

# תוכנה 1 – חורף תשע"ו

## תרגיל מספר 4

### הנחיות כלליות:

- קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
  - יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש stav יקרא הקובץ stav\_hw4.zip). קובץ ה-zip יכיל:
    - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
    - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש.

**הגשת מחלקה עם חבילות:** יש לכווץ בתוך קובץ ה-zip שאתם מגישים את כל היררכיית התיקיות מתחת ל-src באקליפס. למשל, כדי להגיש את המחלקה sw1.pac.MyClass העתיקו את התיקיה sw1 שמתחת ל-src כולל כל מה שבתוכה לתוך קובץ ה-zip.

הערה: התרגיל מנוסח בלשון נקבה ומיועד לסטודנטים משני המינים.

### הקדמה: בדיקות קלט

- החל מתרגיל זה ואילך, יהיה עליכם לבצע בדיקות של תקינות הקלט המתקבל לתכנית. למשל, האם מס' הארגומנטים תקין, האם ערכי הארגומנטים תקינים וכו'. עם זאת:
- אם מצוין בסעיף כלשהו כי ניתן להניח משהו על הקלט, אין צורך לבדוק זאת בקוד של אותו סעיף (הנחת קדם).
  - בפונקציות בהן מתקבל כקלט מסלול לקובץ, אין צורך לבדוק את חוקיות המסלול או אם הוא מצביע לקובץ קיים.
  - אין צורך לבדוק את מה שנאסף כבר ע"י הקומפיילר, למשל את טיפוס הארגומנטים למתודה.

כדי להודיע למשתמשת על ארגומנט לא תקין לתכנית (משורת הפקודה), נשתמש בפקודת throw באופן הבא:

```
throw new Exception("[ERROR] " + message);
```

כאשר המחרוזת [Error] תשמש אותנו להבדיל בין שגיאות שנזרקו ע"י הקוד שלכם לבין שגיאות שנזרקו ע"י מחלקה מהספרייה הסטנדרטית. המחרוזת message תכיל מסר קריא המסביר את הבעיה. השגיאה תיזרק ותגרום לסיום התכנית.

### משחק מילים אינטרקטיבי

בתרגיל זה נממש משחק אינטרקטיבי המורכב משני שלבים:

1. שלב ההגדרות שבו המשתמשת יוצרת מילה שחלק מהאותיות שלה מוסתרות.
2. שלב המשחק שבו על המשתמשת לנחש את האותיות המוסתרות עד לחשיפת המילה בשלמותה.

שימו לב, בתוכנית זאת עליכם להשתמש **במערכים בלבד**, ולא במבני נתונים גנריים כמו Lists/Maps/Sets וכו'. שימוש במבני נתונים גנריים יכול לגרום להורדת ניקוד משמעותית עד כדי ציון נכשל.

כל הקוד שלכם יכתב במחלקה `il.ac.tau.cs.sw1.ex4.WordPuzzle`, שלד המחלקה מצורף לתרגיל הבית.

**הערה:** בשלד המחלקה מופיעות הגדרות קבועים ומימושים של פונקציות המייצרות את הפלט של התוכנית. השתמשו בהם בקוד שלכם, זה יעזור לכם לשמור על פורמט פלט תקין וימנע טעויות בהגדרות קבועים.

(1) **[10 נק']** בשלב הראשון, נוסף למחלקה שירות לטעינת אוצר המילים שלנו מקובץ ע"י מימוש מתודה לקריאת מילים באמצעות `Scanner`. בתוך המחלקה הגדירו את השירות:  
`public static String[] scanVocabulary(Scanner scanner)`

השירות יקבל כקלט עצם מסוג `java.util.Scanner` ויניח שהוא כבר מאותחל לקריאה ממקור כלשהו (למשל, קובץ). עליכם לקרוא את המילים בעזרת ה-`Scanner` ולהחזיר מערך עם כל המילים שנקראו **ממיונות לקסיקוגרפית ולא חזרות**.

