

תוכנה 1

תרגיל מספר 5

הנחיות כלליות:

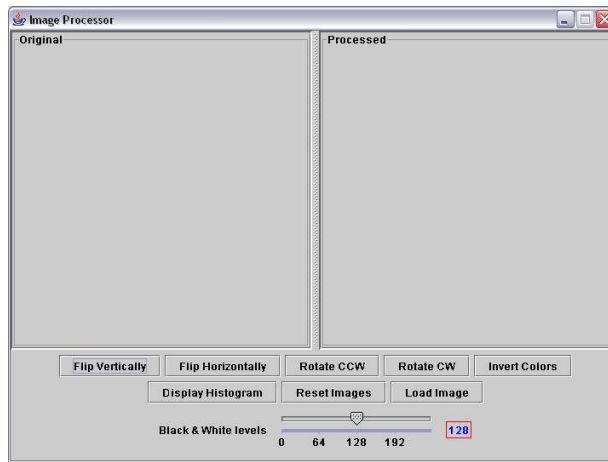
- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה במערכת ה VirtualTAU בלבד (<http://virtual2002.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip) קובץ ה zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
 - ב. קבצי ה java של התוכניות אותם התבקשתם לממש.
 - ג. קובץ טקסט עם העתק של כל קבצי ה java

חלק א':

בחלק זה נממש כלים של עיבוד תמונה. השירותים שנכתבו יסובבו, יהפכו וישנו צבע עבור תמונות בגווני אפור. אנחנו מספקים לכם קובץ jar המכיל שירותים לטעינת תמונות מקובץ וכן ממשק משתמש גרפי להפעלת הפונקציות לעיבוד תמונה.

מה עליכם לעשות

באתר הקורס מופיע שלד של המחלקה ImageProcessing. עליכם לממש את כל המתודות הריקות המוגדרות במחלקה. אין לשנות את מתודת ה-main. תיעוד עבור המחלקה ניתן למצוא [כאן](#). כל אחד משירותי המחלקה מקבל כקלט מערך דו-מימדי של שלמים המייצג תמונה. כדי לפשט את התוכנית, נתמוך רק בתמונות בגווני אפור. כל תא במערך מקביל לפיקסל בתמונה, ומכיל מספר בין 0 ו-255, כאשר 0 הוא שחור, 255 הוא לבן, והמספרים ביניהם הם גווני אפור. כפלט השירותים (חוץ מ-'histogram') מחזירים מערך דו-מימדי חדש המייצג את התמונה לאחר העיבוד המבוקש, ופלט זה מוצג בממשק המשתמש הגרפי במסגרת בצד ימין, שכותרתה "Processed".

