תוכנה 1

תרגיל מספר 10

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 ומספר התרגיל (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer_hw10.zip) קובץ ה zip יכיל: (א) את קבצי ה java של התוכניות אותם התבקשתם לממש (ב) קובץ טקסט בשם answers עם התשובות לשאלות: תרשים לחלק 1 ותשובות לסעיפים ג', ו-ד' בחלק 3.

חלק 1: Expressions

ברצוננו לתאר בתוכנה מגוון של ביטויים חשבוניים. ביטוי חשבוני הוא ישות הניתנת לשערוך כגון מספר או פעולה חשבונית המופעלת על שניים או שלושה ביטויים חשבוניים (הגדרה רקורסיבית). כל ביטוי חשבוני יודע להדפיס את עצמו.

עליכם להגדיר את הטיפוסים הבאים ולממשם כמחלקות קונקרטיות, מחלקות מופשטות או מנשקים. בנוסף הציגו תרשים מחלקות המתאר את היחס בין הטיפוסים שהגדרתם (מכונה גם עץ הורשה או היררכית מחלקות) בקובץ answers.

- **Expression** – טיפוס המייצג ביטוי כלשהו.
- **Literal** – טיפוס המתאר מספר ממשי (double).
- **BinaryExpression** – טיפוס המתאר פעולה בינארית (פעולה על שני ביטויים).
- **TrenaryExpression** – טיפוס המתאר פעולה טרינרית (פעולה על שלושה ביטויים).
- **Sum** – טיפוס המתאר את פעולת סכום של שני ביטויים.
- **Product** - טיפוס המתאר את פעולת המכפלה של שני ביטויים.
- **Exponent** - טיפוס המתאר את פעולת החזקה של שני ביטויים.
- **Conditional** – טיפוס המתאר פעולה על שלושה פרמטרים x, y, z כך שאם ערכו של x שונה מ 0 יוחזר ערכו של y אחרת יוחזר ערכו של z.

בידקו את עצמכם ע"י קוד הלקוח הבא:

```
public class ExpClient {

      public static void main(String[] args) {
            Expression l1 = new Literal(1.0);
            Expression l2 = new Literal(2.0);
            Expression l3 = new Literal(3.0);

            Expression sum = new Sum(l1, l2);
            Expression e1 = new Exponent(l3, sum);
            Expression prod = new Product(l1, l2);
            Expression exp = new Exponent(l2, l3);
            Expression e2= new Conditional(sum, prod, exp);

            System.out.println(e1 + " = " + e1.eval());
            System.out.println(e2 + " = " + e2.eval());
      }

}
```

פלט התוכנית הוא:

```
(3.0 ^ (1.0 + 2.0)) = 27.0
((1.0 + 2.0) ? (1.0 * 2.0) : (2.0 ^ 3.0)) = 2.0
```

הערות:

- כל אחד משמונת הטיפוסים לעיל יגדיר את המתודה: `public double eval();`
- כל הפעולות שהוגדרו פועלות על ביטויים (Expression). שערוך של ביטוי מתבצע ע"י הפונקציה eval.
- ביטויים מורכבים ישוערכו ע"י הפעלה רקורסיבית של eval על הארגומנטים.
- תשובתכם צריכה לבטא **שימוש חוזר** ברכיבי תוכנה **ולמזער את שכפול הקוד**. הדבר יעשה, בין השאר, ע"י בחירה נכונה של מחלקות קונקרטיות, מחלקות מופשטות ומנשקים. קוד עובד הוא תנאי הכרחי אך לא מספיק במקרה זה.
- שימו לב למתודה toString() ולהדפסות הסוגריים. תזכורת: כאשר אופרטור ה- '+' ופונקציית ההדפסה מוצאים עצם שאינו String במקום שבו אמור היה להימצא String, מופעלת המתודה toString() של אותו העצם. במחלקה Object קיים מימוש ברירת מחדל של מתודה זו.

