

# תוכנה 1 – אביב תשע"ג

## תרגיל מספר 9

### סגנון קידוד, מבני נתונים גנריים

#### 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 zvainer יקרא הקובץ zvainer\_hw9.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.
  - ג. קובץ טקסט בשם answers.txt/doc המכיל תשובות לשאלות המילוליות.

### חלק א' (30%): שיפור סגנון קידוד (Java Coding Style)

בשאלה זו נתון לכם קובץ קוד בשם `TextAnalyzer.java` המממש מחלקה לניתוח טקסט. עליכם לשכתב את המחלקה תוך תיקון סגנון הקידוד כך שיעמוד במוסכמות הסגנון, ואף יהיה קריא, מובן, ויעיל ככל האפשר.

המחלקה הנתונה מקבלת שם קובץ כארגומנט בשורה הפקודה, ומבקשת מהמשתמש 2 שמות קבצים שישמשו לפלט. לקובץ הראשון תדפיס את כל המילים שהופיעו בקובץ הקלט ואת מספר המופעים שלהן. לקובץ השני תדפיס התוכנית את כל המילים שהופיעו בקובץ הקלט, כשהן ממוינות בסדר לקסיקוגרפי.

ניתן להניח שהמחלקה פועלת באופן תקין, למרות סגנון הקידוד הלקוי.

עליכם לשפץ את הקוד בהסתמך על המוסכמות לקידוד ג'אווה כפי שהן מופיעות בקישור הבא:  
<http://www.oracle.com/technetwork/java/codeconv-138413.html>

שימו לב במיוחד לנקודות הבאות (שעלו מניתוח מדגמי של תרגילים שהגישו סטודנטים בקורס):

- שמות משמעותיים למחלקות, משתנים ומתודות
  - שם המשתנה אמור לציין את תפקידו ואת התוכן שלו. לא ראשי תיבות, לא מילים בעברית, לא אותיות בודדות (אלא אם כן זה משתנה אינדקס למשל בלולאת For ואז מקובל להשתמש ב-i,j,k)
  - שמות של משתנים בולאניים יהיו מהצורה `isVisible`, `isValid`
  - שמות מחלקות יתחילו באות גדולה, שמות משתנים ומתודות באות קטנה, ולאחר מכן תופיע אות גדולה בתחילת כל מילה (למשל `studentAverage`).
  - שמות קבועים יהיו מהצורה `COURSE_ID`.
  - שם המתודה יהיה לרוב פועל, ויצין את הפונקציה אותה היא מבצעת.
- יש להשתמש בקבועים לאחסון קבועים או ערכים בהם יש שימוש חוזר.
- מניעת שכפול קוד – ארגון הקוד במתודות כאשר כל מתודה מבצעת פונקציה מוגדרת. אין לכתוב את כל הקוד במתודה ה-main.
- טווח ההכרה של המשתנים אמור להיות מינימלי – משתנים אמורים להיות כמה שיותר מקומיים.

- אינדנטציה (הזחה)
- תיעוד – יש לכתוב מעל כל מתודה, משתנה, בלוק קוד מה הם עושים, פירוט לפי רמת המורכבות, עם זאת אין לציין את המובן מאליו.
- הרשאות נראות (private, public) של שדות צריכות להיות מינימליות, וגישה פומבית לשדות (במידה והיא נדרשת) פרטיים צריכה להיעשות ע"י Getters ו-Setters.
- בכל שורה תופיע פקודה אחת. בכל שורה יופיע רק סוגר מסולסל יחיד.
- עדיף להפריד חישובים ולוגיקה מפעולות קלט/פלט.

### בשאלה זו עליכם להגיש:

1. קובץ קוד מתוקן עבור המחלקה הכתוב על פי הכללים המפורטים בקישור הנ"ל. על הקובץ להיות בעל סגנון קידוד מושלם.
2. הסבירו בקצרה מהם 5 התיקונים השונים המשמעותיים ביותר שביצעתם ומדוע הם חשובים (להגיש בקובץ התשובות).

## **חלק ב' (70%): מנוע חיפוש (מבני נתונים גנריים)**

בחלק זה נתרגל עבודה עם אוספים (Collections) ע"י מימוש מנוע חיפוש פשוט. בין קבצי העזר של התרגיל תוכלו למצוא את המחלקות HTMLTokenizer, SimpleSearchEngine ואת המנשק WordIndex שכבר נכתבו עבורכם.

