בחינה בתוכנה

סמסטר ב', מועד א' ,תשס"ט

5/7/2009

ליאור וולף, ליאור שפירא, נעמה מאיר, מתי שמרת

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 (ויותר). יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תפסל. 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.
- אסור השימוש בחומר עזר כלשהוא, כולל מחשבונים או כל מכשיר אחר פרט לעט. בסוף הבחינה צורף לנוחותכם נספח ובו תיעוד מחלקות שימושיות.

שאלה	א	ב	ג	ד	ה	ו	ז	סה"כ	
1									
2									
3									
4									
	ציון בחינה:								

בהצלחה!

שאלה 1 (35 נקודות)

עליכם לממש מערכת לייצוג מטריצות ולביצוע פעולות עליהן . נתון מנשק כללי לייצוג מטריצה

מטריצה לדוגמה (אם שכחתם): $A=\begin{bmatrix} 1 & 4 & 10 & 2 \\ 12 & 0.5 & -1 & 0.1 \\ 2 & 9 & 133 & -4 \end{bmatrix}$

```
/** Represents a matrix */
public interface IMatrix {
	/** return the value at row r and column c */
	public double at(int r,int c);

	/** set the value at row r and column c to be val */
	public void setAt(double val, int r, int c);

	/** transpose the matrix (A'[i][j] = A[j][i]) */
	public void transpose();

	/** this[i][j] = this[i][j]+m[i][j] for each i and j */
	public void add(IMatrix m);

	/** multiply with m and return new matrix as result */
	public IMatrix multiply(IMatrix m);

	/** number of rows */
	public int rows();

	/** number of columns */
	public int cols();
}
```

הערה: האינדקסים במטריצה יהיו [0,c-1] לעמודות ו-[0,r-1] לשורות

<u>תזכורת:</u>

חיבור מטריצות: נסמן ב $C = A+B$ את סכום המטריצות (מאותו גודל) אזי $C_{ij} = A_{ij} + B_{ij}$

כפל מטריצות: A בגודל m x n, Bבגודל n x p המכפלה AB מגודל m x p מוגדרת ע"י:

$$(AB)_{ij} = \sum_{r=0}^{n-1} A_{ir} B_{rj}$$

סעיף א'

הגדירו תנאי קדם ואחר לכל שירות במנשק IMatrix, ומשתמר מחלקה (אם צריך)

```
public interface IMatrix {

    public double at(int r,int c);

    public void setAt(double val, int r, int c);

    public void transpose();

    public void add(IMatrix m);

    public IMatrix multiply(IMatrix m);

    public int rows();

    public int cols();
}
```

סעיף ב '

סטודנט X וסטודנטית Y החליטו לממש את המנשק IMatrix. תחילה הם החליטו לממש את המערכת כך:

public class FullMatrix implements IMatrix {…} – מטריצה מלאה

public class DiagonalMatrix implements IMatrix {…} – מטריצה אשר בה יש אברים רק על האלכסון

מה הבעיה במימוש זה של המערכת?

סעיף ג '

X ו-Y החליטו לשנות את המימוש ויצרו את המנשק הבא, **השלימו את תנאי הקדם והאחר של השירותים get, set ו-isLegal.**

```
public interface IMatrixImpl {
	/** Returns true if values be placed in this cell*/
	//
	//
	//
	//
	//
	//
	public boolean isLegal(int r, int c);

	//
	//
	//
	//
	//
	//
	public double get(int r, int c);

	//
	//
	//
	//
	//
	//
	public void set(double val, int r, int c);

	public int rows();
	public int cols();
}
```

סעיף ד '

X ו-Y כתבו את המחלקה FullMatrix אשר מממשת את IMatrixImpl

```
public class FullMatrix implements IMatrixImpl {

	private double[] m;
	private int cols;

	public FullMatrix(int rows, int cols) {
		m = new double[rows*cols];
		this.cols = cols;
	}

	public double get(int r, int c) {
		return m[r * cols + c];
	}

	public boolean isLegal(int r, int c) {
		return true;
	}

	public void set(double val, int r, int c) {
		m[r * cols + c] = val;
	}

	public int rows() {
		return m.length/cols;
	}

	public int cols() {
		return cols;
	}

}
```

