

פתרון הבחינה בתוכנה 1

סמסטר ב', מועד א' 2013/2014

7.7.2014

ליאור וולף, יעל אמסטרדמר, דביר נתנאלי ולנה דנקין

הוראות (נא לקרוא!)

- משך הבחינה **שלוש שעות**, חלקו את זמנכם ביעילות.
- אסור השימוש בחומר עזר כלשהו, כולל מחשבוני או כל מכשיר אחר פרט לעט. בסוף הבחינה צורף לנחותכם נספח ובו תיעוד מחלקות שימושיות.
- יש לענות על כל השאלות בגוף הבחינה במקום המיועד לכך. המקום המיועד מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס המבחן תיפסל. תשובות במחברת הבחינה לא תיבדקנה. במידת הצורך ניתן לכתוב בגב טופס הבחינה.
- 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.
- ניתן להוסיף הנחות לגבי אופן השימוש בשירותים המופיעים בבחינה, ובלבד שאין הן סותרות את תנאי השאלה. יש לתעד הנחות אלו כחווה (תנאי קדם, תנאי בתר) בתחביר המקובל, שייכתב בתחילת השירות.
- יש להתייחס בכבוד רב לצוות המשיחים.

לשימוש הבודקים בלבד:

שאלה	משקל	א	ב	ג	ד	ה	סה"כ
1	25						
2	30						
3	25						
4	20						

בהצלחה!

כל הזכויות שמורות ©

מבלי לפגוע באמור לעיל, אין להעתיק, לצלם, להקליט, לשדר, לאחסן במאגר מידע, בכל דרך שהיא, בין מכנית ובין אלקטרונית או בכל דרך אחרת כל חלק שהוא מטופס הבחינה.

שאלה 1 (25 נקודות)

השאלה הבאה עוסקת במימוש enum המייצג ומבצע פעולות עבור מבנה אלגברי המכונה **חוג למחצה** (semiring).

אנחנו רגילים לעבוד באלגברה הרגילה מעל המספרים הממשיים בה מוגדרות הפעולות חיבור (סכום), למשל $4 + 5 = 9$, ופעולת הכפל, למשל $4 \times 5 = 20$. האלגברה הרגילה מכונה בשל כך גם **Plus-mult**.

אולם, באופן כללי יותר, ניתן להגדיר אלגברות שבהן שני אופרטורים, $\oplus$ (נכנה אותו osum) ו- $\otimes$ (נכנה אותו omult), המוגדרים באופן שונה. למשל במבנה הידוע כ-**Max-plus**, $\oplus$ הוא אופרטור שמחזיר את המקסימום של שני הארגומנטים, למשל $4 \oplus 5 = \max\{4, 5\} = 5$. האופרטור $\otimes$ מתבצע על ידי סיכום שני הארגומנטים, למשל $4 \otimes 5 = 4 + 5 = 9$.

כלומר, באלגברה הרגילה, $\oplus$ הוא $+$ ו- $\otimes$ הוא $\times$ ואילו ב-**Max-plus** לפי ההגדרה $\oplus$ הוא $\max$ ו- $\otimes$ הוא $+$. קיימים גם מבנים אחרים בהם ההגדרות הן שונות.

באופן פורמאלי יותר, חוג למחצה (SEMIRING) הוא מבנה הכולל קבוצה R עם פעולות osum ו-omult, המוגדרות כך שהן המקיימות מספר תכונות:

1. שתי הפעולות קיבוציות (אסוציאטיביות): $(x \oplus y) \oplus z = x \oplus (y \oplus z)$ וכן"ל לגבי $\otimes$.

2. פעולות החיבור חילופיות (קומוטטיביות): $x \oplus y = y \oplus x$

3. קיים איבר נטרלי לכל אחת משתי הפעולות $x \oplus 0 = x$ ו- $x \otimes 1 = x$

4. מתקיים חוק הפילוג (דיסטריבוטיביות): $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$

$$z \otimes (x \oplus y) = (z \otimes x) \oplus (z \otimes y)$$

התכונות הכלליות של חוג למחצה, וכן של האלגברה ה"רגילה" (Plus-mult) ושל **Max-plus** מסוכמות בטבלה למטה.