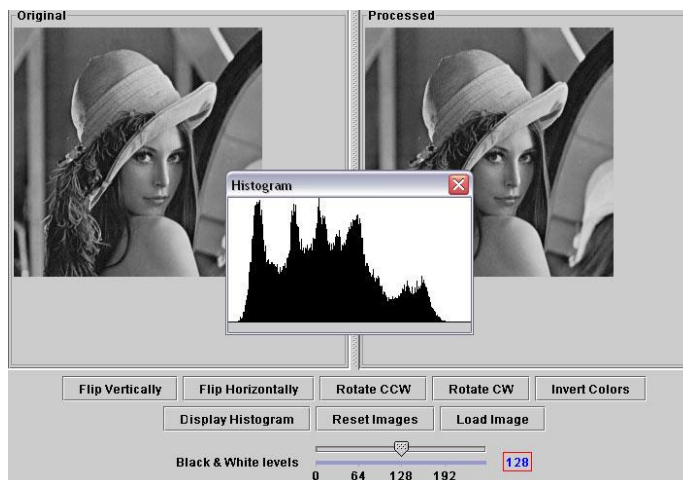


פרטים טכניים

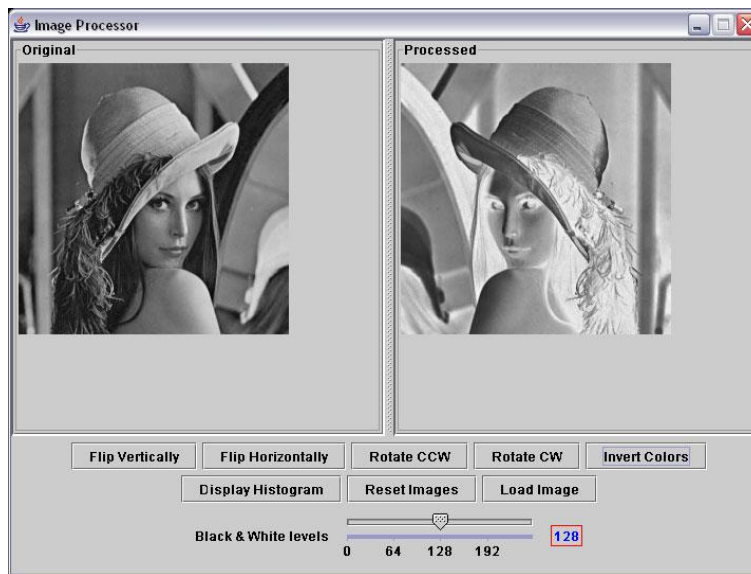
הורידו מאתר הקורס את הקובץ [image_processor.jar](#). הוסיפו אותו ל- `java build path` של הפרוייקט שלכם (כפי שנעשה בתרגיל הקודם). ניתן לבדוק את המימוש שלכם בכל עת על-ידי הרצת המחלקה `ImageProcessing`. ההרצה תפתח חלון של הממשק הגרפי לעריכת תמונה (ראו תמונת מסך למטה). בתחילה, לחיצה על הכפתורים לא תגרום לשום שינוי. ככל שתממשו עוד ועוד מתודות של המחלקה, כפתורים נוספים בחלון יתחילו לפעול (הם יריצו את המימושים שלכם), כך שתוכלו להשתמש בממשק הגרפי כדי לבדוק את נכונות התוכנית שכתבתם.

הממשק הגרפי

- מתוך חלון הממשק הגרפי ניתן לטעון תמונה בגווני אפור בעזרת הכפתור 'Load Image'. קובץ זיפ המכיל שתי תמונות לדוגמא ניתן למצוא [כאן](#).
- המתודה 'histogram' מחזירה מערך חד מימדי של 256 תאים, כאשר התא בעל האינדקס i מכיל את מספר הפיקסלים בתמונה בעלי צבע i (למשל, אם התא בעל אינדקס 0 מכיל את המספר 30, בתמונה הנתונה ישנם 30 פיקסלים שחורים). לחיצה על הכפתור 'Display Histogram' תגרום להצגת ההיסטוגרמה של התמונה.



- המתודה invert הופכת את הצבעים בתמונה כך ש-0 הופך ל-255, 1 הופך ל-254 וכך הלאה.
 - המתודה toBW הופכת את התמונה לתמונה בשחור ולבן בלבד. כל ערך של פיקסל מעל ה-threshold (כולל) הופך ללבן (255) והיתר לשחור (0).
 - המתודה rotate90CW מסובבת את התמונה ב-90 מעלות עם כיוון השעון, ואילו המתודה rotate90CCW מסובבת אותה ב-90 מעלות נגד כיוון השעון.
 - המתודה flipX מבצעת שיקוף של התמונה לפי ציר ה-x, כלומר, היפוך אנכי. בדומה, המתודה flipY מבצעת היפוך אופקי. ראו את האנימציה [הבאה](#) להמחשה.
- לדוגמא, בתמונת המסך הבאה בוצעו על התמונה 'lena.jpg' היפוך אופקי והיפוך צבעים.



חלק ב'

בחלק זה נתמודד עם שתי בעיות הקשורות למשחקי מילים.

1. נרצה לבנות אפליקציה אשר תעזור לנו בפתרון תשבצים. לשם כך נבנה מחלקה אשר, בהינתן מילון ודוגמא (pattern) מחזירה את כל המילים במילון המתאימות למחרוזת דוגמא. נממש את

האפליקציה במחלקה שתקרא: DictionaryHelper

הדוגמא היא מחרוזת אשר מכילה אותיות או את התו '-', אשר יסמן כי איננו יודעים איזו אות מופיעה במקום זה.

i. ממשו את הפונקציה findMatches אשר מקבלת מילון (מערך של מילים) ומחרוזת דוגמא ומחזירה מערך המכיל את כל המילים המתאימות לדוגמא שסופקה. להלן מספר הרצות לדוגמא:

```
findMatches("m-d-m", dictionary) -> { "madam", "modem" }
```

```
findMatches("-n--r-m", dictionary) -> { "anagram", "interim" }
```

```
findMatches("cycl--", dictionary) -> { "cycled", "cycles", "cyclic" }
```

```
public static String[] findMatches(String pattern, String[] dictionary) {  
}
```

ii. השלימו את המימוש כך שניתן יהיה להפעיל את השירות משורת הפקודה (command line). התוכנה הממומשת תקבל שני ארגומנטים: הראשון הוא שם הקובץ המכיל את המילים במילון והשני הוא מחרוזת הדוגמא.

