

# תוכנה 1 – סתיו תשע"ה

## תרגיל מספר 8

### מבני נתונים מקושרים וגנריים

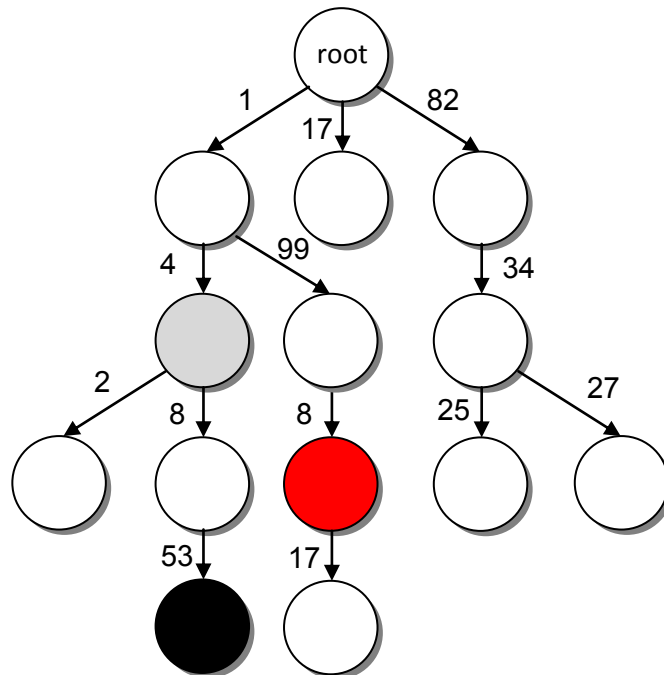
#### הנחיות כלליות:

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\_hw8.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, כולל תיקיות החבילה.

#### חלק א' - מבני נתונים מקושרים (50 נק')

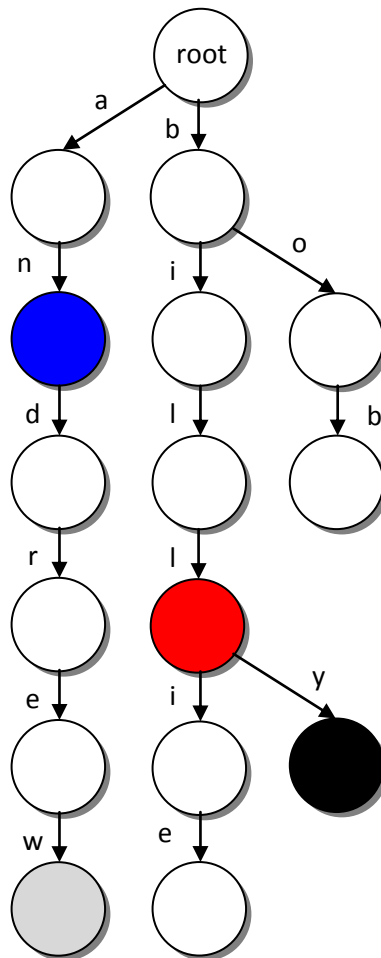
בחלק זה נתרגל עבודה עם מבני נתונים מקושרים, באמצעות מימוש עץ מסוג Trie. מבנה נתונים זה מאפשר לשמור ביעילות **רצפים** (אשר יכנו גם "מפתחות"), ולחפש אותם לפי התחיליות שלהם (prefix). כל רצף בעץ מיוצג ע"י מסלול מצומת השורש אל אחד הצמתים הפנימיים, כאשר כל קשת במסלול משוייכת לאיבר ברצף. לדוגמא, העץ בדוגמא הבאה מייצג Trie של רשימות של מספרים שלמים. המסלול לצומת הצבוע באדום מייצג את הרשימה {1,99,8}, והמסלול לצומת השחור מייצג את הרשימה {1,4,8,53}.



שימו לב, שמכל צומת יכולות לצאת לכל היותר 100 קשתות, עבור 100 המספרים בין 0-99. אם נרצה למצוא את כל הרצפים שמתחילים ב-{1,4}, ניקח את כל המסלולים העוברים דרך הצומת האפור.

