

תוכנה 1

תרגיל מספר 3

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/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש (לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip) קובץ ה zip יכול:
 - 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':

מטרת חלק זה לתרגל כתיבת שירותים סטטיים לא טריוויאליים. ס בהינתן החוזה של השירות. בכל סעיף מוגדר שירות למימוש עם שני חוזים אפשריים. עליכם לממש את השירות פעמיים, כך שיתאים לחוזים. ברוב המקרים ניתן לפתור את התרגיל בקלות ע"י הוספת **פונקציות עזר** (אף שהתרגיל לא דורש זאת במפורש).

הערה: במידת הצורך, עבור כל זוג פונקציות תוכלו להשתמש באחת הפונקציות לצורך מימוש השנייה.

1. בהינתן מערך כקלט, החזר מערך המכיל את אותם מספרים, אך מסודר כך שלאחר כל מופע של 4 יש מופע של 5. מותר להזיז את כל אברי המערך חוץ מהאברים שערכם 4. להלן מספר דוגמאות:

```
fix45({5, 4, 9, 4, 9, 5}) → {9, 4, 5, 4, 5, 9}
```

```
fix45({1, 4, 1, 5}) → {1, 4, 5, 1}
```

```
fix45({1, 4, 1, 5, 5, 4, 1}) → {1, 4, 5, 1, 1, 4, 5}
```

```
/*
 * @pre occurrences(4,arr) == occurrences(5, arr)
 * @pre arr[arr.length - 1] != 4
 * @pre forall 0 <= i < arr.length-2, arr[i] == 4 ==> arr[i+1] != 4
 * @post forall 0 <= i < arr.length-1, $prev(arr[i]) == arr[i]
 * @post $ret != arr
 * @post values in $ret are a permutation of values in arr
 * @post forall 0 <= i < $ret.length-2, $ret[i] == 4 => $ret[i+1] == 5
 * @post forall 0 <= i < arr.length-1, arr[i] == 4 ==> $ret[i] == 4
 */
public static int[] fix45A (int[] arr) {

}
```

```

/*
 * @post ((occurrences(4,arr) == occurrences(5, arr)) AND
 *       (arr[arr.length - 1] != 4) AND
 *       (forall 0 <= i < arr.length-2, arr[i] == 4 ==> arr[i+1] != 4))
 *       ==>
 *       ((forall 0 <= i < arr.length-1, $prev(arr[i]) == arr[i]) AND
 *       ($ret != arr) AND
 *       (values in $ret are permutation of values in arr) AND
 *       (forall 0 <= i < $ret.length-2, $ret[i]==4 ==> $ret[i+1]==5) AND
 *       (forall 0 <= i < arr.length-1, arr[i] == 4 ==> $ret[i] == 4))
 */
public static int[] fix45B (int[] arr) {

}

```

2. בהינתן מערך של מספרים שלמים (integers), האם ניתן לבחור תת קבוצה מתוך המערך כך שסכום תת הקבוצה שווה למספר מטרה מסוים? ניתן לפתור בעיה זו ע"י רקורסיה. טיפ: במקום להסתכל על כל המערך, נסתכל על המערך החל מאינדקס start ועד לסופו. הקורא לשירות זה יכול לציין שברצונו לפתור עבור כל המערך ע"י נתינת ערך 0 ל- start . להלן כמה דוגמאות:

$\text{groupSum}(0, \{2, 4, 8\}, 10) \rightarrow \text{true}$

$\text{groupSum}(0, \{2, 4, 8\}, 14) \rightarrow \text{true}$

$\text{groupSum}(0, \{2, 4, 8\}, 9) \rightarrow \text{false}$

```

/*
 * @pre 0 <= start <= nums.length - 1
 * @pre nums != null
 * @pre  $\forall 0 \leq i \leq \text{nums.length}-1, \text{nums}[i] \in \mathbf{Z}$ 
 * @post  $(\exists S \subseteq \text{nums}, \sum_{s \in S} s = \text{target}) \Rightarrow \$ret == \text{true}$ 
 * @post  $(\neg \exists S \subseteq \text{nums}, \sum_{s \in S} s = \text{target}) \Rightarrow \$ret = \text{false}$ 
 */
public static boolean groupSumA(int start, double[] nums, int target) {

}

