

פתרון הבחינה בתוכנה 1

סמסטר א' ו-ב', מועד ב', תשס"ט
09/09/09

אוהד ברזילי, דן הלפרין, ליאור וולף,
נעמה מאיר, מתי שמרת, ליאור שפירא

הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות** - חלקו את זמנכם ביעילות.
- יש לענות על כל השאלות.
- בשאלות שבהן יש צורך לנמק, תשובה ללא נימוק לא תזכה באף נקודה.
- יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות ואולם במידת הצורך ניתן לכתוב בגב טופס הבחינה.
- יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תפסל. תשובות במחברת הבחינה לא תיבדקנה.
- יש למלא מספר סידורי (מס' מחברת) ומספר ת.ז על כל דף של טופס הבחינה.
- ניתן להניח לאורך השאלה שכל החבילות הדרושות יובאו, ואין צורך לכתוב שורות import.
- במקומות בהם תתבקשו לכתוב מתודה (שירות), ניתן לכתוב גם מתודות עזר.
- אסור השימוש בחומר עזר כלשהוא, כולל מחשבוני או כל מכשיר אחר פרט לעט.

לשימוש הבודקים:

שאלה	א	ב	ג	ד	סה"כ
1	10	4	16	10	40
2	15	15			30
3	4	4	4		12
4	3	15			18
	ציון בחינה: 100				

בהצלחה!

שאלה 1 (40 נקודות)

שאלה זו עוסקת בתכנון ובהגדרת טיפוס נתונים עבור פולינום דליל (sparse polynomial) שמקדמיו שלמים. בשאלה נדון בטיפוס שאינו מקובע (mutable).

- פולינום הוא ביטוי מהצורה: $c_0 + c_1x + c_2x^2 + c_3x^3 + \dots + c_nx^n$.
- הם מספרים שלמים, והם נקראים המקדמים של הפולינום (coefficients) c_0, c_1, \dots, c_n .
- המעריך הגדול ביותר בפולינום (של איבר שהמקדם שלו אינו אפס) נקרא הדרגה (degree) של הפולינום.

לדוגמא: הביטוי $2x^3 - 5x^2 + 13$ הוא פולינום שדרגתו 3 ומקדמיו הם: 2, -5, 0, 13.

- פולינום דליל הוא פולינום אשר רוב המקדמים שלו (c_i) הם 0.

לדוגמא: הביטוי $2x^{15} + x^4 + 13$ הוא פולינום דליל - למרות שדרגתו 15 רק שלושה מקדמים שלו שונים מ-0.

להלן הממשק והמצב המופשט של פולינום (לא בהכרח דליל):

```

/** פולינום שאינו מקובע (mutable) שמקדמיו שלמים
 * @abst:  $c_0 + c_1x + c_2x^2 + c_3x^3 + \dots + c_nx^n$ 
 */
public interface IPolynomial {

    /** השרות מחזיר את דרגת הפולינום
     * @abst: $ret = n
     */
    public int degree ( );

    /** d השרות מחזיר את המקדם של האיבר עם מעריך d
     * @abst: $ret =  $c_d$ 
     */
    public int coeff (int d);

    // more operations: add, mul, sub, minus

}

```

א. (10 נקודות) בסעיף הבא אתם מתבקשים לממש את הבנאים של המחלקה

SparsePolynom וכן את השרות `degree()`:

- אם לצורך המימוש יש צורך בשרותי עזר נוספים, שדות או טיפוסים חדשים ממשו גם אותם.
- על בחירת ייצוג הפולינום להביא בחשבון את דלילותו הן בצריכת הזיכרון של העצם והן בסיבוכיות זמן הריצה של פעולותיו. כלומר: על הסיבוכיות להיות פרופורציונית למספר המקדמים שאינם אפס ולא לדרגת הפולינום.
- עליכם לספק בנאים המאפשרים יצירה נוחה של עצם חדש בהינתן מקדמיו – לפחות אחד הבנאים יוכל לקבל מספר ארגומנטים הפרופורציוני למספר המקדמים של הפולינום שאינם 0.

```
public class SparsePolynom implements IPolynomial {

    private TreeMap<Integer, Integer> monoms = new TreeMap<Integer, Integer>();
    private int degree;

    /**
     * @pre:   coeffs are even number integers, where the elements in odd places
     *        represent coeffs (not equal zero) and the elements in even places
     *        corresponds to their degree. For Example:
     *        the polynomial: 2x^5 - x + 7 will be passed as {2,5,-1,1,7,0}
     */
    public SparsePolynom(int ... coeffs) {

        for (int i = 0; i < coeffs.length; i+=2) {
            this.monoms.put(coeffs[i+1], coeffs[i]);
        }

        for (int i = 1; i < coeffs.length; i+=2) {
            if (degree < coeffs[i])
                degree = coeffs[i];
        }
    }

    /**
     * @pre:   coeffs and degrees are same length. Every element in degrees
     *        represents a degree with non-zero coefficient, which its value is
     *        the corresponding coeffs value. For Example:
     *        the polynomial: 2x^5 - x + 7 will be passed as {2,-1,7} and {5,1,0}
     */
    public SparsePolynom(int[] coeffs, int[] degrees) {

        for (int i = 0; i < coeffs.length; i++) {
            this.monoms.put(degrees[i], coeffs[i]);
        }

        for (int i = 0; i < degrees.length; i++) {
            if (degree < degrees[i])
                degree = degrees[i];
        }
    }

    public int degree() {
        return degree;
    }
}
```

ייצוגי פולינום אפשריים נוספים (ויש גם אחרים):

- ייצוג פולינום כשני מערכים: האחד של מקדמים שאינם אפס והשני של דרגותיהם בהתאמה
- ייצוג פולינום כמערך אחד שבו דרגות ומקדמים לסרוגין
- הגדרת טיפוס עזר לייצוג איבר בפולינום (מקדם, דרגה) וייצוג פולינום כרשימה של טיפוס עזר אלו. במקרה זה היה צריך לספק גם בנאי אשר מקבל טיפוסים פשוטים (למשל int) כדי לאפשר יצירה נוחה של פולינומים, וגם כדי לשמור על הסתרת המידע מהלקוח

ב. (4 נקודות) מהי פונקצית ההפשטה של המימוש אותו בחרתם?

$$\mathbf{AF}(\text{this}) = c_0 + c_1x + c_2x^2 + c_3x^3 + \dots + c_nx^n$$

$$\text{s.t. } c_i = \begin{cases} \text{monoms.get}(i) & \text{if monoms.get}(i) \neq \text{null} \\ 0 & \text{if monoms.get}(i) = \text{null} \end{cases}$$

$$n = \text{this.degree}$$

ג. 1. (4 נקודות) אומה הסתומה מראה לביל הדביל איך היא מימשה את השרות coeff בשורה אחת בלבד. ביל מעיר לה כי למרות שהמימוש מאוד קצר, היא שכחה לציין שהוא מניח כמה הנחות על אופן הקריאה (תנאי קדם - preconditions). ממשו את השרות coeff וציינו את החוזה שלו:

להלן שני פתרונות אפשריים (יש גם אחרים)

```
/**
 * @pre      $ret != 0, "assumes we will only be asked about non-zero coeffs"
 *           d >= 0, "d is non-negative"
 *           d <= degree(), "d is less than max degree (with non-zero coefficient)"
 *
 * @post:    returns the coefficient of the element which its exponent is d
 */
public int coeff1a(int d) {
    return monoms.get(d);
}
```

```
/**
 * @pre:     d >= 0, "d is non-negative"
 *           d <= degree(), "d is less than max degree (with non-zero coefficient)"
 *
 * @post:    returns the coefficient of the element which its exponent is d
 */
public int coeff1b(int d) {
    return monoms.get(d) == null ? 0 : monoms.get(d);
}
```

שגיאות נפוצות:

- בדיקת תנאי קדם בגוף השרות
- שימוש במשתנים פרטיים בהגדרת תנאי הקדם והבתר

2. (4 נקודות) אומה אינה מרוצה מריבוי תנאי הקדם ומעדיפה מימוש ללא תנאי קדם כלל. ממשו את השרות coeff וציינו את החוזה שלו עבור מקרה זה:

```
/**
 * @pre: true (no precondition)
 *
 * @post: returns the coefficient of the element which its exponent is d
 *        d<0 || d>degree() $implies $ret==0
 *
 */
public int coeff2(int d) {
    if(d<0 || d>degree())
        return 0;
    return monoms.get(d) == null ? 0 : monoms.get(d);
}
```

