הנחיות כללית:

• קראו בעיון את קובצי הנוסל הגשת התגרילים אשר נמצאו באתר הקורס.
•時は להגיש קובץ zip אחד עם השם המשתמש שלך (לדוגמה, עבור המשתמש zvainer יקרא הקובץ zvainer.zip)
• הקובץ zip יכיל:

א. קובץ פרטיים باسم details.txt
ב. קבצי ה- java של התוכניות אותם תבקש לממש
ג. קובץ טקסט עם העתק של קבצי ה- java חלק א' חווה של מקדונלד הזקן

Old MacDonald had a farm, E-I-E-I-O
And on his farm he had some chicks, E-I-E-I-O
With a cluck-cluck here and a cluck-cluck there
Here a cluck there a cluck
Everywhere a cluck-cluck

Old MacDonald had a farm, E-I-E-I-O
And on his farm he had some cows, E-I-E-I-O
With a moo-moo here and a moo-moo there
Here a moo there a moo
Everywhere a moo-moo
With a cluck-cluck here and a cluck-cluck there
Here a cluck there a cluck
Everywhere a cluck-cluck

Old MacDonald had a farm, E-I-E-I-O
And on his farm he had some dogs, E-I-E-I-O
With a woof-woof here and a woof-woof there
Here a woof there a woof
Everywhere a woof-woof
With a moo-moo here and a moo-moo there
Here a moo there a moo
Everywhere a moo-moo
With a cluck-cluck here and a cluck-cluck there
Here a cluck there a cluck
Everywhere a cluck-cluck

Old MacDonald had a farm, E-I-E-I-O
Requirements:

In Old MacDonald's farm you can find: **dogs, cows, pigs, chicks** and **horses**. In this exercise you will write an application that receives as input a list of animals in old MacDonald's farm (with possible repetitions). The application prints:

1. **The list of animals in old MacDonald's farm with their sounds.** The order of the animals in this list is exactly the order in the input list. For example: for the input "cow pig chick chick cow" the output is

   cow: moo
   pig: oink
   chick: cluck
   chick: cluck
   cow: moo

2. **The status of old MacDonald's farm:** a two column table where the first column contains animal names (no repetitions!) in alphabetical order and the second column contains the number of animals of this type in old MacDonald's farm. For example: for the input "cow pig chick chick cow" the output is:

   Animal   Count
   ------   -----
   chick    2
   cow      2
   pig      1

3. **The "old MacDonald's had a farm" song for the animals in the farm.** Revise the song to describe only the animals currently in the farm. For every animal type the line "*And on his farm he had some ...*" appears exactly once, the song then continues repeating previous types. The order of appearance of the animal types in the song is the order of appearance in the input list. For example: for the input "cow pig chick chick cow" the output is:

   Old MacDonald had a farm, E-I-E-I-O
   And on his farm he had some cows, E-I-E-I-O
   With a moo-moo here and a moo-moo there
   Here a moo there a moo
   Everywhere a moo-moo
   Old MacDonald had a farm, E-I-E-I-O

   Old MacDonald had a farm, E-I-E-I-O
   And on his farm he had some pigs, E-I-E-I-O
   With an oink-oink here and an oink-oink there
   Here an oink there an oink
   Everywhere an oink-oink
   With a moo-moo here and a moo-moo there
Here a moo there a moo
Everywhere a moo-moo
Old MacDonald had a farm, E-I-E-I-O

Old MacDonald had a farm, E-I-E-I-O
And on his farm he had some chicks, E-I-E-I-O
With a cluck-cluck here and a cluck-cluck there
Here a cluck there a cluck
Everywhere a cluck-cluck
With an oink-oink here and an oink-oink there
Here an oink there an oink
Everywhere an oink-oink
With a moo-moo here and a moo-moo there
Here a moo there a moo
Everywhere a moo-moo
Old MacDonald had a farm, E-I-E-I-O

**Design:**

A schematic description of the interfaces, classes and methods (details might differ slightly from the code).
Resources:

A skeleton for the application was implemented for you and you can download the files from the web site. Some of the classes have a complete implementation and should not be altered. Others are missing some implementation details and it is up to you to add those.

You should not change the signature of public methods, but you may add private methods and fields as you see fit.

Your implementation should rely on classes from the collections framework (Set, List, Map, …). Read the documentation for the various classes and choose the ones you find most suitable for the implementation.

Fully implemented classes: The interface Animal and the classes implementing it (Pig, Cow, Horse, Chick and Dog) all belong to the package il.ac.tau.sw1.oldmac.animals.
The class Main (not shown in the diagram) is the entry point to the application (i.e. its main method should be used). Main and all the classes in il.ac.tau.sw1.oldmac.animals are implemented and should not be altered.