```

```

/*
 * @pre 0 <= start <= nums.length - 1
 * @post  $(\exists S \subseteq [0..nums.length-1], \sum_{s \in S} nums[s] = target) \Rightarrow \$ret == true$ 
 * @post  $(\neg \exists S \subseteq [0..nums.length-1], \sum_{s \in S} nums[s] = target) \Rightarrow \$ret == false$ 
 */
public static boolean groupSumB(int start, int[] nums, int target) {

}

```

3. בהינתן מחרוזת, החזירו true אם מספר המופעים של תת המחרוזת "is" במקום כלשהוא במחרוזת

שווה למספר ההופעות של תת המחרוזת "not" במחרוזת (case sensitive). להלן כמה דוגמאות:

```

equalIsNot("This is not") → false
equalIsNot("This is notnot") → true
equalIsNot("noisxxnotyynotxisi") → true

```

```

/*
 * @pre str != null
 * @post $ret == (occurrences("is") == occurrences("not"))
 */
public static boolean equalIsNotA(String str) {

}

```

```

/*
 * @pre (str != null) AND (occurrences("is") == occurrences("not"))
 * @post $ret == (occurrences("is") == occurrences("not"))
 */
public static boolean equalIsNotB(String str) {

}

```

4. ניתנים כקלט שני מערכים של מחרוזות, a ו-b ללא כפילויות. החזירו את מספר המחרוזות המופיעות בשני המערכים. הפתרון צריך להיות יעיל ככל האפשר, "לינארי" אם המחרוזות ממוינות, ז"א עובר פעם אחת על המערכים. להלן כמה דוגמאות:

sharedStr({"Call", "me", "Ishmael"}, {"Call", "me", "Jonha"}) →

2

sharedStr ({"a", "c", "x"}, {"z", "b", "c", "x", "a"}) → 3

sharedStr ({"a", "b", "c"}, {"a", "b", "c"}) → 3

```

/*
 * @pre a != null AND b != null
 * @pre forall i, j; (0 <= i <= a.length-1 AND 0 <= j <= a.length-1) ==>
 *   ((i+1 == j) ==> (a[i] <= a[j]))
 * @pre forall i, j; (0 <= i <= b.length-1 AND 0 <= j <= b.length-1) ==>
 *   ((i+1 == j) ==> (b[i] <= b[j]))
 * @pre not exists i, j; i != j ==> a[i].equals(a[j])
 * @pre not exists i, j; i != j ==> b[i].equals(b[j])
 * @post |S| where S ≡ {s | ∃ i, j; s.equals(a[i]) ∧ s.equals(b[j])}
 */
public static int sharedStrA(String[] a, String[] b) {

}

```

```

/*
 * @pre not exists i, j; i != j ==> a[i].equals(a[j])
 * @pre not exists i, j; i != j ==> b[i].equals(b[j])
 * @post |S| where S ≡ {s | ∃ i, j; s.equals(a[i]) ∧ s.equals(b[j])}
 */
public static int sharedStrB(String[] a, String[] b) {

}

```

חלק ב':

חלק זה נועד לתרגל כתיבת חוזים עבור שירותים קיימים. בכל סעיף נתונה פונקציה, עליכם כתוב את החוזה (תנאי קדם ואחר) עבור הפונקציה הנתונה. הניחו כי כל הפונקציות אינן בודקות את הקלט.

1. שורש

```
public static double sqrt(double d) {  
    ...  
}
```

2. ערך מוחלט

```
public static double abs(double d) {  
    ...  
}
```

3. עצרת

```
public static int factorial(int n) {  
    ...  
}
```

4. מיון

```
public static void sort(int[] arr) {  
    ...  
}
```

חלק ג':

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.

