon** מגדירה מצולע המורכב מ n צלעות כאשר $n \geq 3$.
הפוליגון מאופיין ע"י n נקודות אשר אין יותר משתי נקודות שנמצאות על אותו ישר.

לפניך תרשים היררכית המחלקות.



לפניך טבלת פעולות המחלקה:

`public Poligon(Point[] points)`	פעולה הבונה פוליגון על-פי מערך נקודות המועברות כפרמטר
`public int getNumberOfSides()`	פעולה המחזירה את מספר הצלעות
`public double getPerimeter()`	פעולה המחזירה את היקף הפוליגון
`public double getMinSide()`	פעולה המחזירה את אורך הצלע הקצרה ביותר
`public String toString()`	פעולה המחזירה מחרוזת המתארת את הפוליגון

(א) יש לכתוב ייצוג למחלקה Poligon כלומר את כותרת המחלקה והתכונות שלה.

(ב) יש לממש את המחלקה Poligon כלומר לכתוב את כל המחלקה בשפת ג'אווה.

(ג) יש לכתוב מחלקת בדיקה TestPoligon הכוללת פעולה ראשית אשר תבצע:

- קליטת את מספר הקדקודים של המצולע, n
- יצירת מערך מטיפוס Point בגודל n בשם points
- הצבת n נקודות בתאי המערך על פי קליטת n זוגות של ערכי x,y .
- יצירת עצם מטיפוס Poligon בשם poli המקבל בפרמטר את מערך points
- הצגה כפלט של העצם poli

(ד) יש להוסיף פעולות בדיקה לעל פעולות המחלקה Poligon בפעולה הראשית במחלקת הבדיקה בליווי פלטים משמעותיים.

למורה

מעבדה מספר 4: הדפסות רקורסיביות

ידע קודם:

- ✓ יסודות מדעי המחשב

מיקום המעבדה

- אחרי שיעור מבוא - מה זה רקורסיה?

המלצה להתחלת השיעור

- עברו על תרגיל מספר 1 ביחד עם התלמידים.
- ניתן לבקש מתלמיד שיקליד את הקוד במחשב שמחובר למקרן במעבדה ולהציג בפני התלמידים את תוצאת ההרצה.
- בקשו התלמיד לשנות את מיקום הזימון הרקורסיבי בפעולה ובקשו מהם "לנחש" מה יהיה הפלט כתוצאה מהשינוי.

מטרת המעבדה:

- להתמודד עם מיומנות של כתיבת פעולה רקורסיבית
- זיהוי תנאי העצירה
- זיהוי הפרמטר לשינוי בפעולת הזימון הרקורסיבי כך שיוביל אותנו לתנאי העצירה
- מעקב רקורסיבי בעזרת "מלבנים"
- הבחנה בין רקורסיה ראש, זנב ומלאה, כלומר מיקום הזימון הרקורסיבי בפעולה.

המלצות

- תנו לתלמידים להתקדם, והסתובבו בינהם
- במידה ונתקלתם בתרגיל שקשה לכולם, עצרו והדגימו ביחד את תהליך הפתרון תוך שיתוף התלמידים

תשומת לב ל-

- כשחלק מהתלמידים יגיעו לפעולות line4, line5 , יש צורך לעצור, לתווך ולהסביר מדוע יש צורך בשני פרמטרים. פרמטר אחד n הוא אורך הסדרה, והפרמטר השני הוא זה שמשתנה בכל זימון רקורסיבי, יתחיל מ 1 ויעצור ב n .
- ניתן היה גם לזמן את הפעולות באופן הבא: line4(4,4) ן line5(4,4). במקרה זה פרמטר שני יקטן עד 0
- תרגיל 12 מצריך זימון רקורסיבי של פעולה רקורסיבית או בשימוש לולאה, כי מתאים ללולאה מקוננת.

מעבדה 4: פעולות הדפסה רקורסיביות

מטרת התרגיל: לתת תחושה של איך לכתוב פעולה רקורסיבית ואיך לזמן אותה.
לזהות את תנאי העצירה, לדעת איזה פרמטר לשנות ובכמה כדי לקבל את התוצאה הרצויה, תוך ניסוי ובדיקה.

