# תרגיל בית מס. 2
# מימוש Triangle

**תכנות OOP ב-Java. העמסה, פולימורפיזם, ועוד.**
**הגשה בזוגות!**
**להגשה במודל עד יום חמישי 05/04 למניינם בשעה 23:00.**
**משקל התרגיל 3 נק' מהציון הסופי**

**אחראית על התרגיל: מיכל הורוביץ**

**בחלק א' נממש מנשק נתון בשתי דרכים שונות. חלק א' שווה 80% מנקודות התרגיל.**
באתר נמצא המנשק: נבדוק שניתן להחליף בין המימושים מבלי לשנות את היישום. כמו כן, נוודא שההתנהגות של שתי המחלקות שניצור זהה, למרות ההבדל באלגוריתמים.

**בחלק ב' תידרשו לענות על שאלות בהקשר למימוש.**
כדאי מאד לקרא את השאלות בחלק ב' לפני תחילת המימוש, שכן השאלות יכולות לכוון אתכם למימוש נכון יותר מבחינת עיצוב התכנה.

## מטרת התרגיל

במטרה ללמוד את ההבדל בין מנשק של מחלקה והמימוש שלה, בתרגיל זה נממש שתי מחלקות בעלות מנשק זהה אך מימוש שונה. שתי המחלקות משמשות לתיאור משולש שווה צלעות במרחב האוקלידי שאחת מצלעותיו מקבילה לציר ה-x, וחיצונית אין ביניהן שום הבדל: קוד העושה בהן שימוש יכול להחליף ביניהן ללא שום הבדל בתוצאה שתתקבל. יחד עם זאת, נראה שהקוד שבתוך שתי המחלקות עשוי להיות נבדל במידה ניכרת. יתר על כן, נראה שהמימוש של חלק מהמנשק בשיטה האחת פשוט יותר מאשר בשנייה, אבל מימושו של יתר המנשק קל יותר דווקא במחלקה השנייה. באתר נמצא המנשק: נבדוק שניתן להחליף בין המימושים מבלי לשנות את היישום. כמו כן, נוודא שההתנהגות של שתי המחלקות שניצור זהה, למרות ההבדל באלגוריתמים.

## אופן ההגשה:
נא עקבו אחרי ההוראות להגשות תרגילים המופיעות תחת לשונית "מדריכים" באתר הקורס.
**הגישו במדויק כנדרש.** שימו לב כי **חובה לבדוק את התרגיל בבודק האוטומטי לפני ההגשה.** בצעו זאת מספיק מוקדם על מנת שתוכלו להספיק לטפל בבעיות באם תקרינה.
שימו לב גם למדיניות האיחורים המפורסמת באתר - **איחור בהגשה יגרור הפחתה של 10 נק' עבור כל יום איחור. מעבר ל-3 ימים התרגיל לא יתקבל.**

את קובץ ה-pdf עם התשובות לשאלות בחלק ב' (קראו לקובץ hw2PartB.pdf) – צרפו יחד עם קבצי המימוש בתוך ה-zip שאתם מגישים בבודק האוטומטי.

## חלק א' - תיאור התרגיל למימוש:

הורידו לתוך סביבת העבודה את קבצי ה-java הנתונים במטלה.
ניתן להוריד את הפרוייקט גם מהגיט
https://github.com/michalHorovitz/OOP2022Public

ב-eclipse השתמשו ב File->Import->Git->CloneURI , ובחרו את הפרוייקט היחיד הנמצא שם ששמו OOP-HW2-Triangle.
אם אתם עובדים ב VDI, מומלץ לשנות את המיקום המוצע לפרויקט בתיקייה כלשהי בכונן H.

שימו-לב שאין לשנות את הקבצים הנתונים כלל. כל הקבצים, הנתונים לכם ואלו שתכתבו בעצמכם, יהיו באופן ישיר תחת תקיית src ולא בחבילות אחרות.

המחלקות הנתונות לכם:

1. מנשק Triangle אותו עליכם לממש לפי ההוראות דלהלן.