חידת סודוקו לדוגמא:

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

ופתרונה:

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

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

- GUI – אחראית על הממשק הגרפי למשתמש. מכילה את פונקציות ה-main של האפליקציה (כלומר, הרצת האפליקציה נעשית ע"י הרצת il.ac.tau.cs.sw1.sudoku.GUI). המימוש של מחלקה זו נתון בשלמותו.

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

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

מתודה זו מקבלת מטריצה בגודל 9x9 של חידת סודוקו 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 and 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 $\text{solveSudoku}(m)==\text{true}$ - return true (recursive call)
5. $m[i][j] = -1$
6. return false

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

הדרכה טכנית

1. המחלקה GUI משתמשת בספרייה חיצונית בשם SWT שכוללת גם קוד ג'אווה וגם קוד תלוי- מערכת הפעלה בשפת C. בגלל שהספרייה 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://www.eclipse.org/swt/>

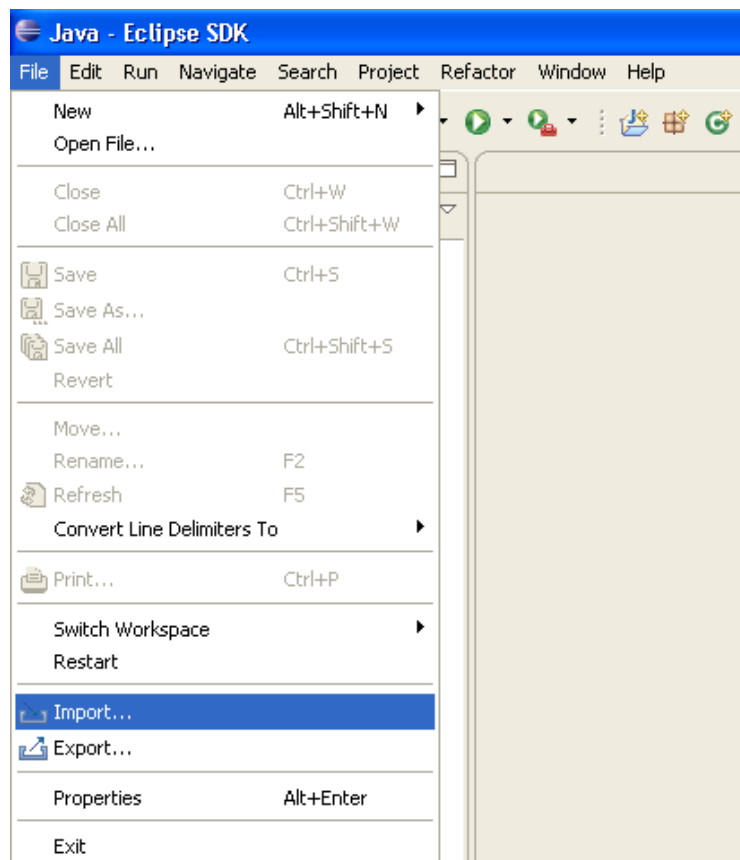
b. עיקבו אחרי הוראות העבודה במסמך Developing SWT applications using Eclipse