בנוסף, מבנה ה-Trie שנממש יאפשר לנו לשמור, בצומת הקצה של מסלול שמייצג רצף, סט של ערכים המשייכים לרצף הזה. למשל, ניתן לשמור עבור כל רשימת מספרים את תאריכי ההוספה שלה. במקרה כזה, בצומת האדום נשמור את סט כל התאריכים בהם התבקשנו להוסיף לעץ את הרשימה {1,99,8} (מדובר בסט, כי ייתכן שזה קרה יותר מפעם אחת).

בתור דוגמה נוספת, נתבונן על Trie של מחרוזות. כאן כל איבר ברצף הוא תו במחרוזת. אם נניח שהמחרוזות שלנו מכילות רק את התווים a-z, מכל צומת ייצאו לכל היותר 26 קשתות. בדוגמה למטה, המסלול לצומת האדום מייצג את המחרוזת bill, המסלול לצומת השחור את המחרוזת billy, והמסלול לצומת האפור את המחרוזת andrew. מכיוון שהמחרוזות הללו מייצגות שמות פרטיים, הערך שנשמור עבור כל מחרוזת יכול להיות, למשל, מחרוזת אחרת שמייצגת את שם המשפחה. במקרה כזה, בצומת האדום נשמור את סט כל שמות המשפחה של אנשים בשם "bill".



באתר הקורס נתונים לכם שני מנשקים גנריים<sup>1</sup>:  $\text{TrieNode}\langle K, V \rangle$  ו- $\text{Trie}\langle K, V \rangle$  המייצגים צומת בעץ ואת העץ בהתאמה.  $K$  הוא טיפוס המפתחות הנשמרים בעץ, ו- $V$  הוא טיפוס הערכים הנשמרים עבור כל מפתח.

למשל, בדוגמה הראשונה למעלה,  $K$  יכול להיות  $\text{List}\langle \text{Integer} \rangle$  (רשימת שלמים) ו- $V$  יכול להיות  $\text{Date}$ , המייצג את תאריכי הוספת הרשימה לעץ. בדוגמה השנייה למעלה,  $K$  הוא  $\text{String}$  שמייצג שמות פרטיים, ו- $V$  הוא  $\text{String}$  שמייצג שמות משפחה.

<sup>1</sup> היזכרו במחלקות גנריות שנלמדו בשיעור 7 ובתרגול 8.

**TrieNode מגדיר את המתודות הבאות:**

**void** addSuffix(K suffix, V value)

- מקבלת סיומת של מפתח מטיפוס K וערך מטיפוס V, ומוסיפה אותם למבנה הנתונים תחת הצומת הנוכחי. לדוגמא, נניח שאנחנו בצומת הכחול בעץ למעלה, וקיבלנו את הסיומת "dy" ואת הערך "smith". לצומת זה כבר יש צומת ילד המחובר בקשת 'd', מן הילד הזה נוסף קשת עם 'y' לצומת ילד חדש, ובצומת החדש נשמור את הערך "smith". כלומר, המסלול מהשורש לצומת החדש ייצג את השם "andy", והערך שישמר עבורו הוא שם המשפחה "smith". בתור דוגמא נוספת, אם אנחנו בצומת האדום בעץ למעלה, וקיבלנו את הסיומת "ie" ואת הערך "jones", אין צורך להוסיף צמתים חדשים כי המסלול המתאים כבר קיים; נותר לנו רק להוסיף את הערך "jones" לצומת בקצה המסלול.

**int** getNumKeys();

- מחזירה את מספר המפתחות שהסתיימו בצומת הנוכחי מבין אלה שנוספו אליו, כולל חזרות. למשל, אם הכנסנו 5 אנשים בשם bill ו-7 אנשים בשם billy לעץ בדוגמא למעלה, קריאה ל- getNumKeys() מהצומת האדום תחזיר 5.

**Set<V>** getValues()

- מחזירה Set עם כל הערכים שנשמרו בצומת הנוכחי. למשל, אם הצומת הנוכחי הוא הצומת האדום מהדוגמא למעלה, נחזיר את כל שמות המשפחה שנשמרו עבור "bill".