הערה ראשונה: ניתן להוריד [מילון לדוגמא](#) מאתר הקורס¹.

הערה שניה: ניתן להיעזר במחלקה [DictionaryReader](#), אשר מופיעה באתר הקורס. למחלקה פונקציה יחידה, readFromFile, אשר מקבלת שם קובץ של מילון ומחזירה מערך עם המחרוזות במילון:

```
String[] dictionary = DictionaryReader.readFromFile("dictionary.txt");
```

¹ המילון הורד מהאתר: <http://www.mieliestronk.com/wordlist.html>

2. נרצה לבנות אפליקציה למציאת אנאגרמה של משפט כללי. האפליקציה תקבל משפט (כלומר מחרוזת המכילה מספר מילים) ותדפיס סדרה של מילים חוקיות המורכבות מהאותיות שהופיעו במשפט המקורי. במימוש שלנו נסתפק בפלט של אנגרמה אחת עבור משפט קלט אחד. שימו לב: רווחים וסימני פיסוק אינם נחשבים כאות וניתן להתעלם מהם.

ע"מ לבצע את המטלה ביעילות, נייצג מילה (או משפט) ע"י מערך של מספרים, כאשר במקום ה- i יופיע מספר הפעמים בה האות ה- i בא"ב הלועזי (אנגלי) מופיעה במילה. להלן מספר דוגמאות לייצוג מחרוזות במערך:

abba

2	2	0	0	...	0
---	---	---	---	-----	---

abcccc

1	1	4	0	...	0
---	---	---	---	-----	---

נשים לב, שבהינתן ייצוגים כנ"ל למשפט s ומילה w , המילה עשויה להיכלל באנאגרמה של המשפט רק אם ערכי כל תא בייצוג המילה קטנים או שווים מהערכים המתאימים בייצוג המשפט. כלומר, לכל תא i מתקיים: $w[i] \leq s[i]$.

בנוסף, נבחין כי אם המילה w היא אכן חלק מאנאגרמה של s אזי נוכל לקבל בעיה "קטנה" יותר בה יש לחפש אנאגרמה עבור משפט אחר, המורכב מאותיות s למעט אלו שהופיעו ב- w . משפט כזה ייוצג ע"י מערך בו ערך התא במקום ה- i הוא בדיוק $s[i]-w[i]$.

על סמך הבחנות אלו נוכל לממש את האלגוריתם הבא עבור משפט s :

- i. נבנה מילון של מילים חוקיות
- ii. נייצג את כל מילים במילון עפ"י הייצוג שתואר לעיל
- iii. נייצג את s עפ"י הייצוג הנ"ל
- iv. נעבור על רשימת המילים החוקיות ונבדוק האם ניתן לשלבן באנאגרמה:

אם המילה הנוכחית עשויה להיות חלק מאנאגרמה, אזי:

$$(1) \text{ נבנה ייצוג חדש } s', \text{ בו } s'[i]=s[i]-w[i]$$

(2) אם בייצוג s' ערכי כל התאים הם 0, אזי מצאנו אנאגרמה חוקית וסיימנו.

(3) אחרת, נפעיל שוב את שלב ד' (ברקורסיה) עם s'.

עליכם לממש את המחלקה AnagramFinder המיישמת את האלגוריתם המוצע. להלן דוגמאות הרצה של המחלקה:

```
> java AnagramFinder dictionary.txt "most of it"
moot fist
> java AnagramFinder dictionary.txt "challenge me"
helm glen ace
> java AnagramFinder dictionary.txt "dinosaurs"
run adios
```

שאלות ותשובות:

i. מהי רשימת המילים החוקיות?

תשובה: נשתמש במילון כמו בסעיף הקודם.

ii. אילו אותיות הן חוקיות?

תשובה: כולן, אולם אנו נקודד רק את האותיות המעניינות אותנו. נשתמש באותיות קטנות ונעזר במתודה isLetter של המחלקה Character.

iii. מה לעשות במקרה שיש מספר תשובות חוקיות?

תשובה: מספיק לתת פתרון חוקי אחד.

iv. מה לעשות אם לא קיימת מילה המתאימה כחלק מהאנאגרמה באחד משלבי הרקורסיה?

