

# תוכנה 1

## תרגיל מספר 5

### הנחיות כלליות:

- קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
  - הגשת התרגיל תעשה במערכת ה VirtualTAU בלבד (<http://virtual2002.tau.ac.il/>).
1. יש להגיש קובץ zip יחיד הנושא את שם המשתמש (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip) קובץ ה zip יכיל:
- א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
  - ב. קבצי ה java של התוכניות אותם התבקשתם לממש.
  - ג. קובץ טקסט עם העתק של כל קבצי ה java
  - ד. קובץ טקסט בשם answers עם התשובות לשאלות

### חלק א':

סודוקו היא חידה בה יש למקם ספרות על לוח משובץ שגודלו 9x9, המורכב מאזורים בגודל 3x3. מטרת המשחק - למקם את הספרות 1 עד 9 על גבי לוח המשחק כך שבאותו טור, באותה שורה ובאותו אזור לא תופיע אותה ספרה יותר מפעם אחת. חלק מהמשבצות בלוח כבר מכילות ספרות. היפרסודוקו (Hypersoduko) הינו סודוקו המכיל בנוסף עוד 4 אזורים צבועים שגם בהם מותר לכל ספרה להופיע פעם אחת בלבד.

חידת היפרסודוקו לדוגמא:

8	4	5	2				7	
9		7	4				1	
							4	5
1	2				7			3
	7				9		5	
	9			2	1			
		4		9				
		2	7	8				1
	1		3			8		6

8	4	5	2	1	3	6	7	9
9	3	7	4	6	5	2	1	8
2	6	1	9	7	8	3	4	5
1	2	8	5	4	7	9	6	3
4	7	6	8	3	9	1	5	2
5	9	3	6	2	1	7	8	4
6	8	4	1	9	2	5	3	7
3	5	2	7	8	6	4	9	1
7	1	9	3	5	4	8	2	6

החבילה hypersudoku אשר ניתנת להורדה מאתר הקורס מכילה תוכנית לחישוב פתרון חידות היפרסודוקו. החבילה מורכבת משתי מחלקות :

- GUI – אחראית על הממשק הגרפי למשתמש. מכילה את פונקציית ה-main של האפליקציה (כלומר, הרצת האפליקציה נעשית ע"י הרצת hypersudoku.GUI). המימוש של מחלקה זו נתון בשלמותו.

- Solution – אחראית על חישוב הפתרון לחידת היפרסודוקו. מחלקה זו מכילה את המתודה :

```
public static boolean calcSolution(int[][] matrix)
```

מתודה זו מקבלת מטריצה בגודל 9x9 של חידת היפרסודוקו כאשר כל כניסה ריקה מכילה את

הערך 1- . המתודה מחזירה true אם קיים פתרון לחידה ו- false אחרת. במידה וקיים פתרון

לחידה, המתודה תמלא את הכניסות הריקות בספרות בהתאם לפתרון. **אחרת, matrix לא**

**תשונה!!!** המתודה calcSolution נקראת ע"י המחלקה GUI. מימושה של calcSolution אינו

נתון.

**עליכם להשלים את מימוש המתודה calcSolution במחלקה Solution.** אתם יכולים להיעזר בקטע

הפסאודו קוד הבא לפתרון נאיבי של חידת סודוקו. אלגוריתם זה מתעלם מבעיות יעילות ולכן ייתכן

כי הרצת תוכנית המבוססת עליו תיקח זמן רב :

### Algorithm solveSudoku (Matrix m)

1. Verify that each digit appears at most once in every row, column, zone and hyper-zone.  
If not - return false (no solution)
2. If there are no empty cells in m - return true.
3. Let  $m[i][j]$  be an empty cell in m.
4. For  $k=1$  to 9:
  - 4.1.  $m[i][j]=k$
  - 4.2. if solveSudoku(m)==true - return true (recursive call)
5.  $m[i][j] = -1$
6. return false

במידת הצורך ניתן להוסיף למחלקה Solution **מתודות עזר** חדשות.