הנחיות

- יש ליצור מחלקה RecursePrint ומחלקת בדיקה TestRecursePrint
- המחלקה תכיל אוסף פעולות רקורסיביות לביצוע ההדפסות שיופיעו בהמשך.
- הפעולה הראשית שבתכנית הבדיקה תזמן את הפעולות ותציג פלטים משמעותיים.

עבור כל פעולת בדיקה תופיע הודעה שמכילה כותרת פעולת הזימון עם הפרמטר שקבעתם ולאחריו הפלט המתקבל מתוצאת זימון הפעולה.

- חובה להשאיר בתכנית הבדיקה את כל הזימונים

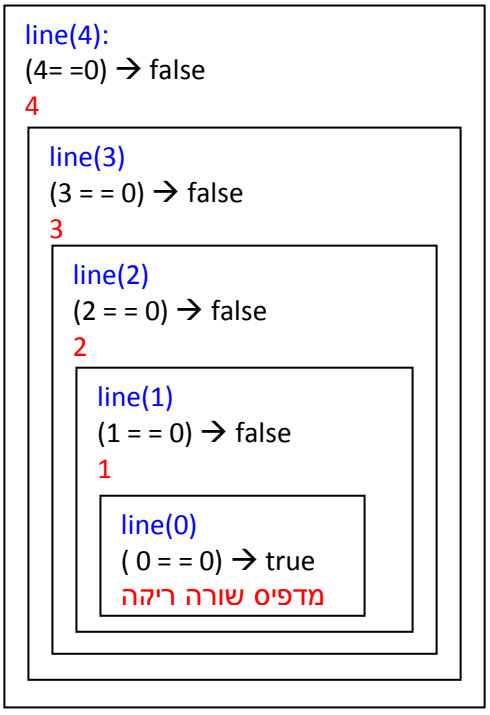
תרגיל 1: מעקב רקורסיבי אשר ישמש בסיס ודוגמה.

א- מה לדעתכם תבצע הפעולה הבאה:

נראה מעקב אחר זימון הפעולה line(4)

```
public static void line(int n)
{
   if (n==0)
      System.out.println();
   else
   {
      System.out.print(n + " ");
      line(n-1);
      }
   }
}
```

זימון הפעולה line(4) יציג כפלט ← 4 3 2 1



ב- מה לדעתכם יקרה אם נחליף את המיקום של שורה ההדפסה בשורה הזימון הרקורסיבי?
שערו, רק אח"כ הקלידו ובדקו ונסו להבין למה. לוו במעקב רקורסיבי.

ג- מה לדעתכם יקרה אם נוסיף שורה הדפסה לפני ואחרי הזימון הרקורסיבי?
שערו, רק אח"כ הקלידו ובדקו ונסו להבין למה. לוו במעקב רקורסיבי.

ד- האם הפלטים ישתנו אם נשנה אם נמחק את התנאי של if (n==0) ובמקום ה else נרשום תנאי if (n > 0) ? מה תנאי העצירה במקרה זה?

<u>תרגיל 2</u>

כתוב את הפעולות הבאות אשר ידאגו לפלט הרצוי:

(1) **line1** , אשר תדפיס : 1 2 3 4 5 עבוד הזימון line1(5) .

(2) **line2** , אשר תדפיס : 5 4 3 2 1 1 2 3 4 5 עבוד הזימון line2(5) .

(3) **line3** , אשר תדפיס : 5 4 3 2 1 2 3 4 5 עבוד הזימון line3(5) .

פסק זמן

- מדוע בפעולות line5 , line4 יש שני פרמטרים?
- מה התפקיד של הפרמטר הראשון?
- מה התפקיד של הפרמטר השני?
- במימוש הפעולה, איזה פרמטר משתנה ואיזה נשאר קבוע?

(4) **line4** , אשר תדפיס : 1 2 3 4 5 5 4 3 2 1 עבוד הזימון line4(5,1) .

(5) **line5** , אשר תדפיס : 5 5 5 5 5 עבוד הזימון line5(5,1) .