חלק 2: Sorted Set Iterator

בחלק זה הנם מתבקשים לממש מחלקות שונות המגדירות פעולות בינאריות על קבוצות, כדוגמת פעולות האיחוד, חיתוך, הפרש סימטרי וכדומה. לשם כך הגדרנו שני מנשקים - SortedSet ו- Iterator –המגדירים קבוצות מסודרות ואיטרטור על הקבוצות. שימו לב, אלו **אינם** המנשקים הנושאים שם זהה הקיימים ב JDK. ראו את התיעוד המלא של המנשקים בקוד המופיע באתר.

```
package il.ac.tau.cs.sw1.sortedset;

/**
 * A Set that provides a total ordering on its elements.
 * All elements must implement the Comparable interface.
 */
public interface SortedSet<T extends Comparable<T>> {

	/**
	 * Return true if this set contains the specified element.
	 */
	public boolean contains(T element);

	/**
	 * Return the maximal (highest) element in this set.
	 */
	public T getMaximum();

	/**
	 * Return the minimal (lowest) element in this set.
	 */
	public T getMinimum();

	/**
	 * Return true if this set contains no elements.
	 */
	public boolean isEmpty();

	/**
	 * Return an iterator over the elements of this set.
	 * The iterator traverses the elements in an ascending
	 * order.
	 */
	public Iterator<T> iterator();

	/**
	 * Return the number of elements in this set (its
	 * cardinality).
	 */
	public int size();

}
```

```java
package il.ac.tau.cs.sw1.sortedset;

/**
 * An iterator over a collection of elements.
 */
public interface Iterator<T> {
	/**
	 * Advances the iterator to the next element.
	 */
	void advance();

	/**
	 * Returns true if the iterator has reached the end of
	 * the collections.
	 */
	boolean atEnd();

	/**
	 * Returns the current element in the iteration.
	 */
	T get();
}
```

האיטרטור הנתון פועל באופן מעט שונה מזה המוכר לנו ב Java, במקום לשאול האם קיים עוד איבר באוסף נברר האם הגענו לסוף. כמו כן, הפונקציה advance המקדמת את האיטרטור **אינה** מחזירה ערך. כדי לקבל את האיבר עליו מצביע האיטרטור נשתמש בפונקציה get, שימוש בפונקציה זו אינו מקדם את האיטרטור. נתון קטע קוד קצר המדגים שימוש באיטרטור החדש.

```java
for (Iterator<T> iter = ...; !iter.atEnd(); iter.advance()) {
    T element = iter.get();
}
```

<u>פעולות על קבוצות:</u>

הינכם נדרשים לממש את ארבע הפעולות הבינאריות של איחוד, חיתוך הפרש והפרש סימטרי כפי שיוגדרו להלן. תוצאת כל הפעולות היא בעצמה קבוצה ממוינת.

- **איחוד** – בהינתן שתי קבוצות A ו-B קבוצת האיחוד היא

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

- **חיתוך** - בהינתן שתי קבוצות A ו-B קבוצת החיתוך היא

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

- **הפרש**- בהינתן שתי קבוצות A ו-B ההפרש של A מ-B הוא קבוצת כל האברים שנמצאים ב-A ולא נמצאים ב-B

$$A \backslash B = \{x | x \in A \wedge x \notin B\}$$

- **הפרש סימטרי**- בהינתן שתי קבוצות A ו-B ההפרש הסימטרי הוא האיחוד של קבוצות ההפרשים (A מ-B ו-B מ-A)

$$A \ominus B = (A \backslash B) \cup (B \backslash A)$$

<u>דרישות המימוש:</u>

המחלקות השונות שהנכם נדרשים לממש ייצגו את תוצאת הפעולות השונות על קבוצות הקלט. קבוצת התוצאה, כמו גם קבוצות הקלט עצמן אינן ניתנות לשינוי לאחר שנוצרו (immutable).

לא ניתן לאחסן את הערך null בקבוצות הנ"ל. ניסיון להוסיף ערך זה לקבוצה (בעת יצירתה) יגרום לזריקת חריג מטיפוס IllegalArgumentException.

את המחלקות הינכם מתבקשים לממש בשני אופנים שונים, גישה "עצלה" (lazy) וגישה "להוטה" (eager).

גישה "להוטה''' – eager:
בגישה זו, בעת יצירת אובייקט של אחת המחלקות המממשות פעולה בינארית על קבוצות (איחוד, חיתוך הפרש או הפרש סימטרי) נחשב את תוצאת הפעולה באופן מיידי ונשמור אותה. מכאן ואילך כל השירותים המוגדרים במנשק יפעלו על אובייקט התוצאה (עליכם לבחור כיצד לייצג אותו).

