תרגיל בית מס. 3
מימוש צורות, פירמידות ומנסרות, ואוספים
פולימורפיזם, static, final, immutable
מערכים, אוספים, equals, compare,
חריגות ותיעוד

להגשה עד יום רביעי, 28.12 למניינם
ההגשה בזוגות במוודל עד השעה 23:00
משקל התרגיל: 7 נק'

התרגיל עוסק במימוש צורות (מלבן ומשולש) במישור (חלק ראשון),
פירמידות ומנסרות במרחב (חלק שני),
מערכים ואוספים (חלק שלישי),
חריגות (חלק רביעי)
ותיעוד קוד (חלק חמישי).

התרגיל כולל מימוש ושאלות על המימוש (בחלקים ב' ו-ג').
יש לממש חלק אחר חלק, שכן כל חלק מתייחס לחלקים הקודמים.
נתונות לכם:
1. מחלקות: Point ו- HW3Utils. אין לשנות מחלקות אלו.
2. מחלקות בדיקות: לשני החלקים הראשונים.
3. דוגמאות קלט-פלט.

<u>אופן ההגשה:</u>
כל המחלקות בתרגיל יהיו תחת התקייה הראשית src.
יש לארוז בקובץ zip אחד את כל קבצי ה-java שכתבתם וקובץ pdf אחד ובו תשובותיכם לשאלות הפתוחות. שם הקובץ צריך להיות: 42_HW3_123456789_987654321.zip
כאשר *123456789* ו-*987654321* הם מספרי הזהות (בני 9 ספרות כל אחד, גם אם מתחילים ב-0) של המגישים.

לא תתקבל עבודה שאינה מתקמפלת בבודק האוטומטי.
בנוסף, בבודק האוטומטי יש מספר טסטים ודוגמאות על מנת שתוכלו לבדוק את הקוד שכתבתם עוד לפני הגשתו. בתחילת הבדיקה של התרגיל שלכם, אנחנו נערוך את אותם הבדיקות האוטומטיות. לפיכך, שימו לב כי הינכם מקבלים OK עבור הבדיקות האלו.

<u>חלק ראשון: מימוש Shape, Circle ו-Rectangle</u>

בשאלה זו עליכם לממש מספר צורות במישור. עליכם להגדיר:
1. מחלקה אבסטרקטית / מנשק בשם Shape – המתאר צורה במישור.
2. שתי מחלקות (רגילות) Circle ו-Rectangle הממשות/מרחיבות את Shape.

```
/** @return the area of the shape */
public double area();

/** @return the perimeter of the shape */
public double perimeter();

/** @return the center point of the shape */
public Point getCenter();

/** Set the center point of the shape */
```

```
public void setCenter(Point p);
```

Circle מתארת מעגל במישור ע"י שני שדות: מרכז המעגל ואורך רדיוס.
Rectangle מתארת מלבן במישור ע"י שלשה שדות: מרכז המלבן, אורך צלע אנכית ואורך צלע אופקית.

א. הוסיפו במחלקות Circle ו-Rectangle דריסה של מתודות **equals** ו-**hashCode** כך שישוו לפי ערכי השדות.

ב. הוסיפו **בנאים** כדלהלן:

1. Circle:
   a. בנאי ריק, היוצר מעגל שמרכזו בראשית הצירים ורדיוסו 1.
   b. בנאי המקבל את נקודת המרכז (Point) והרדיוס (double) לפי סדר זה.
   c. בנאי המקבל מחרוזת המכילה 3 מספרים (double) מופרדים באמצעות פסיק (,) – שני המספרים הראשונים מתארים את נקודת המרכז (x ו-y לפי סדר זה), והמספר השלישי הוא הרדיוס.
   הבנאי בונה את האובייקט לפי זה. בנאי זה יכול להיעזר ב- Scanner עם שימוש במתודה useDelimiter(",") המוגדרת במחלקה Scanner. ניתן לראות הסבר על מחלקה זו ודוגמה בהמשך התרגיל.
2. Rectangle
   a. בנאי ריק, היוצר מלבן שמרכזו בראשית הצירים וצלעותיו 1.
   b. בנאי המקבל את נקודת המרכז (Point) אורך צלע אנכית ואורך צלע אופקית (double) לפי סדר זה.
   c. בנאי המקבל מחרוזת המכילה 4 מספרים (double) מופרדים באמצעות פסיק (,) – שני המספרים הראשונים מתארים את נקודת המרכז (x ו-y לפי סדר זה), המספר השלישי הוא אורך צלע אנכית, והרביעי הוא אורך צלע אופקית.
   היעזרו ב-Scanner.