(6) **line6** , אשר תדפיס : * * * * * עבוד הזימון line6(5, '*') .

(7) **line7** , אשר תדפיס : 8 6 4 2 1 2 3 4 עבוד הזימון line7(4) .

(8) **line8** , אשר תדפיס : 1 -2 3 -4 5 -6 עבוד הזימון line8(6) .

(9) **line9** , אשר תדפיס : 0 11 2 13 4 15 עבוד הזימון line9(5) .

(10) **line10** , אשר תדפיס : 0 11 2 13 4 15 עבוד הזימון line10(5) .

(11) **line11** , אשר תדפיס : 4 13 2 11 0 11 2 13 4 עבוד הזימון line11(4) .

(12) הוסף את הפעולות הבאות, ראה דוגמת פלטים.

<u>הנחיות</u>:

- ✓ במימוש הפעולות, ניתן להשתמש בפעולה line(n) או להשתמש בלולאה להדפסת שורה של מספרים.
- ✓ יש לכתוב את פעולות triUpDown , triDownUp ללא שימוש בפעולות הקודמות, אלא באופן ישיר.
- ✓ כשלב מקדים לפעולת sandClock , באפשרותך להתעלם מהרווחים ורק בשלב שני יש לדאוג לרווחים. ניתן להשתמש בפעולה line6 כאשר הפרמטר השני הוא תו רווח.

```
triUp(6)
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
------------------
triDown(6)
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
------------------
triDownUp(6)
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
------------------
triUpDown(6,1)
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
------------------
sandClock(7,0)
7777777
 55555
  333
   1
  333
 55555
7777777
------------------
```

למורה

מעבדה מספר 5: אוסף פעולות רקורסיביות המחזירות ערך

ידע קודם: יסודות מדעי המחשב

- ✓ פירוק מספר
- ✓ לולאות
- ✓ מערכים
- ✓ מחרוזות

מיקום המעבדה

- אחרי שיעור מבוא - מה זה רקורסיה?
- רצוי אחרי מעבדה 4 – הדפסות רקורסיביות
- אפשר גם אחרי שיעור מבוא

תיאור המעבדה

- מעבדה זו כוללת מחלקת שירות, מחלקה עם אוסף פעולות מחלקה static , פעולות רקורסיביות המחזירות ערך.
- אוסף הפעולות מחולק לקטיגוריות: כללי, פירוק מספר, מחרוזת, מערך, מערך דו ממדי.
- אוסף הפעולות ניתן מהקל לקשה יותר
- יש לפצל מעבדה זו ל 4 מעבדות

המלצות לשיעור מקדים את המעבדה

- רצוי להציג ולפתור מספר בעיות על הלוח עוד לפני המעבדה
- נסו להגיע עם התלמיד להגדרה רקורסיבית לעומת הגדרה איטרטיבית
- הציגו פתרון רקורסיבי המלווה במעקב רקורסיבי
- בפתרון בעיה, ממליצה לתת פתרון איטרטיבי - בעזרת לולאה, לעומת פתרון רקורסיבי
- הדגישו את הדומה והשונה בשני הפתרונות, פתרון בעזרת לולאה לעומת פתרון רקורסיבי
- חדדו את נושא תנאי העצירה
- חדדו את נושא הערך המוחזר
- לפני כל קטיגוריה, הדגימו חשיבה ופתרון על הלוח ביחד עם התלמידים.

מטרת המעבדה:

- להתמודד עם פיתוח חשיבה רקורסיבית
- זיהוי תנאי העצירה
- זיהוי פרמטר בפעולת הזימון הרקורסיבי כך שיוביל אותנו לתנאי העצירה
- מעקב רקורסיבי בעזרת "מלבנים"
- הבחנה בין רקורסיה ראש, זנב ומלאה, כלומר מיקום הזימון הרקורסיבי בפעולה.