Max-plus		Plus-mult		הגדרה	
דוגמא	פעולה	דוגמא	פעולה	סימן	שם
$10 \oplus 3 = \max\{10, 3\} = 10$	Max	$4 \oplus 2 = 4 + 2 = 6$	סכום +	$a \oplus b$	$\oplus$ osum
$10 \otimes 3 = 10 + 3 = 13$	סכום +	$5 \otimes 2 = 5 \times 2 = 10$	כפל $\times$	$a \otimes b$	$\otimes$ omult
$-10 \oplus -\infty = \max\{-10, -\infty\} = -10$	$-\infty$	$10 \oplus 0 = 10 + 0 = 10$	0	$a = 0 \oplus a$	נטרלי חיבור 0
$-10 \otimes 0 = -10 + 0 = -10$	0	$1 \otimes 99 = 1 \times 99 = 99$	1	$a = 1 \otimes a$	נטרלי כפל 1

כעת, נגדיר פעולות על מטריצות מעל האיברים של חוגים למחצה.

osum של מטריצות באותו גודל A ו-B מייצר מטריצה C באותו גודל:

$$[C]_{ij} = [A \oplus B]_{ij} = [A]_{ij} \oplus [B]_{ij}$$

omult על מטריצה A בגודל $m \times p$ ומטריצה B בגודל $p \times n$ מייצר מטריצה C בגודל $m \times n$:
 $(i=1..m, j=1..n)$

$$[C]_{ij} = [A \otimes B]_{ij} = \bigoplus_{k=1}^p [A]_{ik} \otimes [B]_{kj}$$

הפעולות הנ"ל אנלוגיות לחיבור וכפל מטריצות שאנחנו מכירים מן האלגברה ה"רגילה", אך משתמשות בפעולות $\oplus$ ו- $\otimes$ של החוג למחצה שאליו שייכים איברי המטריצה. האופרטור $\oplus_{k=1}^p$ עובר על k מ-1 עד p ומבצע $\oplus$ על הערך המתקבל לכל k (בדומה לפעולה $\sum_{k=1}^p$, בה משתמשים כקיצור לסכום).

הגדירו enum בשם `Semiring` אשר ימנה, במימוש הנוכחי שלכם, שני מופעים המייצגים את החוגים למחצה שראינו, `MAX_PLUS` ו-`PLUS_MULT`. ה- enum יממש את שתי הפעולות של המטריצות, כלומר את שתי המתודות הבאות:

```
/**
 * @pre A!= null, B != null
 * @pre A.getRowDimension() == B.getRowDimension() >= 0
 * @pre A.getColumnDimension() == B.getColumnDimension() >= 0
 * @pre all of the elements in A and B are not Double.NaN
 */
public Matrix matrixOsum(Matrix A, Matrix B)

/**
 * @pre A!= null, B != null
 * @pre B.getRowDimension() == A.getColumnDimension() >= 0
 * @pre A.getRowDimension() >= 0
 * @pre B.getColumnDimension() >= 0
 * @pre all of the elements in A and B are not Double.NaN
 */
public Matrix matrixOmult(Matrix A, Matrix B)
```

המטריצות A, B וערך ההחזרה שייכות למחלקה `Matrix` המוגדרת בנספח ומייצגת מטריצות של `double`.

הנחיה:

על המימוש להשתמש בפולימורפיזם אמיתי, כדי למנוע שכפול קוד ולאפשר הוספה מהירה של חוגים למחצה חדשים ל-`enum`.