ג. הוסיפו במחלקות Circle ו-Rectangle דריסה של מתודה **toString**.
המחרוזת שתתאר מעגל תהיה כדלהלן:

```
"Circle [radius=#radius, center=#center]"
```

המחרוזת שתתאר מלבן תהיה כדלהלן:

```
"Rectangle [length vertical edge=#lengthVertical, length horizontal
edge=#lengthHorizontal, center=#center]"
```

כאשר במקום #radius, #lengthVertical, #lengthHorizontal נשים את הערכים בפורמט של שני מקומות אחרי הנקודה העשרונית (השתמשו במתודה formatDouble הנתונה לכם במחלקה HW3Utils), ובמקום #center נשים את נקודת המרכז (לפי toString של Point הנתונה לכם).

חלק שני: מימוש הגופים

בחלק זה נממש קוד, ונענה על שאלות לגבי המימוש.
הגופים הנתמכים: פירמידה ומנסרה במרחב התלת מימדי אשר בסיסם הינו צורה מטיפוס Shape מקביל (צורה במישור המקביל למישור X-Y).
עבור מנסרה נתאר רק מנסרות ישרות וניצבות (מנסרות אשר שני הבסיסים שלהן מקבילים, חופפים ומצויים בדיוק האחד מעל האחר).

לכל גוף תלת ממדי נשמרים פרטי המידע הבאים:

baseShape: Shape - צורת הבסיס/ים של הגוף (Shape הוגדרה בחלק הקודם)

baseZ: double - ערך ה-z (double) של הבסיס במערכת הצירים.
עבור מנסרה – ערך זה יכיל את ערך ה-z של הבסיס התחתון,
עבור פירמידה – ערך זה יכיל את ערך ה-z של הבסיס היחיד.

z: double - עבור מנסרה יכיל את ערך ה-z של הבסיס השני (הגבוה),
ועבור פירמידה יכיל את ערך ה-z של קדקוד הפירמידה.

בנוסף, עבור פירמידה, נשמור את ערכי (x,y) של קדקוד הפירמידה ע"י שדה מטיפוס Point.

לכל גוף יש מחלקה מתאימה. שמות המחלקות הן Pyramid ו-Prism עבור פירמידה ומנסרה בהתאמה. לכל מחלקה יש **בנאי** המקבל ערכים לכל השדות.
עבור מנסרה: צורת הבסיס ואח"כ גבהי שני הבסיסים.

```
Prism(Shape baseShape, double baseZ, double z)
```

עבור פירמידה: צורת הבסיס, אח"כ גובה הבסיס ואח"כ ערכי (x,y,z) עבור קדקד הפירמידה.

```
Pyramid(Shape baseShape, double baseZ, double x, double y, double z)
```

בנוסף, יש בכל מחלקה בנאי שבמקום לקבל כפרמטר ראשון Shape baseShape, הוא יקבל:
String baseType, String baseParams לפי סדר זה משמאל לימין.
כך ש- baseType הוא "R" או "C" (באות גדולה) עבור Rectangle או Circle בהתאמה,
ו-baseParams הוא מחרוזת המתארת את הצורה (Rectangle או Circle) כנדרש בבנאי המתאים המתואר בחלק א'.

