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 aviv יקרא הקובץ aviv_hw2.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.

הקדמה: בדיקות קלט

החל מתרגיל זה ואילך, יהא 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 שנאכף כבר ע"י הקומפיילר, למשל את טיפוסי הארגומנטים למתודה.

כדי להודיע למשתמש 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 מהספריה הסטנדרטית. המחרוזת message תכיל מסר קריא המסביר את הבעיה. השגיאה תיזרק **ותגרום לסיום התכנית**.

כדי להודיע למשתמש על קלט לא תקין מה-console, נדפיס הודעת אזהרה למשתמש באופן הבא:

```
System.out.println("[WARNING] " + message);
```

הודעה זו תסביר למשתמש כיצד עליו לתקן את הקלט בפעם הבאה.

תוכנית לתיקון שגיאות בטקסט

בתרגיל זה נממש תוכנית אינטרקטיבית המתקנת שגיאות כתיב בטקסט, על סמך אוצר מילים – רשימת מילים חוקיות בשפה- אשר יטען מתוך קובץ.

שימו לב, בתוכנית זאת עליכם להשתמש **במערכים בלבד**, ולא במבני נתונים גנריים כמו Lists/Maps וכו'.

צרו את המחלקה il.ac.tau.cs.sw1.ex4.SpellingCorrector

חשוב: בתרגיל זה תבצעו פעולות השוואה על מחרוזות. הקפידו להשתמש ב equals על מנת להשוות בין שתי מחרוזות, ולא ב ==.

(1) **[10 נק']** בשלב הראשון, נוסיף 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 **ממוינות לקסיקוגרפית** ו**ללא 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.
- **<u>אין לסגור את ה-Scanner בתוך המתודה</u>**.
- ניתן להיעזר בשירותים של המחלקה java.util.Arrays.

(2) **[15 נק']** נוסיף למחלקה CorrectSpelling שירות המבצע השוואה בין שתי מילים ומחזיר ערך מספרי המייצג את המרחק ביניהן. ככל המילים דומות יותר, כך המרחק ביניהן יהיה קטן יותר. בפרט, המרחק בין שתי מילים זהות הוא 0. מרחק זה נקרא מרחק המינג.
(https://en.wikipedia.org/wiki/Hamming_distance)

חישוב המרחק יתבצע על ידי השוואת שתי המילים תו-תו, כאשר שתיהן מיושרות לצד שמאל (כלומר, לתחילת המילה). נשווה תחילה את התו הראשון של שתיהן, לאחר מכן את השני וכן הלאה. אם מצאנו שני תוים שונים, נגדיל את המרחק בין שתיהן ב 1. אם אורכן של שתי המילים שונה, אזי הפרש האורכים גם הוא נסכם יחד עם שאר המרחק. לדוגמא, ההפרש בין המילים "abc" ו "aabc" הוא 3, שכן, רק שתי האותיות הראשונות של המילים זהות, ושאר האותיות שונות. עבור "abba" ו "aaba" המרחק הוא 1, שכן רק האות השניה שונה בין שתי המילים.

הוסיפו ל- SpellingCorrector את המתודה calcHammingDistance בעלת החתימה הבאה:

```
public static int calcHammingDistance(String word1, String word2)
```

המתודה תקבל שתי מילים ותחזיר את המרחק ביניהן.

(3) **[15 נק']** נוסיף שירות, אשר עבור מילה מסויימת word ואוצר מילים מסויים, המיוצג ע"י המערך vocabulary, מייצרת את רשימת כל המילים הקרובות ל word מתוך vocabulary, עד למרחק של 2 בין מילים.
השירות יחזיר מערך אשר מכיל שלושה מערכים. המערך הראשון יכיל את כל המילים במרחק 0 מ word. למעשה, למערך זה יש שתי אופציות: או שיהיה ריק, או שיכיל את המילה word עצמה, במידה והיא נמצאת ב vocabulary. המערך השני יכיל את כל המילים במרחק 1 מ word, וכך הלאה, עד מרחק 2.
שימו לב, סדר המילים במערכים שמתקבלים יהיה לפי סדר הופעתן במילון, כלומר, סדר לקסיקוגרפי.
אורכי המערכים יהיו כמספר המילים במרחק המתאים (כלומר, לא אורך קבוע).

ממשו את המתודה findSimilarWords על פי החתימה הבאה:

```
public static String[][] findSimilarWords(String word, String[] vocabulary)
```

(4) **[10 נק']** נוסיף שירות המקבל משפט, חותך אותו למילים ע"פ הרווחים במשפט ומחזיר את רשימת המילים במשפט (לפי סדר הופעתן) בתוך מערך. במידה ויש יותר מרווח אחד בין שתי מילים, שימוש ב slipt ייצר מחרוזות ריקות, ובמקרה הזה עליכם להתעלם מהן ולא להחזיר אותן במערך, על מנת שלא לנסות ולתקן אותן. את המשפט יש להמיר ל lowercase. אורך המערך שיחזור יהיה כמספר המילים במשפט.

ממשו את המתודה המתודה splitSentence על פי החתימה הבאה:

```
public static String[] splitSentence(String sentence)
```

לדוגמא, אם נריץ את המתודה splitSentence על המשפט "love love me do" נקבל מערך באורך 4 עם המחרוזות הבאות: [love, love, me, do]
עבור המשפט "love love me do" נקבל את אותו המערך, כיוון שאנחנו מתעלמים מרווחים.