מנוע החיפוש שלנו יטפל במספר מצומצם של דפי HTML אותם הוא יקרא מהרשת. דפים אלו מוגדרים מראש ורשימתם נמצאת במחלקה SimpleSearchEngine. אם תרצו תוכלו להוסיף או לשנות את הדפים בהם אתם משתמשים.

לאחר שהורדנו דף HTML מהרשת נתייחס רק לחלק הטקסט שבדף ונפרק חלק זה למילים בודדות. קוד זה כבר מומש עבורכם במחלקה HTMLTokenizer.

בנוסף המחלקה SimpleSearchEngine שבה הקוד המפעיל את המערכת ומתקשר עם המשתמש קיימת אף היא. הקוד במחלקה זו קורא בתחילה למתודה index אשר תאכלס את אינדקס המילים. לאחר מכן, כתלות בקלט המשתמש יתבצע חיפוש של דפים לפי מילת שאילתה, או חיפוש של דפים לפי דמיון לכתובת דף נתון (במקרה זה יש להכניס את כתובת דף השאילתה לאחר הסימן ~, ראו דוגמא בהמשך).

### **מה עליכם לעשות:**

נרצה ליצור אינדקס של כל המילים שהופיעו בכל הדפים שהורדנו מהרשת. קיומו של אינדקס זה יאפשר לנו מאוחר יותר לבצע חיפושים עבור מילה מסוימת. עליכם לממש מחלקה בשם MyWordIndex המממשת את המנשק WordIndex. מחלקה זו שומרת את אינדקס המילים, מאפשרת הוספת מילים לאינדקס ולאחר מכן מאפשרת גם חיפוש באינדקס.

```

import java.util.Collection;
import java.util.List;

/**
 * A word index allows for a quick search over a large collection of words. It
 * supports two services, the first is populating the index with words, the
 * second is searching for a word.
 */
public interface WordIndex {

    /**
     * Add the words originating in the specifies URL.
     *
     * @param words
     *         - collection of words to add to the index
     * @param strURL
     *         - the location of the page containing the words
     */
    void index(Collection<String> words, String strURL);

    /**
     * Search for pages containing a given word in the index
     *
     * @param word
     *         - the word to search
     * @return A list of pages containing the word. The pages are ordered
     *         according to the relative importance of the word within them.
     */
    List<String> search(String word);

    /**
     * Search for pages that are similar to a given URL in the index
     *
     * @param strURL
     *         - a URL for which similar web pages are to be found
     * @return A list of pages that are similar to the given URL based on
     contained words.
     *         The pages are ordered based on similarity to the given URL.
     */
    List<String> findSimilarPages(String strURL);
}

```

הערה: בקוד למעלה התייעוד כתוב בסגנון javadocs: התגית @param משמשת להסבר על פרמטר של המתודה, ו-@return משמשת להסבר על ערך החזרה. לא מדובר בחוזה!

## המתודות השונות:

- המתודה index**
  - מתודה זו אחראית על אכלוס מבנה הנתונים שלכם. המתודה מקבלת אוסף של מילים (ייתכנו חזרות) ואת כתובת האינטרנט (URL) של הדף מהן הגיעו. עליכם לבחור את מבני הנתונים שבהם תשתמשו ולדאוג לשמור על הקשרים הבאים:
    - לכל מילה באילו דפים היא מופיעה וכמה פעמים בכל דף
    - לכל דף כמה מילים בסה"כ מופיעות בו (עם חזרות)
    - לכל דף אילו מילים ייחודיות (שונות) מופיעות בו.
  - הערה: רשימת המילים בקלט היא כפי שהופיעה בדף המקורי, אולם יש לשמור גרסת lowercase של המילים.
- המתודה search**
  - מקבלת מילה לחיפוש ומחזירה רשימה ממוינת של כתובות אינטרנט בהן המילה מופיעה. נמיון את הרשימה כך שכלל שהמשקל היחסי של מילה בדף גבוה יותר כך הכתובת תופיע במקום גבוה יותר ברשימה.
  - המשקל היחסי של מילה בדף יהיה מספר המופעים של המילה בדף חלקי מספר המילים הכולל באותו דף. לא נחזיר דפים בהם המילה אינה מופיעה כלל.
- המתודה findSimilarPages**
  - מקבלת כתובת של דף אינטרנט (URL) ומחזירה רשימה של כתובות אינטרנט ממוינות לפי רמת הדמיון לדף הנתון. לצורך חישוב רמת הדמיון בין שני דפי אינטרנט, נשתמש ב-Jaccard Index ([http://en.wikipedia.org/wiki/Jaccard\\_index](http://en.wikipedia.org/wiki/Jaccard_index)) המחושב כך: נחשב את קבוצת המילים הייחודיות בכל אחד מהדפים אותם נרצה להשוות, ואז נחלק את גודל חיתוך הקבוצות בגודל איחוד הקבוצות.