גישה ב' – lazy:
בגישה זו לא נבצע חישוב בטרם נדרש לו, למשל לא נחשב את איבר הבא בתוצאה בטרם נצרך לקדם את האיטרטור, ובכל מקרה בשום שלב לא נחשב את תוצאת הפעולה בכללותה. בעת יצירת האובייקט נשמור הפניות לשתי קבוצות הקלט (מובטח לנו שהן לא ישתנו בהמשך בזכות ה-immutability) אך לא נבצע כל חישוב. בכל קריאה לשירות נחשב בעזרת האיטרטורים של קבוצות הקלט את המינימום הנדרש לשירות.
עבור השירותים size, getMinimum ו-getMaximum נחשב את תוצאתם בפעם הראשונה שנדרש לכך ונשמור תוצאה זו כדי שלא נאלץ לחזור על החישוב בפעמים הבאות שנדרש לכך (memorization). מכיוון שהקבוצות אינן ניתנות לשינוי מובטח לנו שתוצאת החישוב לא תשתנה. בכל אחת מהמחלקות נממש איטרטור חכם שכאשר נקדם אותו הוא ייתן את האשליה שהוא מתקדם על קבוצת התוצאה ולא על קבוצות הקלט. את האיטרטורים יש לממש כמחלקות פנימיות של מחלקות הקבוצה השונות. שימו לב שאם יש שכפול קוד בין האיטרטורים עליכם להעבירו למחלקה משותפת.

הינכם נדרשים לממש את המחלקות הבאות (ראו שלד פתרון באתר):

א. **UnionSortedSet** – מחלקה זו מייצגת את תוצר פעולת האיחוד בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים באיחוד של A ו-B.

ב. **IntersectionSortedSet** – מחלקה זו מייצגת את תוצר פעולת החיתוך בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים בחיתוך של A ו-B.

ג. **DifferenceSortedSet**– מחלקה זו מייצגת את תוצר פעולת ההפרד בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים בהפרש של A ו-B (A\B).

ד. **SymetricDifferenceSortedSet** – מחלקה זו מייצגת את תוצר פעולת ההפרש הסימטרי בין שתי קבוצות. המחלקה נבנית באמצעות שתי קבוצות ממוינות A ו-B, ומאפשרת גישה רק לעצמים שנמצאים בהפרש הסימטרי של A ו-B

בנוסף למחלקות הקונקרטיות הללו הינכם נדרשים לממש את המחלקה האבסטרקטית BinaryOpSortedSet. מחלקה מופשטת זו מייצגת קבוצה ממוינת שהיא תוצר של פעולה בינארית על קבוצות ממוינות. יש לממש במחלקה זו פונקציונאליות רבה ככל שניתן (כל הפונקציות שלא קשורות במימוש הפעולה הבינארית הספציפית). השירותים אותם הינכם נדרשים לממש **באופן מלא** במחלקה זו הינם:

- `final public int size();`
- `final public boolean isEmpty();`
- `final public T getMinimum();`
- `final public T getMaximum();`

ייתכנו כמובן שירותים ושדות נוספים במחלקה זו. בנוסף, במידה ותתקלו בצורך לשתף קוד בין מחלקות האיטרטורים השונות המקום הטבעי הוא במחלקה פנימית אבסטרקטית במחלקה BinaryOpSortedSet.
כפי שצוין לעיל הנכם נדרשים לספק שני מימושים שונים למחלקות אלו. מימושים אלו יהיו בלתי תלויים זה בזה **ואין לשתף בינהם קוד**, גם לא דרך מחלקות אבסטרקטיות משותפות.

<u>הערות:</u>

את הפרוייקט sortedset תמצאו ב-resources.zip באתר. ייבאו אותו לאקליפס (File->Import…).

בקבצים שסיפקנו לכם תמצאו את המחלקה SimpleSortedSet המממשת את המנשק SortedSet. מחלקה זו תשמש לכם כדוגמה בלבד ולצרכי בדיקה של הקוד, אין לעשות בה שימוש במימוש שלכם. כלומר כדי לבדוק שהפעולות הבינאריות שמימשתם נכונות תוכלו להשתמש במחלקה זו, אך אין לעשות בה שימוש במימוש הפנימי של המחלקות השונות.