המלצות

- לגבי פתרונות רקורסיביים של פעולות בוליאניות, ממליצה לטפל במקרה של 'אמת' , 'שקר' או להפך ורק אח"כ לבצע את הזימון הרקורסיבי, כך לא יקרה מקרה שהתלמיד לא יטפל באחד המקרים.
- דונו עם התלמידים , מתי המקרה שיש להוסיף פרמטר לפעולה הרקורסיבית ומתי לא.
- דברו אתם על פעולה עוטפת, למשל בפעולות על מערכים.
- פעולה המקבלת מערך, תעטוף פעולה רקורסיבית אשר תקבל את המערך ומספר נוסף שיציין את גודל המערך, או תא אחרון או תא ראשון, תלוי בזימון הפעולה הרקורסיבית. דונו אתם על נושא זה והסבירו.
- חיזרו על תבניות "**כל הערכים מקיימים תנאי**" ו "**ערך אחד לפחות מקיים תנאי**" וחדדו מהו הערך המוחזר במקרה הקצה, תנאי העצירה בעל אחד מהתבניות.

מעבדה 5: פעולות רקורסיביות עם החזרת ערך

מטרת התרגיל: פתרון רקורסיבי לבעיות מוכרות , מהקל לכבד, מחולק לנושאים

הנחיות

- יש ליצור מחלקה RecurseReturn ומחלקת בדיקה TestRecurseReturn
- המחלקה תכיל אוסף פעולות רקורסיביות אשר מחזירות ערך
- הפעולה הראשית במחלקת הבדיקה תזמן את הפעולות ותציג פלטים משמעותיים.

פעולות רקורסיביות המחזירות ערך

1	הפעולה מקבלת מספר טבעי או אפס ומחזירה את n!	`public static int factorial(int n)`
2	הפעולה מחזירה את x*y , y>0 שלם	`public static int multiple(int x, int y)`
3	הפעולה מחזירה את x^y , y>0 שלם	`public static int power(int x, int y)`
4	הפעולה מחזירה את a/b	`public static int div(int a, int b)`
5	הפעולה מחזירה את a%b	`public static int mod(int a, int b)`
6	הפעולה מחזירה את סכום המספרים מ 1 עד n. n טבעי	`public static int sum1ToN(int n)`
7	הפעולה מחזירה את סכום המחזירים שהם כפולות של 3 בין 1 ל n . n טבעי	`public static int sum1ToN3(int n)`

פעולות רקורסיביות עם מספר טבעי

8	הפעולה מחזירה את מספר הספרות במספר num	`public static int countNum(int num)`
9	הפעולה מחזירה את סכום הספרות של num	`public static int sumDigits(int num)`
10	הפעולה מחזירה את מכפלת הספרות של num	`public static int multDigits(int num)`
11	הפעולה מחזירה את סכום הספרות הזוגיות של num	`public static int sumDigitsEven(int num)`
12	הפעולה מחזירה 'אמת' אם המספר num מכיל את הספרה 5 , 'שקר' – אחרת	`public static boolean isContain5 (int num)`
13	הפעולה מחזירה 'אמת' אם כל ספרות המספר זוגיות, 'שקר' – אחרת	`public static boolean isAllEven(int num)`
14	הפעולה מקבלת 2 מספרים טבעיים ומחזירה את ההפרש של מספר הספרות שלהם.	`public static int diffCountDigits (int num1, int num2)`
15	הפעולה מחזירה 'אמת' אם המספר הוא בסדר ספרות עולה, 'שקר' – אחרת	`public static boolean isUp(int num)`
16	הפעולה מחזירה את המספר בסדר ספרות הפוך זימון הפעולה יהיה עם rev=0	`public static int reverseNum (int num, int rev)`

פעולות רקורסיביות עם מחרוזות

17	הפעולה מחזירה מחרוזת שהיא שיכפול "ראי" של עצמה. למשל עבור "abc" יתקבל "abccba"	**public static** String mirror(String str)
18	הפעולה מחזירה מחרוזת כך שאחרי כל אות קטנה מופיעה האות הגדולה. למשל עבור "abc" יתקבל "aAbBcC"	**public static void** smallBig(String str)
19	הפעולה מחזירה כמה פעמים תת-מחרוזת sub נמצאת בתוך מחרוזת str	**public static int** timeSubInStr (String str, String sub)
20	הפעולה מחזירה 'אמת' אם מחרוזת היא פלינדרום , 'שקר' - אחרת	**public static boolean** isPalindrom(String str)