- נגדיר "מילה" כרצף של תווים שמופרד ע"י רווחים לבנים (whitespaces) מהרצפים האחרים. ניתן להניח שמפריד ברירת המחדל של `Scanner` הוא לפי רווחים לבנים.
- כל מילה שאינה מחרוזת ריקה ("") תחשב למילה חוקית.
- את כל המילים יש להמיר ל `lowercase`.
- יש להחזיר רק את 3000 המילים ה"חוקיות" השונות הראשונות. בפרט, גודל המערך המוחזר לא יעלה על 3000.
- אם בסיום הקריאה מצאתם פחות מ-3000 מילים שונות, גודל המערך המוחזר צריך להיות בהתאם. למשל, אם קראתם 2463 מילים, החזירו מערך בגודל 2463 שמכיל אותן.
- **אין לסגור את ה-Scanner בתוך המתודה**.
- ניתן להיעזר בשירותים של המחלקה `java.util.Arrays`.

(2) **[5 נק']** נוסף את השירות `isInVocabulary` אשר מקבל מערך מחרוזות המייצג את האוצר המילים וכן מחרוזת נוספת, ובודק אם מחרוזת זו קיימת באוצר המילים. השירות יחזיר ערך בוליאני. ממשו את המתודה `isInVocabulary` על פי החתימה הבאה:

```
public static boolean isInVocabulary(String[] vocabulary, String word)
```

(3) **[10 נק']** יצירת חידה (מילה עם אותיות מוסתרות) מתבצעת באופן הבא: תחילה, נבחר מילה אותה נרצה להצפין. מילה זו אמורה להופיע באוצר המילים אותו יצרנו ע"י הפונקציה מסעיף (1).

לאחר מכן, נייצר מחרוזת המייצגת תבנית הסתרה. אורכה של התבנית יהיה זהה לאורך המילה אותה בחרנו להצפין, והתווים שלה יהיו אחד מהשניים: `*` ו `_` (כוכבית ומקף תחתון). אם באינדקס `i` של התבנית מופיע התו `*`, זה אומר שהאות באינדקס `i` במילה אותה בחרנו תופיע כמו שהיא. אחרת, אם באינדקס `i` מופיע התו `_`, זה אומר שאות זו תהיה מוסתרת.

לדוגמה, עבור המילה `guitar` והתבנית `**_**_*` נקבל את החידה `g_i_t_ar` (האותיות `g, i, t` מוסתרות) תבנית היא תבנית חוקית עבור מילה `word` אם היא מקיימת את ארבעת התנאים הבאים:

- אורכה זהה לאורך המילה `word`.
- התבנית מסתירה לפחות אות אחת.
- התבנית מכילה רק את התווים `*` ו `_`.
- אם אות כלשהי מופיעה יותר מפעם אחת במילה `word`, כל המופעים שלה יהיו או גלויים או חשופים (כלומר, עבור המילה `wheeps` והתבנית `**_***`, התבנית אינה חוקית כיוון שאחד המופעים של התו `e` הוא מוסתר, והשני גלוי).

ממשו את השירות `checkPattern` אשר מקבל מילה `word` ותבנית `pattern` ומחזיר ערך בוליאני: `true` אם התבנית חוקית עבור `word`, ו `false` אחרת. חתימת השירות:

```
public static boolean checkPattern(String word, String pattern)
```

(4) [5 נק'] נרצה להוסיף שירות אשר סופר את מספר התווים המוסתרים בתבנית `pattern` כלשהי, כלומר, מספר הפעמים שהתו `_` מופיע בתבנית.

ממשו את המתודה `countBlanksInPattern`

```
public static int countBlanksInPattern(String pattern)
```

(5) [10 נק'] כעת, נרצה לייצר חידה מתוך המילה והתבנית שהגדרנו. החידה מיוצגת באמצעות מערך של תווים (`char`). במקום ה `i` במערך יופיע התו `i` במילה `word` אם הוא גלוי בתבנית `pattern`, ואחרת יופיע הסימן `_`. שימו לב: הפונקציה יכולה להניח שהתבנית `pattern` היא חוקית עבור המילה `word`, כך שאין צורך לבצע קריאה ל `checkPattern` בתוך המימוש. פירושו של דבר הוא שמי שמשתמשת בפונקציה `createPuzzle` צריכה לבצע בדיקת חוקיות על התבנית והמילה לפני הקריאה לפונקציה, כיוון שפונקציה זו מצפה לקבל קלט חוקי.