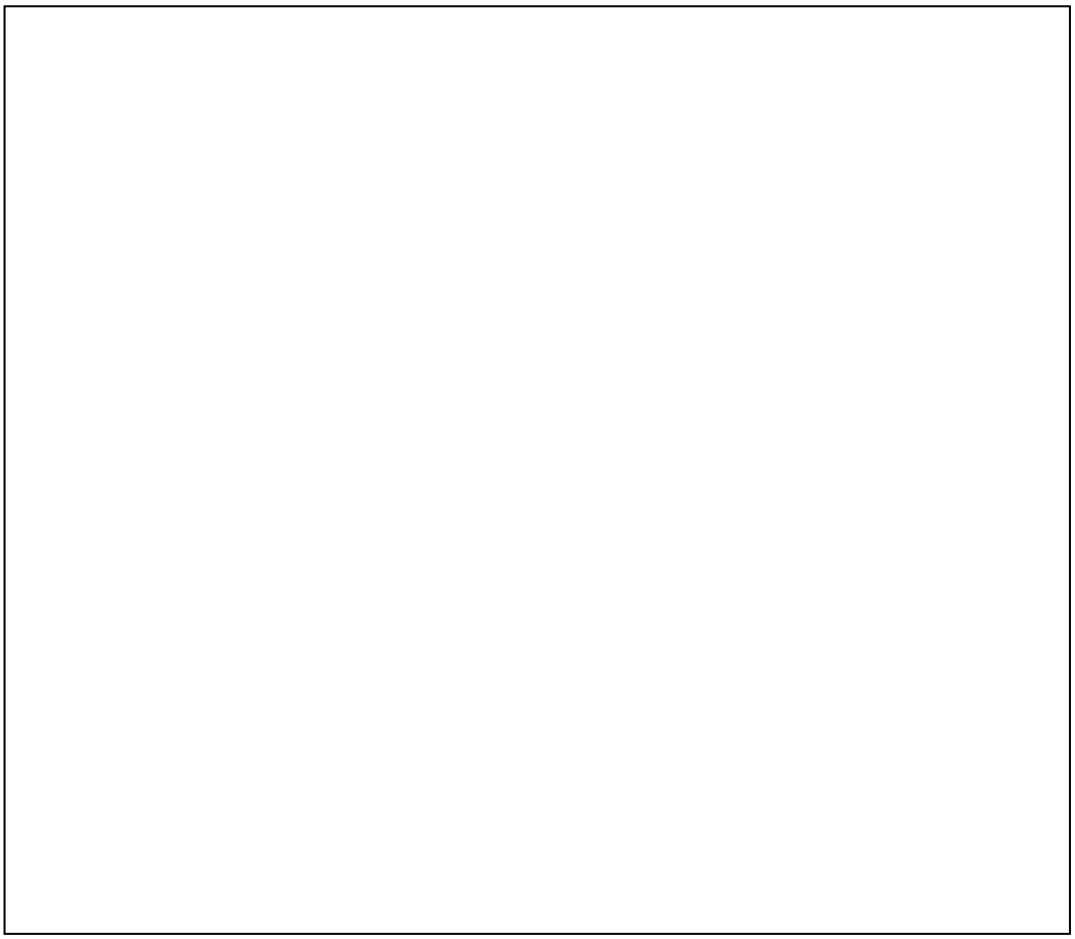
השלימו את המחלקה DiagonalMatrix אשר מממשת את אותו מנשק

```
public class DiagonalMatrix implements IMatrixImpl {

};
```

סעיף ה '

בהנחה שיהיה מס' מימושים רב של IMatrixImpl, הציעו שיטה כיצד ניצור מופעים של IMatrixImpl (ז"א מופעים של מחלקות המממשות את המנשק), **כיתבו את הגדרת המחלקה (ו/או השירותים) הנדרשים כדי לממש את הצעתכם (אין צורך לממש)**



סעיף ו '

בנוסף הם כתבו את המחלקה Matrix אשר מממשת את IMatrix. **השלימו את השירותים הבאים במחלקה:**

