# **Lab #8**: Map

The main aim of the lab is to get familiar with some methods in **Map** interface.

==========

# Task 1: Word Count

==========

This task aims at using **implementations of Map** to write a program for counting words in a text file. Methods serving for such a program are defined in **MyWordCountApp.java** as follows:

```
public class MyWordCountApp {
	// public static final String fileName = "data/hamlet.txt";
	public static final String fileName = "data/fit.txt";
	// <word, its occurences>
	private Map<String, Integer> map = new HashMap<String, Integer>();

// Load data from fileName into above map (containing <word, its
occurences>)
	// using the guide given in TestReadFile.java
	public void loadData() {
		// TODO
	}

// Returns the number of distinct tokens in the file data/hamlet.txt or
fit.txt
	public int countDistinct() {
		// TODO
		return 0;
	}

// Prints out the number of times each unique-token appears in the file
	// data/hamlet.txt (or fit.txt)
	// In this method, we do not consider the order of tokens.
	public void printWordCounts() throws FileNotFoundException {
		// TODO
	}

// Prints out the number of times each unique token appears in the file
	// data/hamlet.txt (or fit.txt) according ascending order of tokens
	// Example: An - 3, Bug - 10, ...
	public void printWordCountsAlphabet() {
		// TODO
	}

}
```

Consider using ***appropriate implementations of Map*** to develop above methods.

==========

# Task 2: Map and Text file

==========

For a given text file, suppose it consists of the following lines:

```
What have I
 1     2  3                        <-- word position in text file
What have I
 4     5  6                        <-- word position in text file
What have I done to deserve this
 7     8  9  10  11   12     13    <-- word position in text file
```

For such an assumption, consider the following class using Map to store each pair ***<word, its positions>*.** We use ArrayList for the positions of each word in the text file.

```
public class TextAnalyzer {
//   <word, its positions>
     private Map<String, ArrayList<Integer>> map = new
HashMap<String, ArrayList<Integer>>();
//load words in the text file given by fileName and store into
map by using add method in Task 2.1.
// Using BufferedReader as in file TextFileUtils.java
public void load(String fileName) {
          // TODO
     }
}
```

We would form a map that maps each unique word to a list of word positions in the text file. NOTE: The word position for a word at the end of a line is stored as a negative integer rather than a positive integer so you can recreate the text file later when you iterate through the words in the map. (files **short.txt** or **length.txt** in **data** folder)

**Task 2.1:** In the following method, if the word is not in the map, then adding that word to the map containing the position of the word in the file. If the word is already in the map, then its word position is added to the list of word positions for this word.

Remember to negate the word position if the word is at the end of a line in the text file.

```
public void add(String word, int position) {
     // TODO

}
```

==========

**Task 2.2**: This method should display the words of the text file along with the positions of each word, one word per line, in **alphabetical order**.

```
public void displayWords() {
            // TODO
      }
```

Sample map for the text file above (order of words may vary):

```
Word        Word Position(s)
============================
WHAT        1, 4, 7
HAVE        2, 5, 8
I           -3, -6, 9
DONE        10
TO          11
DESERVE     12
THIS        -13

============================================================
```

**Task 2.3**: This method will display the content of the text file stored in the map.

```
public void displayText() {
            // TODO
      }
```

```
============================================================
```

**Task 2.4**: This method will return the word that occurs most frequently in the text file.

```
public String mostFrequentWord() {
      // TODO
      return null;
}
```