בנוסף, על המחלקות לממש את המנשק הבא:

+ getHeight(): double	//מחזיר גובה
+ volume(): double	//מחזיר נפח
+ getBaseShape(): Shape	//מחזיר את מצולע הבסיס
+ toString(): String	//מחזיר מחרוזת המתארת את הגוף(*)
+ equals(Object o): boolean	// מחזיר true אם הפרמטר מתאר גוף בעל גיאומטריה זהה לגוף שעליו המתודה נקראת //

(*) המחרוזת שתתאר את הגוף תהיה כדלהלן:
עבור מנסרה
"Prism: Base shape=#baseShape. z-values for bases=#z1,#z2"
עבור פירמידה
"Pyramid: Base shape=#baseShap. z-base shape =#zBase. Apex=(#x,#y,#z)"
כאשר #baseShape מסוג String שווה לתוצאה של הפעלת המתודה toString() על האובייקט המתאר את בסיס הגוף שהוא מסוג Polygon.
#z1,#z2 אלו גבהי הבסיס של המנסרה כך ש- z1<z2.
#zBase הוא גובה הבסיס של הפירמידה.
(#x,#y,#z) אלו ערכי נקודת קדקוד הפירמידה.
תצוגת כל ערכי ה-double היא עם שני מקומות אחרי הנקודה העשרונית (השתמשו במתודה formatDouble הנתונה לכם במחלקה HW3Utils).

הוסיפו מחלקה אבסטרקטית/מנשק בשם PrismPyramid אשר Pyramid ו-Prism יורשות ממנו או ממשות אותו. המטרה "לתפוס" את המשותף שבמחלקות Pyramid ו-Prism במחלקה PrismPyramid.

הוסיפו מתודות נוספות אם יש צורך.

שאלות - החלק השני:

1. תנו דוגמה בתרגיל ליחס הורשה, ליחס הפשטה, ליחס הכלה, לדריסה (overriding) של מתודה, לפולימורפיזם, ולהעמסה (overloading) של מתודה.
אם אין לכם דוגמה לאחד מן הדברים, ציינו זאת.
2. האם ניתן להגדיר יחסי הורשה בין המחלקות Prism ו-Pyramid? הסבירו.
3. האם PrismPyramid שהגדרתם היא מחלקה אבסטרקטית או מנשק? מדוע בחרתם בדרך הזו? מה ההבדל בין שני הדברים?
4. אובייקט נקרא Immutable אם הוא לא ניתן לשינוי אחרי יצירתו.

א. אובייקט ניתן לשינוי אחרי יצירה דרך מתודות המיועדות לכך (למשל set) או דרך שדות בעלי הרשאה שאינה פרטית. חישבו על דרכים נוספות לשינוי אובייקט אחרי יצירה (היזכרו בתרגיל בית 2).

ב. מבין המחלקות שמימשתם עד עתה, מאלו מחלקות ניתן ליצור אובייקטים ניתנים לשינוי ומאלו מחלקות ניתן ליצור אובייקטים שאינם ניתנים לשינוי?

ג. מה היתרון בתכונת Immutable? ומהו החיסרון בכך?

חלק שלישי: קלט ומערך

בחלק זה המערכת תקלוט רשימה של גופים מתוך קובץ לפי פורמט מיוחד, ותבצע מספר דברים.
כמו-כן הנכם נדרשים לענות על שתי שאלות לגבי המימוש (מופיעות בסוף החלק).
בסיום החלק הזה מופיע הסבר ועזרה לניהול קלט-פלט עבור התרגיל.
קובץ הקלט הינו קובץ טקסט, בו כל שורה מגדירה גוף אחד (מנסרה או פירמידה).
הפרמטרים בשורה מופרדים באמצעות #, והסדר הוא כדלהלן:
האות הראשונה בכל שורה אומרת באיזה סוג מדובר: Y (-pyramid), R (-prism).
אח"כ # מפריד.
אח"כ, נקבל את ערכי צורת הבסיס, כדהלן:
תחילה: C (circle) או R (rectangle), אח"כ: (נקודותיים), אח"כ ערכי צורת הבסיס מופרדים באמצעות פסיק. שני ערכי double ראשונים הם ערכי x,y של נקודת המרכז, ואח"כ עבור מעגל ערך מספרי נוסף עבור הרדיוס, ועבור מלבן שני ערכים מספרים נוספים עבור אורך צלע אנכית ואורך צלע אופקית לפי סדר זה.
אח"כ # מפריד.
אח"כ נקבל את ערכי תלת המימד של הגוף -