### הדרכה טכנית

המחלקה GUI משתמשת בספרייה חיצונית בשם SWT שכוללת גם קוד ג'אווה וגם קוד תלוי- מערכת הפעלה בשפת C. בגלל שהספריות הללו אינן מותקנות ביחד עם ג'אווה, צריך להתקין אותן כדי להשתמש ב-SWT. לצורך ההתקנה בצעו את הפעולות הבאות:

a. הורדת קובץ zip של ספריית SWT המתאים למערכת ההפעלה בה אתם עובדים מהאתר

<http://eclipse.org/swt/>

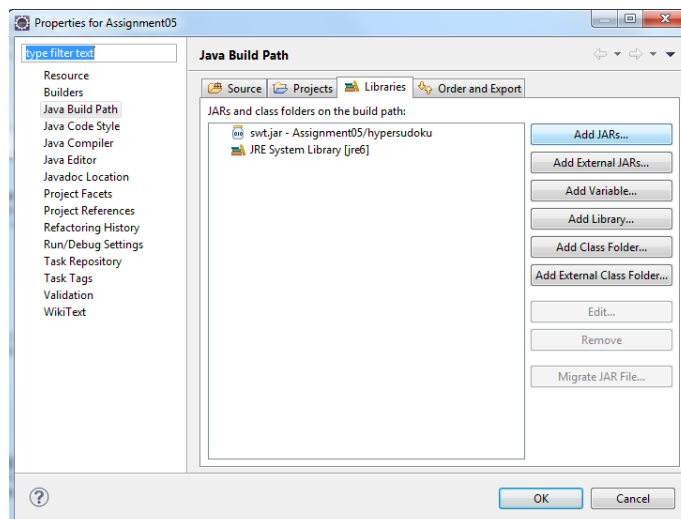
b. מתוך קובץ ה-zip העתיקו את קובץ swt.jar לתוך הפרויקט שלכם.

c. הוסיפו את ספריית swt.jar לתוך ה-build path של הפרויקט שלכם בעזרת לחיצה

ימנית על הפרויקט ולחיצה על properties ולאחר מכן Java Build Path וכאשר אתם

נמצאים בטאב Libraries בחרו Add Jars ובחרו את קובץ ה-swt.jar שהעתקתם לתוך

הפרויקט.



בחלק זה נשתמש בשרשראות מרקוב על מנת לכתוב תוכנית שהופכת קובץ טקסט לסיפור מצחיק.



התוכנית מורכבת מ- 2 שלבים :

1. בניית מילון בעזרת זוגות מילים עוקבות בטקסט.

2. הדפסת סיפור אקראי על פי המילון שבנינו.

**שלב א' - בניית המילון:**

נעבור על קובץ טקסט נתון ועבור כל צמד מילים עוקבות נשמור אותן במבנה WordTuple כאשר המילה הראשונה בצמד תהיה ה- prevWord והמילה השנייה תהיה currWord.  
עבור כל WordTuple נשמור את רשימת המילים שהופיעו לאחר צמד המילים בטקסט המקורי.  
לדוגמא : אם הטקסט המקורי הוא Roses are red and Roses are blue, עבור הצמד [Roses,are] נחזיק מערך שיכיל מצביע לצמד [are,red] וגם מצביע לצמד [are,blue].  
בנוסף עבור כל WordTuple נשמור גם מצביע אליו ממערך שנקרא לו מילון (dict).  
עליכם לממש את המחלקה WordTuple שמייצגת צמד מילים, ואת המתודה buildDict במחלקה Mimic שמקבלת שם של קובץ טקסט ובונה מילון של WordTuples.

על מנת לבנות את המילון נבצע:

1. נעבור על כל המילים בטקסט המקורי.
  2. עבור כל זוג מילים עוקבות נבדוק האם הוא כבר קיים במילון. במידה וכן נמצא את ה- WordTuple המתאים, במידה ולא ניצור WordTuple חדש.
  3. נניח שסדר המילים בטקסט הוא  $w_1 w_2 w_3 w_4 \dots$ , וה- WordTuple הנוכחי הוא  $[w_1, w_2]$ . נשתמש במילה  $w_3$  ונחפש אותה ברשימת המילים העוקבות של  $[w_1, w_2]$ . אם מצאנו אותה אז נמשיך הלאה. אם לא מצאנו אותה, ניצור WordTuple חדש שיכיל את את המילה העוקבת והמילה הנוכחית -  $[w_2, w_3]$  ונוסיף אותו למערך המילים העוקבות של הצמד  $[w_1, w_2]$ . במידה ומערך המילים העוקבות מלא, לא נוסיף לתוכו איברים נוספים.
  4. נחזור על הצעד הקודם עבור הצמד  $[w_2, w_3]$  והמילה העוקבת  $w_4$ .
- שימו לב, שאתם מקבלים מתודות עזר מוכנות כגון קריאה מקובץ, בנוסף ניתן להגדיר מתודות עזר נוספות.
- ניתן להניח כי גודל המערך following WordTuples במחלקה WordTuple יהיה מוגבל ל- 20 תאים, אך את המערך dict יש להקצות בצורה שיוכל להכיל את כל זוגות המילים לפי הטקסט הנתון.



#### שלב ב' - הדפסת סיפור:

הרעיון הוא להתחיל מצמד מילים התחלתי ולבחור אקראית מילה שתמשיך אותו מתוך רשימת המילים העוקבות לצמד זה בטקסט המקורי, ולהמשיך באופן זה עבור הצמד החדש שנוצר וכך הלאה. לדוגמא אם הטקסט המקורי הוא *Roses are red and Roses are blue* נבצע:

1. נתחיל מלמצוא WordTuple המתאים לזוג מילות ההתחלה שקיבלנו בתור ארגומנט. נניח שקיבלנו את *Roses are*.
2. נדפיס את המילה הנוכחית בצמד - עפ"י הדוגמא: עבור הצמד  $[Roses, are]$  נדפיס את *are*.
3. בעזרת רשימת המילים העוקבות, נבחר אקראית אחת מהן. עפ"י הדוגמא: המילים העוקבות האפשריות הן *red, blue* ולכן בוחרים אקראית אחת מהן למשל *blue*.
4. נקבל צמד מילים חדש המורכב מהמילה הנוכחית בצמד הקודם והמילה העוקבת שבחרנו אקראית - עפ"י הדוגמא: נקבל את הצמד החדש  $[are, blue]$ .
5. נחזור לצעד 2 עבור הצמד החדש ונבצע שוב את השלבים, עד שנדפיס את כמות המילים הדרושה.

6. במידה ונתקעים (לא ניתן למצוא את הצמד במילון), יש לחזור לצמד שמכיל את המילות ההתחלה שהתקבלו כארגומנט מהמשתמש ולהמשיך משם.

עליכם לממש את המתודה `printMimic` המקבלת מילון (בסיוס שלב אי), מילה ראשונה ושנייה להתחיל איתן, ואת מספר המילים שעלינו להדפיס. המתודה תדפיס בהתאם לאלגוריתם הנ"ל. בנוסף עליכם לממש את המתודה `main` אשר תקבל ארגומנטים מהמשתמש, תיצור מילון ותדפיס את הטקסט החדש לפי המילון. התוכנית תופעל בצורה הבאה:

```
Mimic <filename> <startWord1> <startWord2> <wordCount>
```

### הערות:

באתר הקורס תוכלו להוריד את הקבצים הדרושים על מנת להתחיל לעבוד, ובנוסף 2 קבצי טקסט בשביל בדיקות. הקבצים הם:

`WordTuple` - מחלקה עבור צמד מילים. השלימו את המתודות הריקות.

`Mimic` - המחלקה הראשית שבונה את המילון ומדפיסה סיפור חדש המבוסס על המילון.

`alice.txt`, `small.txt` - קבצי טקסט לצרכי בדיקה.



### דרישות נוספות לתרגיל:

החל מתרגיל זה אנו מצפים שתבצעו:

- בדיקת קלט - עבור כל מתודה יש לבצע בדיקת קלט לכל ארגומנט בקלט. אך אם יש התייחסות לכך בדרישות קדם אין צורך לבדוק זאת בתוך המתודה. למשל במתודה `main` יש לבדוק שהמערך `args` הוא לא ריק לפני שניגשים לתאים שלו.
- `Code conventions` - יש לקרוא ולפעול בהתאם למסמך המתואר על ידי Oracle:

<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

**בהצלחה!**