W05 Practical

To Access main Program

```
git checkout master
```

To access Extension 1

```
git checkout Extension1
```

To access Extension 2

```
git checkout Extension2
```

Overview

Initial Practical

The specification required that a program which takes in two arguments.

- `Argument 1`: The location of the diction of words.
- `Arguments 2`: A string of text

The program then removes 1 consonant from each word in the string of text and checks that that string of that had the consonant removed is contained in the dictionary of words and if it is then it adds it to the list of alternatives for that word. The program then outputs all the alternatives for the words to the

Problem Decomposition

- Read the dictionary file
- Make the the dictionary of words into a set in which comparisons can be made to.
- Read the text.
- Find consonants in the text.
- Delete consonants and compare that word to words in the dictionary.
- It has also been noted words in the dictionary vary in capitalisation so for valid comparison words in the dictionary and string will never to be lowercase before comparison. This is assumption is based on results from `stacscheck`.

Extension 1: Introduction of same process for vowels

The extension asks that a second version of the program be produced which lists lost vowel alternatives.

Assumptions

- second version indicates it has alternative functionality to the original so this is not an additive incremental addition of functionality but alteration of a new program to do the exact same process for

vowels.

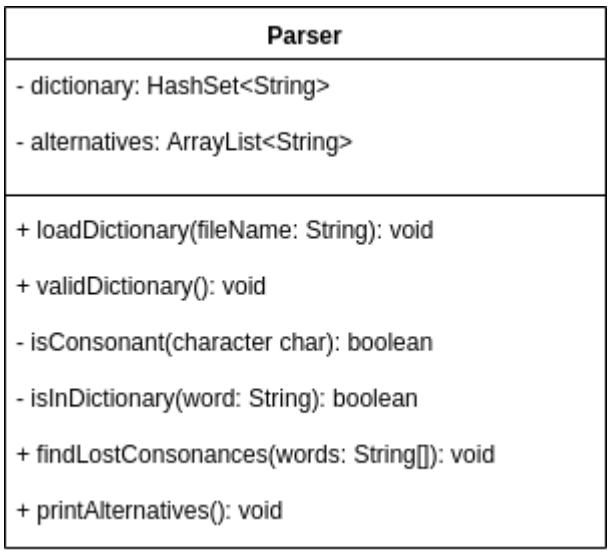
Extension 2: Adding Consonants

The program adds consonants to a word an tries to determine if they are present in the dictionary.

Assumptions

- The must find all the possible ways to add constants around every character in a word.
- compares them to those in the supplied dictionary.
- Multiple constants can be added at once.

Design



Parser

A two class implementation was decided to fulfil the requirements of the practical. To increase code readability the functionality in which the program requires that implemented in the Parser. The general features in which the program required were abstracted into functions. Then these functions could then be called in the main class. This made the program more readable as the functionality currently occurring could be deduced from name of function and parameters.

It was decided to made the dictionary into a `HashSet` as:

- elements in the dictionary will be unique.
- the order of the elements in dictionary doesn't matter.

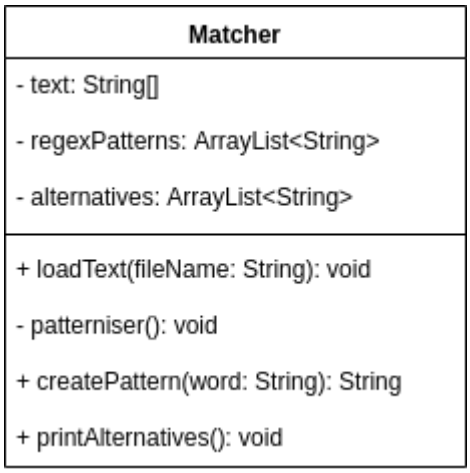
it was also decided to store list of alternatives that need printing in an `ArrayList` instead of directly printing them when they are found. This is because I could the get the length of the `ArrayList` at the end on finding Consonants is ran to determine how many alternatives were found.

