Overview

For the second practical we had to write a program which could analyse patterns in which characters occur in a file of text, it was required that the program should count the number of occurrences of all the possible groups of three consecutive characters. It was required that the program ignore punctuation and numerals as well as converting all the letter to lower case. Where a trigram was at the end of a word but consisted of only 2 letters an underscore should be added.

Design

I have put all the program comments in the extension project.
Please not the program arguments should be as follows:
test.txt nGrams.csv wordGrams.csv dictionary.txt UTF-8

The design of my project for reading data in from a file bears, close resemblance to my previous practical, due the simplicity and efficiency of the structre in my previous did not feel the need to write a new structure

A new instance of the nGramInfo class is initiated, and the print buffers are set up. A while loop is then entered; the terminating condition is met when the line of the text file is null e.g. end of file. When the while loop is entered the function from the nGram class is called, 'hasNextnGram'. This function returns a Boolean value, which is true when the end of the file is met, otherwise the returned value will be false. Within this function another function is called, findNextnGram. The find findNextnGram creates a new instance of the nGram class, which is a sub class (nested class) of the nGramInfo class. The nGram class contains the following attributes: nGramString, word, predictedWord and occurrences. The instaces of this class hold the information on each n-gram found. The findNextnGram then reads in a line of the text file. The line is split up into words and each word is converted to lower case and all non letter characters are removed. The words are then added to an array. A loop then goes through each word in the array, as long as the item in the array is not just a blank space, and hands the word to numerous other function including: 'findNGram' and 'checkNGramOccurrences'. The find nGram and findWordGram function are quite similar. Both functions take an input string, which is split up into either characters or words. The findNGram function is also recursive for efficiency. The start character or word is then removed from the front of the string and the input is then handed back into the function. Based on what the user has input (x) each collection of x characters or words are finally handed to another function called 'checkNGramOccurrences'. However, because word-grams in theory could go over multiple lines, I first needed to read every word in the I put file and add it to an array, then with all the words in the array I can look for word-grams. The 'checkNGramOccurrences' function will check an arraylist of object nGram, for object with the same string as the input. If one is found the attribute 'occournces' is incremented for that object. Otherwise a new nGram object is created, where the nGramString attribute is set to the input of the function and then added to the array list. Finally, when all the nGrams and word grams are found the array lists are ordred based on occournecs and the data is output to a CSV file.

The garbage collection system built into the java programming language means the tempory instances of the nGram subclass are automatically deleted if they are not added to the array of nGrams.

I have tried also to implement predictive text functionality, which works by having the user input an n-gram via the console, the program then adds all a dictionary of words each with an assigned score (based on how frequently they are used in common English) to an array. When the user inputs a n-gram the program looks through the array to try and find any words staring with the input n-gram. If there are any matches, the found words are then ordered by score and the top 5 are output to the console.

Testing

To test the core functionality of my program I will use the various files supplied as an input and make sure the generated nGrams are firstly the correct size and also that the occurrences of each of the different nGrams found match the actual number of occurrences in the input file.

To carry out erroneous and extreme testing I will do the following tests:

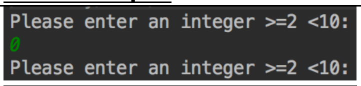
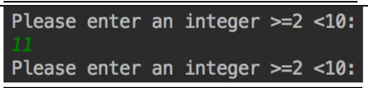
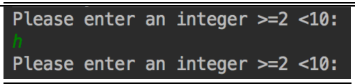
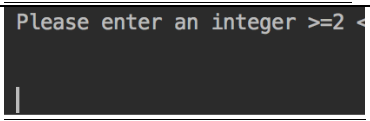
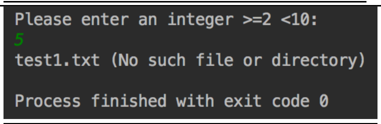
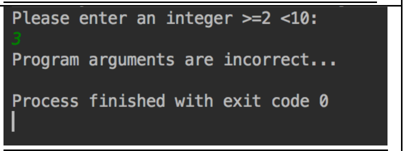
Try and input an invalid file

Try and input an invalid number when the user input is required.

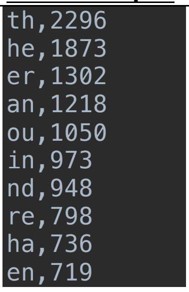
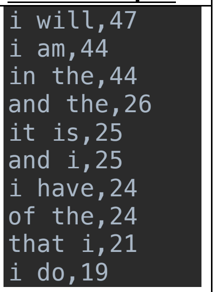
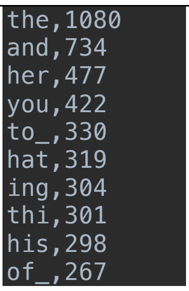
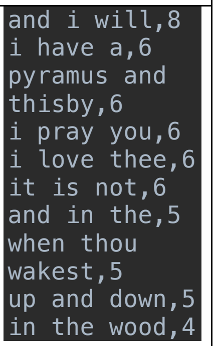
To test the word prediction functionality, I will output the actual word the program trying to be predicted and the word which was predicted. I can then compare the two words and calculate the accuracy of the predictions over multiple different files.