אין צורך לממש את כל הפעולות של חוג למחצה, אלא רק את אלו הנחוצות לצורך מימוש הפעולות על מטריצות.

```
public enum Semiring {

    PLUS_SUM {

        public double getOsumNeutralElement() {
            return 0;
        }

        public double osum(double elem1, double elem2) {
            return elem1 + elem2;
        }

        public double omult(double elem1, double elem2) {
            return elem1 * elem2;
        }

    },

    MAX_PLUS {

        public double getOsumNeutralElement() {
            return Double.NEGATIVE_INFINITY;
        }

    }

}
```

```

    public double osum(double elem1, double elem2) {
        return Math.max(elem1, elem2);
    }

    public double omult(double elem1, double elem2) {
        return elem1 + elem2;
    }

};

public Matrix matrixOsum(Matrix A, Matrix B) {
    int rowDimension = A.getRowDimension();
    int columnDimension = A.getColumnDimension();
    Matrix C = new Matrix(rowDimension, columnDimension,
        getOsumNeutralElement());
    for (int i = 0; i < rowDimension; i++) {
        for (int j = 0; j < columnDimension; j++) {
            C.set(i, j, osum(A.get(i, j), B.get(i, j)));
        }
    }
    return C;
}

public Matrix matrixOmult(Matrix A, Matrix B) {
    int rowDimension = A.getRowDimension();
    int columnDimension = B.getColumnDimension();
    int p = A.getColumnDimension();
    Matrix C = new Matrix(rowDimension, columnDimension, getOsumNeutralElement());
    for (int i = 0; i < rowDimension; i++) {
        for (int j = 0; j < columnDimension; j++) {
            double tempOsum = getOsumNeutralElement();
            for (int k = 0; k < p; k++) {
                tempOsum = osum(tempOsum,
                    omult(A.get(i, k), B.get(k, j)));
            }
            C.set(i, j, tempOsum);
        }
    }
    return C;
}

public abstract double getOsumNeutralElement();

public abstract double osum(double elem1, double elem2);

public abstract double omult(double elem1, double elem2);
}

```

Class Matrix

extends java.lang.Object

A simple double array matrix implementation.

Constructor Summary

(int rows, int cols, double initval) [Matrix](#)
Constructs a matrix of doubles with the specified dimensions, where all its elements are initialized to initval.

Method Summary

double	get (int row, int col) Returns the element at the specified row and column.
int	getColumnDimension () Returns the column dimension.
int	getRowDimension () Returns the row dimension.
void	set (int row, int col, double elem) Sets the element at the specified row and column.

java.lang

Class Double

Field Summary

static double	NaN A constant holding a Not-a-Number (NaN) value of type double.
static double	NEGATIVE_INFINITY A constant holding the negative infinity of type double.

שאלה 2 (30 נקודות)

בשאלה זו עליכם להשלים את מימוש המתודה **predictTextTopic** אשר חוזרת את הנושא בו עוסק טקסט בקובץ קלט נתון.

לרשות המתודה עומדים קבצי-עזר אשר בכל אחד מהם רשימת מילים המאפיינות נושא מסוים ואת משקלן (בכל שורה מופיעים מילה, רווח, ומספר שלם בטווח 1-5 המציין את משקל חשיבות המילה באפיון נושא זה).

המתודה תחשב ציון התאמה עבור הטקסט מול כל אחד מהנושאים (על בסיס המילים האופייניות לכל נושא), ולבסוף תחזיר את שמו של הנושא בעל ציון ההתאמה המקסימלי.

כדי לחשב את ציון ההתאמה של הטקסט הנתון מול נושא מסוים, נסכום את משקלן של כל המילים המאפיינות את הנושא ונמצאות בטקסט הקלט (**ללא חזרות**) – משקלה של מילה הנמצאת מספר פעמים בטקסט הקלט ייסכום רק פעם אחת).

דוגמא:

עבור קובץ קלט המכיל את הטקסט הבא:

The party law regarding food is to eat an apple for dinner

ותוך שימוש בקבצי-העזר הבאים:

Science.txt	Food.txt	Politics.txt	שם הקובץ:
Science 5 Lab 4 Physics 5 Measure 4 Variable 3 Fact 3 Data 4 Law 1	Food 5 Orange 1 Apple 3 Eat 4 Recipe 4 Dinner 3 Cook 4	Politics 5 Minister 4 Party 2 Elections 5 Poll 2 Vote 4 Law 2 Knesset 4	תוכן הקובץ:

תחזיר המתודה את המחרוזת "Food", כי נושא זה קיבל את ציון ההתאמה המקסימלי עבור הטקסט הנתון:

Science: law (1) = 1
Food: food (5) + eat (4) + apple (3) + dinner (3) = **15**
Politics: party (2) + law (2) = 4

חתימת המתודה:

```
public static String predictTextTopic (String inputFileName,
                                       List<String> topicNames)
```

קלט המתודה:

inputFileName – שם של קובץ טקסט אותו נרצה לנתח (למשל "input.txt").
 topicNames – רשימת מחרוזות המציינות שמות של נושאים אפשריים (למשל {"Science", "Food", "Politics"})

הערות:

- ניתן להניח שרשימת הנושאים אינה ריקה ושקובץ הטקסט מכיל לפחות מילה אחת.
- ניתן להניח שעבור כל נושא ברשימת הנושאים, קיים קובץ-עזר מתאים באותו שם (וסיימת ".txt").
- ניתן להניח שכל הקבצים נמצאים בספירת העבודה של התוכנית.
- עליכם להשתמש במבני נתונים יעילים על מנת לתמוך במספר גדול של נושאים ומילים המאפיינות כל נושא.
- עליכם להשלים קוד ב-10 מלבנים בהמשך (לנוחיותכם סימנו בכוך כל מקום בו עליכם להשלים קוד).

סעיף א' (4 נק')

השלימו את הגדרת הטיפוסים במתודה `predictTextTopic` ע"י מילוי החלקים הריקים (עיינו בהמשך השאלה וקבעו מה יהיו טיפוסיהם) של מתודות העזר, הקפידו לשמור על תאימות בהמשך):

```
public static String predictTextTopic (String inputFileName,
                                       List<String> topicNames) throws IOException {
    Set<String> inputFileWords =
        loadInputWords(inputFileName);

    Map<String, Map<String, Integer>> topicsWordWeights =
        loadTopicFiles(topicNames);

    Map<String, Integer> topicScores =
        calcTopicScoresForTextWords(inputFileWords,
                                     topicsWordWeights);

    String maxScoreTopic = getMaxScoreTopic (topicScores);

    return maxScoreTopic;
}
```



סעיף ב' (7 נק')

השלימו את טיפוס ערך ההחזרה ואת מימוש המתודה `loadInputWords` אשר מחזירה אוסף נתונים ובו כל המילים הנמצאות בקובץ `inputFile`. על האוסף המוחזר להכיל את המילים כשהן ללא חזרות, ובאותיות קטנות.
 ✓ ניתן להניח שהקובץ נמצא בתיקיה הנכחית ומכיל רק אותיות לועזיות ורווחים.
 ✓ אין צורך לטפל בחריגות בתוך המתודה (עליהן להיזרק הלאה למתודה הקוראת), אבל יש לסגור משאבים בצורה מסודרת גם במקרה של שגיאה.

```
private static Set<String> loadInputWords (
    String inputFile) throws IOException {
```



```
    Scanner scanner = null;
    try {
        scanner = new Scanner (new File(inputFile));
        HashSet<String> wordSet = new HashSet<String>();
        while (scanner.hasNext())
            wordSet.add(scanner.next().toLowerCase());
        return wordSet;
    }
    finally {
        if (scanner!=null)
            scanner.close();
    }
}
```

```
}
```

סעיף ג' (7 נק')

השלימו את טיפוס ערך ההחזרה ואת מימוש המתודה loadTopicFiles אשר מקבלת רשימת מחזורות המייצגות נושאים, ומחזירה אוסף נתונים המאחסן עבור כל שם של נושא את רשימת המילים המאפיינות אותו (באותיות קטנות) ואת משקלן.