תשובה: יש להחזיר את האנאגרמה החלקית שנמצאה (ייתכן שהיא תהיה ריקה). למשל, ייתכן שעבור "dinosaurs z" נחזיר רק "run adios" ועבור "zzz" נחזיר "".

v. נראה כי ניתן לייעל את הקוד, לדוגמא, מילה ארוכה לא יכולה להיות אנאגרמה של מילה קצרה יותר.

תשובה: נכון, ניתן להשתמש באורך המחרוזת כדי לייעל את המימוש. ניתן, לדוגמא, להוסיף תא למערך שיכיל את מספר האותיות במילה במיוצגת ולהשתמש בערך זה במהלך החישוב.

חלק ג':

בשאלה זו תממשו תוכנית אשר מייצרת תמונה גרפית פשוטה. אינכם נדרשים לממש את החלק הגרפי בעצמכם, אלא להשתמש במחלקה Turtle (המופיעה באתר הקורס).

Turtle graphics ו-LOGO – הקדמה

LOGO היא שפת תכנות המשמשת לעתים לצורך עריכת היכרות עם מושגי תכנות בסיסיים וגיאומטריה של המישור לילדים. סביבת LOGO כוללת חלון שמייצג את המישור וצב שנע על המישור. לצב יש זנב אשר יכול להיות מורם או מורד. אם הצב הולך כאשר זנבו מורד, הוא מצויר אחריו קו. כאשר זנבו מורם, לא נוצר קו. המטרה ב-LOGO היא לצייר תמונות שונות באמצעות מתן הוראות לצב.

לצב יש מיקום במישור וכיוון אליו ראשו פונה. ניתן להורות לצב להתקדם בקו ישר קדימה או לשנות את כיוונו, וכך ולגרום לו ליצור תמונה. למשל, ההוראה 'forward 30' תגרום לצב להתקדם 30 יחידות מרחק קדימה (בכיוון אליו הוא פונה). ההוראה 'left 45' תגרום לצב להסתובב ארבעים וחמש מעלות נגד כיוון השעון מכיוונו הנוכחי. להלן רשימה של הוראות אפשריות לצב:

forward x	advance x units forwards
backwards x	move x units backwards
left x	turn x degrees counter-clockwise
right x	turn x degrees clockwise
tail up	lifts the turtle's tail
tail down	lowers the turtle's tail

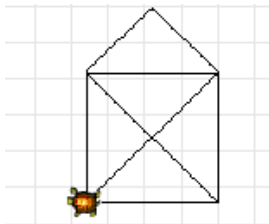
מה אינכם נדרשים לממש

מימוש לאמולציה של LOGO נתון לכם באתר הקורס, דרך המחלקה Turtle. מחלקה זו מגדירה את כל הפעולות של צב LOGO. ניתן לייצר עצם מסוג צב, ולתת לו הנחיות באמצעות קריאה למתודות המפורטות בטבלה הבאה. מומלץ גם לקרוא את [התיעוד של המחלקה Turtle](#).

moveForward(x)	advance x units forwards
moveBackward(x)	move x units backwards
turnLeft(x)	turn x degrees counter-clockwise
turnRight(x)	turn x degrees clockwise
tailUp()	lifts the turtle's tail
tailDown()	lowers the turtle's tail

המחלקה [Pentagon](#) היא דוגמה לתכנית המשתמשת במחלקה Turtle כדי ליצור ציור "מחומש". להלן חלק ממימוש המחלקה עם מסי הערות.

```
class Pentagon {
    public static void main(String[] args) {
        Turtle leonardo = new Turtle(); // Creates the turtle
        leonardo.tailDown();           // Start painting
        leonardo.moveForward(100);     // Advances the turtle
                                        // forward by 100 units
        // ...
    }
}
```