What you should implement:

Complete the implementation of the classes Farm, FarmBuilder and Song in the package il.ac.tau.sw1.oldmac as described below.

FarmBuilder class:

Builds a Farm object out of a list of animals. Implements a single method:

- public static Farm buildFarm(String[] animalNames)

The method receives a list of animal types then returns a new Farm populated with those animals.

Farm class:

Represents a farm. implements the following methods:

- public void addAnimal(Animal animal)
  
  Adds a new animal to the farm.
• public Iterator<Animal> iterator()

  Returns an iterator over all the animals in the farm. The order is the same as the order in which the animals were inserted to the farm.

• public Iterator<Animal> iteratorUnique()

  Returns an iterator over all the animals in the farm without repetitions. The iterator iterates over the animals in the farm by the order of their addition to the farm. For example, if the animals added to the farm were: cow, pig, chick, chick, cow (in this order), then the order of iteration is cow, pig, chick.

• public void printStatus()

  Prints the status of the farm as described in the second requirement.

Song class:

• public static void printSong(Farm farm)

  Prints the "Old MacDonald had a farm" song as described in the third requirement.

You may add any methods and fields as you deem necessary to those three classes. In your implementation you should use classes (and interfaces) from the Java Collection Framework.

You may assume that:

• The list of arguments to the application is not empty and that every argument is one of the following: "cow", "chick", "horse", "dog" or "pig".
• The method Iterator.remove() is never called for the two iterators of class Farm.
In this part, you are required to implement a simple search engine.

Parts of the code have already been written for you and are available for download on the course site.

Our search engine will handle a limited number of HTML pages that it will read from the internet. These pages are fixed in advance.

You can find the list in the SearchEngine class.

You can add or change the pages you use.

After downloading an HTML page from the internet, we only refer to the part of the text in the page and break this part into individual words.

This code is already implemented for you in the HTMLTokenizer class.

In addition, the SearchEngine class in which the code operates the system and communicates with the user is also available.

What you need to do:

We want to create an index of all the words that appeared in all the pages we downloaded from the internet.

The existence of this index will allow us to perform searches for a specific word later.

You need to implement the MyWordIndex class that implements the WordIndex interface.

The class stores the word index, allows adding words to the index and searching in it.

```java
package il.ac.tau.cs.sw1.simplesearch;

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

public interface WordIndex {

    /**
     * Add the words originating in the specified 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 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);
}
```
The methods and techniques:

**Index Method**

This method is responsible for populating the structure of your data. The method accepts a collection of words (which may repeat) and an internet address from which they were received.

You must choose the structures you will use and ensure the following relationships:

- For each word, count the number of pages it appears in and how many times it appears in each page,
- For each page, count the number of words that contain the word (with repeats).

The received list of words is as it appeared in the original page, but it must be stored in lowercase.

**Search Method**

The method takes a word for search and returns a sorted list of internet addresses on which the word appears. The list is sorted so that the higher the relative weight of the word in the page, the higher the address appears in the list.

The relative weight of a word in a page will be the number of occurrences of the word in the page divided by the total number of words in the same page.

**Hints:**

Use classes of the Java Collections framework, especially the `Map` and `HashMap` classes, which may be useful.

Sort the list of internet addresses using the `Collections.sort(...)`. Remember that the "natural sort" of the addresses (String) is a lexicographical sort and not as required.

In order to change the sorting method, you must implement the `Comparator` interface.

Note: The words received from the HTML file are not "clean" (contain punctuation and other symbols).

You do not need to perform any action on the words, except using them as they are.

In order to test your plan, write a class and a `main` function that creates an `SearchEngine` object of the `SimpleSearchEngine` class and generates a `MyWordIndex` object that implements the `MyWordIndex` interface.

After creating the object, you must call the `run` method. For example:

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

**Examples:**

For a list of internet addresses in the `Main` file (pre-generated, see below), the search term "java" will receive the output:

```
> java
```
הערות כלליות:
בשני חלקי התרגיל חלק משקוד בכל גת עננים עילכים מעלים מספרים שלご覧ים מ_codigo הקים.坦克
לייבא את הקוד הקים ע"י שחריון את קבצי הקוד המופיעה בנויה אלכלים במושב. כעד חוכל
ליבא את הפרויקט עד לפורים הייבוב (File - import) של אקליפס.
 الليبيים ברויצים של פרויקט קים (Existing Projects into Workspace) ליאורמקו והסיע את קובי
וזה ב用自己的 ובוית הקים המ vazים.

8