- ***אם צריך ליצור מופע חדש של IMatrixImpl עשו זאת בעזרת מה שכתבתם בסעיף ה'.***
- ***שימו לב שייתכן וקיימים בנאים ושירותים אחרים במחלקה, שימושו בעתיד.***

```
public class Matrix implements IMatrix {
	protected IMatrixImpl impl;

	public Matrix(int rows, int cols) {









	}
	public double at(int r, int c) {









	}
	public void setAt(double val, int r, int c) {









	}
	// ...  rest of the methods
}
```

סעיף ז '

סטודנט Z הציע להרחיב את המחלקה Matrix כך שתדע לחשב דטרמיננטה של מטריצה. **השלימו את הגדרת המחלקה AdvancedMatrix וממשו את השירות determinant**

דטרמיננטה של מטריצה בגודל 2 על 2: $A=\begin{bmatrix} a & b \\ c & d \end{bmatrix}, \det A = ad - bc$

מטריצה בגודל 3 על 3: $A=\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \det A = a(ei - fh) - b(di - fg) + c(dh - eg)$

מטריצה כללית תחושב בצורה רקורסיבית

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1k} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & a_{k3} & \cdots & a_{kk} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k2} & a_{k3} & \cdots & a_{kk} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k3} & \cdots & a_{kk} \end{vmatrix} + \ldots \pm a_{1k} \begin{vmatrix} a_{21} & a_{22} & \cdots & a_{2(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{k(k-1)} \end{vmatrix}.$$

ז"א $C_{ij} = (-1)^{i+j} M_{ij}$, $|A| = \sum_{i=1}^{k} a_{ij} C_{ij}$ ו- M_{ij} היא הדטרמיננטה של מטריצה הנוצרת ממחיקת שורה i ועמודה j ממטריצה A.

הערה: (i) ניתן לחשב דטרמיננטה גם בדרכים אחרות, אך אנו ממליצים על שיטה זו לצורך התרגיל (ii) דטרמיננטה של מטריצה לא ריבועית תוגדר כ-0 לצורך השאלה (iii) ניתן להוסיף שירותי עזר במידת הצורך

```
public class AdvancedMatrix _____ {

       …

       public double determinant() {
```

שאלה 2 (30 נקודות)

בשאלה הבא עליכם לבנות **servlet** אשר מקבל בתור פרמטר מחרוזת המכילה URL ומדפיס בחזרה (לדפדפן של המשתמש) את ה-URL הדומה ביותר בתוכו לURL החדש ואשר נשלחו בעבר באופן דומה אל ה-servlet . אין צורך בהוספת תגי HTML לדף החוזר.

מציאת התוכן הדומה ביותר ייעשה בשיטת **MIN-HASH**. בשיטה זו כל מסמך מיוצג ע"י וקטור. ב-וקטור זה N מספרים מסוג **DOUBLE** אשר מקודדים את התוכן של המסמך. הקידוד מבוסס על מתן ערכים רנדומאליים שנקבעו מראש למילים במילון שנקבע מראש.

משמעות "מילון" בהקשר הזה היא אוסף של מילים.

נניח ש N=1 . הקידוד נעשה באופן הבא :לפני הקידוד משייכים לכל מילה במילון ערך רנדומאלי בין 0 ל-1. בהינתן מסמך מסתכלים על הערכים של כל המילים מתוך המילון שמופיעות בו ורושמים את הערך המינימאלי בתור הקידוד של המסמך.

אם N = 2 פשוט חוזרים על התהליך פעמיים. כאשר לכל קורדינטה יש אוסף ערכים רנדומאליים נפרד (בכל באוסף שכזה יש ערך לכל מילה). לכל אוסף ערכים מחשבים את המינימום על כל המילים שמופיעים במסמך כדי לקודד את המסמך. שימו לב: הערכים למילים במילון אינם משתנים כאשר עוברים ממסמך אחד לשני. (במילה קורדינטה אנו מתכוונים למקום מסויים בוקטור)

עבור N כללי פשוט חוזרים על התהליך N פעמים. כל קורדינטה בלתי תלויה באחרות.

