

At this lab section, we will experiment  
different implementation of the ADT  
Dictionary in Java.

## Dictionaries

Asst. Prof. Dr. Feriştah DALKILIÇ  
Res. Asst. Fatih DİCLE

---

## PART 1 – Dictionaries

A dictionary provides a powerful way to organize searchable data as finding a word’s definition, a friend’s address, or someone’s telephone number.

The ADT dictionary—also called a map, table, or associative array—contains entries that each have two parts:

- A keyword—usually called a search key—such as an English word or a person’s name
- A value—such as a definition, an address, or a telephone number—associated with that key

The ADT dictionary has the same major operations:

- Add a new entry to the dictionary, given a search key and associated value
- Remove an entry, given its associated search key
- Retrieve a value associated with a given search key
- See whether the dictionary contains a given search key
- Traverse all the search keys in the dictionary
- Traverse all the values in the dictionary
- Detect whether a dictionary is empty
- Get the number of entries in the dictionary
- Remove all entries from the dictionary

You can implement a dictionary by using either an array or a chain of linked nodes. Some dictionaries do sort their entries by search key, while other dictionaries have unsorted entries. The worst-case efficiencies of the dictionary operations for array-based and linked implementations are given in the following table. Using an array to implement a sorted dictionary allows for an efficient retrieval operation because you can use a binary search.

	Array-Based		Linked	
	Unsorted	Sorted	Unsorted	Sorted
Addition	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Removal	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Retrieval	$O(n)$	$O(\log n)$	$O(n)$	$O(n)$
Traversal	$O(n)$	$O(n)$	$O(n)$	$O(n)$

### Exercise – 1

In this section, you will experiment with array-based ADT Dictionary implementation.

#### Step – 1

Create a new Java Project. Add the interface “DictionaryInterface.java” and “ArrayDictionary.java” given in *src* folder.

#### Step – 2

Add a new class with the name of “Test.java”. Create an instance of ArrayDictionary and add the given *contact\_name – phone\_number* pairs into the dictionary.

contact_name	phone_number
"Dirk"	"555-1234"
"Abel"	"555-5678"
"Miguel"	"555-9012"
"Tabbie"	"555-3456"
"Tom"	"555-5555"
"Sam"	"555-7890"
"Reiss"	"555-2345"
"Bette"	"555-7891"
"Carole"	"555-7892"
"Derek"	"555-7893"
"Nancy"	"555-7894"

#### Step – 3

Add the following method into “Test.java” and display the current content of the dictionary.

```
public static void display(DictionaryInterface<String, String> dictionary)
{
    Iterator<String> keyIterator = dictionary.getKeyIterator();
    Iterator<String> valueIterator = dictionary.getValueIterator();

    while (keyIterator.hasNext() && valueIterator.hasNext())
        System.out.println(keyIterator.next() + " : " + valueIterator.next());
    System.out.println();
} // end display
```

#### Step – 4

In Test.java, perform the operations given below:

- Display the phone book.
- Show the contact count in your phone book.
- Retrieve the Sam’s phone number.
- Query whether Bo in your contact list.
- Update the Miguel’s phone number as "555-9015".
- Remove Reiss from your contacts.
- Display your current phone book.
- Delete your all contacts.

Your Test.java
<pre>import java.util.Iterator; public class Main {     public static void display(DictionaryInterface&lt;String,String&gt;dictionary){     Iterator&lt;String&gt; keyIterator = dictionary.getKeyIterator();     Iterator&lt;String&gt; valueIterator = dictionary.getValueIterator();     while (keyIterator.hasNext() &amp;&amp; valueIterator.hasNext()){         System.out.println(keyIterator.next()+" : "+ valueIterator.next());     }     System.out.println(); }     public static void main(String[] args) {         ArrayDictionary phoneBook = new ArrayDictionary&lt;&gt;();</pre>

```

        phoneBook.add("Dirk", "555-1234");
        phoneBook.add("Abel", "555-5678");
        phoneBook.add("Miguel", "555-9012");
        phoneBook.add("Tabbie", "555-3456");
        phoneBook.add("Tom", "555-5555");
        phoneBook.add("Sam", "555-7890");
        phoneBook.add("Reiss", "555-2345");
        phoneBook.add("Bette", "555-7891");
        phoneBook.add("Carole", "555-7892");
        phoneBook.add("Derek", "555-7893");
        phoneBook.add("Nancy", "555-7894");

        display(phoneBook);
        System.out.println(phoneBook.getSize());
        System.out.println(phoneBook.getValue("Sam"));
        System.out.println(phoneBook.contains("Bo"));
        phoneBook.remove("Reiss");
        phoneBook.remove("Miguel");
        phoneBook.add("Miguel", "555-9015");
        display(phoneBook);
        phoneBook.clear();
    }
}

