**HashMap Workshop**

The purposes of this workshop is to make sure that the students control the List class, the basics of HashMap And prepared them well for HW10.

**List and introduction to HashMap**

* Start by reminding the students about the list methods and make sure they control them.
* Let the students solve #Exercise1 and discuss about it.
* Introduce and remind the students about the following HashMap methods and attributes:
* <key , value> - every HashMap contain a pair of elements- a key element and it's value. In that way we can create a “map” and by a given key we can find his value in the map!
* get(object key) - Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
* keyset() - Returns a set view of the keys contained in this map.
* put(key k , value val) - Associates the specified value with the specified key in this map.
* values() - Returns a collection view of the values contained in this map.

These are the important methods to this workshop, but you are more than welcome to show the students the full API of HashMap.

**Exercises:**

**#Exercise1**

1. Implement **public void addLast(char val)** in the List class. This method enable us to add an element to the end of the list.
2. Implement **public void addLast(char val)**  in the List class using the “last” field.

**Points for discussion:**

1. What will happen if we will add to our List class a field which saves our last Node in the list?
2. Notice that by adding “last” field we should adjust our addFirst and delete methods. For example in addFirst method we should check if the list is empty. If the list is empty our last and first element are both the same!

**#Exercise 2**

Implement **AddToMap(String word)** method which gets a word.

The method purpose is to create a list of letters of the given word and add to the HashMap the word and it's list of letters as key and value elements.

**#Exercise 3**

Implement **public static char[] letters(String word)** method which gets a word.

The method purpose is to check if the given word is in the map.

If the word is in the map, return an array of it's letters (think of HashMap method).

If the given word isn't in the map return null.

**Points for discussion:**

1. Don't forget the list methods such as list to array. They can be very helpful and save us lots of time and code, so don’t rush.
2. Notice that the HashMap get method is useful for getting the value of a specific key and also for checking if a given key is in the HashMap.

**#Exercise 4**

Implement **public static void print()** method which should print all the elements in the HashMap with the given format: “key1 : {k , e , y , 1}” “key2 : {k , e , y , 2}” …

**Points for discussion:**

1. Notice that this method displays us the HashMap. Each key has it's own value and it points to it.
2. Notice that this is a good example of using “for each” syntax.

**#Exercise 5 if you have time**

Implement **public static List allLetters()** method

The method purpose is to create a big list of all the letters from all the list of letters in the map.

For example if the map look like this: {key:[k,e,y] , dhs:[d,h,s]} than the returning list will look like this: {s,h,s,y,e,k}. notice that there is no need to check duplicates and order.

**Points for discussion:**

1. Notice that we use a different iterator for the different lists.

The solutions for the list methods are in the list class.