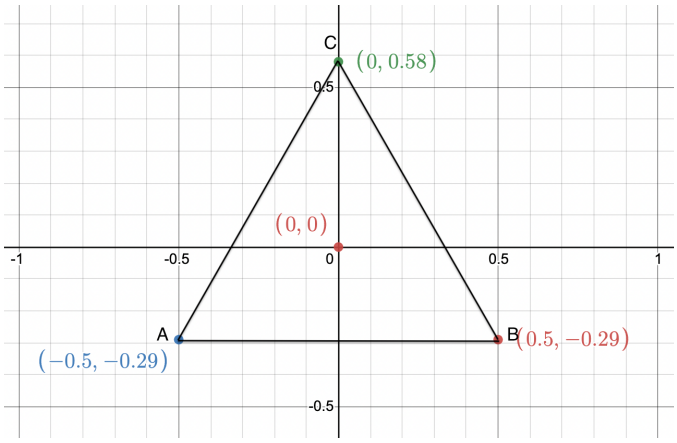
בנוסף נתונות מחלקות עזר:

2. Point – מתארת נקודה במערכת הצירים, ויש שם מתודות שימושיות.
3. HW2Utils – מכילה מתודות סטטיות לעזר. שימוש במתודות אלו יחסוך לכם הרבה עבודה וימנע טעויות במימוש.
4. מחלקות בדיקות:

TestGetters ,TestSetters ,TestUpdateMethods ,TestOtherMethods , TestBonus

הסבר על אופן הרצת מתודות בדיקות ב-Junit, יש הסבר תחת לשונית "מדריכים".
כל מתודת test מזכה בבודק האוטומטי ב-4 נקודות.

המחלקות שתצרו מייצגות משולש שווה צלעות כך שאחת מצלעותיו מקבילה לציר ה-x.
על משולש שווה צלעות ונוסחאות נוחות ניתן לראות בויקיפדיה – משולש שווה צלעות.
נסמן את שני הקודקדים המתארים את הצלע המקבילה לציר ה-x, ב- A עבור הקדקוד השמאלי, וב- B עבור הקדקוד הימני. הקדקוד השלישי יסומן ב-C.
נכנה את נקודת מפגש התיכונים ב"מרכז המשולש". שימו לב כי במשולש כזה כל תיכון הוא גם גובה וגם חוצה זווית.
למשל, הציור להלן מתאר משולש שווה צלעות שצלע AB מקבילה לציר ה-x, מרכז המשולש הוא (0,0) ואורך הצלע היא 1.




בדוגמה לעיל הנקודה C מעל הצלע AB ונכנה משולש זה כ-up. אפשרות נוספת היא שהמשולש "הפוך", כלומר נקודה C נמצאת מתחת לצלע AB ונכנה משולש זה ב-under. לתכונה זו של המשולש נקרא "כיוון".

המנשק, שעל שתי המחלקות לממש, נקרא Triangle. המנשק Triangle נמצא באתר. **אין לשנות מנשק זה.** עליכם לממש מנשק זה בשתי דרכים שונות המתוארות להלן.

**<u>עליכם לממש את המנשק Triangle בשתי דרכים שונות.</u>**

מחלקה אחת נקראת **Triangle1**, והיא מייצגת את המשולש בעזרת שני שדות:
שדה אחד מטיפוס Point המתאר את נקודת "מרכז המשולש",
ושדה שני מטיפוס double המתאר את גובה המשולש. בנוסף הסימן (חיובי/שלילי) של השדה השני מסמל אם מדובר במשולש up או under. סימן חיובי מתאר משולש up (כלומר C מעל הצלע AB), וסימן שלילי מתאר משולש under (כלומר, C מתחת הצלע AB). הגובה צריך להיות שונה מ-0.

מחלקה שניה נקראת **Triangle2**, והיא מייצגת את המשולש בעזרת שלשה שדות:
שדה אחד מסוג Point מתאר את הנקודה השמאלית (A),
שדה שני מתאר את אורך הצלע. ערך שדה זה צריך להיות גדול מ-0.
שדה בוליאני המתאר אם המשולש up או under (כלומר אם נקודה C מעל/מתחת הצלע AB).

כלפי חוץ, שתי המחלקות, **Triangle1 ו-Triangle2,** מציגות את אותו המנשק: זה המתואר ב-Triangle.