**TrieNode<K, V>** getNext(K suffix)

- מחזירה את הצומת הבא על המסלול המתאים לסיומת הרצף הנתונה. למשל, הפעלת המתודה על הצומת האדום עם הסיומת "yabc" תחזיר את הצומת השחור. אם לא קיים צומת מתאים בעץ, יוחזר null.

**List<TrieNode<K, V>>** getNexts()

- מחזירה את רשימת כל הצמתים-ילדים של הצומת הנוכחי (כלומר, יש קשת מן הצומת הנוכחי אליהם). אין חשיבות לסדר הצמתים ברשימה, והיא יכולה להכיל ערכי null.

**Trie מגדיר את המתודות הבאות:**

**boolean** addKey(K key, V value)

- מקבלת מפתח לעץ וערך המשויך אליו, ומוסיפה אותם לעץ. מחזירה false אם ההוספה נכשלה משום שהמפתח אינו תקין (למשל, null).

**Set<V>** searchByPrefix(K prefix);

- מחפשת את כל המפתחות בעץ עם תחילית מסוימת, ומחזירה Set עם כל הערכים שנשמרו עבור המפתחות הללו. למשל, בדוגמא למעלה, בחיפוש התחילית "bill", המתודה תחזיר את כל שמות המשפחה שנשמרו עבור bill, billie, billy וכו'.

אנו נכתוב מימוש לשני המנשקים שבו K מוגדר להיות String, כלומר, נייצג Trie של מחרוזות.

1. באתר הקורס נתון לכם שלד המחלקה

```
public class StringTrieNode<V> implements TrieNode<String, V>
```

שימו לב - מכיוון שטיפוס המפתח נקבע להיות String, למחלקה זו רק פרמטר גנרי אחד - V, טיפוס הערך הנשמר עבור כל מפתח.

לנוחיותכם, בשלד המחלקה כבר נתונים לכם שדות המחלקה, ובנאי.

השדה `nexts` - ישמור מצביעים לילדים בעץ. מכיוון שלכל צומת יש לכל היותר 26 ילדים, הבנאי הנתון מאתחל את רשימת הצמתים הבאים ב- 26 שומרי מקום לצמתים (null). באינדקס 0 יישמר מצביע לילד המחובר בקשת עם 'a', באינדקס 1 מצביע לילד המחובר ב-'b' וכו'. ניתן להוסיף/לגשת לילד באינדקס ה- i תוך שימוש במתודות `nexts.get(i)` ו- `nexts.set(i, child)`.

השלימו את מימוש המחלקה.

#### הערות:

- אנו נניח שכל הרצפים הנשמרים בעץ הם של אותיות a-z קטנות בלבד.
- אינכם צריכים לבדוק את תקינות הקלט למתודות - אנו נניח שבדיקת התקינות היא באחריות המחלקה `StringTrie`.
- אל תשתמשו ב- `Map` בפתרון הסעיף הנוכחי.
- מעבר לכך, אתם רשאים לשנות ולהוסיף למימוש הפנימי של `StringTrieNode` (השדות, הבנאי).
- אתם רשאים להוסיף מתודות ומחלקות עזר לפי הצורך, אך אל תשנו את המנשקים הנתונים.
- שימו לב שאינכם יודעים מראש מה יהיה הטיפוס הקונקרטי שנציב במקום `V (String ?Integer?)`, אך גם אין לכם צורך לדעת זאת - הפעולות היחידות שתצטרכו לבצע על ערכים מטיפוס V הן להוסיף אותם לאוספים שונים.