- ✓ ניתן להניח שעבור כל נושא קיים בספרית העבודה הנוכחית קובץ עזר עם שם הנושא וסיומת ".txt".
- ✓ ניתן להניח שבכל שורה בקבצי העזר יופיעו מילה (אותיות לועזיות בלבד), אח"כ רווח, ואח"כ מספר שלם בטווח 1-5.
- ✓ אין צורך לטפל בחריגות בתוך המתודה (עליהן להיזרק הלאה למתודה הקוראת), אבל יש לסגור משאבים בצורה מסודרת גם במקרה של שגיאה.

```
private static Map<String, Map<String, Integer>> loadTopicFiles(
    List<String> topicNames) throws IOException {
```

```
    Map<String, Map<String, Integer>> result = new HashMap<>();
    for (String fileName : topicNames) {
        Map<String, Integer> wordInFileMap = new HashMap<>();

        try (Scanner scanner = new Scanner (new File(fileName + ".txt"))) {
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] tokens = line.split(" ");
                wordInFileMap.put(tokens[0].toLowerCase(),
                                   Integer.parseInt(tokens[1]));
            }
        } // automatically closing scanner
        result.put(fileName, wordInFileMap);
    }

    return result;
}
```

```
}
```

הערות:

- מבחינת סגירת משאבים במקרה של חריגה, בסעיף ג' נעשה שימוש במבנה החדש של try with resources אשר מקביל ל- try-finally בו נעשה שימוש בסעיף ב' (וכלל סגירה אוטומטית של משאבים בסוף מקטע הקוד).
- שגיאות נפוצות בסעיף זה:
 - בעיות שונות ביצירת מפה המחזיקה מפות כערכים.
 - המרה שגויה בין מחזורות למספר.
 - לא סוגרים כל קובץ שנפתח.
 - שימוש ב- TreeMap במקום ב- HashMap (מיותר כי לא נעשה שימוש בתכונת המיון שלו).
 - תפיסה של חריגות בתוך המתודה בניגוד למוגדר (מתודות אמורות לחלחל למתודה הקוראת).

סעיף ד' (7 נק')

השלימו את טיפוסי הארגומנטים ואת מימוש המתודה calcTopicScoresForTextWords, המחזירה עבור כל שם נושא את ציון ההתאמה שחושב לו בהסתמך על המילים בטקסט הקלט. ראו לעיל את אופן חישוב ציון ההתאמה.

private static Map<String, Integer> calcTopicScoresForTextWords(

Set<String>

inputFileWords,



Map<String, Map<String, Integer>>

topicsWordWeights) {



```
Map<String, Integer> topicScores = new HashMap<String, Integer>();
for (String topic: topicsWordWeights.keySet()) {
    int topicScore = 0;
    for (String word: inputFileWords) {
        if (topicsWordWeights.get(topic).containsKey(word)) {
            topicScore += topicsWordWeights.get(topic).get(word);
        }
    }
    topicScores.put(topic, new Integer(topicScore));
}
return topicScores;
```



}

סעיף ה' (5 נק')

השלימו את מימוש המתודה getMaxScoreTopic המחזירה מחרוזת המייצגת את הנושא בעל ציון ההתאמה המקסימלי.

✓ במידה וקיימים מספר נושאים עם אותו ציון התאמה, יש להחזיר אחד מהם.

✓ במידה והקלט ריק יש להחזיר null.

private static String getMaxScoreTopic(Map<String, Integer> topicScores) {

```
Integer maxScore = Collections.max(topicScores.values());
for (String topic : topicScores.keySet()) {
    if (topicScores.get(topic).equals(maxScore)) {
        return topic;
    }
}
return null;
```



}

שאלה 3 (25 נקודות)

סעיף א' (16 נק')

השלימו את מימוש הפונקציה הרקורסיבית שמחשבת את כל תתי הקבוצות של מספרים מ 1 עד n (כולל 1 ו n).
 הפונק' מקבלת את n ומחזירה רשימה של קבוצות (Set), כל Set מייצג תת קבוצה אחרת. הפונק' אינה מוגדרת עבור n שלילי, עבור $n = 0$ תת הקבוצה היחידה היא הקבוצה הריקה.
 לדוגמא, עבור $n = 2$ הפונק' תחזיר רשימה המכילה 4 קבוצות: $[], [1], [2], [2, 1]$ (קבוצה ריקה).

```
public static List<Set<Integer>> calcSubSets(int n){
    ArrayList<Set<Integer>> res = new ArrayList<Set<Integer>>();
    if (n == 0){
        /*res.add(new HashSet<Integer>());*/
    }
    else{
        List<Set<Integer>> recursionRes = calcSubSets(n-1);
        /*for (Set<Integer> l : recursionRes){
            Set<Integer> withNSet = new HashSet<Integer>();
            Set<Integer> withoutNSet = new HashSet<Integer>();
            withNSet.addAll(l);
            withNSet.add(n);
            withoutNSet.addAll(l);
            res.add(withNSet);
            res.add(withoutNSet);
        }*/
    }
    return res;
}
```