(5) **[5 נקודות]** נוסיף את השירות buildSentence אשר מקבל מערך של מילים, בונה מהן משפט לפי הסדר ומחזיר את המחרוזת המתקבלת. כל שתי מילים במשפט יופרדו על ידי רווח בודד. ניתן להניח שברשימת המילים words אין מילה ריקה. העזרו ב trim() במידה ואתם רוצים להוריד רווחים בסוף המילה שנוצרת.

```
public static String buildSentence(String[] words)
```

לדוגמא, עבור המערך [love, love, me, do] נקבל את המחרוזת "love love me do"

(6) **[5 נק']** נוסיף את השירות isInVocabulary אשר מקבל מערך מחרוזות המייצגת את האוצר המילים וכן מחרוזת, ובודק אם המילה קיימת באוצר המילים. השירות יחזיר ערך בוליאני.
ממשו את המתודה isInVocabulary על פי החתימה הבאה:

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

(7) **[40 נק']** כעת, נוסיף למחלקה SpellingCorrector מתודת main המבצעת תיקון אוטומטי של משפטים הנקראים מה-console (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 תתנהל אינטראקציה עם המשתמש (ראו מצגת תרגול 4 לגבי יצירת Scanner וחיבורו ל-System.*in*) באופן הבא:

i. התוכנית תבקש מהמשתמש להזין משפט קלט (המשפט מסתיים בירידת שורה).

ii. התוכנית תחלק את המשפט למילים ע"י שימוש ב splitSentence
iii. התכנית תבצע בדיקת איות לכל מילה ע"י שימוש בשירות findSimilarWords באופן הבא:
   a. במידה והמילה חוקית, כלומר, קיימת ב vocabulary, התוכנית ממשיכה למילה הבאה.
   b. במידה והמילה היא לא חוקית, התוכנית מדפיסה את השורות הבאות:

```
The word (word) is incorrect:
(x) corrections of distance 1
(y) corrections of distance 2
```

כאשר (word), (x), (y) מייצגים את המילה, מספר התיקונים במרחק 1 ומספר התיקונים באורך 2 (בהתאמה)

   c. לאחר מכן, התוכנית תדפיס את 8 הצעות התיקון הראשונות, החל מהתיקונים באורך 1. סדר התיקונים יהיה לפי סדר הרשימות המתקבלות מ findSimilarWords . במידה ונגמרו התיקונים באורך 1, התוכנית תעבור תיקונים באורך 2 (לפי הסדר) התיקונים ימוספרו מ 1 עד 8 (ראה דוגמת הרצה בסוף התרגיל). לאחר מכן תודפס השורה:

```
Enter your choice:
```

   d. בשלב זה המשתמש מתבקש להזין את בחירתו – מספר בין 1 – 8.
   e. עליכם להתייחס למקרה שבו יש פחות מ 8 תיקונים אפשריים, למשל, סה"כ קיבלנו 3 תיקונים במרחק 1 ו 4 תיקונים במרחק 2. במקרה זה, יודפסו רק 7 אופציות לתיקון והאופציה 8 לא תהיה בחירה חוקית.
   f. לאחר מכן, התוכנית תעבור למילה הבאה ותבצע את אותו התהליך.
iv. לאחר סיום המעבר על כל המשפט, התוכנית תדפיס את המשפט המתוקן וכן את מספר המילים שתוקנו (ראו פלט ריצת התוכנית). השתמשו בשירות buildSentence בשביל לייצר את המשפט.
v. בכל שלב בתוכנית, כאשר המשתמש יקליד את הפקודה "quit", התוכנית תסתיים. זכרו לסגור בסוף התכנית את ה-Scanner –ים שפתחתם.

לנוחותכם, מצורף שלד למחלקה 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 זה לא חובה.

טסטר:

לתרגיל זה מצורפת מחלקת טסטר בשם SpellingCorrectorTest. מחלקה זו מיועדת לרוץ מאותו ה package של המחלקה SpellingCorrector אותה אתם מגישים. המחלקה תתקמפל רק לאחר שתממשו את כל המתודות הנדרשות בתרגיל. הריצו את הטסטר לאחר סיום המימוש - במידה וכל הבדיקות עברו, פלט הריצה של הטסטר יהיה "done!" בלבד, אחרת תודפס 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. וודאו שהודעות אלה משמעותיות ומסבירות את הבעיה למשתמש.
קובץ הקלט בדוגמא למטה הוא blackbird.txt וניתן להניח שבהרצה זו הקובץ נמצא בתקיית ההרצה. קלט מהמשתמש מסומן בכחול.

```
Read 43 words from blackbird.txt
Enter your sentence:
Blackbird singing ir the dead of nighe
the word ir is incorrect
6 corrections of distance 1
5 corrections of distance 2
1. in
2. is
3. if
4. it
5. or
6. ip
7. of
8. to
Enter your choice:
2
the word nighe is incorrect
1 corrections of distance 1
1 corrections of distance 2
1. night
2. light
Enter your choice:
1
The correct sentence is: blackbird singing is the dead of night
Corrected 2 words.
Enter your sentence:
quit
```

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

בהצלחה!