In this lab, you will learn to use **HashMaps**.

Problem statement: NewDictionary.txt has words and their meanings as shown in Figure 1. You need to write a program that reads this data into a data structure. It then takes user input to search for a word and then if the word is found, prints the word and its meaning(s) (Fig.2 and 3). This problem has two parts.

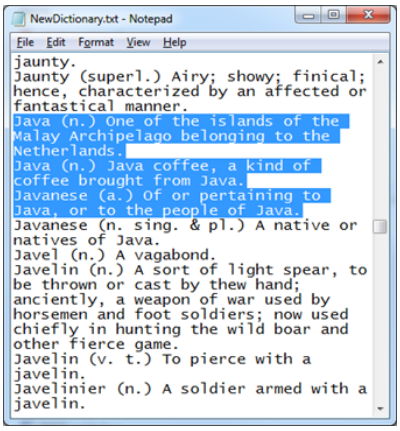

Figure 1: NewDictionary.txt

```
Enter search word
java
-----------WordList Search----------
Java: (n.) One of the islands of the Malay Archipelago belonging to the Netherlands.
Java: (n.) Java coffee, a kind of coffee brought from Java.
******************************
-----------SingleMap Search----------
Java: (n.) Java coffee, a kind of coffee brought from Java.
******************************
-----------MultiMap Search----------
Java: (n.) One of the islands of the Malay Archipelago belonging to the Netherlands.
Java: (n.) Java coffee, a kind of coffee brought from Java.
******************************
```

Figure 2: Existing word search

```
Enter search word
xyz
-----------WordList Search------
Sorry! xyz not found!
******************************
-----------SingleMap Search-----
Sorry! xyz not found!
******************************
-----------MultiMap Search------
Sorry! xyz not found!
******************************
```

Figure 3: Missing word search

Part1 (required for 10 points): In this part, you will use an ArrayList and a HashMap to store and search for the words.

- Use the ArrayList **wordlist** to search for words and their meanings. wordList requires searching all Words in the list.
- Use HashMap<String, Word> named **singleMap** and then search for words and their meanings. As singleMap can have only one meaning stored as its value, words with more than one meaning will have only their last meaning stored as the previous ones will get overwritten.

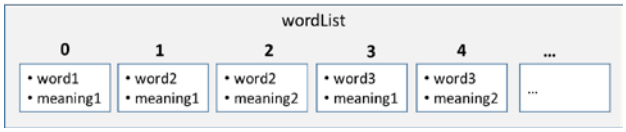

Figure 4: ArrayList. Note that word2 and word3 have two meanings each

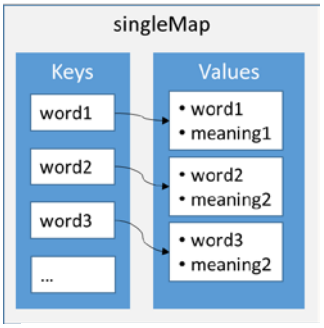

Figure 5: HashMap. Note that word2 and word3 can keep only the meaning loaded last

Part2 (optional for 2 bonus points): In this part, you will use a multimap to store and search for words. A multimap is like a Map but it can map each key to multiple values. (More at: https://docs.oracle.com/javase/tutorial/collections/interfaces/map.html)

- **Use** HashMap<String, List<Word>> named **multiMap** to store data and then search for words and their meanings. Note that the value in key-value pair is a List of strings, which means it can store multiple meanings. So if a word has more than one meaning, you need to keep adding the meaning to the list attached to the key that is the word, thereby preserving the meanings found previously.

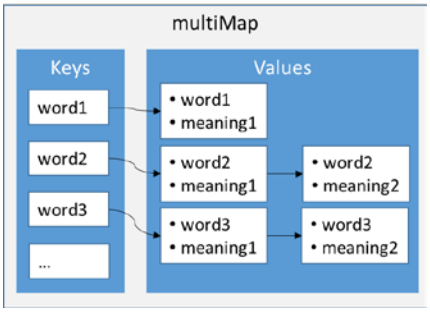

Figure 6: MultiMap. Note that word2 and word3 have two meanings each

Note: Sometimes when downloading txt files from the web (e.g. Canvas), some characters do not transfer well, corrupting the text-file. To check that the entire file has been downloaded and read correctly for you, insert the following statement as the last line in loadWordList() method.

System.out.println(wordList.size());

It should print 36,964. If your number comes out different, then open NewDictionary.txt in Eclipse and go to the word number it says. Search for some strange looking character in the text. Once you find it, copy and then use the edit to copy-replace all its occurrences in the file. This should solve the problem.

Solution Design: You are given Dictionary.java, and Word.java. Word.java is fully coded. Dictionary.java has 8 methods, of which main(), loadWords(), and getWord() methods are fully coded. loadWords() method reads data from txt file and loads the wordlist. It uses getword() as a helper method to create Word objects, which you need not worry about.

You need to write the remaining five methods:

1. loadSingleMap(): populate singleMap with words from wordList
2. loadMultiMap(): populate multiMap with words from wordList
3. searchWordList(): Search for a word in wordlist
4. searchSingleMap(): Search for a word in singleMap
5. searchMultiMap(): Search for a word in multiMap

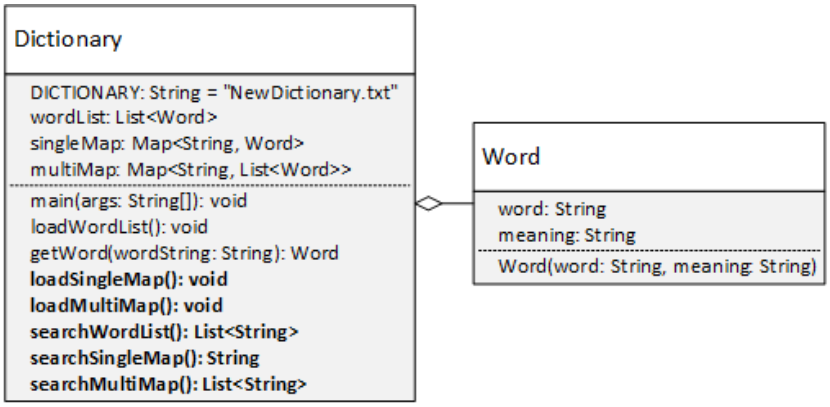

Figure 7: Class diagram

Lab8 setup

1. Download Dictionary.java, Word.java, TestDictionary.java, and NewDictionary.txt from Canvas
2. Create a package named lab8 in Labs project.
3. Import java files into this package and NewDictionary.txt into the 'Labs' project folder
4. Complete Dictionary.java and test it using TestDictionary.java
5. Write your name and Andrew ID in the first line of Dictionary.java
6. Submit only Dictionary.java on Canvas.

Rubric:

Part1: 10 points	**Part2: 2 points**
Test1 to test6: 6 points (1 point each)	Test7 to test10: 1 point (0.25 points each)
Console I/O • 1st part: 2 points • 2nd part: 2 points	Console I/O • 3rd part: 1 point