

# פתרון מבחן בתוכנה 1 והערות בדיקה

מועד א' סמסטר א' תשס"ו, 17 בפברואר 2006

שאלה מספר 1

```
public abstract class AbstComparable implements MyComparable {  
  
    public abstract boolean lessThanEqual (MyComparable other);  
  
    public boolean greaterThan (MyComparable other) {  
        return other.lessThanEqual(this);  
    }  
  
    public boolean lessThan (MyComparable other) {  
        return other.greaterThan(this);  
    }  
  
    public boolean equal (MyComparable other) {  
        return lessThanEqual(other) && other.lessThanEqual(this);  
    }  
  
    public boolean greaterThanEqual (MyComparable other) {  
        return greaterThan(other) || equal(other);  
    }  
  
    public boolean notEqual (MyComparable other) {  
        return !equal(other);  
    }  
}
```

•

ב. שני הפתרונות הבאים קיבלו ניקוד מלא:

```
public class MyNormComparablePoint
    extends MyComparablePoint {
    public MyNormComparablePoint (int x, int y) {
        super(x,y);
    }
    public boolean lessThanEqual (MyComparable other) {
        MyNormComparablePoint otherPoint =
            (MyNormComparablePoint)other ;
        return (x*x + y*y) <=
            otherPoint.x*otherPoint.x + otherPoint.y*otherPoint.y ;
    }
}
```

חטרון: יחס הירושה שגוי. `MyNormComparablePoint` אינו מקיים יחס `is-a`

המחלקה `MyComparablePoint`. בפרט החוזה של המתודה `lessThanEqual` אינו עומד בכללי עיצוב על פי חוזה

יתרון: חסכון בקוד.

גם הפתרון הבא קיבל ניקוד מלא (אם נומק כהלכה):

```
public class MyNormComparablePoint
    extends AbstComparable {

    public MyNormComparablePoint (int x, int y) {

        this.x = x;
        this.y = y;
    }

    public boolean lessThanEqual (MyComparable other) {
        MyNormComparablePoint otherPoint =
            (MyNormComparablePoint)other;

        return (x*x + y*y) <=
            (otherPoint.x*otherPoint.x + otherPoint.y*otherPoint.y);
    }

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

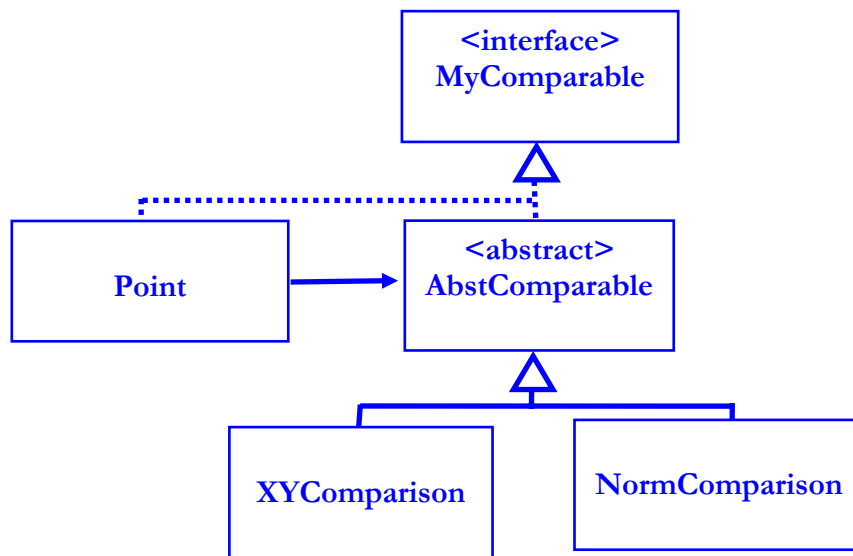
    protected int x;
    protected int y;
}
```

חסרון: שכפול קוד. המתודה translate והשדות x,y נכתבו פעמיים.  
ירושה לצורך מימוש – נקודה אינה הרחבה של תכונות ההשוואה.  
פגיעה במודולריות – מקשה על הרחבה עתידית של תכונות המחלקה

יתרון: הירושה עומדת בכללי עיצוב לפי חוזה. יחס הירושה רע פחות  
מהירושה מ- MyComparable.