סעיף ב' (5 נק')

```

public static void randomPrint(){
    Random rand = new Random();
    List<Integer> l1 = new ArrayList<Integer>();
    List<Integer> l2 = new ArrayList<Integer>();
    for (int i = 1; i < 5; i++){
        if (rand.nextDouble() > 0.5){
            l1.add(i);
        }
        else{
            l2.add(i);
        }
    }
    Iterator<Integer> it1 = l1.iterator();
    Iterator<Integer> it2 = l2.iterator();
    while (it1.hasNext() || it2.hasNext()){
        if(rand.nextDouble() > 0.5 && it1.hasNext()){
            System.out.print(it1.next() + " ");
        }
        else if (it2.hasNext()){
            System.out.print(it2.next() + " ");
        }
    }
}

```

איזה מבין הפלטים הבאים לא ניתן לקבל בהרצת קוד זה?

- א. 1 2 3 4
- ב. 1 3 2 4
- ג. 1 4 3 2
- ד. 2 1 4 3
- ה. 3 4 1 2

סעיף ג' (4 נק')

מהו הפלט של הרצת הפונקציה הבאה?

```

public void foo(){
    int sum = 0;
    for (int i = 0; i < 5; i++){
        sum += (i++);
    }
    System.out.println(sum);
}

```

- א. 6
- ב. 9
- ג. 10
- ד. 12
- ה. 15

שאלה 4 (20 נקודות)

קראו את קוד המחלקות הבאות וענו על השאלות.

```
import java.util.ArrayList;
import java.util.List;

public class A1 {

    public void bar() {
        System.out.println("A1 bar");
    }

    public static void main(String[] args) {
        List<A1> list = new ArrayList<>();
        list.add(new B());

        /* */
    }
}
```

```
import java.util.List;

public interface A2 {

    void foo(List<A1> list, int idx);
}
```

```
import java.util.List;

public class B extends A1 implements A2 {

    public void foo(List<A1> list, int idx) {
        A1 a1 = list.get(idx);
        if (a1 instanceof B) {
            System.out.println("It's a B!");
        } else {
            a1.bar();
        }
    }

    public void bar() {
        System.out.println("B bar");
    }
}
```

בכל אחד מהסעיפים הבאים מוחלף סימון ה- `/* */` בפונקציית ה-main במס' שורות קוד. הנכם מתבקשים לציין מהו הפלט של הרצת פונקציית ה-main בכל אחד מהמקרים. אם לדעתכם התכנית אינה עוברת קומפילציה או זורקת שגיאה בזמן ריצה, הסבירו מה סוג השגיאה (זמן קומפילציה או זמן ריצה) וממה היא נובעת. **בכל מקרה** (שגיאה או ריצה תקינה) יש לכתוב הסבר קצר.

סעיף א' (4 נק')

/* */ מוחלף ב-

```
list.add(new A1());
A2 a = (A2) list.get(1);
a.foo(list, 0);
```

תתקבל שגיאה בזמן ריצה.

האיבר באינדקס 1 ברשימה הוא מטיפוס A1, ולכן נסיון לבצע עליו casting ל-A2 ייכשל בזמן ריצה. התקבלו פתרונות שהניחו (באופן עקבי, כלומר, גם בסעיפים ג' וד') כי הקומפיילר אינו מאפשר המרה בין A1 ל-A2, למרות שהנחה זו שגויה.

