

תוכנה 1 – סתיו תשע"ד

תרגיל מספר 7

מנשקים Interfaces

הנחיות כלליות:

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

- הגשת התרגיל תעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש aviv יקרא הקובץ aviv_hw7.zip). קובץ ה-zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
 - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש, כולל תיקיות החבילה.

חלק א' (50 נק') - צורות

בחלק זה נתרגל כתיבת מחלקות המממשות מנשק נתון. לאחר מכן, נשתמש בהן בעזרת תכונת הפולימורפיזם של תכנות מונחה עצמים ב-Java.

נתון המנשק Shape:

```
public interface Shape {

    // Returns the shape's area
    public float getArea();

    // Returns the shape's perimeter
    public float getPerimeter();

    // Returns a string representation of the shape details.
    // The returned string would include the shape's name, its fields
    // and their corresponding values.
    public String getDetails();

}
```

1. עליכם לכתוב שלוש מחלקות המממשות את המנשק Shape. להלן פירוט חתימת הבנאי של כל אחת מהן (כל בנאי מקבל כפרמטרים את הנתונים הנדרשים להגדרת הצורה אותה הוא יוצר):

```
// x and y are the coordinates of the upper left corner
public Rectangle (int x, int y, int width, int height)

// x and y are the coordinates of the upper left corner
public Square (int x, int y, int width)

// x and y are the coordinates of the ellipse center
// semiMajorAxis is the 'large radius' of the ellipse
// semiMinorAxis is the 'small radius' of the ellipse
public Ellipse(int x, int y, int semiMajorAxis, int semiMinorAxis)
```

- כל המחלקות בחלק זה ימומשו כחלק מחבילה בשם `il.ac.tau.cs.sw1.shapes`.
- על כל מחלקה להכיל שדות פרטיים אליהם ניתן לגשת עם מתודות `getter` ו-`setter` ציבוריות.
- כל אחת משלוש המחלקות מממשת את הממשק `Shape` ועל כן צריכה לממש את המתודות המוגדרות בו.
- המתודה `getDetails` תחזיר מחרוזת המייצגת את פרטי הצורה ותכיל את שם הצורה, ואת כל השדות שבה לצד הערכים שלהם. דוגמא למחרוזת שתוחזרנה עבור מלבן, ריבוע ואליפסה בהתאמה:

"Rectangle: X=100, Y=150, Width=50, Height=50"

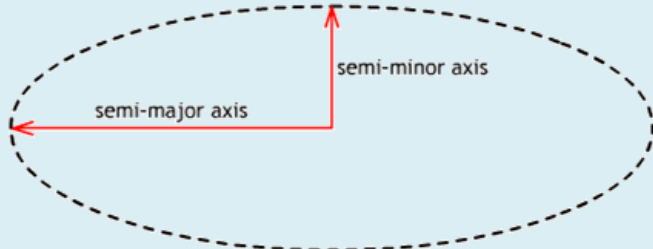
"Square: X=100, Y=150, Width=50"

"Ellipse: X=100, Y=150, SemiMajorAxis=50, SemiMinorAxis=30"

- המתודות `getArea` ו-`getPerimeter` יחזירו את שטח והיקף הצורה בהתאמה (שימו לב שלצורך החישוב יש להשתמש בקבוע `Math.PI` לקבלת הערך פאי).

אליפסה

S = $\pi * \text{SemiMajorAxis} * \text{SemiMinorAxis}$: שטח האליפסה יחושב ע"י הנוסחה:
P = $\pi * (\text{SemiMajorAxis} + \text{SemiMinorAxis})$: היקף האליפסה יחושב ע"י הקירוב הבא:



The diagram shows a dashed ellipse. Two red arrows originate from the center. One arrow points horizontally to the left and is labeled "semi-major axis". The other arrow points vertically upwards and is labeled "semi-minor axis".

