משימה 7 # בקורס תוכנה 1

חלק א': המילים של מקדונלד הצהוב

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

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

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.** 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
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 the public methods, but you may add private methods and fields as you see fit.

Your implementation should rely on the collection classes mentioned in class (Set, List, Map, …). Read the documentation for the various classes and choose the ones you need for your implementation.

Fully implemented classes: The interface IAnimal and the classes implementing it (Pig, Cow, Horse, Chick and Dog) all belong to 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(IAnimal animal)

  Add a new animal to the farm.
• public Iterator<IAnimal> iterator()

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

• public Iterator<IAnimal> iteratorUnique()

    Return an iterator over all animals in the farm without repetitions. The iterator iterates 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 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.
חלק ב: חפש אתני

بحד אל נושא זה студентים וסטודנטית של דר. אוטו וידבר המחברת. דפה זו
מנוע החיפוש שלה שולח י手下 בקש מוצמדת של דר. אוטו וידבר המחברת. דפים
של ת déco מרכז את الشريفים בקופים. בחרה שידברת דף מ_typeof נגזרת של דף
הⲪוס במקלדת שידברת דף מ_typeof נגזרת של דף html ת déco מרכז את الشريفים
בנוסף המ长沙市 שידברת דף מ_typeof נגזרת של דף html ת déco מרכז את الشريفים
הם עליכם לעשות:

נרצה ליצור אינדקס של כל המילים שהורדו מהרשת

וכ殪ה של אינדקס זה

יאפשר לנו מאוחר יותר לבצע חיפושים עבור מילה מסוימת

עליכם לממש את המחלקה WordIndex.

מחלקה זו שומרת את אינדקס המילים,

מאפשרת הוספת מילים לאינדקס וחיפוש בו.

המתודה index

המתודה אחראית על אכלוס מבנה הנתוניםนำไป

המתודה מקבלת אוסף של מילים ( ייתכנו חזרות)

וכ永利ים בודדים. מזזים המילים במילים במילים

רשימת המילים במילים במילים במילים במילים

בקלט היא בהתאם ביד המחבר ויליאם

lowercase של המילים

public class WordIndex {  
   public WordIndex() {    
...  
   }  
   /**  
* Add the words originating in the specifies URL.  
* @param words - collection of words to add  
* @param strURL - the location of the page containing the words  
*/  
   public 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.  
*/  
   public List<String> search(String word) {    
...  
   }   
}
The method search takes a word to search and returns an ordered list of internet pages where the word appears.

We rank the list so that the higher the relative weight of a word on a page, the higher the page will appear in the list.

The relative weight of a word on a page is considered to be the number of times the word appears on the page divided by the total number of words on the page.

**Example:**

For the list of pages appearing in the file Main and the search word "java," the output will be:


**Instructions:**

1. Read the instructions on the course website, VirtualTAU (http://virtual2002.tau.ac.il/).
2. Submit your solutions using the VirtualTAU system.
3. Submit your solutions by creating a zip file named after your user name. For example, for the user zvainer, the file name would be zvainer.zip.
4. The zip file should contain:
   - A file named details.txt containing your name and ID.
   - Java files of the programs you have written.
   - A text file containing a copy of all the Java files.

**Java commands:**

```java
> java
```