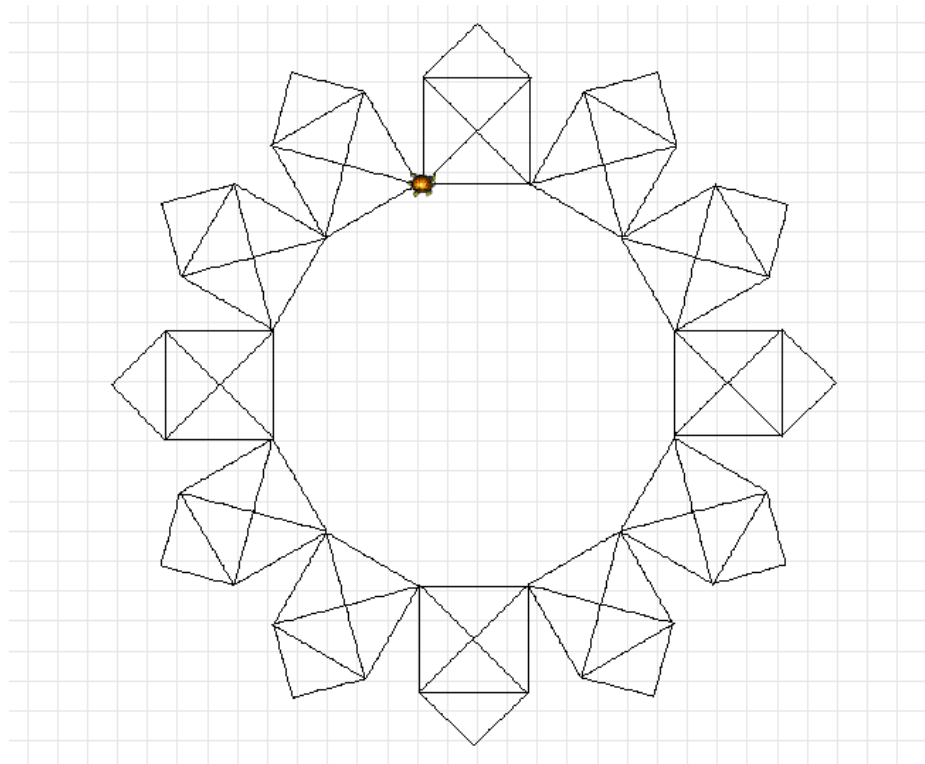
ציור אוילר מאופייין בכך שניתן לציירו מבלי להרים את העט מהנייר. במילים אחרות, זהו ציור שניתן לצייר מבלי לחזור על אותו קו פעמיים. זוהי דוגמה לציור אוילר "מחומש" שניתן ליצור בעזרת הצב.

מה עליכם לעשות

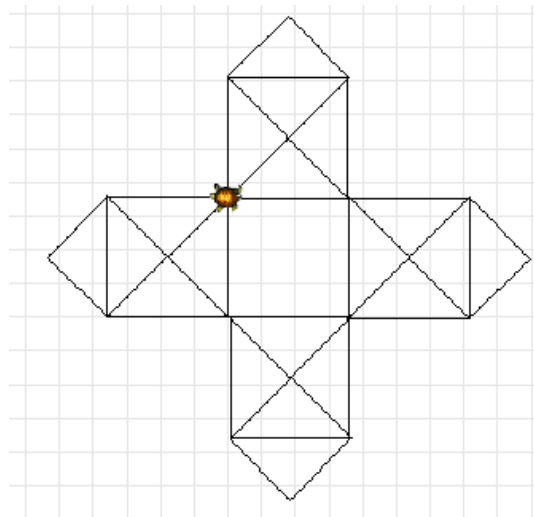
כתבו מחלקה בשם TurtleDrawing ובה מתודת main אשר מקבלת מהמשתמש מספר שלם n ומציירת את "מחומש" אוילר הנ"ל n פעמים. זאת, באמצעות יצירת עצם מסוג צב ושימוש במתודות כדי לתת לו פקודות. אחרי כל מחומש הצב מסתובב כך שकेבור n פעמים הוא משלים מעגל. ניתן להניח שהקלט מהמשתמש תקין (מספר שלם, גדול או שווה ל-3, ניתן להמרה ל- integer וכו'). **שימו לב!** הקלט מהמשתמש לא יתקבל כארגומנט של התוכנית אלא דרך הקלט הסטנדרטי (System.in).² השתמשו במחלקה [java.util.Scanner](#) (שנלמדה בתרגול) על מנת לקרוא את הקלט. להלן הפלט של

TurtleDrawing עבור n==12:

² תזכורת: כדי להעביר לתוכנית קלט דרך System.in ב-Eclipse יש לכתוב בחלון ה-console במהלך ריצת התכנית, ולסיום להקיש Enter.



זהה הפלט עבור $n=4$:



בתום הציור, השתמשו בפקודה `hide()` כדי להעלים את הצב.

פרטים טכניים

כדי להשתמש במחלקה Turtle, הוסיפו את `logo_turtle.jar` ל-Build path של הפרוייקט.