<u>דוגמא</u>: המילון לדוגמא מכיל את המילים "אבא" "אמא" "ילד" "ילדה", והקידוד לדוגמא הוא דו מימדי (N=2, שתי קורדינטות) לפי טבלאות הקידוד הבאות:

טבלה 1

מילה	ערך
אבא	0.91
אמא	0.62
ילד	0.76
ילדה	0.22

טבלה 2

מילה	ערך
אבא	0.52
אמא	0.26
ילד	0.17
ילדה	0.29

המשפט (שיקרא להלן משפט א') "דני הרביץ לדנה כי היא קיללה את אמא שלו" מקבל את הערך הוקטורי (0.62,0.26).

המשפט (להלן משפט ב') "הילד הביט בילדה שהביטה בילד" מקבל את הערך הוקטורי (0.22,0.17).

המשפט (להלן משפט ג') "אבא אמא קצת יותר לאט, אבא אמא תחכו מעט" מקבל את הערך הוקטורי (0.62,0.26).

המשפט (להלן משפט ד') "ילדה טובה כמו אבא" מקבל את הערך הוקטורי (0.22,0.29).

מידת הדמיון בין שני מסמכים נקבעת ע"י מספר המקומות בוקטורים המקודדים כל מסמך בהם הוקטורים שווים. להלן טבלת הדמיון של המשפטים בדוגמא:

	משפט א'	משפט ב'	משפט ג'	משפט ד'
משפט א'	2	0	2	0
משפט ב'	0	2	0	1
משפט ג'	2	0	2	0
משפט ד'	0	1	0	2

המילון נתון לכם בתור קובץ המכיל רשימה של מילים. שם הקובץ **DICT.TXT**. את סט הערכים למילים במילון עליכם לבנות בעצמכם כך שכל מסמך ייוצג ע"י N=999 ערכים.

מימוש ה-SERVLET יכלול מחלקת עזר **Helper** אשר אין עליכם לממש. עליכם לרשום מנשק בשם **IHelper** בלבד. מחלקה זו תכיל את הפונקציות הבאות:

1. randdouble שמחזירה מספר רנדומלי מסוג double בין 0 ל-1.
2. url2strings שמקבלת מחרוזת URL ומחזירה מבנה נתונים לבחירתכם שמכיל את כל המילים.
3. filename2strings שמקבלת שם של קובץ ומחזירה מבנה נתונים המכיל את רשימת המילים בקובץ.

משמעות המשפט "מבנה נתונים לפי בחירתכם" לעיל הוא: אחד ממבני הנתונים בנספח א' או מבנה נתונים שאתם ממשים בעצמכם על טופס הבחינה.

סעיף א '

רשמו את המנשק ***IHelper***

```
public interface IHelper
```

סעיף ב '

רשמו את המחלקה MINHASH אשר מקבלת לבנאי שלה שם של קובץ מילון ואולי פריטים אחרים וממשת את הפונקציונאליות של קידוד ה-MINHASH (לא כולל אחסון קידודים שחושבו בעבר)

סעיף ג '

רשמו את המחלקה MINHASHDATABASE אשר מממשת בסיס נתונים אשר אוגר אוסף של מסמכים ומאפשר שליפה של מסמכים דומים לפי שיטת ה-MINHASH. באוגר הכוונה היא לא לאחסן את המידע הטקסטואלי עצמו, אלא מידע מזהה בתוספת מידע לצורך מציאת דמיון בין מסמכים.
שימו לב, אם יש כמה מסמכים עם אותה מידת דמיון מקסימאלי ת, מספיק לשלוף אחד מהם.
שימו לב, יתכן שהדמיון המקסימאלי הוא דמיון 0

סעיף ד'

רשמו את המחלקה MINHASHSERVLET אשר מממשת את servlet כפי שנדרש לעיל.
שימו לב -- אם עדיין לא נשלח אף URL אל ה-servlet אזי יש להחזיר הודעה בנוסח "empty database."
לתשומת ליבכם – בסיס הנתונים משותף לכל המשתמשים של הסרולט, אבל הוא קיים רק כל עוד הסרוולט קיים. כדי לסייע, הוספנו בסוף נספח א' מעט מידע על המנשקים הרלוונטיים.