2. עתה נעבור לכתיבת התוכנית (המחלקה) `ShapeDimensionCalculator` שתעשה שימוש בטיפוסי הצורות שיצרתם בסעיף הקודם. התוכנית תקבל מהמשתמש (דרך ה-`System.in`) סדרה של נתוני צורות, תחשב עבור כל צורה את שטחה ואת היקפה, ולבסוף תדפיס דו"ח מסכם לקובץ הפלט.

נתחיל בכתיבת המתודה `getShapesFromUser` (בתוך המחלקה `ShapeDimensionCalculator`) שהינה בעלת החתימה הבאה:

```
public static Shape[] getShapesFromUser()
```

המתודה תדפיס למסך תפריט שיאפשר למשתמש להגדיר צורות, תקלוט את נתוני הצורות דרך המקלדת, ותחזיר לבסוף מערך מטיפוס `Shape` (שימו לב שמערך מטיפוס של הממשק `Shape` יכול להכיל מופעים של מחלקות שונות המממשות את הממשק).

עבור כל צורה שנבחרה ע"י המשתמש, תבקש המתודה את הנתונים הנדרשים לבנייתה. בגמר הכנסת הנתונים עבור צורה מסויימת, המתודה תיצור אובייקט מהמחלקה המתאימה (אליפסה,

מלבן או ריבוע) ותוסיף אותו למערך הצורות. לאחר מכן תדפיס המתודה הודעה שהצורה התווספה (תוך שימוש במתודת ה-getDetails) של אובייקט הצורה.

דוגמא לתוכן חלון ה-Console בגמר הרצת המתודה (קלט המשתמש מופיע בירוק):

```
Shape Dimension Calculator

Please choose shape type:
E - Ellipse
R - Rectangle
S - Square
X - Exit
E
Please enter X coordinate: 100
Please enter Y coordinate: 100
Please enter semi-major axis length: 50
Please enter semi-minor axis length: 30
Shape added: [Ellipse: X=100, Y=100, SemiMajorAxis=50, SemiMinorAxis=30]

Please choose shape type:
E - Ellipse
R - Rectangle
S - Square
X - Exit
R
Please enter X coordinate: 200
Please enter Y coordinate: 200
Please enter width: 50
Please enter height: 30
Shape added: [Rectangle: X=200, Y=200, Width=50, Height=30]

Please choose shape type:
E - Ellipse
R - Rectangle
S - Square
X - Exit
S
Please enter X coordinate: 300
Please enter Y coordinate: 300
Please enter width: 50
Shape added: [Square: X=300, Y=300, Width=50]

Please choose shape type:
E - Ellipse
R - Rectangle
S - Square
X - Exit
K
Unknown command. Please try again.

Please choose shape type:
E - Ellipse
R - Rectangle
S - Square
X - Exit
X
```

עליכם לממש את המתודה על פי דוגמת הפלט המופיע לעיל.

בדוגמת הרצה זו, המתודה תחזיר מערך מטיפוס Shape המכיל 3 אובייקטים שנוצרו על פי נתוני המשתמש (המעריך יכול להיות גדול יותר ושאר ערכיו יהיו Null).

הערות:

- המשתמש יכול להכניס 20 צורות לכל היותר.
- ריצת המתודה תסתיים אם המשתמש בחר באפשרות היציאה בתפריט (הקיש על 'X'), או אם הוכנסו 20 צורות.
- ניתן להניח שהמשתמש מכניס קלט חוקי (אותיות או מספרים שלמים כנדרש), אך אם הוקש תו שאינו כלול בתפריט, יש להדפיס למסך "Unknown command. Please try again.", ולהדפיס מחדש את התפריט (ראו קבצי הדגמה).

3. ממשו את המתודה writeShapesToFile אשר מקבלת מחרוזת המייצגת שם קובץ-פלט ומערך מטיפוס Shape המחזיק אובייקטים של צורות, וכותבת דו"ח מסכם לקובץ על פי הדוגמה המובאת בהמשך.