על כל אחת מהמחלקות לממש את **<u>כל המתודות</u>** שבמנשק.

כמו כן, יש לממש בכל אחת מהן **<u>שני בנאים</u>**.
בנאי אחד המקבל את הפרמטרים שמתאימים למחלקה לפי הסדר המצוין לעיל,
ובנאי שני ללא פרמטרים, היוצר מופע ברירת מחדל: משולש שווה צלעות שמרכזו בראשית הצירים ואורך צלעו 1. ראו דוגמה להלן.

<u>אם אחד מהפרמטרים שנשלחים לבנאי/למתודה כלשיה אינם תקינים</u> (למשל מספר שלילי המייצג אורך צלע, גובה 0), אז הבנאי ייצור מופע ברירת מחדל (משולש שווה צלעות שמרכזו בראשית הצירים ואורך צלעו 1).
<u>במימוש של מתודות ה-get עבור השדות מסוג Point</u>, יש ליצור אובייקט חדש שהוא העתק של הנקודה (ע"י מתודת ()copy הנמצאת במחלקה Point) ולהחזיר מצביע לעותק.
<u>באופן דומה במימוש של מתודות ה-set או ה-update</u> עבור הקדקודים, יש להציב בשדה עותק של האובייקט המתקבל כפרמטר ולא את הפרמטר בעצמו.

<u>שימו לב כי מתודות העדכון</u> (מלבד עדכון המרכז) משנות את התכונה הנדרשת ללא שינוי של מרכז המשולש. עקבו בצורה מדויקת אחרי התיעוד הנתון לכם במנשק Triangle.java,
למשל, המתודה updateHeight מקבלת רק גובה חיובי (אחרת לא מבצעת כלום), תעדכן את גובה המשולש, אבל לא משנה את מרכז המשולש, ולא את הכיוון של המשולש (up/under).
מתודת עדכון מרכז המשולש, תשנה את מרכז המשולש ללא שינוי של שאר הפרמטרים: גובה, אורך צלע, כיוון.

<u>כמו כן, זכרו כי ההתנהגות בשני המימושים צריכה להיות זהה.</u> כלומר, אם נתונים לנו שני עצמים שונים המתארים את אותו המשולש, עצם אחד מסוג Triangle1 ועצם שני מסוג Triangle2, כל מתודה שנפעיל על שניהם תשנה את העצמים הנ"ל כך שעדיין ייצגו את אותו משולש ויחזירו את אותו ערך בדיוק. בפרט, <u>קריאה למתודה updateHeight(x) עבור עצם ממחלקה</u> Triangle2 תשנה את השדות של העצם כך שיתאר משולש שגובהו השתנה, אך נקודת המרכז והכיוון לא השתנו.

אם יש לכם <u>קטע קוד שמשתמשים בו במתודות שונות באותה מחלקה</u>, ניתן (וכדאי) לאחד אותם במתודה משותפת שהיא private. מתודה המוגדרת private היא מתודה פרטית של המחלקה ואינה ניתנת לגישה מחוץ למחלקה. מתודות אלו משמשות כפונקציות עזר במחלקה למניעת כפילויות של קוד בתוך המחלקה ולכתיבת קוד קריא ויפה יותר.

**אין להשתמש במחלקה אחת שממשתם בתוך המחלקה השנייה אותה אתם ממשים!**

**<u>חשוב</u>: אם <u>יש מתודות שניתן לממש באופן זהה בשתי המחלקות</u>, חישבו כיצד לאחד את המימוש במקום אחד בשיטות 'תכנות מונחה עצמים' שלמדנו.**

המחלקה HW2Utils תסייע לכם רבות ותמנע מכם את הצורך בחישובים 'מייגעים'.

**<u>פתרו באופן מסודר:</u>**

א. התחילו במימוש בנאים ומתודות get (5 מתודות ראשונות במנשק). ואז בדקו על ידי מחלקת הבדיקות TestGetters.
הסבר על אופן הרצת מתודות בדיקות ב-Junit, יש הסבר תחת לשונית "מדריכים".

