

# תוכנה 1 – סתיו תשע"ג

## תרגיל מספר 9

### 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 אחד עם העתק של כל קבצי ה-java.

### חלק א' (50%): איטרטורים ומחלקות פנימיות

בחלק זה נתרגל עבודה עם המנשקים Iterator ו- Iterable וכתובת מחלקות פנימיות. ברצוננו לכתוב רב-אוספים (multi-collections) אשר ניתן להוסיף אליהם מס' אוספים כרצוננו (תתי-אוספים), ואז לעבור על איברי תתי-האוספים הללו בסדר מסוים.

נתונים לכם באתר הקורס שני מנשקים.

- .I `MultiIterator<T>` - מנשק היורש מן המנשק `Iterator` ומוסיף לו פונקציה אחת, `getNextIdx()`, אשר מחזירה את אינדקס תת-האוסף שממנו יילקח האיבר שיוחזר בקריאה הבאה ל-`next()`, אך אינה מקדמת את האיטרטור כמו `next()`.
- .II `MultiCollection<T>` - מנשק המייצג רב אוסף ומאפשר מעבר על איבריו. לשם כך, הוא יורש מ-`Iterable<T>` ומוסיף לו מס' פונקציות. שימו לב כי טיפוס האיטרטור המוחזר ע"י מחלקות המממשות את המנשק הנ"ל הוא תמיד מטיפוס `MultiIterator`.

עליכם לכתוב שלוש מחלקות אשר מממשות את המנשק `MultiCollection<T>`, כדלקמן:

1. `ConcatMultiCollection<T>` - מדמה שרשור של תתי-האוספים לפי סדר הוספתם. סדר האיברים המתקבל: כל איברי תת-האוסף הראשון לפי הסדר, אח"כ כל איברי תת-האוסף השני לפי הסדר וכך הלאה.
2. `RoundRobinMultiCollection<T>` - לוקח בכל פעם איבר אחד מכל תת-אוסף לפי סדר הוספתם, בצורה מעגלית. סדר האיברים המתקבל: כל האיברים הראשונים של כל תתי-האוספים, אח"כ כל האיברים השניים וכך הלאה.
3. `MajorityMultiCollection<T>` - לוקח בכל פעם את כל האיברים במקום ה-`i` מכל תתי-האוספים, ומחזיר את הערך הנפוץ ביותר מביניהם. אם יש מס' ערכים שמופיעים מס' פעמים שווה, יוחזר הערך הראשון מביניהם (לפי סדר הוספת תתי-האוספים). `getNextIdx()` של האיטרטור יחזיר את אינדקס תת-האוסף הראשון שבו נמצא הערך המוחזר (במקום ה-`i`). סדר האיברים המתקבל: האיבר הראשון הנפוץ ביותר, האיבר השני הנפוץ ביותר וכך הלאה.

**דוגמא**

נניח שאנו מוסיפים לרב-האוסף את תתי-האוספים הבאים לפי הסדר:

```
[a11, a12, a13]
[a21, a22, a23, a24, a25]
[a31, a32, a33]
```

הסדר שיתקבל ב- ConcatMultiCollection:

```
a11, a12, a13, a21, a22, a23, a24, a25, a31, a32, a33
```

הסדר שיתקבל ב- RoundRobinMultiCollection:

```
a11, a21, a31, a12, a22, a32, a13, a23, a33, a24, a25
```

הסדר שיתקבל ב- MajorityMultiCollection:

```
a11, a12, a13, a24, a25
```

**דוגמא נוספת**

נניח שאנחנו מוסיפים לרב האוסף את האוספים הבאים לפי הסדר:

```
[salad, noodles, ice-cream]
[soup, steak, apple]
[soup, pasta]
```

הסדר שיתקבל ב- ConcatMultiCollection:

```
salad, noodles, ice-cream, soup, steak, apple, soup, pasta
```

הסדר שיתקבל ב- RoundRobinMultiCollection:

```
salad, soup, soup, noodles, steak, pasta, ice-cream, apple
```

הסדר שיתקבל ב- MajorityMultiCollection:

```
soup, noodles, ice-cream
```

**הערות**

- **חשוב:** המעבר על איברי האוספים ייעשה בצורה "עצלה" (lazy): אנו ניגש לאיבר באחד מתתי-האוספים רק כשנידרש לו במהלך המעבר על רב-האוסף. בפרט, לא ניתן לצרף את איברי כל תתי-האוספים לאוסף אחד בעת הוספתם לרב-האוסף או בעת יצירת מופע של האיטרטור. גישה זו עשויה לחסוך חישובים לא נחוצים, והיא יעילה במיוחד במקרים בהם לא נרצה בהכרח לעבור על כל איברי רב-האוסף.
- מומלץ להשתמש באיטרטורים של תתי-האוספים לצורך מעבר על איברי רב-האוסף.
- ניתן להוסיף פונקציות עזר, מנשקים ומחלקות כרצונכם. בפרט מומלץ לחשוב כיצד לשתף קוד בין המחלקות השונות.
- הצעה לאופן פעולת האיטרטור: בעת יצירת האיטרטור חשבו מהו האיבר הראשון ושמרו אותו. בכל קריאה ל-()next החזירו את האיבר ששמרתם, חשבו ושימרו את האיבר שיבוא אחריו. במהלך חישוב האיבר שימרו את אינדקס תת האוסף ממנו הוא נלקח.
- הקפידו על קונבנציות כתיבת הקוד!

**קובץ בדיקה**

באתר הקורס מצורפת תכנית בדיקה בסיסית עבור המחלקות הנ"ל, SmallTestMultiCollections. עליכם לוודא שכל הבדיקות עבור הקוד שכתבתם עוברות בהצלחה. ניתן להשתמש בקוד זה כבסיס לכתיבת בדיקות נוספות.

**יש להגיש**

את שלושת הקבצים הנתונים לכם באתר הקורס (כולל תכנית הבדיקה), את שלוש המחלקות שנדרשתם לממש, ומחלקות\מנשקים נוספים, אם כתבתם כאלה כחלק פתרון התרגיל. שימו את הקבצים בחבילה הנתונה `il.ac.tau.cs.software1.mutli_collection`.

**חלק ב' (50%): 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 ההחזרה. 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.

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\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Java_(programming_language))
3. <http://en.wikipedia.org/wiki/Java>
4. <http://www.java.net>

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://haaretz.com>
3. <http://www.space.com>
4. <http://www.nasa.gov>
5. <http://www.java.net>
6. <http://www.ynetnews.com>
7. <http://www.oracle.com>
8. <http://en.wikipedia.org/wiki/Java>
9. [http://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Java_(programming_language))