Project Description

We want to develop a tool for storing a list of words. The words will be used in an application that plays a word game. The list of words is loaded from a dictionary of words stored in a file.

The words in the file are stored as a series of lines. Each line can be blank, or it can contain one or more words. If a line contains more than one word then each word is separated from the next using a comma. This is commonly known as Comma Separated Values format or CSV format. As a consequence, the word files used in this project all have .CSV extensions.

> **TIP**: The String class provides a method called **`split`**, which can be used to partition a String using specified delimiters (e.g. comma).

For example, a file called SAMPLE.CSV might contain the following lines.

> NOTE: The line numbers are included here for convenience – they are NOT included in the file.

SAMPLE.CSV

1. `electric,amusement,brass,elastic,hospital,rail,hole,trouble,mark`
2.
3. `increase`
4. `but  ,art,jump,  hanging,shame`
5. `edge,iron,take,opposite,thin,sun,fixed,field,drawer,twist,curtain,soap`
6. `card`
7. `happy`
8.
9. `attack,drain, after,interest,cold,summer  ,apple,soft,connection,ray`
10. `fire,mine,stomach,steel,worm,reason,nation,knot,waste,father,train,hope`
11. `unit,destruction,town,credit,hair,chain`
12. `support,waiting,disease,name,bread,violent,plough,sea,tree,knee,finger`
13. `writing`
14. `bite`

> **TIP**: Words may be preceded or followed by spaces. For example, line 4 above includes the word "`but  `" and "`  hanging`"; and line 9 has "` after`" and "`summer `". The String class provides a method called **`trim`**, that can be used to remove leading and trailing spaces from a string.

The following is a skeletal structure for the Dictionary class

```
public class Dictionary {

  private ArrayList<String> words ;

  public Dictionary(String filepath, int shortest, int longest) {

   // This method reads words from the specified file

   // and adds them to the 'words' list. See details below.
```

```
    }
    // You will also need to include the other methods specified below.
}
```

Methods Required

The class should support/provide the following operations

- The Dictionary class constructor method has the following header

  `public Dictionary(String filepath, int shortest, int longest)`

  The method reads the words from the specified CSV file. Only words with lengths between shortest and longest INCLUSIVE should be added to the 'words' list. All words stored in the list should be in uppercase. Duplicates are not allowed so each word should only appear once in the list. The words in the dictionary should be stored in alphabetical order.

  If there are no words in the file that match the specified length requirements then the method should return an empty list.

- The class provides a method to add additional words to the list using the following header

  `public boolean add(String word)`

  If the specified word is not already in the list the method adds it to the list and returns true. Otherwise it returns false.

  NOTE: When new words are added the list should be maintained in alphabetical order.

  TIP: The `Collections` `binarySearch` operation might be helpful here.

- The class provides a method to randomly choose a word from the dictionary using the following header

  `public String nextWord()`

  If the dictionary is empty then the method returns an empty string (i.e. ""). Otherwise the method returns a randomly chosen word from the list.

- The class provides a method to verify if a specified word appears in the dictionary using the following header

  `public boolean inDictionary(String word)`

  The method returns true if the dictionary contains the word. Otherwise it returns false. NOTE: The word may be passed in lower, upper or mixed case.

Driver Class

The DictionaryDriver class will be used to test the Dictionary class. The DictionaryDriver class is passed three 'command line' parameters as follows

`filepath` is the filepath for the CSV file containing the words

`shortest` is the length of the shortest word to be used

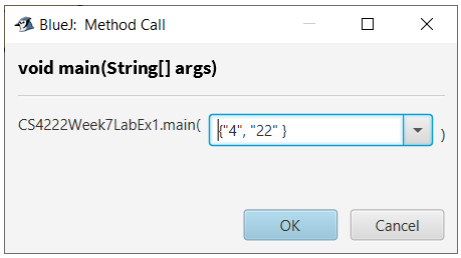
`longest` is the length of the longest word to be used

For example, when using the DictionaryDriver program from the command line you have to type something like this

```
java DictionaryDriver basicenglish.csv 3 10
java DictionaryDriver   technicalterms.csv   4     15
java DictionaryDriver middleenglish.csv 5 5
```

The three parameters MUST BE PRESENT. If the command line does not contain the three parameters then the program should display a suitable message and stop. You can assume that when the three parameters are present they are specified in the correct order.

NOTE: If you are using BlueJ then you can specify 'command line' parameters inside the curly brackets in the Method Call dialog box. Remember, ALL the parameters have to be specified as Strings, even the numeric ones. See the examples in the Lab Sheet for Week 7 which include the following



Submission Requirements

Your solution to the assignment should be submitted on Sulis on or before **16h00 Tuesday 5th April 2022**.

You should submit **TWO** files (1) **Dictionary.java** that implements the operations specified for the class, and (2) **DictionaryDriver.java** that tests the various operations to verify that they work correctly. In each file you should include prominent comments that contain your ID number and your Name. Sulis will accept any filenames you use but it is extremely important that you adhere to the file name conventions.

You should NOT submit zip files.

This part of the assignment is worth 25% of the total of 100%.

The marks will be allocated as follows

Component	Allocation
Dictionary Constructor	9
add method	5
nextWord method	2
inDictionary method	1
Driver program	8
Total	25