יש להגיש את קבצי ה-Java כפי שצורפו לפרוייקט: שתי ספריות lazy ו-eager שמכילות את קבצי הקוד

חלק 3 – ComparablePoint

ברצוננו להוסיף למחלקות קיימות את האפשרות להשוות בין שני עצמים מאותה המחלקה. על ההשוואה לתמוך בששת היחסים הבאים: $\geq, \leq, \neq, =, <, >$.
השוואה בין שני עצמים מאותו טיפוס תעשה ע"י המנשק MyComparable הנתון להלן:

```
public interface MyComparable {
      boolean lessThanEqual (MyComparable other);
      boolean lessThan (MyComparable other);
      boolean greaterThanEqual (MyComparable other);
      boolean greaterThan (MyComparable other);
      boolean equal (MyComparable other);
      boolean notEqual (MyComparable other);
}
```

מתכנתת הגדירה מחלקה, MyComparablePoint, המייצגת נקודות במישור שניתן להשוות ביניהן. קריטריון ההשוואה הוגדר להיות שיעור ה- x של הנקודות, כלומר נקודה היא קטנה מנקודה אחרת עם ערך ה-x שלה קטן מערך ה-x של הנקודה האחרת. במקרה שלשתי הנקודות שיעור x זהה ההשוואה תעשה לפי שיעור ה- y. להלן הקוד המלא של המחלקה הקונקרטית MyComparablePoint:

```
public class MyComparablePoint extends AbstComparable {
    public MyComparablePoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public boolean lessThanEqual(MyComparable other) {
        MyComparablePoint otherPoint = (MyComparablePoint)other;
        return (x != otherPoint.x) ? x < otherPoint.x
                                   : y <= otherPoint.y;
    }

    public void translate(int dx, int dy) {
        x += dx;
        y += dy;
    }

    protected int x, y;
}
```

א. כדי למנוע את שכפול הקוד הגדירה אותה מתכנתת מחלקה מופשטת (abstract class), **AbstComparable**, המממשת את המנשק MyComparable ומפשטת את הגדרת המחלקות הקונקרטיות.
ממשו את המחלקה **AbstComparable** המופיעה בקוד למטה. על המחלקה להיות כללית מספיק כך שתשמש בסיס לכתיבת מגוון רחב של מחלקות (ולא רק למחלקה **MyComparablePoint**).
הימנעו ככל הניתן משכפול קוד:

```
public abstract class AbstComparable implements MyComparable {

    public abstract boolean lessThanEqual (MyComparable other);

    public boolean lessThan (MyComparable other) {...}
    public boolean greaterThanEqual (MyComparable other) {...}
    public boolean greaterThan (MyComparable other) {...}
    public boolean equal (MyComparable other) {...}
    public boolean notEqual (MyComparable other) {...}
}
```

ב. נרצה להגדיר את המחלקה **MyNormComparablePoint**. המחלקה תהיה זהה למחלקה **MyComparablePoint** מהסעיף הקודם פרט לקריטריון ההשוואה בין הנקודות. נקודה מטיפוס **MyNormComparablePoint** מוגדרת להיות קטנה מנקודה אחרת (מאותו טיפוס) אם הנורמה שלה קטנה מהנורמה של הנקודה האחרת. הנורמה של נקודה (x, y) מוגדרת להיות $\sqrt{x^2 + y^2}$.
ממשו את המחלקה MyNormComparablePoint בעזרת ירושה**.** שימו לב – על המחלקה לתמוך בכל השירותים שאותם סיפקה MyComparablePoint.

ג. למימוש המחלקה MyNormComparablePoint בעזרת ירושה חסרונות ויתרונות. מהו החסרון המרכזי של מימוש זה ומהו היתרון המרכזי שלו? **השלימו את התשובה בקובץ doc (או txt) וצרפו להגשה**

ד. הציעו מימוש חלופי למחלקות MyComparablePoint ו- MyNormComparablePoint שיפתור את הבעייתיות של שיוצרת הירושה במקרה זה. אין צורך לספק את קוד המחלקות אלא רק לתאר את העיצוב החלופי. יש לספק תאור מילולי וכן תרשים מחלקות (מלבנים וחיצים) לעיצוב החדש. אין צורך לציין בתרשים פרטים שאינם מהותיים לעיצוב החדש. **השלימו את התשובה בקובץ doc (או txt) וצרפו להגשה**

<u>הערות:</u>

את הפרויקט mycomperable תמצאו ב-resources.zip באתר. ייבאו אותו לאקליפס (File->Import…).