- אם מדובר בפירמידה, נקבל 4 ערכי double המופרדים באמצעות פסיק: הערך הראשון הוא גובה הבסיס, ושלשה הערכים שאח"כ הם ערכי x,y,z של קדקוד הפירמידה.
- אם מדובר במנסרה, נקבל 2 ערכי double המופרדים באמצעות פסיק עבור גבהי הבסיסים.

למשל:

- Y#R.1.5,-1.2,3,4#-1,0,1,2

  מתאר פירמידה (Y) עם הפרמטרים הבאים:
  - הבסיס מלבן שמרכזו (1.5,-1.2), אורך צלע אנכית 3, ואורך צלע אופקית 4.
  - הבסיס בגובה 1- במערכת הצירים,
  - קדקוד פירמידה (0,1,2)

- R#C.2,-2,3.1#0,1

  מתאר מנסרה (R) ישרה עם הפרמטרים הבאים:
  - הבסיס מעגל שמרכזו (2,-2) ורדיוסו 3.1.
  - בסיס אחד בגובה 0 במערכת הצירים
  - בסיס שני בגובה 1 במערכת הצירים

ערכי ה-double מופיעים בייצוג עשרוני.

הגדירו מחלקה ShapesHandler ובה מתודה main אשר מקבלת כארגומנט ראשון את שם קובץ הקלט וכארגומנט שני את מספר השורות בקובץ. תפקידה של מתודה זו לקרוא את קובץ הקלט, לבנות את כל הגופים על פי ההוראות שבו, לשמור אותם במיכלים ואז להדפיס את כל הגופים באופנים שונים. כל גוף יודפס בשורה נפרדת. המחרוזת המודפת המתארת את הגוף מתקבלת מהפעלת מתודת toString() על הגוף.
בשלב זה ניתן להניח כי הפרמטרים הנשלחים ל-main תקינים, וכן שפורמט הקובץ תקין, כולל ערכי ה-double תקינים.

נעשה שימוש בשלשה סוגים של מיכלים:

```
1. PrismPyramid[] arr
2. List<PrismPyramid> list
3. Map<Double, List<PrismPyramid>> map;
```

<u>שלב א': קלט</u>
יש לקרא את הקובץ פעם אחת, ותוך כדי הקריאה למלא את שלשת המיכלים האלו באופן הבא.
arr: יכיל את כל הגופים לפי סדר קריאתם.
list: יכיל את כל הגופים בסדר הפוך לסדר קריאתם.
map: המפתחות הם מסוג Double ומכילים את הגבהים של הגופים שנוצרו מהקובץ.

יתכנו מספר גופים באותו גובה, ולכן הערך עבור כל מפתח הוא **רשימה** של גופים.
הרשימה הזו צריכה להיות לפי הסדר בקובץ.
המפה צריכה להיות ממוינת לפי המפתחות (הגבהים) - מהקטן לגדול.

אסור לקרוא את הקובץ יותר מפעם אחת, וכן אסור להגדיר מיכלים נוספים עבור הקלט (מלבד לטיפול בחריגות – ראה חלק 4 בתרגיל).

שלב ב': כתיבת קבצי פלט:

1. הדפסת כל הגופים לפי סדר הופעתם בקובץ הקלט. הדפסה זו תעשה ע"י ה-arr:

```
for (PrismPyramid p : arr) {
      if (p==null) break;
      //write a line with p to the file
}
```

ההדפסה תהיה לקובץ shapesOutArr.txt. שימו לב כי אינכם מדפיסים שורות null.

2. הדפסת כל הגופים לפי סדר הפוך מהופעתם בקובץ הקלט. הדפסה זו תעשה ע"י ה-list:

```
for (PrismPyramid p : list) {//write a line with p to the file}
```

הדפסה תהיה לקובץ shapesOutList.txt

שימו לב כי עליכם להגדיר נכונה את list ואת arr כך שהסדר הנכון אמור להתקבל מהדפסה באמצעות לולאות for דלעיל.