לדוגמא, עבור המילה `guitar` והתבנית `**_**` נקבל את המערך `[_, u, _, _, a, r]`

ממשו את המתודה `createPuzzle` על פי החתימה הבאה:

```
public static char[] createPuzzle(String word, String pattern)
```

(6) [10 נק'] בדיקה נוספת שנרצה לערוך היא אם קיים פתרון יחיד עבור תבנית מסוימת (`pattern`), החידה שיצרנו ממנה (`puzzle`), ואוצר המילים שלנו (`vocabulary`). לדוגמא, אם באוצר המילים שלנו מופיעות המילים `at` ו `am` והתבנית שבחרנו היא `*_*`, הרי שהחידה שתיווצר היא `[a, _]`, ולה קיימים שני פתרונות אפשריים באוצר המילים `vocabulary`. אם הפתרון הוא יחיד, התוכנית תחזיר את הערך `true`, ואחרת, תחזיר `false`.

ממשו את המתודה `hasUniqueSolution` על פי החתימה הבאה:

```
public static boolean hasUniqueSolution(String pattern, char[] puzzle, String[] vocabulary)
```

(7) [10 נק'] נוסיף שירות אשר מקבל תו (`guess`), מילה (`word`) והחידה שנוצרה ממילה זו (`puzzle`). במידה וישנם מופעים מוסתרים של התו `guess` בחידה `puzzle`, נחשוף את המופעים הללו ע"י עדכון המקומות המתאימים `puzzle` מהתו `'_'` לתו `guess`. הפונקציה תחזיר את מספר המקומות שעודכנו.

למשל, עבור החידה `[w, _, _, _, p]` שנוצרה מהמילה `wheep`, והתו `e`, הפונקציה תעדכן את החידה ל `[w, _, e, e, p]` ותחזיר את הערך 2. אם עבור אותן חידה ומילה היינו מעבירים את התו `f`, האובייקט `puzzle` לא היה משתנה והפונקציה הייתה מחזירה את הערך 0.

ממשו את המתודה `applyGuess` על פי החתימה הבאה:

```
public static int applyGuess(char guess, String word, char[] puzzle)
```

הנחות:

ניתן להניח ש `guess` יהיה תמיד אות חוקית בא"ב האנגלית ב `lowercase`.

(8) [40 נק'] כעת, נוסף למחלקה WordPuzzle מתודת main שתפעיל את המשחק האינטרקטיבי באמצעות קבלת קלט מהconsole (כלומר, `System.in`). התכנית תקבל כארגומנט משורת הפקודה את המסלול לקובץ אוצר המילים, תפעיל את השירות `scanVocabulary` בכדי לקבל את אוצר המילים, ותדפיס את מס' המילים שנקראו בפורמט: "Read DDD words from SSS", כאשר DDD הוא מספר המילים באוצר המילים שנוצר ו SSS הוא שם הקובץ כפי שהועבר ברשימת הארגומנטים. במידה וחסר ארגומנט או שהקובץ אינו תקין, יש להדפיס הודעת שגיאה לבחירתכם (בפורמט המצוין בעמוד הראשון) ולצאת מהתוכנית.

לאחר מכן האינטראקציה עם המשתמשת תתנהל באופן הבא (ראו מצגת תרגול 4 לגבי יצירת Scanner וחיבורו ל-`System.in`):

```

i. שלב ההגדרות:
    a. התוכנית תדפיס את השורה --- Settings stage ---
    b. לאחר מכן, המשתמשת תתבקש להזין את המילה אותה ממנה תרצה לייצר חידה.
    התוכנית תדפיס את ההודעה: Enter your word: ותמתין לקלט מהמשתמשת.
        i. במידה והמילה אותה הזינה המשתמשת מופיעה באוצר המילים, התוכנית תעבור לשלב c.
        ii. אחרת, התוכנית תדפיס את ההודעה Illegal word! ותחזיר לשלב b.
        c. התוכנית תדפיס את ההודעה: Enter your pattern: ותמתין לקלט מהמשתמשת.
        i. במידה והתבנית אותה הזינה המשתמשת היא חוקית עבור המילה וגם מייצרת חידה בעלת פתרון יחיד, התוכנית תעבור לשלב המשחק.
        ii. אחרת, התוכנית תדפיס את ההודעה Illegal pattern! ותחזור לשלב c.
    ii. שלב המשחק:
        a. התוכנית תדפיס את השורה: --- Game stage ---
        b. כל משחק מתחיל עם מספר נסיונות השווה למספר התוים החסרים בתבנית + 3. כלומר, אם התבנית מסתירה 4 מקומות, למשתמשת יהיו סה"כ 7 נסיונות.
        c. התוכנית מדפיסה את החידה, ולאחר מכן, בשורה חדשה, את ההודעה:
            "Enter your guess:" וממתינה לקלט מהמשתמשת. ניתן להניח שהמשתמשת תזין תו אחד ואחריו תקיש על Enter.
        d. התוכנית בודקת אם האות שהוזנה מופיעה כאות מוסתרת במילה.
            i. אם אות זו מופיעה כאות מוסתרת במילה:
                1. התוכנית מדפיסה את ההודעה הבאה (XXX - מס' הניחושים שנותרו)
                Correct Guess, xxx guesses left
                2. התוכנית בודקת אם כל החידה פוענחה (כל האותיות גלויות). אם כן, תודפס ההודעה
                Congratulations! You solved the puzzle
            ii. אחרת, (האות לא מופיעה או שהיא מופיעה וגלויה), מודפסת ההודעה:
                Wrong Guess, xxx guesses left
        e. במידה ונשארו עוד ניחושים (כל ניחוש מוריד 1 ממספר הנסיונות), התוכנית חוזרת לשלב c. אחרת, מודפסת ההודעה Game over! והתוכנית מסתיימת.