2. באתר הקורס נתון לכם שלד המחלקה

```
public class StringTrie<V> implements Trie<String, V>
```

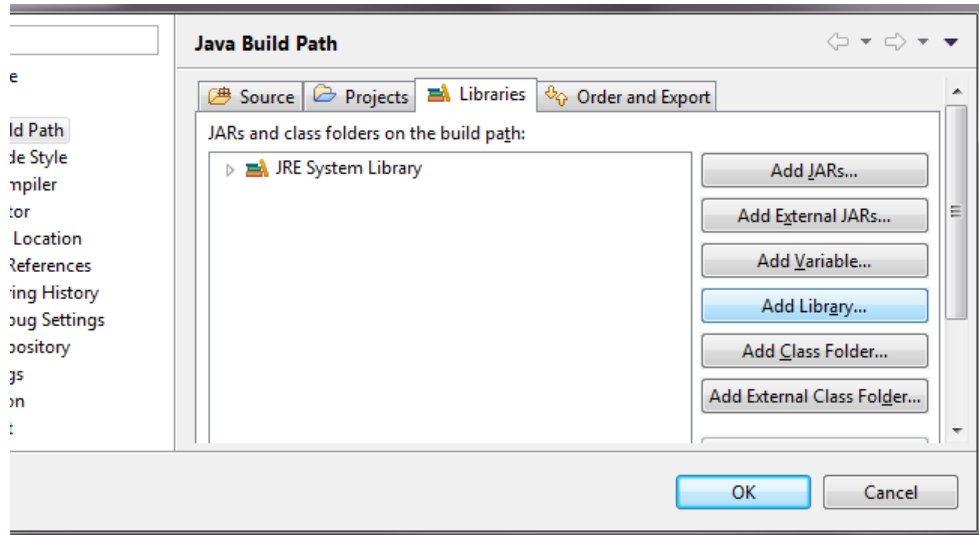
השלימו את מימוש המחלקה.

#### הערות:

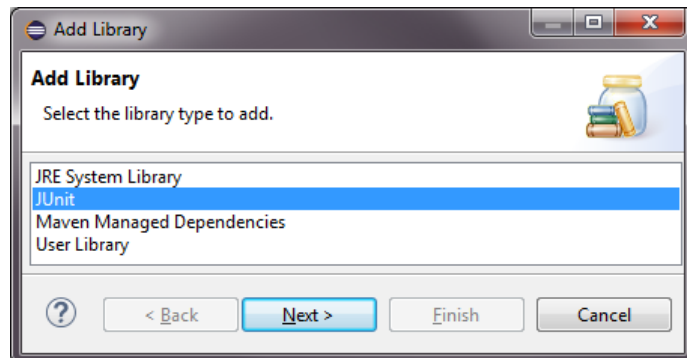
- נרצה שכל הרצפים הנשמרים בעץ הם של אותיות a-z קטנות בלבד. ב- `StringTrie` עליכם לבדוק את תקינות הקלט למתודות, ולוודא שהקלט שאתם מעבירים בקריאות למתודות של `StringTrieNode` תקין. אם הוא אינו תקין יש לזרוק שגיאה (ניתן להיעזר ב- `ErrorHelper` לשם כך).
- המפתח הריק "" תקין ובמקרה כזה יש לשמור את הערכים המשויכים אליו בשורש העץ. null אינו תקין.
- ניתן להעזר במתודה `addAll` של `Set` במימוש `searchByPrefix`.

## בדיקות עם JUnit

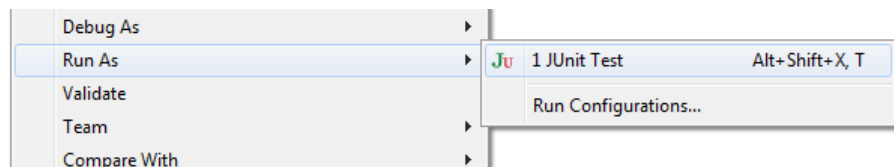
בין קבצי התרגיל נתונה לכם גם מחלקת `TrietestCase`, בה ניתן להיעזר כדי לבדוק את המימוש שלכם. כדי להריץ את הבדיקה יש להוסיף את המחלקה לפרוייקט באקליפס, וכן להוסיף את ספריית JUnit. לאחר מכן, יש להריץ את המחלקה כ- `JUnit test case`. הוספת הספרייה: קליק ימני על הפרוייקט `java build path < properties < Libraries` לחצו על `Add Library`.