ב. עיצוב חלופי:  
נחליף את הקשר בין נקודה והשוואות מירושה להאצלה.  
המחלקה נקודה תכיל הפנייה לעצם משווה (מטיפוס המחלקה המופשטת  
או המנשק ) ותממש את המנשק MyComparable בעזרת הפניית הבקשות  
לעצם המשווה.  
מכיוון שהמשווה צריך הכרות אינטימית עם הנקודה, ניתן לממש אותו  
כמחלקה פנימית (אולי אנונימית)

תרשים מחלקות:



## שאלה מספר 2

start()	@post current element is first	[text]	א.
forward()	@pre !atLast() @post move one element forward	[text]	
atLast()	@return true if current is at last element	[text]	
set()	@post get() == x	[text]	
get()	@return the value of the current element	[text]	

0 <= current < size && size = cont.length ב.

start()	current=0;		ג.
forward()	current++;		
atLast()	return (current==size-1);		
set()	cont[current]=x;		
get()	return cont[current];		

ד. לא תקין, כי תנאי הקדם של get() במחלקה חזק יותר מהתנאי במנשק

ה. תקין. תנאי הקדם של forward() במחלקה חלש מזה שבמנשק. יש לשנות את תנאי האחר כדי שיתייחס למקרה שלא כוסה.

@post \$pre atLast() ==> move to first element [ OR do not move ]  
 @post \$pre !atLast() ==> move one element forward

ו.

```
public class ExtendedFixedList extends SimpleFixedList {
```

```

public ExtendedFixedList(int n) {
    super(n);
}
@return true iff there is an element at position i
public boolean legalIndex(int i) {
    return (0 < i) && (i <= size);
}

@pre legalIndex(i)
@return element at position i
public int get(int i){
    return cont[i-1];
}
}

```

ההערה "חסרה שאילתא" פרושה שיש צורך להוסיף שאילתא כדי שתנאי הקדם של השרות `get(i)` יוכל להיבדק. עוד מרכיב חיוני: בנאי עם פרמטר, שקורא לבנאי של ה `super` (בנאי ברירת מחדל לא יעבוד).  
הערות כלליות לשאלה 2 הורדו נקודות על התייחסות לפקודות בתנאים שונים של חוזה, על התייחסות באינוריסטה לרכיבים שאינם רלבנטיים שם.

### שאלה מספר 3

.א.

1.

```
public int foo() {  
    return 1;  
}
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין  
method overloading

2

```
public int foo (float i) {  
    return i;  
}
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין  
Explicit casting is required from float to int.

3

```
public float foo(float i) {  
    return i;  
}
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין

method overloading

4.

```
public float foo(int i) {  
    return i;  
}
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין

method overloading is not allowed since the signature of the two methods is the same except for the returned value.

.ב.

1. a.foo();

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין

ידפיס "in B.foo" דריסה + שיגור דינאמי.

2. a.moo1();

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין

---

ידפיס "in A.soo" המתודה soo היא סטטית ולכן אין שיגור דינאמי.

3. a.moo2();

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין  
The method moo2() is undefined for the type A

4. b.foo();

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין  
ידפיס 'in B.foo' . דריסה.

5. b.moo1();

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין  
ידפיס 'in A.soo' . המתודה moo1 מוגדרת ב-A ולכן המתודה הסטטית soo של A תופעל.

6. b.moo2();

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין  
ידפיס 'in B.soo' . moo2 מוגדרת ב-B ולכן soo של B תופעל.

ג.

פיתרון אפשרי:

```
public class B extends A implements I3 {  
    public void bar1() {};  
    protected void bar4() {};  
}
```