

## 2. LAB B – BUILDER

---

### Purpose

Given a text file with many words, we want to read it word by word and keep track of it in the following ways:

Create two builders:

1. Create a dictionary for these words, sorted alphabetically (eliminate duplicates)
2. Create a word counting map (for each word the number of times it appears in the text).

<p>Lab Scenario (<u>WHAT</u> to do )</p>	<p>Implement the builder design pattern for a word processing application.</p> <p>Develop the following:</p> <ol style="list-style-type: none"> <li>1. Base builder interface will be developed for easy maintenance</li> <li>2. Concrete builders : <ul style="list-style-type: none"> <li>- CounterDictionaryBuilder which counts words</li> <li>- ClassicDictionaryBuilder which sorts words</li> </ul> </li> </ol> <p>1 test class</p>
<p>Implementation Steps ( <u>HOW</u> to do it )</p>	<p><b>Code the DictionaryBuilder interface</b></p> <pre>public void addWord(String word); public Collection getDictionary();</pre> <p><b>Code the CounterDictionaryBuilder</b></p> <ul style="list-style-type: none"> <li>- implements DictionaryBuilder</li> <li>- contains a Map member variable (holds the words and their count)</li> <li>- implement addWord(String word) (method should update the count for the word added in the map)</li> <li>- implement getDictionary (simply return the map keySet)</li> </ul> <p><b>Code the ClassicDictionaryBuilder</b></p> <ul style="list-style-type: none"> <li>- implements DictionaryBuilder</li> <li>- contains a TreeSet member variable</li> <li>- implement addWord (since TreeSet already knows how to sort words all we have to do is add the word)</li> <li>- implement getDictionary (simply return the treeSet)</li> </ul> <p><b>Code the Test class</b></p> <p>Load a collection of strings into the builder and call getDictionary().</p> <p>Iterate on the collection and print it.</p>

## Review of the Builder

4 classes will be created:

1. Interface DictionaryBuilder will serve as base of concrete Builders
2. CounterDictionaryBuilder is a concrete builder class which counts the appearances of each word added
3. ClassicDictionaryBuilder is a concrete builder which sorts the words added, and maintains only one of each word (no duplicates)
4. DictionaryDirector tests the 2 builders. It loads words (optionally from a file) and adds them to the builder.

### 2.1.1. Create a new interface – DictionaryBuilder

Serves as the base for concrete builders. Makes the application flexible; new concrete builders can be added later on

1. Create the interface DictionaryBuilder
  - a. Implement public void addWord(String). Clients use this method to add words
  - b. Implement public Collection getDictionary(). Clients use this method to get the collection of words added different builders may return different collections

### 2.1.2. Create a new class CounterDictionaryBuilder

This builder will implement the DictionaryBuilder interface. The builder counts words added, by using a map data structure.

1. Create the class CounterDictionaryBuilder
2. Extend DictionaryBuilder
3. Implement a Map member variable  
Map will store each word with a counter
4. Implement the addWord(String) method  
Search for the word in the map
  - If the word isn't found, store it with a value of 1 indicating we have one of that word
  - If the word is found in the map, fetch its value, add 1 to it and store it (increasing the count for that word)
5. Implement the getDictionary() method  
Return the map entrySet: EntrySet is a collection containing pairs : all the keys and values from the map

### 2.1.3. Create a new class ClassicDictionaryBuilder

This builder will implement the DictionaryBuilder. The builder will sort words added, and will prevent / duplicates

1. Create the class ClassicDictionaryBuilder
2. Extend DictionaryBuilder
3. Implement a TreeSet member variable: TreeSet is a sorted set data structure
4. Implement the addWord(String) method: Add the word to the TreeSet
5. Implement the getDictionary() method: Return the TreeSet

### 2.1.4. Create the class DictionaryDirector

This class tests the builders we developed, by loading words, then adding them to the builder, and finally calling getDictionary() on the builder and examining the results

1. Create the class DictionaryDirector
2. Create the main method
3. Test the WordsCountBuilder
  - a. Create a WordsCountBuilder object
  - b. Add some words to it
  - c. Call getDictionary and print the collection received
  - d. Words should each have a count
4. test the ClassicDictionaryBuilder
  - a. Create a DictionaryBuilder object
  - b. Add some words to it
  - c. Call getDictionary and print the collection received
  - d. Examine the output – words should be sorted and unique

**Note:** The solution for this exercise is available in the ‘solutions’ directory