30.04.2006

תכנות מתקדם בשפת Java
בית הספר למדעי המחשב
אוניברסיטת תל אביב

תרגיל בית מספר 4

שאלה 1

בשאלה זו עליך לתת Design (Prototypes) בלבד.

מנהל המרינה בעיר הנמל הציורית גוצ'ק ביקש ממך לממש אפליקציה לניהול כלי השייט העוגנים במרינה.
כלי השייט נחלקים ל:

1) סירות מרוץ
2) סירות משוטים
3) יאכטות

עבור כל כלי שייט יש לשמור את הפרטים הבאים:

א) שם כלי השייט
ב) אורך
ג) רוחב
ד) משקל
ה) שנת יצור
ו) שם החברה שיצרה את כלי השייט
ז) החומר ממנו בנוי כלי השייט – יכול להיות: עץ, ברזל, פיברגלס או בטון

עבור סירות מרוץ יש לשמור גם מהירות מקסימלית. עבור סירות משוטים יש לשמור גם את מספר המשוטים. עבור יאכטות יש לשמור גם את מחירן.
יאכטות נחלקות לשני סוגים: יאכטות מנועיות ויאכטות מפרש. עבור יאכטות מנועיות יש לשמור גם את עוצמת המנוע (בכוחות סוס), ועבור יאכטות מפרש יש לשמור את מספר התרנים, גובה תורן מקסימלי ושטח המפרשים הכולל.
כמו כן, קיימות במרינה מספר יאכטות משולבות (דהיינו יאכטות מנועיות עם מפרשים).
לכל כלי השייט, ממשק (לא בהכרח java interface) המאפשר לעדכן את פרטי כלי השייט.

כל כלי שייט צריך לעגון ברציף.
במרינה קיימים 100 רציפי עגינה קטנים (ממוספרים מ-1 עד 100), ו-26 רציפים גדולים (מסומנים ב-A עד Z). בכל רציף יכול לעגון לכל היותר כלי שייט אחד. ברציף קטן יכול לעגון כלי שייט שאורכו עד 20 מטר ובגדול עד 100 מטר. בחלק מהרציפים (בגלל רוח חזקה) אסור לעגון יאכטות עם מפרשים (יש לאפשר למנהל המרינה לעדכן נתון זה עבור כל רציף).

מערכת הניהול מאפשרת:

1) להוסיף ולמחוק כלי שייט מהרשימה. כאשר מוסיפים כלי שייט יש להקצות לו רציף מתאים לעגינה (או להודיע אם אין מקום מתאים).
2) לעדכן ולהדפיס פרטים של כלי שייט קיים.
3) להחליף את מקום העגינה בין שתי יאכטות (או להודיע אם אי אפשר).
4) להדפיס את רשימת כל כלי השייט, ממוינים לפי אורכם, ועבור כל כלי שייט לציין גם את רציף העגינה.
5) להדפיס את רשימת כל רציפי העגינה המלאים, ממוינים לפי מספרם (או סימנם – עבור רציפים גדולים), ועבור כל רציף לציין את שם כלי השייט שעוגן בו.
6) להדפיס את רשימת כל רציפי העגינה הפנויים (שוב, בצורה ממוינת).

עליכם לספק מסמכי Javadoc (softcopy בלבד – אין צורך להדפיס) שיתארו את המחלקות הממשיות, המחלקות המופשטות והמנשקים הדרושים למימוש התוכנית. כמו כן יש לצרף תרשים מחלקות המתאר את היחסים בין הישויות השונות (ראה http://www.classdraw.com/Help.htm) ודיון קצר לגבי הנימוקים ליחסים כפי שנבחרו.

<u>אין צורך לממש בקוד מפורש אף מתודה</u>.

שאלה 2

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 (כן – ההגדרה רקורסיבית). כל ביטוי חשבוני יודע להדפיס את עצמו.

עליך להגדיר את הישויות הבאות (מחלקות קונקרטיות, מחלקות מופשטות ומנשקים):

1. Expression – ישות המייצגת ביטוי כלשהו.
2. Literal – ישות המתארת מספר בודד (`double`).
3. BinaryOp – ישות המתארת פעולה בינארית (פעולה על שני ביטויים).
4. TrenaryOp – ישות המתארת פעולה טרינארית (פעולה על שלושה ביטויים).
5. Sum – ישות המתארת סכום.
6. Product - ישות המתארת מכפלה.
7. Exponent - ישות המתארת חזקה.
8. CondExp – ישות המתארת פעולה על שלושה פרמטרים a, b, c שהיא a ? b : c . שערוך הביטוי מתבצע כך: אם ערכו של a שונה מ 0.0 יוחזר ערכו של b אחרת יוחזר ערכו של c.

בידקו את עצמכם ע"י קוד הלקוח הבא:

```
public class Client
{
	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 CondExp(sum, prod, exp);

		System.out.println(e1 + " = " + e1.eval());  // using toString()
		System.out.println(e2 + " = " + e2.eval());  // using toString()
	}
}
```

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 אחת מ-8 הישויות לעיל תכיל את המתודה: **`public double eval()`**
- כל הפעולות לעיל פועלות על ביטויים (`Expression`)
- שערוך של ביטוי מתבצע ע"י הפונקציה `eval`
- ביטויים מורכבים ישוערכו ע"י הפעלה רקורסיבית של `eval` על הארגומנטים
- תשובתך 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.