Examples

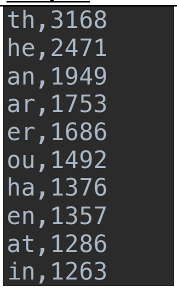
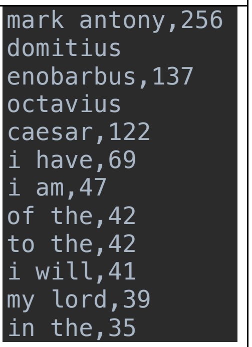
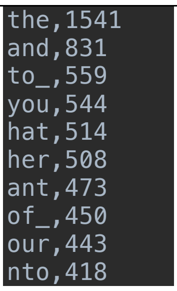
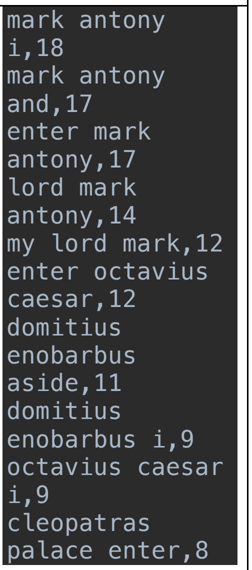
Test with errounous input

Input	**Expected output**	**Actual output**
nGram value: 0	Ask user to re-enter a valid number	 Please enter an integer >=2 <10: 0 Please enter an integer >=2 <10:
nGram value: 0	Ask user to re-enter a valid number	 Please enter an integer >=2 <10: 11 Please enter an integer >=2 <10:
nGram value: h	Ask user to re-enter a valid number	 Please enter an integer >=2 <10: h Please enter an integer >=2 <10:
nGram value: <blank>	Ask user to re-enter a valid number	 Please enter an integer >=2 <
Corrupt input file	Notify the user the input file is not valid	 Please enter an integer >=2 <10: 5 test1.txt (No such file or directory) Process finished with exit code 0
Incorrect number of program arguments	Notify the user the program arguments are incorrect	 Please enter an integer >=2 <10: 3 Program arguments are incorrect... Process finished with exit code 0

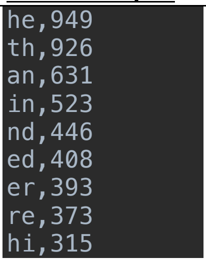
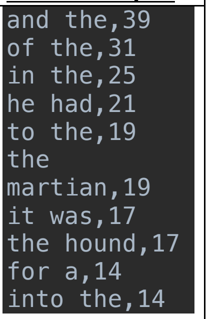
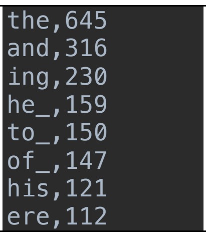
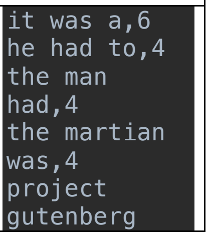
Test with file: A Midsummer Night's Dream

Input	**Expected output**	**Actual output**	**Expected output**	**Actual output**
nGram value:2	th,2296 he,1873 er,1302 an,1218 ou,1050 in,973 nd,948 re,798 ha,736 en,719	 th,2296 he,1873 er,1302 an,1218 ou,1050 in,973 nd,948 re,798 ha,736 en,719	i will,47 i am,44 in the,44 and the,26 it is,25 and i,25 i have,24 of the,24 that i,21 i do,19	 i will,47 i am,44 in the,44 and the,26 it is,25 and i,25 i have,24 of the,24 that i,21 i do,19
nGram value:3	the,1080 and,734 her,477 you,422 to_,330 hat,319 ing,304 thi,301 his,298 of_,267	 the,1080 and,734 her,477 you,422 to_,330 hat,319 ing,304 thi,301 his,298 of_,267	and i will,8 i have a,6 pyramus and thisby,6 i pray you,6 i love thee,6 it is not,6 and in the,5 when thou wakest,5 up and down,5 in the wood,4	 and i will,8 i have a,6 pyramus and thisby,6 i pray you,6 i love thee,6 it is not,6 and in the,5 when thou wakest,5 up and down,5 in the wood,4