חתימת המתודה:

public static void writeShapesToFile(String outputFilename, Shape[] shapes)

המתודה תכתוב לקובץ את רשימת הצורות ביחד עם ערכי ההיקף והשטח של כל צורה. בסיום הקובץ יודפס סה"כ מספר הצורות הכלולות במערך, סכום ההיקפים שלהם וסכום השטחים שלהם.

עליכם לממש את המתודה כך שתשמור לקובץ את הפלט הבא עבור הנתונים שהוכנסו בדוגמה של הסעיף הקודם:

Shape Dimension Calculator

Ellipse: X=100, Y=100, SemiMajorAxis=50, SemiMinorAxis=30
Area: 4712.39, Perimeter: 251.33

Rectangle: X=200, Y=200, Width=50, Height=30
Area: 1500.00, Perimeter: 160.00

Square: X=300, Y=300, Width=50
Area: 2500.00, Perimeter: 200.00

Total number of shapes: 3
Total Area sum: 8712.39
Total Perimeter sum: 611.33

הערות:

- עבור על צורה יופיעו 2 שורות בקובץ הפלט: השורה הראשונה היא תוצר של מתודת ה- getDetails של הצורה, והשורה השניה מציגה את ערכי השטח וההיקף כפי שהתקבלו מהפעלת המתודות המתאימות בכל אחד מהצורות.
- ייתכן והמעריך אותו תקבל המתודה יהיה גדול יותר ממספר הצורות אותן הוא מכיל (כלומר החל ממקום מסויים במערך יתכן וערך התאים הוא Null). ניתן להניח שלפחות במקום הראשון המערך יש צורה אחת שאינה null.
- יש להדפיס את כל השברים עם רמת דיוק של 2 ספרות לאחר הנקודה.

4. להלן מנשק נוסף בשם RectangleBoundable המייצג צורה שניתן לחסום במלבן:

```
public interface RectangleBoundable {

    // Returns a Rectangle object which represents the bounding rectangle of the shape
    public Rectangle getBoundingRectangle();

    // Updates the position of the shape such that the upper left corner of its
    // bounding rectangle would be located at (x,y)
    public void setUpperLeftPoint(int x, int y);

}
```

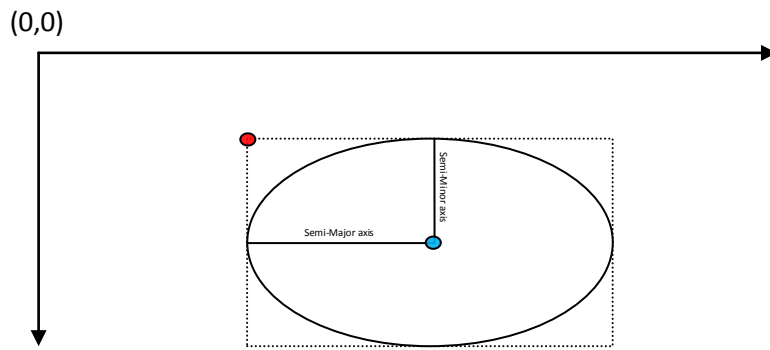
- המתודה getBoundingRectangle תחזיר אובייקט חדש מסוג מלבן, המייצג את המלבן החוסם של הצורה הנוכחית.
- המתודה setUpperLeftPoint תעדכן את ערכי המיקום של הצורה הנוכחית, כך שהפינה השמאלית העליונה של המלבן החוסם שלה תוצב בנקודה (x,y).

עדכנו את המנשק Shape כך שירחיב את המנשק RectangleBoundable. הכותרת של המנשק Shape לאחר השינוי אמורה להיות:

```
public interface Shape extends RectangleBoundable
```

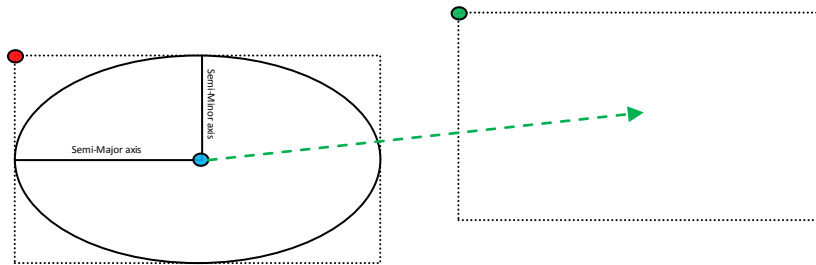
עתה יכלול המנשק Shape גם את המתודות שהוגדרו במנשק RectangleBoundable. לכן, כל מחלקה המצהירה שהיא מממשת את המנשק Shape, תצטרך לממש גם את המתודות של RectangleBoundable.