סעיף ב' (4 נק')

/* */ מוחלף ב-

```
list.add(new B());
A1 a = list.get(1);
a.foo(list, 0);
```

תתקבל שגיאת קומפילציה.

במחלקה A1 לא מוגדרת המתודה foo כלל.

סעיף ג' (4 נק')

/* */ מוחלף ב-

```
list.add(new A1());
A2 a = (A2) list.get(0);
a.foo(list, 1);
```

הפלט יהיה: A1 bar

בקריאה ל-foo יופעל המימוש המוגדר במחלקה B. מימוש זה ישלוף את האיבר באינדקס 1 של הרשימה, שטיפוסו A1 ולא B. לכן ייקרא הקוד של else, המפעיל את a1.bar(). מימוש המתודה ב-A1 מדפיס את הפלט שלעיל.

סעיף ד' (4 נק')

/* */ מוחלף ב-

```
list.add(new A1() {
    public void bar() {
        System.out.println("A1 baz");
    }
});
A2 a = (A2) list.get(0);
a.foo(list, 1);
```

הפלט יהיה: A1 baz

דומה מאד לסעיף ג', רק שכאן מוסיפים לרשימה איבר מטיפוס אנונימי שיושם מ-A1. שוב ייקרא בלוק ה-else בפונקציה foo, והפעם תיקרא המתודה bar של הטיפוס האנונימי שדורסת את המתודה bar ב-A1.

סעיף ה' (4 נק')

/* */ מוחלף ב-

```
list.add(new B());
A2 a = new A2() {
    public void foo(List<A1> list, int idx) {
        B b = (B) list.get(idx);
        b.foo(list, 1);
    }
};
a.foo(list, 0);
```

הפלט יהיה: It's a B!

a הוא עצם מטיפוס אנונימי שמממש את A2. הוא מוציא את האיבר מהרשימה במקום ה-0 (מטיפוס B) וקורא שוב ל-foo. כעת ייקרא המימוש במחלקה B. מכיוון שהאיבר במקום 1 ברשימה גם הוא מטיפוס B, תנאי ה-if יתקיים ולכן יודפס הפלט שלעיל.

נספח לבחינה בתוכנה 1

API חלקי של מחלקות שימושיות

Class Random

Method Summary	
double	nextDouble() Returns the next pseudorandom, uniformlydistributed double value between 0.0 and 1.0.

Class File

Constructor Summary	
File (String pathname)	Creates a new File instance by converting the given pathname string into an abstract pathname.

Class FileReader

Constructor Summary	
FileReader (File file)	Creates a new FileReader, given the File to read from.
FileReader (String fileName)	Creates a new FileReader, given the name of the file to read from.
Method Summary	
Methods inherited from class java.io. InputStreamReader	
close , getEncoding , read , read , ready	
Methods inherited from class java.io. Reader	
mark , markSupported , read , read , reset , skip	

Class BufferedReader

Constructor Summary	
BufferedReader (Reader in)	Create a buffering character-input stream that uses a default-sized input buffer.
Method Summary	
void	close () Close the stream.
int	read () Read a single character.
int	read (char[] cbuf, int off, int len) Read characters into a portion of an array.
String	readLine () Read a line of text.

Class Scanner

Constructor Summary

[Scanner](#)([File](#) source)

Constructs a new Scanner that produces values scanned from the specified file.

[Scanner](#)([InputStream](#) source)

Constructs a new Scanner that produces values scanned from the specified input stream.

Method Summary

void	close ()	Closes this scanner.
Pattern	delimiter ()	Returns the Pattern this Scanner is currently using to match delimiters.
boolean	hasNext ()	Returns true if this scanner has another token in its input.
boolean	hasNextLine ()	Returns true if there is another line in the input of this scanner.
IOException	ioException ()	Returns the IOException last thrown by this Scanner 's underlying Readable.
String	next ()	Finds and returns the next complete token from this scanner.
String	nextLine ()	Advances this scanner past the current line and returns the input that was skipped.
Scanner	useDelimiter (String pattern)	Sets this scanner's delimiting pattern to a pattern constructed from the specified String .