בנוסחה הבאה, A עשויה להיות קבוצת המילים הייחודיות של דף השאילתה, ו-B קבוצת המילים הייחודיות של דף אחר אליו נרצה לחשב רמת דמיון:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

שימו לב: בתוצאות החיפוש שיוחזרו ע"י המתודה יכללו רק דפים שחיתוך המילים המשותפות להם ולדף השאילתה מכיל לפחות 30 מילים (כדי לא להציג דפים שבהם יש חיתוך נמוך מאוד של מילים לדף השאילתה).

## הדרכה:

השתמשו במחלקות מ-Java Collections, בפרט המנשק Map והמחלקה HashMap יהיו שימושיים לאחסון המילים באינדקס. גם המנשקים List, Set והמחלקות ArrayList, HashSet עשויים להועיל לעיבוד הנתונים. שימו לב למתודות `addAll()` ו-`retainAll()` המאפשרות חישוב חיתוך ואיחוד של אוספים (בהתאמה).

את מיון הרשימה של כתובות האינטרנט בצעו בעזרת הפונקציה `Collections.sort(...)`. שימו לב שה"מיון הטבעי" של הכתובות (String) הוא מיון לקסיקוגרפי ולא כפי שמתבקש בשאלה. כדי לשנות את שיטת המיון עליכם לכתוב מחלקת עזר המממשת את המנשק Comparator (מומלץ לחפש באינטרנט דוגמאות למימוש מנשק זה). שימו לב שתוצאות ההשוואה תלויות במילת החיפוש הנוכחית.

הערה: המילים המתקבלות מדף ה-HTML אינן "נקיות" (כוללות סימני פיסוק וכדומה). אין צורך לבצע כל פעולה על מילים אלא להשתמש בהן כמו שהן.

כדי לבדוק את התכנית שלכם השתמשו במחלקה Main המייצרת אובייקט חדש מטיפוס SimpleSearchEngine ומעבירה לו בבנאי מופע של המחלקה MyWordIndex שמימשתם. לאחר יצירת האובייקט נקרא השירות run.

```
public class Main {
    public static void main(String[] args) {
        SimpleSearchEngine searchEngine =
            new SimpleSearchEngine(new MyWordIndex());
        searchEngine.run();
    }
}
```

שימו לב:

- נתון לכם הקוד של הקבצים הבאים: Main.java, SimpleSearchEngine.java, WordIndex.java, HTMLTokenizer
- עליכם לכתוב ולהגיש את המחלקה MyWordIndex שמממשת את המנשק WordIndex. ניתן להוסיף מחלקות עזר באותו קובץ או בקבצים נוספים.
- אין לשנות את הקבצים הנתונים.
- לקבלת מלוא הנקודות על התרגיל יש להשתמש במבני הנתונים המתאימים לבעיה ובאופן ראוי (לא מספיק שהפלט יהיה נכון).

**דוגמא (בהתבסס על רשימת הכתובות המקודדת בקוד שניתן לכם):**

פלט עבור חיפוש דפים על פי המילה "java":

```
> java
Searching for pages containing "java"...
1. http://www.java.com/en/
2. http://en.wikipedia.org/wiki/Java_(programming_language)
3. http://en.wikipedia.org/wiki/Java
4. http://www.oracle.com
```

פלט עבור חיפוש דפים הדומים לדף בכתובת "http://cnn.com" (שימו לב ששאלתת חיפוש דפים על פי דמיון תתחיל תמיד בתו ~ שאינו נכלל בשאלתת עצמה):

```
> ~http://cnn.com
Searching for pages similar to URL "http://cnn.com"...
1. http://bbc.com
2. http://www.space.com
3. http://haaretz.com
4. http://www.nasa.gov
5. http://www.ynetnews.com
6. http://www.oracle.com
7. http://en.wikipedia.org/wiki/Java
8. http://en.wikipedia.org/wiki/Java_(programming_language)
```

שימו לב שהתוצאות עשויות להשתנות כתלות בהתעדכנות האתרים שנסרקו. אתם מוזמנים להשוות את הפלט עם חברים.

בהצלחה!