עדכנו את שלושת המחלקות **Ellipse, Rectangle ו-Square** כך שיכללו מימוש למתודות של המנשק **RectangleBoundable**.



דוגמאות:

את האליפסה שבאיור לעיל, נוכל לייצג כאובייקט מסוג `Ellipse` אשר מכיל את ערכי x ו- y המייצגים את מרכז האליפסה (הנקודה הכחולה) ואת שני הרדיוסים שלה. הפעלת המתודה `getBoundingRectangle` על האובייקט שמייצג את האליפסה באיור, תחזיר אובייקט חדש של מלבן אשר ערכי x ו- y שלו ייצגו את הנקודה האדומה באיור, שהיא הפינה **השמאלית** העליונה של המלבן החוסם את האליפסה.



אם נפעיל על האובייקט שמייצג את האליפסה שבאיור את המתודה `setUpperLeftPoint(x,y)` עם פרמטרים המייצגים את מיקום הנקודה הירוקה, האליפסה תוזז כך שהפינה **השמאלית** העליונה של המלבן החוסם שלה תשב על הנקודה הירוקה.

5. הוסיפו את מימוש המתודה `arrangeShapesHorizontally` למחלקה `ShapeDimensionCalculator`.

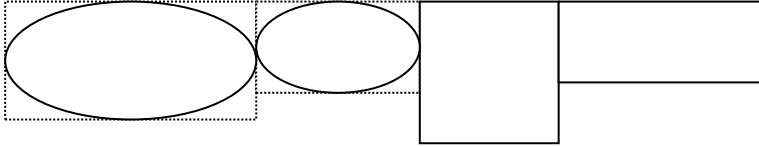
חתימת המתודה:

`public static void arrangeShapesHorizontally(RectangleBoundable[] shapes)`

המתודה תסדר את הצורות הכלולות במערך בשורה אופקית כאשר המלבנים החוסמים של הצורות צמודים זה לזה, וממוקמים בגובה של הנקודה השמאלית העליונה של המלבן החוסם של הצורה הראשונה במערך.
פירוט:

- הצורה הראשונה במערך לא תוזז.
- עבור כל צורה במערך שאינה null, החל מהצורה השניה:
 - שיעור ה- y של המלבן החוסם את הצורה יעודכן לשיעור ה- y של המלבן החוסם את הצורה הראשונה במערך.
 - שיעור ה- x של המלבן החוסם את הצורה יעודכן לשיעור ה- x של הצלע הימנית של המלבן החוסם את הצורה שמופיעה לפני הצורה הנוכחית במערך.

דוגמא לסידור אופקי של אליפסה, אליפסה, ריבוע ומלבן:



במידה והמערך מכיל פחות מ-2 צורות, המערך לא ישונה כלל.

אם נפעיל את המתודה `arrangeShapesHorizontally` על מערך הצורות שקיבלנו בסעיף 2, ולאחר מכן נשתמש במתודה `writeShapesToFile` שכתבתם בסעיף 3, נקבל קובץ פלט המכיל את הטקסט הבא (ראו קוד בפונקציה ה-`main` וקבצי הדגמה מצורפים):

Shape Dimension Calculator

Ellipse: X=100, Y=100, SemiMajorAxis=50, SemiMinorAxis=30
Area: 4712.39, Perimeter: 251.33