פעולות רקורסיביות עם מערכים

21	הפעולה מקבלת מערך ומחזירה אותו עם ערכים שנקלטו מהמשתמש (לא חובה רקורסיבי)	**public static void fillArr** (**int**[] a)
22	הפעולה מדפיסה את אברי המערך בשורה	**public static void ptintArr**(**int**[] a)
23	הפעולה מחזירה את סכום n אברי המערך הראשונים	**public static int sumArr** (**int**[] a, **int** n)
24	הפעולה מחזירה 'אמת' אם n אברי המערך הראשונים הם בסדר עולה, 'שקר'- אחרת	**public static boolean isUp** (**int**[] a, **int** n)
25	הפעולה מחזירה 'אמת' אם n אברי המערך הראשונים מהווים סדרה חשבונית, 'שקר'- אחרת. **סדרה חשבונית** זו סדרה שההפרש בין אחבר לאיבר הוא קבוע.	**public static boolean isArithmetic** (**int**[] a)
26	הפעולה מחזירה את הערך הגדול ביותר במערך ב n התאים הראשונים.	**public static int maxArray** (**int**[] a, **int** n)
27	הפעולה מחזירה 'אמת' אם x נמצא במערך בין אינדקסים 0<=k<=p<a.length . k,p	**public static boolean isExist** (**int**[] a, **int** k, **int** p, **int** x)
28	הפעולה מחזירה 'אמת' אם x נמצא במערך ממוין מקטן לגדול, בין אינדקסים 0<=k<=p<a.length . k,p	**public static boolean isExistSort** (**int**[] a, **int** k, **int** p, **int** x)
29	הפעולה מחזירה 'אמת' אם סכום הערכים בתאים באינדקסים זוגיים שווה לסכום הערכים בתאים האי-זוגיים, 'שקר' – אחרת.	**public static boolean isSumEvenOdd** (**int**[] a)
30	הפעולה מדפיסה את איברי המערך הקטנים מהאיבר העוקב להם.	**public static void printSmallerSuccessive** (**int**[] a)

רשות - למתקדמים

פעולות רקורסיביות עם מערך דו-מימדי

31	פעולה מחזירה מערך דו-מימדי של שלמים עם מספרים רנדומליים בגודל nRowXnCol	public static int[][] **randMat** (int nRow, int nCol)
32	פעולה מחזירה מערך דו-מימדי של שלמים בגודל nRowXnCol עם מספרים שנקלטים מהמשתמש	public static int[][] **build** (int nRow, int nCol)
33	הפעולה מדפיסה את שורה row של המערך הדו-מימדי. הנח שורה row קיימת.	public static void **printRow** (int[][] m, int row)
34	הפעולה מדפיסה אברי המטריצה בשורה r מעמודה c	public static void **printRowFrom** (int[][] m, int r, int c)
35	הפועלה מדפיסה תת-מטריצה מאיבר (r,c) ימינה ולמטה.	public static void **printSubMat** (int[][] m,int r, int c)
36	הפעולה מקבלת מערך דו-מימדי של מספרים שלמים ומחזירה את מספר המסלולים השונים שאפשר להגיע מפינה שמאלית עליונה של המטריצה לפינה ימנית תחתונה של המטריצה. פעולות מותרות: תזוזה ימינה, למטה או באלכסון ימני.	public static int **countPath** (int[][]m, int n, int r, int c)
37	הפעולה מקבלת מטריצה של 0-ים ו 1-ים, מספר שורה ומספר עמודה . הפעולה מחזירה 'אמת' אם יש מסלול של 1-ים ממיקום [row][col] עד פינה שמאלית תחתונה. פעולות מותרות: תזוזה ימינה, למטה או באלכסון ימני.	public static boolean **isPath1** (int[][] m, int row, int col)

למורה

דף עבודה מספר 6

לעתים שיעור נולד מתוך משהו שגילינו בכתה, כדאי לתפוס רגעים כאלה, ולעשות מהם שיעור.
דף העבודה הבא נוצר בעקבות טעויות שנכתבו על ידי תלמידים בבוחן של רשימה.