הצג תרשים מחלקות המתאר את היחס בין הישויות שהגדרת (מכונה גם עץ הורשה או היררכית מחלקות).

<u>שאלה מספר 3</u>

בשאלה הבאה עליכם לכתוב קוד בגוף דף התרגיל (בכתב יד). אין צורך להגיש אותו Online. ניתן לפתור את השאלה פתרון מלא ע"י מילוי המלבנים הריקים שבקטעי הקוד.

ברצוננו להוסיף למחלקות קיימות את האפשרות להשוות בין שני עצמים מאותה המחלקה. על ההשוואה לתמוך ב-6 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** (בעמוד הבא) המייצגת נקודות במישור שניתן להשוות ביניהן. קריטריון ההשוואה הוגדר להיות שיעור ה- x של הנקודות ובמקרה שלשתי הנקודות שיעור x זהה ההשוואה תעשה לפי שיעור ה- y. כדי למנוע את שכפול הקוד הגדירה אותה מתכנתת מחלקה מופשטת (abstract class) **AbstComparable** המממשת את המנשק **MyComparable** ומפשטת את הגדרת המחלקות הקונקרטיות.

להלן הקוד המלא של המחלקה **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;

                if (x == otherPoint.x)
                        return y <= otherPoint.y;
                return x < otherPoint.x;
        }

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

        protected int x;
        protected int y;
}
```

א. ממשו את המחלקה **AbstComparable** המופיעה בקוד לעיל. 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 greaterThan (MyComparable other) {
        
    }

    public boolean lessThan (MyComparable other) {
        
    }

    public boolean equal (MyComparable other) {
        
    }

    public boolean greaterThanEqual (MyComparable other) {
        
    }

    public boolean notEqual (MyComparable other) {
        
    }
}
```

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 **<u>הנורמה שלה</u>** קטנה מהנורמה של הנקודה האחרת. (הנורמה של (x,y) מוגדרת להיות $\sqrt{x^2+y^2}$).

ב.

ממשו את המחלקה **MyNormComparablePoint** בעזרת ירושה. שימו לב – על המחלקה לתמוך <u>בכל</u> השרותים שאותם סיפקה **MyComparablePoint**.

```
                    MyNormComparablePoint
                                          {
    public MyNormComparablePoint (int x, int y) {

    }

}
```

למימוש המחלקה **MyNormComparablePoint** בעזרת ירושה חסרונות ויתרונות. מהו החסרון המרכזי של מימוש זה ומהו היתרון המרכזי שלו?

חסרון: _____

_____

_____

יתרון: _____

_____

_____

ג. הציעו מימוש חלופי למחלקות **MyComparablePoint** ו-**MyNormComparablePoint** שיפתור את הבעייתיות שיוצרת הירושה במקרה זה. אין צורך לספק את קוד המחלקות אלא רק לתאר את העיצוב החלופי. יש לספק תאור מילולי וכן תרשים מחלקות לעיצוב החדש (לכל מחלקה או מנשק - מלבן שמכיל רק את שם המחלקה, וחיצים בין המלבנים לייצוג קשרים ביניהן). אין צורך לציין בתרשים פרטים שאינם מהותיים לעיצוב החדש.

עיצוב חלופי:

_____
_____
_____
_____
_____
_____

תרשים מחלקות:

<u>מה להגיש ?</u>

1) קבצים: הקובץ `ex4.jar` יכיל את הקבצים הבאים:

`il.ac.tau.cs.advJava.ex4`
`src/`
- `Expression.java`
- `Literal.java`
- `BinaryOp.java`
- `TrenaryOp.java`
- `Sum.java`
- `Product.java`
- `Exponent.java`
- `CondExp.java`

`doc/`
- **`html files for q1 and q2`**

`bin/`

הקובץ ימוקם בתיקייה ~/advJava06b עם הרשאות גישה מתאימות כפי שמפורט במסמך הנחיות ההגשה. יש להקפיד על הפרדת הקבצים לחבילות כמפורט לעיל.

2) תדפיסים:

אין צורך להדפיס את קובצי ה- Javadoc בפתרון שאלה 1. יש להדפיס תרשים מחלקות שיתאר בעזרת מלבנים וחיצים את הקשרים בין הישויות השונות. יש לצרף לתרשים דיון קצר שיסביר מדוע נבחרו הקשרים כפי שנבחרו.

פתרון שאלה 2 צריך להכיל את תדפיסי כל הקבצים שהוגשו בקובץ ה `jar` לעיל וכן תרשים מחלקות (מלבנים וחיצים) המתאר את הקשרים בין הישויות השונות. אין צורך בתדפיסי Javadoc עבור שאלה זו.

פתרון שאלה 3 יוגש רק כ `hardcopy`. יש להדפיס את עמודים 6-8 ולמלא את תשובותיכם במקום המיועד לכך.

ליצירת תרשימי המחלקות העזרו במדריך http://www.classdraw.com/Help.htm המסכם את סוגי היחסים האפשריים בין מחלקות ומנשקים וכן בשקפי ההרצאות.