3. (4 נקודות) ביל מזהיר את אומה, כי מתכנתים אשר לא מורגלים בקריאת postconditions עלולים להתבלבל מהערכים המוחזרים, ומציע לשנות את המימוש כך שיזרקו חריגים במקרי הקצה. ביל מציע להשתמש בחריג שאינו נבדק (unchecked exception) בעל שם משמעותי כדי שהלקוחות יקבלו חיווי איכותי לגבי השימוש השגוי שלהם בשרות. ממשו את השרות coeff וציינו את החוזה שלו עבור מקרה זה. אם יש צורך להגדיר חריגים חדשים ציינו גם אותם.

```
/**
 * @pre: true (no precondition)
 *
 * @post: returns the coefficient of the element which its exponent is d
 *
 * @throws DegreeOutOfRangeException when d<0 || d>degree()
 *
 */
public int coeff3(int d) {
    if(d<0 || d>degree())
        throw new DegreeOutOfRangeException();
    return monoms.get(d) == null ? 0 : monoms.get(d);
}

class DegreeOutOfRangeException extends RuntimeException {}
```

4. (4 נקודות) האם ניתן היה להשתמש בחריג נבדק (checked exception) בסעיף הקודם?
 - אם לא – מדוע?
 - אם כן – מדוע העדיף ביל חריג שאינו נבדק?

לא ניתן. זריקת checked exception מחייבת שינוי החתימה במנשק.

ד. (10 נקודות) ממשו את השרות toString של המחלקה SparsePolynom המחזיר ייצוג מחרוזת של הפולינום הנוכחי. לדוגמא: עבור הפולינום $2x^3 - 5x^2 + 13$ תוחזר המחרוזת: "2x^3-5x^2+13".

```
public class SparsePolynom implements IPolynomial {

    //...

    @Override
    public String toString() {
        String result = "";

        for (Map.Entry<Integer, Integer> monom : monoms.descendingMap().entrySet()) {
            if (monom.getKey() != degree() && monom.getValue() > 0)
                result += "+";
            if (monom.getValue() == -1)
                result += "-";
            else if (monom.getValue() != 1)
                result += monom.getValue();

            if (monom.getKey() == 1)
                result += "x";
            else if (monom.getKey() != 0)
                result += "x^" + monom.getKey();
        }
        return result;
    }
}
```

שאלה 2 (30 נקודות)

בשאלה זו נממש סרקן-אינטרנט-עצל (`LazyInternetCrawler`) אשר סורק את האינטרנט אתר אחר אתר וימומש כ-`Iterator`.

האינטרנט (`WWW`) היא רשת אתרים (גרף) אשר לכל אחד מהם כתובת ייחודית (`URL`), ואשר עשויים להכיל בתוכם קישורים (לינקים) לאתרים אחרים. בשאלה זו נתייחס לכתובת של אתר מחרוזת (`String`), לדוגמא: כתובת אתר הבית של הקורס תיוצג ע"י המחרוזת:
`"http://courses.cs.tau.ac.il/software1/0809a/index.html"`

- כשמאתחלים את הסרקן הוא מקבל מחרוזת שהיא כתובת חוקית של אתר כלשהו, שתשמש נקודת ההתחלה לסריקת האינטרנט. כתובת זו תאוחסן בתור של כתובות שאותן יש לסרוק.
- בזימון השרות `next` הסרקן מתקדם לאתר הבא בתור, מוסיף את כל הלינקים החדשים המופיעים באתר זה לתור הכתובות, ומחזיר את כתובת האתר. כלומר אם לינק המופיע באתר מסוים היה קיים כבר בתור לא מוסיפים אותו, אם הוא לא קיים (חדש) אז מוסיפים אותו.

לנוחיותכם, סיפקנו עבורכם את מחלקת השרות הבאה, ובה פונקציה אשר בהינתן כתובת `URL` מחזירה את כל הכתובות של הלינקים בעמוד:

```
public class InternetUtil {
    public static Set<String> linkedURLs(String url) {
        // returns a Set of all URLs that appear as links in url...
    }
}
```