Extension: vowel implementation

The `isConsonant` method replaced with a `letterType` method and this method now takes in two parameters `letterType(char character, boolean isVowel)` if `isVowel` is true the method shall search for vowels and if it is false it shall search for consonants. It shall return true or false if the type it is search for is found. It was design this was to allow for extension of the functionality of the program while making minimal changes to the original.

The user now enters third argument `-v` if they wish to search for vowels.

Extension: Matcher



Since the functionality of this extension was greatly different to the original it was implemented as a new class. REGEX was used to determine where to add consonants. Every word character in the word of a text is surrounded by regex and searched in the file. e.g.

"WAKE ME UP"

```
{"[b-df-hj-np-tv-z]*[w][b-df-hj-np-tv-z]*[a][b-df-hj-np-tv-z]*[k][b-df-hj-np-tv-z]*[e]]
[b-df-hj-np-tv-z]*", "[b-df-hj-np-tv-z]*[M][b-df-hj-np-tv-z]*[E], [b-df-hj-np-tv-z]*[U]
[b-df-hj-np-tv-z]*[P][b-df-hj-np-tv-z]*"}
```

For every word in the dictionary if it matches the word pattern then it is selected to be added to the list of alternatives.

To make the program run faster the since the file is stream read and each process happens with respect to each line in the dictionary.

Testing

Initial Specification

Test Case 1: Incorrect Arguments supplied

Here it is tested that the program properly reacts to the incorrect number of arguments supplied

Expected Output

It is expected that the program outputs an error message along with the number of arguments supplied.

Actual Output

```
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants
Expected 2 command line arguments, but got 0.
Please provide the path to the dictionary file as the first argument and a sentence as the second argument.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants arg1
Expected 2 command line arguments, but got 1.
Please provide the path to the dictionary file as the first argument and a sentence as the second argument.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants arg1 arg2 arg3
Expected 2 command line arguments, but got 3.
Please provide the path to the dictionary file as the first argument and a sentence as the second argument.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $
```

Test Case 2: Incorrect file Path

Here is it is test that the program correctly reacts to when a file which does not exist is made into dictionary for the program.

Expected Output

It is expected that the program outputs an error message along with incorrect path that was let in

Actual Output

```
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ ls
LostConsonants.class  LostConsonants.java  Parser.class  Parser.java
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants this_path_does_not_exist "honestly it doesnt"
File not found: this_path_does_not_exist (No such file or directory)
Invalid dictionary, aborting.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $
```

Test Case 3: Correct Functionality Achieved

Here it is tested the program gives the correct output given valid data.

Expected Output

I made a new file containing the 5 possible variations of words produced when consonants are removed and one which should never be found since it not produced when consonant is removed.

1. Expect all 6 alternatives since the word `apple chair` has been entered.
2. No alternatives should be found as none of these variations exist in `nope`.
3. Expect all 6 alternatives since the program cares not about punctuation and capitalisation.
4. Expected 3 alternatives as `aple` can be produce for removal of either p and `appe` from the removal of an L.

Actual Output

```
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ touch data.txt
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ echo "
aple
appe
hair
cair
char
chai
" > data.txt
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt "apple chair"
aple chair
aple chair
appe chair
apple hair
apple cair
apple chai
Found 6 alternatives.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt "NOPE"
Could not find any alternatives.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt "ApPlE,,, cHaIr..."
APlE,,, cHaIr...
AplE,,, cHaIr...
ApPE,,, cHaIr...
ApPlE,,, HaIr...
ApPlE,,, caIr...
ApPlE,,, cHaI...
Found 6 alternatives.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt "aPPLE"
aPLE
aPLE
aPPE
Found 3 alternatives.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $
```

Extension 1

To test if the program correctly finds vowels.

Expected Output

Here it can be seen that 5 possible [note: vale appears twice] variations of the text "Water has value." has been expected that all 5 are found.