Rectangle: X=150, Y=70, Width=50, Height=30
Area: 1500.00, Perimeter: 160.00

Square: X=200, Y=70, Width=50
Area: 2500.00, Perimeter: 200.00

Total number of shapes: 3
Total Area sum: 8712.39
Total Perimeter sum: 611.33

שימו לב: בחלק זה יש להגיש את הקבצים הבאים (ניתן להוסיף קבצי עזר בהם השתמשתם):

ShapeDimensionCalculator.java, Ellipse.java, Rectangle.java, Square.java

חלק ב' (50 נק') – כתובת IP

המנשק IPAddress המופיע למטה מייצג כתובת של Internet Protocol (IP). דוגמאות לכתובות IP הן:

127.0.0.1

192.168.1.10

כתובת IP, כפי שניתן לראות, מורכבת מארבעה חלקים. ערכו של כל אחד מהחלקים הוא מספר שלם בין 0 ל-255. בסעיף זה נממש את המנשק IPAddress על ידי שלושה ייצוגים שונים: הראשון עושה שימוש במחרוזת, השני במערך של מספרים מסוג short ואילו השלישי משתמש ב-int יחיד (הסבר מפורט בהמשך).

א. כתבו **שלוש** מחלקות שונות המממשות את המנשק IPAddress המוגדר למטה:

1. מחלקה בשם IPAddressString, המממשת את המנשק בעזרת ייצוג פנימי של String.
2. מחלקה בשם IPAddressShort, המממשת את המנשק בעזרת ייצוג פנימי של מערך בגודל 4 של short. כל תא במערך יחזיק מספר בתחום 0..255.
3. מחלקה בשם IPAddressInt, ה מממשת את המנשק בעזרת int יחיד.

לכל אחת מהמחלקות יהיה בנאי המתאים לייצוג הפנימי שלה וכמובן כל אחת מהן מממשת את המנשק. ניתן להניח בחוזה שהבנאים מקבלים קלט תקין ליצירת כתובת IP (בהתאם לייצוג הפנימי של כל מחלקה).

```
public interface IPAddress {

    /**
     * Returns a string representation of the IP address, e.g.
     * "192.168.0.1"
     */
    public String toString();

    /**
     * Compares this IPAddress to the specified object
     * @param other
     *         the IPAddress to compare the current against
     * @return true if both IPAddress objects represent the same
     *         IP address, false otherwise.
     */
    public boolean equals(IPAddress other);

    /**
     * Returns one of the four parts of the IP address. The parts
     * are indexed from left to right. For example, in the IP
     * address 192.168.0.1 part 0 is 192, part 1 is 168,
     * part 2 is 0 and part 3 is 1.
     * (Each part is called an octet as its representation
     * requires 8 bits.)
     * @param index
     *         The index of the IP address part (0, 1, 2 or 3)
     * @return The value of the specified part.
     */
    public int getOctet(int index);
}
```



```

/**
 * There are four classes of private networks
 * (http://en.wikipedia.org/wiki/IPv4#Private_networks)
 * 10.0.0.0 - 10.255.255.255
 * 172.16.0.0 - 172.31.255.255
 * 192.168.0.0 - 192.168.255.255
 * 169.254.0.0 - 169.254.255.255
 *
 * This query returns true if this object is a private network
 * address
 */
public boolean isPrivateNetwork();
}

```

להלן פירוט לגבי אופן מימוש הייצוגים השונים:

1. מחוזות – יש להשתמש במחוזות יחידה לצורך ייצוג כתובת ה IP . כל הפעולות יבוצעו בעזרת מחוזות ז.
2. מערך – כל אחד מחלקי הכתובת (מספר שלם 0-255) יוחזק בתא במערך.
3. int – נשים לב שכל אחד מחלקי כתובת ה-IP הוא מספר שלם בתחום 0-255 (כולל), לפיכך ניתן לייצג אותו בעזרת 8 ביטים. על כן, את ארבעת חלקי הכתובת ניתן לייצג באמצעות 32 ביטים (4 בתים) וזהו בדיוק גודלו של int.