Class String

Method Summary

char	charAt (int index)	Returns the char value at the specified index.
int	compareTo (String anotherString)	Compares two strings lexicographically.
int	compareToIgnoreCase (String str)	Compares two strings lexicographically, ignoring case differences.
boolean	contains (CharSequence s)	Returns true if and only if this string contains the specified sequence of char values.
boolean	endsWith (String suffix)	Tests if this string ends with the specified suffix.
boolean	equals (Object anObject)	Compares this string to the specified object.
boolean	equalsIgnoreCase (String anotherString)	Compares this String to another String , ignoring case considerations.
int	indexOf (String str)	Returns the index within this string of the first occurrence of the specified substring.
int	lastIndexOf (String str, int fromIndex)	Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
int	length ()	Returns the length of this string.

String	replaceAll (String regex, String replacement) Replaces each substring of this string that matches the given <u>regular expression</u> with the given replacement.
String	replaceFirst (String regex, String replacement) Replaces the first substring of this string that matches the given <u>regular expression</u> with the given replacement.
String[]	split (String regex) Splits this string around matches of the given <u>regular expression</u> .
boolean	startsWith (String prefix) Tests if this string starts with the specified prefix.
String	substring (int beginIndex, int endIndex) Returns a new string that is a substring of this string.
String	toLowerCase () Converts all of the characters in this String to lower case using the rules of the default locale.
String	toUpperCase () Converts all of the characters in this String to upper case using the rules of the default locale.
String	trim () Returns a copy of the string, with leading and trailing whitespace omitted.

Collections

Interface	Common implementations
List<E>	ArrayList<E>, LinkedList<E>
Set<E>	HashSet<E>, TreeSet<E>
Map<K,V>	HashMap<K,V>, TreeMap<K,V>

Class Collections

Method Summary	
static boolean	addAll (Collection <? super I > c, I ... elements)
static void	copy (List <? super I > dest, List <? extends I > src) - Copies all of the elements from one list into another. The destination list must be at least as long as the source list. If it is longer, the remaining elements in the destination list are unaffected.
static int	frequency (Collection <?> c, Object o)
static void	reverse (List <?> list)
static void	sort (List < I > list) Sorts the specified list into ascending order, according to the <i>natural ordering</i> of its elements.
static void	sort (List < T > list, Comparator <? super I > c) Sorts the specified list according to the order induced by the specified comparator.
static T	max (Collection <? extends I > coll) Returns the maximum element of the given collection, according to the <i>natural ordering</i> of its elements.
static T	min (Collection <? extends I > coll) Returns the minimum element of the given collection, according to the <i>natural ordering</i> of its elements.

Interface List<E>

Method Summary	
boolean	add(E e)
void	add(int index, E element)
boolean	addAll(Collection<? extends E> c)
void	clear()
boolean	contains(Object o)
E	get(int index)
int	indexOf(Object o)
boolean	isEmpty()
Iterator<E>	iterator()
int	lastIndexOf(Object o)
ListIterator<E>	listIterator()
E	remove(int index)
boolean	remove(Object o)
boolean	removeAll(Collection<?> c)
boolean	retainAll(Collection<?> c)
E	set(int index, E element)
int	size()

Interface Set<E>

Method Summary	
boolean	add(E e)
boolean	addAll(Collection<? extends E> c)
void	clear()
boolean	contains(Object o)
boolean	isEmpty()
Iterator<E>	iterator()
boolean	remove(Object o)
boolean	removeAll(Collection<?> c)
boolean	retainAll(Collection<?> c)
int	size()

Interface Map<K,V>

Method Summary		
void	clear()	
boolean	containsKey(Object key)	
boolean	containsValue(Object value)	
Set<Map.Entry<K,V>>	entrySet()	
V	get(Object key)	
boolean	isEmpty()	
Set<K>	keySet()	
V	put(K key, V value)	
V	remove(Object key)	
int	size()	
Collection<V>	values()	Returns a Collection view of the values contained in this map.
Nested Class Summary		
static interface	Map.Entry<K,V>	A map entry (key-value pair).