שאלה 3 (20 נקודות)

נתונות המחלקות הבאות:

```
public class Base {
	public Base() {
		x = 0;
		bar();
	}

	public Base(int x) {
		this.x = x;
		foo();
	}

	public void foo() {
		System.out.println("Base.foo : " + x);
	}

	private void bar() {
		System.out.println("Base.bar:" + x.toString());
	}

	protected Integer x;
}

public class Derived extends Base {
	public Derived() {
		bar();
	}

	public Derived(int x, int y) {
		super(x);
		this.y = y;
	}

	public void foo() {
		System.out.println("Derived.foo : " + x + ", " + y);
	}

	public void bar() {
		System.out.println("Derived.bar:" +
				x.toString() + ", " + y.toString());
	}

	private Integer y;
}


public class Main {
	public static void main(String[] args) {
		<**BODY OF MAIN**>
	}
}
```

ענו על שלשת הסעיפים הבאים. בכל סעיף יש לסמן תשובה אחת בלבד.

סעיף א '

מה יתרחש כשגוף ה main יהיה: `Base b = new Derived(10, 20);`

1. יודפס `Derived.foo : 10, 0`
2. התכנית לא עוברת קומפילציה
3. יודפס `Derived.foo : 10, 20`
4. יודפס `Derived.foo : 10, null`
5. במהלך ריצת התכנית ייזרק חריג
6. יודפס `Base.foo : 10`

סעיף ב '

מה יתרחש כשגוף ה main יהיה: `Derived d = new Base(10);`

1. יודפס `Derived.foo : 10, null`
2. יודפס `Derived.foo : 10, 0`
3. יודפס `Base.foo : 10`
4. התכנית לא עוברת קומפילציה
5. יודפס `Derived.foo : 10, 20`
6. במהלך ריצת התכנית ייזרק חריג

סעיף ג '

מה יתרחש כשגוף ה main יהיה: `Base b = new Derived();`

1. יודפס
   ```
   Derived.bar : 0, null
   Derived.bar : 0, null
   ```
2. התכנית לא עוברת קומפילציה
3. לא יודפס פלט וייזרק חריג במהלך ביצוע התכנית
4. יודפס `Base.bar : 0` ולאחר מכן ייזרק חריג
5. יודפס
   ```
   Base.bar : 0
   Derived.bar : 0, null
   ```
6. יודפס
   ```
   Base.bar : 0
   Base.bar : 0
   ```

שאלה 4 (15 נקודות)

נתונה המחלקה הגנרית Box.

```
1.  public class Box<T> {
2.      public T get() {
3.          return element;
4.      }
5.
6.      public void put(T element) {
7.          this.element = element;
8.      }
9.
10.     public void put(Box<T> box) {
11.         put(box.get());
12.     }
13.
14.     private T element;
15. }
```

וכן נתונה המחלקה BoxClient העושה שימוש במחלקה Box.

```
1.  public class BoxClient {
2.
3.      public static void main(String[] args) {
4.          Box<Number> nBox = new Box<Number>();
5.          Box<Integer> iBox = new Box<Integer>();
6.          nBox.put(iBox);
7.      }
8.
9.  }
```

סעיף א '

קוד הלקוח **אינו** עובר קומפילציה, אולם ניתן לשנות את קוד **המחלקה Box** כך שקוד הלקוח יעבור קומפילציה.
בצעו את השינויים הדרושים במחלקה Box שיאפשרו למחלקה BoxClient לעבור קומפילציה.
שימו לב, אין לשנות את קוד הלקוח וכמובן יש לשמר את הגנריות של המחלקה Box. אין צורך להעתיק את כל המחלקה, מספיק לציין את מספרי השורות בהן אתם מבצעים שינוי ולציין מהו השינוי.

סעיף ב '

נרצה להוסיף למחלקה `Box` את השירות הסטטי copy

```
public static<T> void copy(*1* from, *2* to) {

          to.put(from.get());

}
```

מהם יהיו הטיפוסים של הפרמטרים from ו- to כך שיאפשרו גמישות בשימוש בשירות זה? לדוגמה, נרצה לאפשר את קוד הלקוח הבא

```
public class BoxClient {

      public static void main(String[] args) {

            Box<Number> nBox = new Box<Number>();

            Box<Integer> iBox = new Box<Integer>();

            Box.copy(iBox, nBox);

      }

}
```

הטיפוס של from הוא: _____

הטיפוס של to הוא: _____

בהצלחה!