Test with file: Antony & Cleopatra

Input	**Expected output**	**Actual output**	**Expected output**	**Actual output**
nGram value:2	th,3168 he,2471 an,1949 ar,1753 er,1686 ou,1492 ha,1376 en,1357 at,1286 in,1263	 th,3168 he,2471 an,1949 ar,1753 er,1686 ou,1492 ha,1376 en,1357 at,1286 in,1263	mark antony,256 domitius enobarbus,137 octavius caesar,122 i have,69 i am,47 of the,42 to the,42 i will,41 my lord,39 in the,35	 mark antony,256 domitius enobarbus,137 octavius caesar,122 i have,69 i am,47 of the,42 to the,42 i will,41 my lord,39 in the,35
nGram value:3	the,1541 and,831 to_,559 you,544 hat,514 her,508 ant,473 of_,450 our,443 nto,418	 the,1541 and,831 to_,559 you,544 hat,514 her,508 ant,473 of_,450 our,443 nto,418	mark antony i,18 mark antony and,17 enter mark antony,17 lord mark antony,14 my lord mark,12 enter octavius caesar,12 domitius enobarbus aside,11 domitius enobarbus i,9 octavius caesar i,9 cleopatras palace enter,8	 mark antony i,18 mark antony and,17 enter mark antony,17 lord mark antony,14 my lord mark,12 enter octavius caesar,12 domitius enobarbus aside,11 domitius enobarbus i,9 octavius caesar i,9 cleopatras palace enter,8

Test with file: Alls Well That Ends Well

Input	**Expected output**	**Actual output**	**Expected output**	**Actual output**
nGram value:2	he,949 th,926 an,631 in,523 nd,446 ed,408 er,393 re,373 hi,315	 he,949 th,926 an,631 in,523 nd,446 ed,408 er,393 re,373 hi,315	and the,39 of the,31 in the,25 he had,21 to the,19 the martian,19 it was,17 the hound,17 for a,14 into the,14	 and the,39 of the,31 in the,25 he had,21 to the,19 the martian,19 it was,17 the hound,17 for a,14 into the,14
nGram value:3	the,645 and,316 ing,230 he_,159 to_,150 of_,147	 the,645 and,316 ing,230 he_,159 to_,150 of_,147 his,121 ere,112	it was a,6 he had to,4 the man had,4 the martian was,4 project gutenberg ebook,3	 it was a,6 he had to,4 the man had,4 the martian was,4 project gutenberg

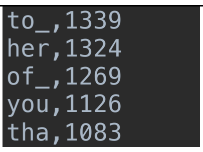
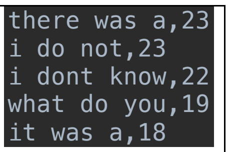
	his,121 ere,112 was,100 in_,98	was,100 in_,98	duel on syrtis,3 top of the,3 a long time,3 but he didnt,3 it would be,3	ebook,3 duel on syrtis,3 top of the,3 a long time,3 but he didnt,3 it would be,3

