

## משימה #7 בקורס תוכנה 1

### חלק א': החווה 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

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
```

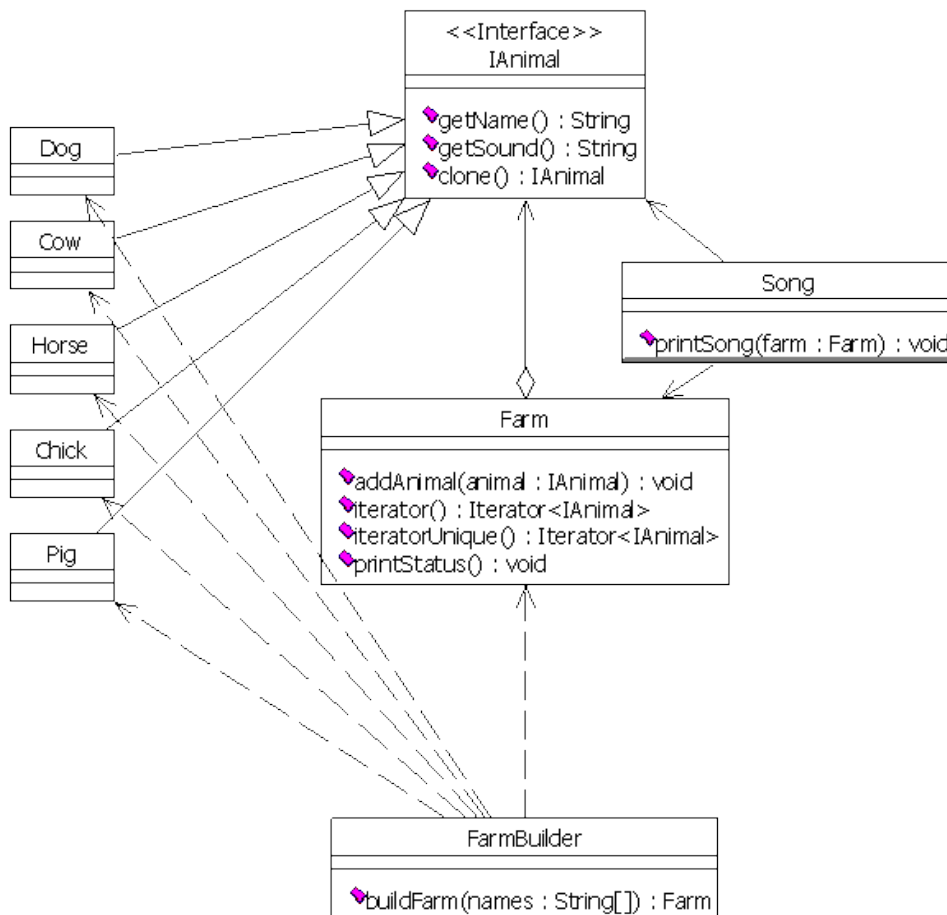
```
Old MacDonald had a farm, E-I-E-I-O
And on his farm he had some pigs, E-I-E-I-O
With an oink-oink here and an oink-oink there
Here an oink there an oink
Everywhere an oink-oink
With a moo-moo here and a moo-moo there
Here a moo there a moo
```

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<IAAnimal> 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<IAAnimal> 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 אותי

בחלק זה אתם נדרשים לממש מנוע חיפוש פשוט. חלקים מהקוד כבר נכתבו עבורכם והם זמינים להורדה באתר הקורס.

מנוע החיפוש שלנו יטפל במספר מצומצם של דפי HTML אותם הוא יקרא מהרשת. דפים אלו קבועים מראש. תוכלו למצוא את הרשימה בקובץ Main.java. 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 Main שבה הקוד המפעיל את המערכת ומתקשר עם המשתמש קיימת אף היא.

### 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 את המחלקה WordIndex. מחלקה זו שומרת את אינדקס המילים, מאפשרת הוספת מילים לאינדקס וחיפוש בו.

```
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) {
        ...
    }
    ...
}
```

### • המתודה index

מתודה זו אחראית על אכלוס מבנה הנתונים שלכם. המתודה מקבלת אוסף של מילים (ייתכנו חזרות) ואת כתובת האינטרנט 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). 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 שככל שהמשקל היחסי של מילה בדף גבוה יותר כך הכתובות תופיע במקום גבוה יותר ברשימה. את המשקל היחסי של מילה בדף נחשב לפי מספר המופעים של המילה בדף מחולק במספר המילים הכולל באותו דף.

## הזרחה:

השתמשו במחלקות מ Java Collections, בפרט המנשק Map והמחלקה HashMap יכולים להועיל.

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 מחלקה המממשת את המנשק Comparator. כדי להקל את המימוש כתבו מחלקה זו כמחלקה פנימית במחלקה Wordindex (זו המלצה בלבד). שימו לב שתוצאות ההשוואה תלויות במלת החיפוש הנוכחית.

שימו לב שהמילים המתקבלות מדף ה-HTML אינן "נקיות" (כוללות סימני פיסוק וכדומה). אין צורך לבצע על המילים אלא להשתמש בהן כמו שהן.

## דוגמאות:

עבור רשימת הכתובות המופיעה בקובץ ה Main ומילת החיפוש "java" יתקבל הפלט:

```
> 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.gutenberg.org/files/27152/27152-h/27152-h.htm
```

## הוראות הגשה:

1. קראו בעיון את קובץ נוהלי הגשת התרגילים אשר נמצא באתר הקורס.
2. הגשת התרגיל תעשה ע"י המערכת VirtualTAU (<http://virtual2002.tau.ac.il/>).
3. הגשת התרגיל תתבצע ע"י יצירת קובץ zip שנושא את שם המשתמש. לדוגמא, עבור המשתמש zvainer יקרא הקובץ zvainer.zip. קובץ ה zip יכיל:
  - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז. הזהות שלכם.
  - ב. קבצי ה-java של התכניות שהתבקשתם לכתוב.
  - ג. קובץ טקסט עם העתק של כל קבצי ה Java.