ב. ממשו מתודות עדכון שדות משתי המחלקות (4 מתודות הבאות במנשק), ובדקו את המימוש על ידי מחלקת הבדיקות TestSetters.

ג. המשיכו למימוש 4 המתודות הבאות (scale ו-*move), ובדקו את המימוש על ידי מחלקת הבדיקות TestUpdateMethods.

ד. ממשו את שאר המתודות (מלבד contains) ובדקו את המימוש על ידי מחלקת הבדיקות TestOtherMethods.

ה. **<u>בונוס: 8 נקודות.</u>** ממשו את מתודות contains, ובדקו את המימוש על ידי מחלקת הבדיקות TestBonus.

ו. אחרי שעברתם את כל הבדיקות, בצעו zip לכל התרגיל על פי ההוראות הנתונות לכם בלשונית "מדריכים", ובדקו את התרגיל בבודק האוטומטי.
הבדיקות האוטומטיות מבצעות את אותם הבדיקות הנתונות לכם.

ז. אל תשכחו לצרף לזיפ המוגש pdf של חלק ב'. חשוב! לפני הגשת הקובץ הסופי, בדקו שוב בבודק האוטומטי לוודא שהכל תקין.

**אין להגיש קוד שאינו מתקמפל. קוד שאינו עובד נכונה עבור הבדיקות הנתונות ינוקד בהתאם.**

**בנוסף, <u>הקוד צריך להיכתב בצורה נכונה</u> כפי שלמדנו (שמירה על סדר הגדרות, הזחה נכונה, שמות מתאימים וכו'). חלק מהניקוד ניתן על דברים אלו.**

**<u>כמה עזרים ב-eclipse :</u>**

1. כשמגדירים למחלקה מנשק למימוש עושים זאת ע"י הו "implements Triangle" בהגדרת המחלקה (ראה דוגמאות במצגת של יחידה 3).
כאשר רושמים זאת, eclipse מיד מציין שגיאה אם המימוש אינו כולל את כל המתודות של המנשק (מה שקרוב לוודאי המצב אם המחלקה חדשה וריקה). לחיצה כפולה על סימן השגיאה האדום בשולי הקוד פותח חלון של אפשרויות תיקון אוטומטיות. אחת מהן (לרוב הראשונה) היא להגדיר את כל המתודות הדרושות באופן ריק. אתם מוזמנים להשתמש באופציה זו תחילה, ואחר כך לעבור על כל אחת מהמתודות הריקות שנוצרו ולמלא אותן בתוכן מתאים. אבל אל תשאירו מתודות ללא מימוש! כל אחת מהן צריכה תוכן.
2. ב-eclpise יש אפשרות לסידור ההזחות במחלקה באופן אוטומטי. מומלץ מאד לבצע זאת לכל מחלקה לפני ההגשה. Source → Format (ctrl+shift+F)

## **חלק ב' – שאלות על המימוש:**

בחלק הראשון של תרגיל בית 2 מימשתם את הממשק Triangle.
בחלק זה של התרגיל, עליכם לענות על השאלות הבאות.

1. (10 נק') האם ישנה העמסה (overloading)? אם כן, היכן?
2. (15 נק') היכן בא לידי ביטוי רב-תצורתיות (פולימורפיזם)?
3. (15 נק') היכן בא לידי ביטוי מנגנון ההסתרה (hiding)?
4. (20 נק') מה היה קורה לו המתודה getCenter() במחלקה Triangle1 הייתה מחזירה מצביע לשדה ולא מצביע לאובייקט חדש?
   הראו קטע קוד (main) קצר ככל האפשר המסביר בעיה כלשהי שעלולה להיווצר.
5. תכנון קוד בצורה נכונה:
6. (40 נק') **אם <u>יש מתודות שניתן לממש באופן זהה בשתי המחלקות</u>,** התבקשתם לאחד את המימוש במקום אחד בשיטות 'תכנות מונחה עצמים' שלמדנו.
   א. (30 נק') אלו מתודות בחרתם לממש במקום משותף, ואלו לא, מדוע?
   ב. (15 נק') איזו דרך בחרתם למימוש המשותף?
   ג. (5 נק') ציינו חסרון בולט באיחוד המימוש של המתודות.

# בהצלחה!