Test with file: <u>Duel on Syrtis</u>

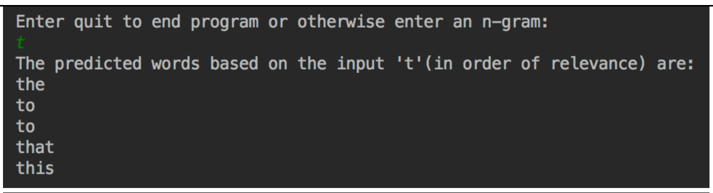
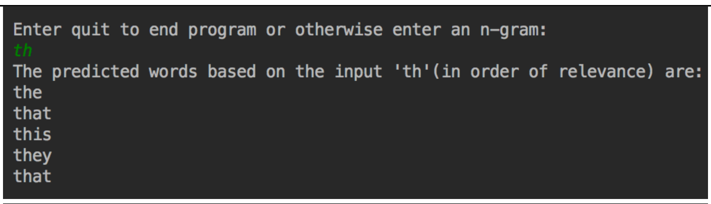
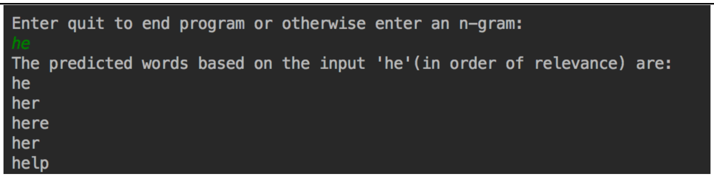
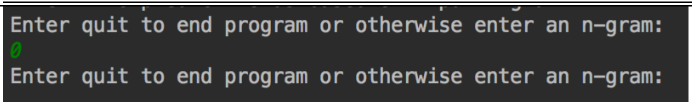
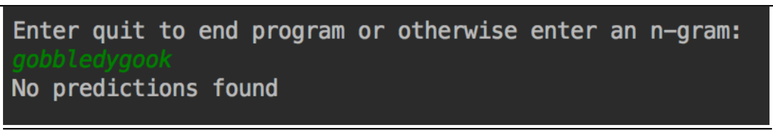
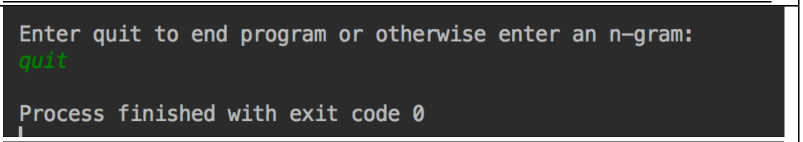
Input	Expected output	Actual output	Expected output	Actual output
nGram value:2	th,7522 he,6802 in,4052 er,3859 an,3758 ha,3442 ou,3307 re,3296 on,2950 at,2930	th,7522 he,6802 in,4052 er,3859 an,3758 ha,3442 ou,3307 re,3296 on,2950 at,2930	of the,434 in the,250 it was,170 it is,156 i have,146 the moor,139 sir henry,139 upon the,133 to the,131	of the,434 in the,250 it was,170 it is,156 i have,146 the moor,139 sir henry,139 upon the,133 to the,131
nGram value:3	the,4466 and,1919 of_,1599 ing,1408 to_,1407 hat,1381 tha,1250 you,1068 his,1052 her,1023	the,4466 and,1919 of_,1599 ing,1408 to_,1407 hat,1381 tha,1250 you,1068 his,1052 her,1023	upon the moor,45 that it was,27 of the moor,26 out of the,23 it was a,22 on the moor,22 sir henry baskerville,22 i could not,20 in front of,19 it is a,19	upon the moor,45 that it was,27 of the moor,26 out of the,23 it was a,22 on the moor,22 sir henry baskerville,22 i could not,20 in front of,19 it is a,19

Test with file: <u>Hound of the Baskervilles</u>

Input	Expected output	Actual output	Expected output	Actual output
nGram value:2	th,6728 he,6363 in,4612 er,3952 re,3157 ha,3055 ed,3024 ou,2987 an,2903 at,2719	th,6728 he,6363 in,4612 er,3952 re,3157 ha,3055 ed,3024 ou,2987 an,2903 at,2719	of the,302 in the,228 it was,180 on the,160 to the,140 it is,124 mrs inglethorp,115 i was,109 at the,99 had been,97	of the,302 in the,228 it was,180 on the,160 to the,140 it is,124 mrs inglethorp,115 i was,109 at the,99 had been,97
nGram value:3	the,3777 ing,1963 and,1448	the,3777 ing,1963 and,1448 hat,1346	that it was,32 i did not,27 out of the,27	that it was,32 i did not,27 out of the,27 one of the,24

	hat,1346 to_,1339 her,1324 of_,1269 you,1126 tha,1083	to_,1339 her,1324 of_,1269 you,1126 tha,1083	one of the,24 there was a,23 i do not,23 i dont know,22 what do you,19 it was a,18	there was a,23 i do not,23 i dont know,22 what do you,19 it was a,18

Test word prediction

Input	**Expected output**	**Actual output**
Input nGram: t	the to to that this	Enter quit to end program or otherwise enter an n-gram: t The predicted words based on the input 't'(in order of relevance) are: the to to that this
Input nGram: th	the that this they that	Enter quit to end program or otherwise enter an n-gram: th The predicted words based on the input 'th'(in order of relevance) are: the that this they that
Input nGram: he	he her here her help	Enter quit to end program or otherwise enter an n-gram: he The predicted words based on the input 'he'(in order of relevance) are: he her here her help
Input nGram: 0	Program should wait for valid input	Enter quit to end program or otherwise enter an n-gram: 0 Enter quit to end program or otherwise enter an n-gram:
Input nGram: gobbledygook	No predictons found	Enter quit to end program or otherwise enter an n-gram: gobbledygook No predictions found
Input nGram: quit	Program exits	Enter quit to end program or otherwise enter an n-gram: quit Process finished with exit code 0

Evaluation

After testing my program, I pleased to say it works as expected and is able to process large data files quickly and efficiently. I have used multiple classes and universal methods which take in various parameters making the program overall more modular and adaptable.

The output files that are generated contain the correct required data as well as other information that is generated based on the input files.

Conclusion

In conclusion I'm satisfied with end program, it is able to carry out the required tasks with ease and is able to handle erroneous data without crashing. The added extensions massively increase the overall functionality and makes the program far more flexible. The word prediction extension did however prove troublesome due to the ambiguity of the extension requirement.