א. (15 נקודות) ממשו את המחלקה `LazyInternetCrawler` כאיטרטור:

```
public class LazyInternetCrawler implements Iterator<String> {
    List<String> allurls = new ArrayList<String>();
    int currentPosition = 0;

    public LazyInternetCrawler(String url) {
        allurls.add(url);
    }

    protected void addURLlinks(String url) {
        Set<String> l = InternetUtils.linkedURLs(url);
        for (String a : l) {
            if (!this.allurls.contains(a))
                this.allurls.add(a);
        }
    }

    @Override
    public boolean hasNext() {
        return (currentPosition < allurls.size());
    }

    @Override
    public void remove() {}

    @Override
    public String next() {
        String url = allurls.get(currentPosition);
        currentPosition++;
        addURLlinks(url);
        return url;
    }
}
```


ב. (15 נקודות) Page Importance הוא מדד לחשיבות אתר אינטרנט. חשיבות של אתר מושפעת ממספר האתרים שמצביעים אליו (יש בהם לינק אליו) לפי החישוב הבא:

נסמן ב- $outLinks(x)$ את מספר הלינקים שמכיל אתר אינטרנט x (מספר האתרים שהוא מצביע אליהם). ונניח שעד שלב מסוים ראינו שלאחר אינטרנט מסוים A יש n מצביעים (יש לינקים מתוך) B_1, B_2, \dots, B_n . אזי:

$$PageImportance(A) = \frac{1}{outLinks(B_1)} + \frac{1}{outLinks(B_2)} + \dots + \frac{1}{outLinks(B_n)}$$

כלומר ככל שבאתר כלשהו יש לינקים ליותר אתרים, כך הוא תורם ל- Page Importance של כל אחד מהם פחות.

בסעיף זה נממש בעזרת `LazyInternetCrawler` מהסעיף הקודם סרקן משוכלל, `LazyPICrawler`, עם שרות נוסף בשם `pageImportance` אשר בהנתן URL מחזירה את ערך Page Importance של העמוד.

להלן דוגמת שימוש בסרקן המשוכלל:

```
LazyPICrawler crawler = new LazyPICrawler("http://www.ynet.co.il");
System.out.println("Before started crawling CS school website is ranked: " +
    crawler.pageImportance("http://www.cs.tau.ac.il/"));

for (int i = 0; i < 1000000 && crawler.hasNext(); i++) {
    crawler.next();
}

System.out.println("After crawling 1000000 websites CS school is ranked: " +
    crawler.pageImportance("http://www.cs.tau.ac.il/"));

String url = null;
for (int i = 0; i < 1000000 && crawler.hasNext(); i++) {
    url = crawler.next();
}

System.out.println("Currently crawling " + url + " which is ranked " +
    crawler.pageImportance(url));

System.out.println("CS school website is now ranked: " +
    crawler.pageImportance("http://www.cs.tau.ac.il/"));
```

שימו לב: חישוב ה Page Importance מושפע מהתקדמות הסריקה. יתכן והסרקן סרק את ה URL הנ"ל, או נתקל בו רק בתור לינק מתוך אתר אחר, או לא נתקל בו כלל. במקרה האחרון יש להחזיר 0.

אם לצורך מימוש הסרקן עליכם לערוך שינוי בתשובתכם לסעיף א' אנא ציינו מה השינוי, אבל אל לכם להוסיף פונקציונליות של הסרקן המשוכלל לתוך תשובת סעיף א'.

ממשו בעמוד הבא את המחלקה `LazyPICrawler`.

```
public class LazyPICrawler extends LazyInternetCrawler {

    private HashMap<String, Double> importanceValues = new HashMap<String, Double>();

    public LazyPICrawler(String url) {
        super(url);
    }

    protected void addURLlinks(String url) {
        super.addURLlinks(url);
        Set<String> l = InternetUtils.linkedURLS(url);
        for (String a : l) {
            double currentImportance =
                importanceValues.get(a) == null ? 0 : importanceValues.get(a);

            importanceValues.put(a, currentImportance + 1.0 / l.size());
        }
    }

    public double pageImportance (String url) {
        if (!importanceValues.containsKey(url))
            return 0;
        return importanceValues.get(url);
    }
}
```

שאלה 3 (12 נקודות)

לפניכם 2 קטעי קוד ואחריהם 3 שאלות המתייחסות אליהם:

```
public class A {
    public A() {
        foo();
    }

    private void foo() {
        System.out.print("A::foo ");
        goo();
    }

    public void goo() {
        System.out.print("A::goo ");
    }
}
```

```
public class B extends A {
    public B() {
        foo();
    }

    public void foo() {
        System.out.print("B::foo ");
    }

    public void goo() {
        System.out.print("B::goo ");
    }

    public static void main(String[] args) {

        *****
    }
}
```

א. מחליפים את שורת הכוכביות בקוד הבא:

```
A a = new B();
```

מה מודפס בהרצת main?