```

Your Output
<p> <b>Dirk : 555-1234</b>  <b>Abel : 555-5678</b>  <b>Miguel : 555-9012</b>  <b>Tabbie : 555-3456</b>  <b>Tom : 555-5555</b>  <b>Sam : 555-7890</b>  <b>Reiss : 555-2345</b>  <b>Bette : 555-7891</b>  <b>Carole : 555-7892</b>  <b>Derek : 555-7893</b>  <b>Nancy : 555-7894</b> </p> <p> <b>11</b>  <b>555-7890</b>  <b>false</b>  <b>Dirk : 555-1234</b>  <b>Abel : 555-5678</b>  <b>Derek : 555-7893</b>  <b>Tabbie : 555-3456</b>  <b>Tom : 555-5555</b>  <b>Sam : 555-7890</b>  <b>Nancy : 555-7894</b>  <b>Bette : 555-7891</b>  <b>Carole : 555-7892</b>  <b>Miguel : 555-9015</b> </p>

In this section, you will experiment with sorted-array-based ADT Dictionary implementation.

#### Step – 1

Add the “SortedArrayDictionary.java” given in src folder. Experiment the same operations in Exercise - 1 by using sorted-array-based ADT Dictionary implementation.

Your Test.java
<pre>import java.util.Iterator; public class Main {     public static void display(DictionaryInterface&lt;String,String&gt;dictionary){     Iterator&lt;String&gt; keyIterator = dictionary.getKeyIterator();     Iterator&lt;String&gt; valueIterator = dictionary.getValueIterator();     while (keyIterator.hasNext() &amp;&amp; valueIterator.hasNext()){         System.out.println(keyIterator.next()+" : "+ valueIterator.next());     }     System.out.println(); }     public static void main(String[] args) {         SortedArrayDictionary phoneBook = new SortedArrayDictionary&lt;&gt;();         phoneBook.add("Dirk","555-1234");         phoneBook.add("Abel","555-5678");         phoneBook.add("Miguel","555-9012");         phoneBook.add("Tabbie","555-3456");         phoneBook.add("Tom","555-5555");         phoneBook.add("Sam","555-7890");         phoneBook.add("Reiss","555-2345");         phoneBook.add("Bette","555-7891");         phoneBook.add("Carole","555-7892");         phoneBook.add("Derek","555-7893");         phoneBook.add("Nancy","555-7894");          display(phoneBook);         System.out.println(phoneBook.getSize());         System.out.println(phoneBook.getValue("Sam"));         System.out.println(phoneBook.contains("Bo"));         phoneBook.remove("Reiss");         phoneBook.remove("Miguel");         phoneBook.add("Miguel","555-9015");         display(phoneBook);         phoneBook.clear();     } }</pre>

Your Output
<pre>Abel : 555-5678 Bette : 555-7891 Carole : 555-7892 Derek : 555-7893 Dirk : 555-1234 Miguel : 555-9012 Nancy : 555-7894 Reiss : 555-2345 Sam : 555-7890 Tabbie : 555-3456 Tom : 555-5555  11 555-7890 false</pre>

```

Abel : 555-5678
Bette : 555-7891
Carole : 555-7892
Derek : 555-7893
Dirk : 555-1234
Miguel : 555-9015
Nancy : 555-7894
Sam : 555-7890
Tabbie : 555-3456
Tom : 555-5555

```

### Step – 2

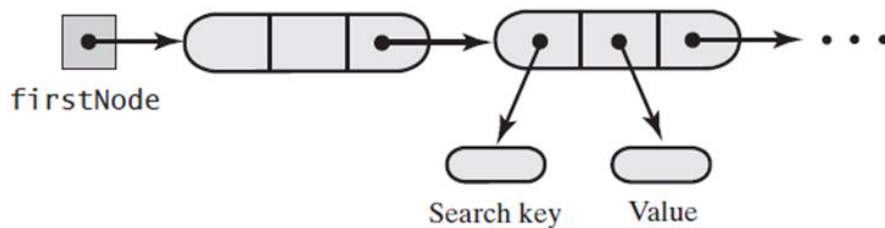
Compare the `locateIndex` methods in “`ArrayDictionary.java`” and “`SortedArrayDictionary.java`”. What is the difference between the two methods? Can we use binary search technique in “`ArrayDictionary.java`”, as well?

#### Your answer

When array dictionary uses sequential search, sorted array dictionary uses a method which is search until either it finds an entry containing key or discover that it doesn't not exist.

### Exercise – 3

In this section, you will experiment with linked-based ADT Dictionary implementation. One of the possible ways to use linked nodes to represent the entries is a chain of nodes that each reference a search key and a value as shown in the figure below.



### Step – 1

Add the “`SortedLinkedDictionary.java`” given in `src` folder.

### Step – 2

You are given `TR_SuperLeague_19_20.txt` that stores the Turkish Football Super League match results of 2019-2020 season. Using a `SortedLinkedDictionary` calculate and display the final points of all the teams. Indicate the champion team of the season.

Hint: Teams get 3 points for a win, one point for a draw, and zero for a defeat.

#### Your code

I run out of time 😞

--

Your Output