כעת בחרו בהוספת ספריית JUnit ולחצו על `Next` ולבסוף על `Finish`.



כדי להריץ את התכנית כ-JUnit לחצו על קליק ימני `< Run as... < JUnit Test`.



מבנה ה-view שאמור להיפתח אוטומטית (JUnit):

The screenshot shows the JUnit test runner interface in an IDE. The top status bar indicates 'Finished after 0.02 seconds', 'Runs: 6/6', 'Errors: 1', and 'Failures: 2'. The test results list includes:

- testFactorialZero (0.000 s) - Success
- testChoosePositive (0.000 s) - Failure
- testChooseZero (0.000 s) - Success
- testChooseNegative (0.000 s) - Success
- testFactorialPositive (0.001 s) - Success
- testFactorialNegative (0.000 s) - Success

The 'Failure Trace' section shows the following error:

```
java.lang.AssertionError: Factorial over 6 expected:<720> but was:<0>
    at ExampleTestCase.testChoosePositive(ExampleTestCase.java:29)
```

Callouts in Hebrew explain the following elements:

- הרץ מחדש (Run again)
- מספר הטסטים שנכשלו (Number of failed tests)
- פירוט הטסטים שרצו (Details of tests that passed)
- מספר הטסטים שהורצו (Number of tests that were run)
- מספר הטסטים שזרקו שגיאה (Number of tests that threw an error)
- פירוט לגבי כישלון הטסט המסומן (Details about the failed test)
- לחיצה כפולה מובילה למקום המתאים בקוד (Double click leads to the code location)

ראו גם שיעור 7 - מצגת ודוגמאות בדיקות, ו- [tutorial](#) (בייחוד רלוונטיים פרקים 5, ו7.1-7.3).

אין צורך להגיש את מחלקת הבדיקה `TrieTestCase` או את `JUnit.jar`.

## חלק ב' - Java collection framework (50 נק')

בחלק זה נתרגל עבודה עם אוספים (Collections) ע"י מימוש מנוע חיפוש פשוט. בין קבצי העזר של התרגיל תוכלו למצוא את המחלקות HTMLTokenizer, SearchEngine, Main 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 SearchEngine. אם תרצו תוכלו להוסיף או לשנות את הדפים בהם אתם משתמשים לצרכי בדיקה.

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

בנוסף, נתונה המחלקה SearchEngine שבה הקוד המפעיל את המערכת ומתקשר עם המשתמש. הקוד במחלקה זו קורא בתחילה למתודה index אשר תאכלס את אינדקס המילים. לאחר מכן, כתלות בקלט המשתמש יתבצע חיפוש של דפים לפי מילת חיפוש שלמה או תחילית של מילה, המסומנת ע"י הוספת \* בסופה (ראו דוגמאות למטה).

### מה עליכם לעשות:

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.

```
public interface WordIndex {

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

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

    /**
     * Search for pages containing a given word in the index
     *
     * @param exact - 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> searchExact(String exact);
}
```

הערה: בקוד למעלה התיעוד 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 על אותה מילה מס' פעמים) ואת כתובת האינטרנט (URL) של הדף מהן הגיעו. עליכם לבחור את מבני הנתונים שבהם תשתמשו ולדאוג לשמור על הקשרים הבאים:
  - לכל תחילית של מילה ומילה שלמה באילו דפים היא מופיעה וכמה פעמים בכל דף (למשל, בדף שמכיל רק את המילה java מופיעות התחיליות j, ja, jav ו- java, פעם אחת כל אחת).
  - לכל דף כמה מילים בסה"כ מופיעות בו (עם חזרות)

**שימו לב:**

- רשימת המילים בקלט היא כפי שהוחזרה ע"י HTMLTokenizer, אולם יש לשמור גרסת lowercase של המילים.
- אין צורך לנקות סימני פיסוק, רווחים וכו'.
- אין לשמור את המילה הריקה "" או null, לא כתחילית ולא כמילה שלמה.