1. A::foo B::goo B::foo
2. B::foo B::goo B::foo
3. A::foo A::goo B::foo
4. A::foo B::goo A::foo

ב. מחליפים את שורת הכוכביות בקוד הבא:

```
A b = new B() {
    public void foo() {System.out.print("Anonymous::foo ");}
    public void goo() {(B) this}.foo();}
};
```

מה מודפס בהרצת main?

1. Anonymous::foo Anonymous::foo Anonymous::foo
2. A::foo Anonymous::foo Anonymous::foo
3. A::foo B::goo B::foo
4. Anonymous::foo B::goo B::foo

ג. מחליפים את שורת הכוכביות בקוד הבא:

```
A a = new A() {
    public void foo() {System.out.print("Anonymous::foo ");}
    public void goo() {
        System.out.println((this instanceof A ?
            "I am A " :
            "I am Anonymous "));
    }
};
```

מה מודפס בהרצת main?

1. A::foo I am Anonymous
2. Anonymous::foo I am Anonymous
3. Anonymous::foo I am A
4. A::foo I am A

שאלה 4 (18 נקודות)

לפניכם 2 קטעי קוד ואחריהם שאלות המתייחסות אליהם:

```
1. public class SomeClass {
2.
3.     public void someMethod() {}
4. }
```

```
5. public class SomeOtherClass {
6.
7.     public void someOtherMethod(SomeClass ref) {
8.         ref.someMethod();
9.     }
10. }
```

א. בשורה 8 מתבצע זימון של השרות `someMethod`. האם השרות שיקרא הוא בהכרח השרות המופיע בשורה 3? נמקו בקצרה.

לא. השרות `someOtherMethod` עשוי לקבל מופע של מחלקה אשר יורשת מהמחלקה `SomeClass` ודורסת את השרות `someMethod`

ב. עבור כל אחד מהשינויים הבאים ציינו האם הוא מבטיח כי בשורה 8 מתבצע זימון של השרות המופיע בשורה 3? הסעיפים אינם תלויים זה בזה. נמקו בקצרה:

- שינוי שורה 1 ל- `public final class SomeClass {`

כן. לא ניתן לרשת ממחלקה המוגדרת `final` ולכן לא ניתן יהיה לא ניתן לדרוס השרות `someMethod`. כמו כן, מובטח שהארגומנט הוא מטיפוס סטטי `SomeClass` ומכיוון שלא ניתן לרשת ממנה אז גם הטיפוס הדינמי יהיה `SomeClass`

- שינוי שורה 3 ל- `protected void someMethod() {}`

לא. השרות `someOtherMethod` עשוי לקבל מופע של מחלקה אשר יורשת מהמחלקה `SomeClass` ודורסת את השרות `someMethod`

- שינוי שורה 3 ל- `public static void someMethod() {}`

כן. בקריאה לשרות המוגדר כ `static` מתבצע בזמן קומפילציה קישור לשרות לפי הטיפוס הסטטי של ההפניה (ומכאן השם...). גם אם בפועל יועבר מופע של מחלקה אשר יורשת מהמחלקה `SomeClass` ודורסת את השרות `someMethod` ה- `JVM` יתעלם מהשרות הדורס.

• שינוי שורה 7 ל- `public void someOtherMethod(final SomeClass ref) {`

לא. השרות `someOtherMethod` עשוי לקבל מופע של מחלקה אשר יורשת מהמחלקה `SomeClass` ודורסת את השרות `someMethod`. המציין `final` במקרה זה אינו מונע דריסה של המחלקה במקום אחר (אלא רק מונע השמה להפנייה).

• החלפת שורה 5 ב- `class SomeOtherClass {`
והעברת קטע הקוד השני (שורות 10-5) לסוף הקובץ `SomeClass.java`

לא. השרות `someOtherMethod` עשוי לקבל מופע של מחלקה אשר יורשת מהמחלקה `SomeClass` ודורסת את השרות `someMethod`. העובדה ששתי המחלקות מופיעות באותו הקובץ אינה משפיעה על עובדה זו.

ושוב, בהצלחה!