3. הדפסת כל הגופים לפי הסדר הנקבע ע"י הגובה שלהם, מהקטן לגדול. גופים עם אותו גובה יודפסו אחד אחרי השני **על פי סדר הופעתם בקובץ הקלט**.
הדפסה זו תעשה שלש פעמים בשלשה אופנים שונים:
   a. ע"י ה-map.
   הדפסה זו היא לקובץ בשם #inputFileName_shapesSortOutMap.txt
   b. ע"י ה-list. הדפסה זו היא לקובץ בשם #inputFileName_shapesSortOutList.txt
   c. ע"י ה-arr. הדפסה זו היא לקובץ בשם #inputFileName_shapesSortOutArr.txt
   כאשר #inputFileName הוא שם קובץ הקלט.
   מיקום הקבצים באותה תיקייה של קובץ ה-input.

אם הגדרתם נכון את map אז הסדר הנכון אמור להתקבל מהדפסה באמצעות לולאות for כדלהלן:

```
for (List<PrismPyramid> pList : map.values()) {
      for (PrismPyramid p : pList) {
            //write a line with p to the file
            }
}
```

לגבי ה-list. המטרה היא לשנות את הסדר ב-list כך שהפלט של הלולאה

```
for (PrismPyramid p : list) {//write a line with p to the file}
```

יהיה על פי גודל הגבהים, כנדרש.
אין להגדיר מערכי עזר או אוספי עזר נוספים עבור הקלט.
ניתן לבצע זאת באמצעות שתי מתודות במחלקה Collections:
`Collections.reverse(List<?> list)` הופכת את הסדר ב-list.
`Collections.sort(List<T> list, Comparator<? super T> c)` ממיינת את ה-list.
לצורך מיון זה, נשלח מימוש מנשק Comperator.
הטיפול ב- arr דומה לטיפול ב-list, ע"י שימוש ב - Arrays.sort במקום ב-Collections.

שאלות חלק ג:

1. הארגומנט השני ב-main מכיל את מספר השורות בקובץ.
למה נתון זה עוזר לנו? (חשבו על בניית המערך לעומת אוסף אחר).
2. אם המחלקה PrismPyramid מממשת את Comparable<PrismPyramid>, אזי ניתן לבצע מיון ע"י הקריאה - Collections.sort(List<T> list).

מדוע אם כן לא בחרנו בדרך זו, אלא השתמשנו במתודה sort המקבלת כפרמטר Comparator?

קבצי קלט-פלט לדוגמה מופיעים באתר.
<u>קבצי הטקסט כולם יהיו באותה תיקייה של קבצי ה-java.</u>
בהמשך הקורס, נלמד על קלט-פלט בצורה יסודית, וכן על טיפול בשגיאות - קובץ לא נמצא וכד'.
בתרגיל זה עליכם להשתמש בקלט-פלט על פי ההוראות דלהלן.

<u>הסבר והוראות עבור השימוש במחלקה Scanner ובקלט:</u>
כדי להשתמש במחלקה Scanner יש לכתוב בראש הקובץ import java.util.Scanner;.
כדי לקרא מקבוץ נבצע

```
Scanner sc = new Scanner ( new File( filename.txt ) );
```

וכדי "לקרא" ממחרוזת str נבצע

```
Scanner sc = new Scanner ( str );
```

ניתן לקרא שורה אחר שורה באופן הבא:

```
while (sc.hasNextLine())
        String line = sc.nextLine();
```