לפיכך, נשתמש ב-int לא כמספר, אלא כרצף בינארי של 32 ביטים. לדוגמא, הכתובת 127.0.0.1 תיוצג ע"י רצף הביטים 01111111000000000000000000000001.

החלק הראשון ייוצג ע"י הביטים במקומות 0-7 (משמאל לימין), החלק השני ע"י הביטים 8-15, השלישי ע"י 16-23 והרביעי ע"י ביטים 24-31.

```

127 <- 01111111 (ביטים 0-7 משמאל לימין)
0 <- 00000000 (ביטים 8-15)
0 <- 00000000 (ביטים 16-23)
1 <- 00000001 (ביטים 24-31)

```

הנחיה: על מנת להשתמש בייצוג זה, עליכם לדעת איך לחלץ את ערכו של כל בית (8 ביט) מהרצף הבינארי באורך 4 הבתים (32 ביטים) שמרכיב את ה-int. נציע 2 דרכים אפשריות:

1. שימוש באופרטורים על ביטים (<<, >>, ~, &, |) להזזה או למיסוך של הביטים ברצף הבינארי כך שיאופסו כל הביטים מלבד אלו השייכים לבית הרצוי.

2. שימוש במחלקה [ByteBuffer](#)

- צרו אובייקט בגודל 4 בתים ע"י `ByteBuffer.allocate(4)`
 - היעזרו במתודות `put(i)/get(i)` להצבה ולחילוץ של בית במיקום i.
 - שימו לב שהמתודה `get(i)` תחזיר את הבית באינדקס i באובייקט ה-`ByteBuffer` שיצרתם כמשתנה מטיפוס `byte` עם טווח ערכים של `(-128..+127)`.
- כדי לבצע המרה של בית b מטיפוס `byte` למשתנה מטיפוס `int` עם טווח הערכים 0..255 לו אנו זקוקים, ניתן להשתמש בטריק הבא:
- ```
int i = (int) (b & 0xFF);
```

**הערה חשובה:** עליכם לממש את המתודות באופן שונה בכל מחלקה בהתאם לייצוג הפנימי. אין להמיר את הייצוג הפנימי לייצוג אחר לצורך מימוש פעולה (רק לצורך פלט). המחלקות השונות לא "ייעזרו" זו בזו (למשל, אסור להשתמש ב- `IPAddressString` על מנת לממש את `IPAddressInt`).

בנוסף, ממשו את המחלקה `IPAddressFactory` המגדירה את המתודות הבאות:

---

```
public class IPAddressFactory {
 public static IPAddress createAddress(String ip) {
 ...
 }

 public static IPAddress createAddress(short[] ip) {
 ...
 }

 public static IPAddress createAddress(int ip) {
 ...
 }
}
```

---

כל אחת מהמתודות הסטטיות יוצרת אובייקט מטיפוס `IPAddress`, כשהאובייקט הקונקרטי נקבע על סמך טיפוס הקלט.

**הערה:** מחלקה שתפקידה היחיד הוא יצור אובייקטים של מחלקות אחרות נקראת *factory class*. מחלקות אלו מסתירות את פרטי יצור האובייקטים מלקוחות של אובייקטים אלו. השימוש בטכניקה זו נועד להסתיר את המחלקות הקונקרטיות שמממשות ממשק.

להלן תכנית המדגימה את השימוש במחלקה `IPAddressFactory` ובממשק.

---

```
public class TestIPAddress {
 public static void main(String[] args) {
 int address1 = -1062731775; // 192.168.0.1
 short[] address2 = { 10, 1, 255, 1 }; // 10.1.255.1

 IPAddress ip1 = IPAddressFactory.createAddress(address1);
 IPAddress ip2 = IPAddressFactory.createAddress(address2);
 IPAddress ip3 = IPAddressFactory.createAddress("127.0.0.1");

 for (int i = 0; i < 4; i++) {
 System.out.println(ip1.getOctet(i));
 }

 System.out.println("equals: " + ip1.equals(ip2));
 }
}
```

---