```

לנוחותכם, מצורף שלד המחלקה בו תוכלו להשלים את המימוש שלכם. השלד כולל מתודות אשר מבצעות את כל ההדפסות הנדרשות בתרגיל. מומלץ להשתמש בהן על מנת לוודא שאתם שומרים על הפורמט הנדרש, אך אין זה חובה.

הנחות נוספות:

1. בשלב ההגדרות, המשתמשת יכול להזין כל קלט עבור מילה/תבנית.
2. בשלב המשחק, ניתן להניח שהקלט מהמשתמשת הוא חוקי, כלומר, בכל תור המשתמשת תזין אות אנגלית אחת ב lowercase.

## טסטר:

לתרגיל זה מצורפת מחלקת טסטר בשם WordPuzzleTester. מחלקה זו מיועדת לרוץ מאותו ה package של המחלקה WordPuzzle אותה אתם מגישים. הריצו את הטסטר לאחר סיום המימוש, במידה וכל הבדיקות עברו, פלט הריצה של הטסטר יהיה "done!" בלבד, אחרת יודפס מספר השגיאה.

מחלקה זו תשמש גם לבדיקת נכונות החתימות של המתודות שתממשו (אם המחלקה לא מתקמפלת, חסרה מתודה או שחתימת אחת המתודות לא נכונה). וגם לבדיקה שטחית של נכונות המימוש. אל תסתפקו בבדיקות שמופיעות בטסטר. הוסיפו בדיקות משלכם על מנת לוודא שהקוד עובד באופן תקין לכל קלט. אין צורך להגיש את הטסטר שלכם.

להלן אינטראקציה לדוגמא המדגימה את ההדפסות בכל שלב. שימו לב לנוסחים של ההודעות שהתוכנית מדפיסה.

קובץ הקלט בדוגמא למטה הוא vocabulary.txt וניתן להניח שבהרצה זו הקובץ נמצא בתיקיה resources/hw4. קלט מהמשתמשת מסומן בכחול. בתיקית קבצי התרגיל תוכלו למצוא פלט ריצה נוסף, בקובץ output1.out.

```
Read 15 words from resources/hw4/vocabulary.txt
--- Settings stage ---
Enter your word:
while
Enter your pattern:
_____
Illegal pattern!
Enter your pattern:
* * *
  _ _ _
--- Game stage ---
w_i_e
Enter your guess:
e
Wrong Guess, 4 guesses left
w_i_e
Enter your guess:
g
Wrong Guess, 3 guesses left
w_i_e
Enter your guess:
h
Correct Guess, 2 guesses left
whi_e
Enter your guess:
t
Wrong Guess, 1 guesses left
whi_e
Enter your guess:
x
Correct Guess, 0 guesses left
Congratulations! You solved the puzzle
```

(כאשר אתם מריצים את התכנית שלכם לצרכי בדיקה, עליכם להעביר כארגומנטים כתובות של קבצים השמורים על המחשב שלכם).

**בהצלחה!**