<u>הנחיה</u>

תנו לתלמידים את דף העבודה, בקשו מהם לבדוק ולנסות לשים את האצבע על הטעויות.

<u>מטרות</u>

- התלמידים יחושו במה המורה עובר כאשר בודק בחנים או מבחנים.
- התלמידים ירגישו שזו משימה כלל לא פשוטה.
- הם יבצעו תהליך למידה מתוך הטעויות של תלמידים אחרים
- לחדד את נושא ההפניות לחוליות ברשימה.

<u>המלצות</u>

- רצוי לספר לתלמידים שאילו פתרונות של תלמידים ולא אני , המורה, חיברתי אותם.
- בקשו שיתמודדו עם עט ונייר ויבצעו מעקב אחר הקוד כדי לזהות אם הקוד תקין או לא ולגלות את השגיאות
- בשלב הבא, אם התלמיד רוצה , שיקליד , יריץ , יתקן ויבדוק שתיקן
- חשוב שיסביר את מהות הטעויות וכיצד לתקן

דף עבודה שניתן לתלמידים בעקבות בוחן בנושא רשימה

תלמידים קבלו לכתוב פעולה המקבלת רשימה של מספרים שלמים עם מספרים שחוזרים על עצמם ברצף. הפעולה תשאיר מכל רצף מספרים מספר אחד ולאחריו חוליה שבה מספר החזרות של המספר.

לפניך פעולות שתלמידים כתבו. עקבו אחר הפעולות וקבעו אם הן מבצעות את הנדרש. אם לא, הראו מה הבעיה ותקנו אותה. יכולות להיות מספר שגיאות בפעולה אחת.

זכרו, אלגוריתם חייב למצות את כל האפשרויות לכן כאשר אתם כותבים אלגוריתם לוו אותו בתרשים עצמים ובדקו מה קורה בכל אפשרות.

פתרון מספר 1

```
public static void compress1(Node<Integer> chain)
{
   Node<Integer> pos = list;
  Node<Integer> temp;
  int count = 1;
  while (list.hasNext())
  {
       if (list.getValue() == list.getNext().getValue())
       {
          count++;
          temp = pos;
          pos = pos.getNext();
          temp.setNext(null);
       }
       else
       {
          Node<Integer> node = new Node<Integer>(count, list.getNext());
          pos.setNext(node);
          count = 1;
          pos = pos.getNext().getNext();
       }
    }
}
```

פתרון מספר 2

```
public static void compress2(Node<Integer> chain)
{
    Node<Integer> temp = list;
   int count = 1;
   while (temp.hasNext())
   {
      count = 1;
      Node<Integer> pos = temp;
      while (pos.hasNext() && pos.getValue() == pos.getNext().getValue())
      {
        count++;
        pos = pos.getNext();
      }
      if (count > 1)
         temp.setNext(new Node<Integer>(count, pos.getNext()));
      temp = temp.getNext();
   }
}
```

פתרון מספר 3

```
public static void compress3(Node<Integer> chain)
{
  Node<Integer> pos = list;
  Node<Integer> change = list.getNext();
  int count = 1;
  while (pos.hasNext())
  {
    if (pos.getValue() == pos.getNext().getValue())
    {
      count++;
      pos = pos.getNext();
    }
    else
    {
      change.setValue(count);
      count = 1;
      if (pos != change)
        pos.setValue(pos.getNext().getValue());
      change = change.getNext().getNext();
      pos = pos.getNext();
    }
  }
  change.setNext(null);
```

פתרון מספר 4

```
public static void compress4(Node<Integer> chain)
{
  Node<Integer> pos, tmp;
  pos = list;
  tmp = list;
  int count = 1;
  while (tmp != null && tmp.hasNext())
  {
    while (tmp.getNext() != null &&
                tmp.getValue() == tmp.getNext().getValue())
    {
      count++;
      tmp = tmp.getNext();
    }
    pos.setNext(new Node<Integer>(count, tmp.getNext()));
    pos = pos.getNext();
    tmp = tmp.getNext();
}
```