בסיום השימוש באובייקט מטיפוס Scanner יש לסגור אותו ע"י הרצת

```
sc.close();
```

המחלקה Scanner יכולה לסייע גם בקריאת הפרמטרים שבשורה המופרדים באמצעות תווים מיוחדים. אם מעבירים ל-Scanner את המחרוזת עצמה, אפשר "לקרוא" אותה בדומה לקריאת קובץ, באמצעות המתודות, hasNext() ו-next(). ניתן להשתמש גם במתודה nextDouble() שקוראת את המספר הממשי הבא ומחזירה אותו כבר מוכן כ-double, וכן במתודה hasNextDouble() על מנת לוודא שהערך הבא הוא double. יש מתודות דומות נוספות. תיעוד המחלקה מופיע בקישור הבא:
https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
למשל, הרצת קטע הקוד הבא תדפיס: x=1, y=2, z=3

```
Scanner lineScan = new Scanner("1#2#3");
lineScan.useDelimiter("#")
double x = lineScan.nextDouble();
double y = lineScan. nextDouble ();
double z = lineScan. nextDouble ();
System.out.println("x="+x+", y="+y +", z="+z);
```

כדי לפענח את הקלט ולבנות את הגופים, סדר הדברים הוא כזה:

א. יש לקרוא את קובץ הקלט שורה אחר שורה.
את קובץ הקלט ניתן לקרוא באמצעות המחלקה Scanner.
במתודה ה-main ניתן לרשום:

```
Scanner sc = new Scanner ( new File( filename.txt ) );
```

כאשר fileName הוא מחרוזת המכילה את שם קובץ הקלט, למשל "shapesIn.txt",
והקובץ נמצא בתיקייה שבה נמצאים קבצי ה-java.
אם מתקבלת הודעה שגיאה על קובץ לא קיים, ניתן לנסות לבצע את השורה הבאה במקום השורה הקודמת:

```
Scanner sc = new Scanner(new File("./src/"+ filename.txt));
```

בנוסף נוסיף את שורות הייבוא הבאות:

```
import java.io.File;
import java.io.FileNotFoundException;
```

כדי לקרוא שורה מהקובץ ולשים את תכנה במשתנה line אפשר לכתוב:

```
String line;
while (sc.hasNextLine())
        line = sc.nextLine();
```

(אם מתעוררת בעיה הנוגעת ל-unhandled exception,
תנו ל-eclipse (או ל-intelij) לפתור לכם את הבעיה: לחצו על add throws declaration.
כלומר נקבל כי חתימת המתודה main בינתיים היא כדלהלן:

```
public static void main(String[] args) throws FileNotFoundException
```

בהמשך עבור הפלט נוסיף ונשנה חתימה זו להיות
public static void main(String[] args) throws IOException

ב. ניתוח וקריאה של השורה. אתם רשאים לפתור זאת בכל דרך הנראית לכם נכונה, אך גם כאן מומלץ להשתמש במחלקה Scanner כפי הדוגמה לעיל.

<u>הסבר והוראות עבור השימוש ב-פלט בתרגיל:</u>
בחלק זה נשתמש במחלקה Writer.
כדי להשתמש במחלקה זו יש לכתוב בראש הקובץ:

```
import java.io.Writer;
import java.io.FileWriter;
```

במתודה ה-main ניתן לרשום:

```
Writer wr = new FileWriter ("shapesOut.txt");
```

כתיבה נעשית באמצעות המתודה write למשל:

```
wr.write("Hello");
wr.write("\n"); //moves the cursor to a new line
```

בסיום יש לבצע שתי שורות

```
wr.flush();
wr.close();
```

כמו-כן, מימוש זה ידרוש שינוי של החתימה של המתודה main:
public static void main(String[] args) throws IOException
אם הקובץ נכתב אצלכם לתיקייה אחרת, נסו את השורה הבאה:
Writer wr = new FileWriter ("./src/"+"shapesOut.txt");
במקום השורה המופיעה לעיל.

חלק ד' –טיפול בחריגות

בחלק זה נוסיף טיפול במקרי שגיאה בנתונים שבקובץ באמצעות חריגות (ירושה מ-Exception).
סוגי השגיאות בהן יש לטפל, למשל:

1. שורה בה משתמשים באות שונה מ- R,Y לציון סוג הגוף (מנסרה או פרמידה).
2. שורה בה מספר הנתונים המספריים עבור הגדרת הבסיס של הגוף אינו נכון.
3. שורה בפורמט אחר מהנדרש, למשל, הנתונים מופרדים באמצעות " :" במקום באמצעות # ופסיק), שורה בה הנתונים אינם מספריים כנדרש, שורה בה יש יותר מידי/פחות מידי נתונים, ועוד ועוד.
4. רדיוס/אורך לא חיובי וכד'.