Actual Output

```
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ touch data.txt
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ echo "
wter
watr
hs
vale
vlue
vale
LOOOOOOOOOOOOOOOL" >data.txt
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt "Water has value." -v
Wter has value.
Watr has value.
Water hs value.
Water has vlue.
Water has vale.
Found 5 alternatives.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt "Water has value." -v
```

Extension 2

To test if the extended program can correctly add find consonant alternatives in the dictionary.

Expected Output

I have 6 possibilities of ways in which consonants can be added around every character in a text and one in which it is identical. It is expected only six alternatives are found.

Actual Output

```
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ javac LostConsonants.java
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ touch data.txt
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ echo "
banana
aaab
baaa
abaa
aaba
AAA
NaNaNa
" > data.txt
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $ java LostConsonants data.txt aaa -A
banana
aaab
baaa
abaa
aaba
nanana
Found 6 alternatives.
    @pc5-034-l:~/Documents/CS1003/Practicals/W05Practical/src $
```

Stacscheck Output

```
    @pc5-036-l:~/Documents/CS1003/Practicals/W05Practical/src $ stacscheck /cs/studres/CS1003/Practicals/W05/Tests
Testing CS1003 W05
- Looking for submission in a directory called 'src': Already in it!
* BUILD TEST - basic/build : pass
* COMPARISON TEST - basic/01_badargs/01_noArgs/prog-expected.out : pass
* COMPARISON TEST - basic/01_badargs/02_1arg/prog-expected.out : pass
* COMPARISON TEST - basic/01_badargs/03_3args/prog-expected.out : pass
* COMPARISON TEST - basic/01_badargs/04_badfile/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/accident/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/barking/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/bride/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/doorbell/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/leopard/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/permanently/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/prince/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/therapist/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/tracking/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/truth/prog-expected.out : pass
* COMPARISON TEST - basic/02_words/william/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/00_dog/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/01_dog/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/02_hunter/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/03_leopard/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/04_cruel/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/05_woks/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/06_roubles/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/07_brie/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/08_fog/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/09_fiction/prog-expected.out : pass
* COMPARISON TEST - basic/03_sentences/10_accident/prog-expected.out : pass
* INFO - basic/04_style/infoCheckStyle : pass
--- submission output ---
Starting audit...
Audit done.
---

28 out of 28 tests passed
    @pc5-036-l:~/Documents/CS1003/Practicals/W05Practical/src $
```

Evaluation

The specification required that a program which would take in two command line arguments:

1. The location of the dictionary file. If the file was unable to found then an error message would be produced. Through testing it can be seen that the program is able to produce this output.

```
File not found: [fileName] (No such file or directory)
Invalid dictionary, aborting.
```

2. The string being processed.

If these arguments are not received that an appropriate error message be produced and sent to the user.

1. As can be seen from testing when no arguments or a greater amount that are required are entered the error message. Through testing it was able to be seen that the program is able to produce this error message.

```
Expected 2 command line arguments, but got [args.length].
Please provide the path to the dictionary file as the first argument and a sentence
as the second argument.
```

As can be seen from testing the program is able to successfully delete all the constants in the string of input produced and delete them and then determine if an alternative for that word exists in the dictionary of words and also then output the number and all these alternatives to the terminal.

Conclusion

In this practical a program which was able to delete consonants and compare them to words in the dictionary was produced.

Difficulties

- It was found when a word was compared to those in the dictionary some would not compare even though they were present in the dictionary. This was found to be cause by the capitalisation of the words in the dictionary and the capitalisation of the input. So before comparison both were made lowercase to allow all the words which exists in the dictionary to be found.
- It was found when a words which were followed with punctuation were compared to those in the dictionary some would not be found to be in the dictionary. So to fix this error all punctuation had to be removed before the comparison.

Given More Time

- A GUI could be have been prepared to allow for easier interaction of the user with the program could have been prepared.
- Stacscheck for each extension could have been prepared to fully test their functionality.