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

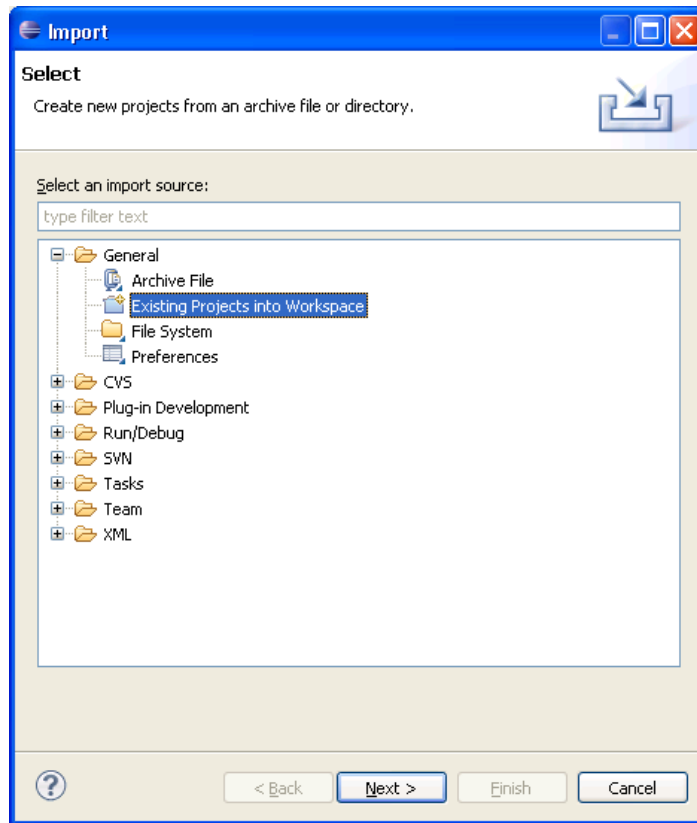
* אם יש בעיית מקום (או quota) ניתן למחוק את קובץ ה- zip שהורדתם לאחר סיום ההוראות לעיל.

2. הורידו את החבילה sudoku מאתר הקורס ושימרו אותה באופן מקומי. עיקבו אחר הפעולות הבאות:

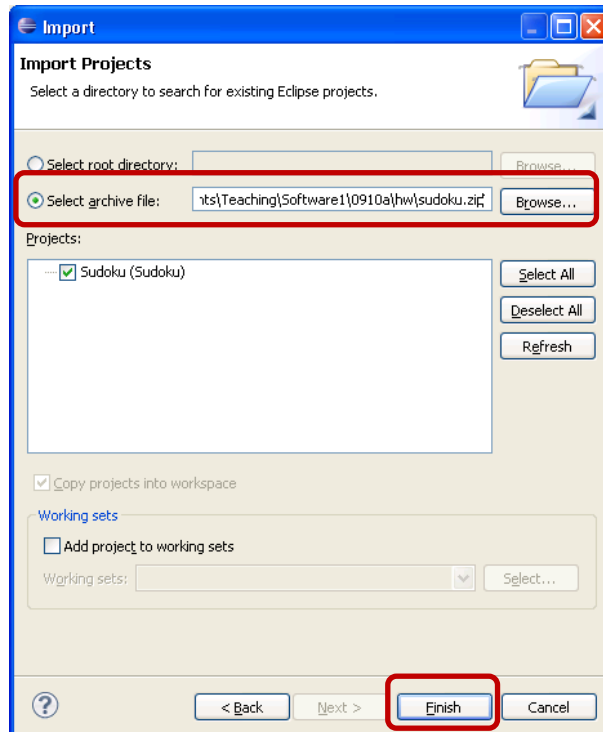
a. יבאו את הפרוקט לאליפס



b. ביחרו באפשרות "Existing projects..."



c. מצאו את קובץ הזיפ ולחצו על Finish



חלק ד':

ממשו את השרות `maxRun` אשר בהינתן מחרוזת (`string`) מחזיר את הריצה (`run`) הארוכה ביותר במחרוזת. **ריצה** מוגדרת בתור מספר הפעמים שבו מופיע אותו התו ברצף (השרות מחזיר את אורך הרצף הארוך ביותר במחרוזת).

להלן כמה דוגמאות:

`maxRun("hoopla") → 2`

`maxRun("$") → 1`

`maxRun("abbCCCddBBBxx") → 3`

`maxRun("") → 0`

ניתן להגדיר מבני עזר או שרותים חדשים לצורך המימוש.

```
public static int maxRun(String str) {  
    // your code..  
}
```