הוסיפו למחלקות שכתבתם בחלקים הקודמים טיפול בשגיאות אלו.
המטרה שעבור כל שורת קלט לא תקינה – תיזרק שגיאה.
התכנית לא תעצור בגלל שורות לא חוקיות. התכנית תדלג על שורה שבגינה נזרקה חריגה ותמשיך לקרא את הקובץ ולטפל בשורות הבאות על אף שמתגלות חריגות בחלק מהשורות.
בשורה שיש מספר חריגות, תיזרק אחת מהחריגות הקיימות בה – לבחירתכם.

עליכם לכתוב מחלקה בשם HW3Exception, ובכל מקרה של תקלה בקלט לזרוק חריגה זו (עם הודעה מתאימה המתארת את הבעיה).
התכנית תוציא כפלט קובץ טקסט עם כל השגיאות. הקובץ יכיל את כל השורות הבעייתיות עם פרטים על השגיאה שנוצרה בעקבות זאת (תיאור השגיאה שנוצרה. מספר שורה בקובץ הקלט).
שם הקובץ יהיה #inputFileName_shapesErrors.txt, כאשר #inputFileName הוא שם קובץ הקלט. מיקום הקובץ יחד עם כל קבצי ה-output.

שאר הדברים, כמו הארגומנטים הנשלחים ל-main וכד', ניתן להניח כי הם תקינים.
באתר ישנה דוגמה לקובץ קלט ולקבציי פלט ושגיאות המתאימים לקלט. קבצי החריגות שלכם לא חייבים להיות זהים לגמרי.

חלק ה' - תיעוד הקוד

ישנם שתי דרכים לתעד את הקוד. שתי הדרכים משמשות יחדיו **למטרות שונות**.
דרך אחת באמצעות java-doc – ראו תיעוד של Rhombus בתרגיל בית 2. תיעוד זה נחשף יחד עם חתימת המתודות ה-public. מטרת תיעוד זה היא להודיע למשתמש במתודות על אופן השימוש במתודות ומה מטרתן. במקרה של מתודה דורסת, ניתן להסתפק לפעמים בתיעוד כזה רק במחלקת האב (ואז מבצעים הפניה לתיעוד זה במחלקת הבן).
יש כלי אוטומטי ב- eclipse המוסיף Javadoc. סמנו את המתודה/משתנה/מחלקה בתוך קובץ ה-java ואז לחצן ימני בעכבר – Source ואז Generate Element Content. בד"כ קיצור המקלדת לכך באקליפס הוא ctrl+alt+j. פעולה זו יוצרת Javadoc בסיסי לאלמנט הנבחר, אותו עליכם לערוך.
במקרה של מתודה דורסת, פעולה אוטומטית זו יוצרת הפניה לתיעוד במחלקת האב.

אופן התיעוד האחר הינו ע"י הוספת שורה שלמה או רק סוף שורה לאחר שני קווים נטויים, למשל:

```
int heigt; //this is a field which represents the ….
```

תיעוד זה נועד להסביר למתכנת חלקים קשים יותר בקוד, נקודות עדינות במימוש וכד'.

הוסיפו תיעוד מהסוג הראשון במחלקה Shape מחלק א'. הוסיפו תיעוד לכל המתודות והבנאים וכן תיעוד בראש הקובץ למחלקה.
בנוסף, תעדו באופן התיעוד השני את המחלקה ShapesHandler מחלק ג'.
אין להגזים בתיעוד (בפרט לא בסוג השני). בחירת שמות נכונים, וארגון הקוד בצורה נכונה חוסכים המון שורות תיעוד מיותרות. אם יש צורך בתיעוד ארוך בכדי להבין את הקוד, סימן שהקוד אינו מאורגן בצורה מיטבית.

בהצלחה!