• **המתודה searchPrefix**

- מקבלת תחילית של מילה לחיפוש ומחזירה רשימה ממוינת של כתובות אינטרנט בהן מופיעות מילים המתחילות באותה תחילית. נמיון את הרשימה כך שכלל שהמשקל היחסי של התחילית בדף גבוה יותר כך הכתובת תופיע במקום גבוה יותר ברשימה. משקל זה מוגדר כך: תהיה X רשימת כל המילים שמכילות את התחילית הנתונה (עם חזרות) בדף מסוים. תהיה Y רשימת כל המילים באותו דף (עם חזרות). המשקל היחסי של התחילית בדף הוא  $\frac{X.size()}{Y.size()}$ . לא נחזיר דפים בהם התחילית אינה מופיעה כלל.

- שימו לב:** כדי להשוות טיפוסים פרימיטיביים ניתן להשתמש בשירות compare של הטיפוס העוטף. לדוגמא: Double.compare(double d1, double d2)

• **המתודה searchExact**

- פועלת בדומה ל- searchPrefix אך מחפשת מילה שלמה ולא תחילית. המשקל היחסי של מילה בדף מוגדר באופן דומה: יהיה x מס' המופעים של המילה הנתונה בדף מסוים. תהיה Y רשימת כל המילים באותו דף (עם חזרות). המשקל היחסי של המילה בדף הוא  $\frac{x}{Y.size()}$  ושוב, לא נחזיר דפים בהם המילה אינה מופיעה כלל.

**הנחיות:**

- עליכם לכתוב ולהגיש את המחלקה MyWordIndex שממשת את המנשק WordIndex. ניתן להוסיף מחלקות ומתודות עזר.
- אין לשנות את הקבצים הנתונים באתר הקורס.
- לקבלת מלוא הנקודות על התרגיל יש להשתמש במבני הנתונים המתאימים לבעיה מתוך הספרייה הסטנדרטית ובאופן ראוי (נכונות הפלט אינה הקריטריון היחיד). לדוגמא, כדי לשמור ערכים ללא חזרות יש להשתמש ב-Set או Map.
- בפרט, אל תשתמשו בפתרון חלק א' עבור חלק ב' (Trie).



- 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). מחלקה זו תיעזר בנתונים שנשמרו באינדקס לצורך מיון הכתובות. שימו לב שתוצאות ההשוואה תלויות במילת החיפוש הנוכחית.
- כדי לבדוק את התכנית שלכם השתמשו במחלקה `Main` המייצרת אובייקט חדש מטיפוס `SearchEngine` ומעבירה לו בבנאי מופע של המחלקה `MyWordIndex` שמימשתם. לאחר יצירת האובייקט נקרא השירות `.run`.

### דוגמא (בהתבסס על רשימת הכתובות בקוד שניתן לכם):

```

Indexing "http://en.wikipedia.org/wiki/Java_(programming_language)" ...found 7978
tokens.
Indexing "http://en.wikipedia.org/wiki/Java" ...found 5586 tokens.
Indexing
"http://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html"
...found 1968 tokens.
Indexing "http://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html" ...found 15705
tokens.
Indexing "http://en.wikipedia.org/wiki/United_States" ...found 29661 tokens.
Indexing "http://en.wikipedia.org/wiki/World_War_II" ...found 24866 tokens.
Indexing "http://en.wikipedia.org/wiki/World_War_I" ...found 31195 tokens.
> java
Searching for pages containing the word "java"...
1. http://en.wikipedia.org/wiki/Java_(programming_language)
2. http://en.wikipedia.org/wiki/Java
3. http://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html
4. http://en.wikipedia.org/wiki/World_War_II
> wor
Searching for pages containing the word "wor"...
Unable to find relevant pages.
> wor*
Searching for pages containing the prefix "wor*"...
1. http://en.wikipedia.org/wiki/World_War_I
2. http://en.wikipedia.org/wiki/World_War_II
3. http://en.wikipedia.org/wiki/United_States
4. http://en.wikipedia.org/wiki/Java_(programming_language)
5. http://en.wikipedia.org/wiki/Java
6. http://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html
7. http://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html
> (exit)
Bye!

```

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

## בהצלחה !