Documentation

Requirements

Functional Requirements

[These should deal with the function of the system, i.e. this will be the main body of requirements that clearly specify what requirements the system shall, should or may meet]

R. 1. The program shall translate from English to French.
  R.1.1 Shall translate from both text file and command line input
R. 2. The program shall translate from French to English.
  R.2.1 Shall translate from both text file and command line input
R. 3. Program shall have a navigation menu.
R. 4. Program shall have a help/instruction.
R. 5. Program shall allow the user to read from/write to file
  R. 5.1 Shall allow user to read/write file after any changes made to the initial dictionary.
R. 6. Program shall have a toggle option to allow user to add unfound words to the dictionary.
R. 7. Program shall allow the user to remove words or phrases from the dictionary.
R. 8. Program shall allow the user to display the dictionary.
R. 9. Program shall allow user to run the program without IDE.
R. 10. Program may run the program on GUI.
R. 11. Program may have an additional language support (Italian)
R. 12. Program may deal with grammar laws.

Non-functional Requirements

[These relate to non-functional aspects of the system such as usability, performance or system hardware constraints (e.g. minimum hardware specification), required software etc.]

R. 1. Shall run on Java Runtime Environment (JRE)
R. 2. Shall be in English.
R. 3. Should comply with Copyright, Design & Patent Act 1988.
  Rational Does not use copyrighted material.
R. 4. Should run on up to date operating system.

Use Cases

[Consider who or what will use the system and how they will interact with it. Give a descriptive sentence for each use case then the USER: SYSTEM dialogue]

Data Dictionary: User wants to translate X language text file to Y language			**Alternatives**
1	User	Starts the program	
2	System	Opens start menu	
3	User	Selects translate X language to Y language via text file	
4	System	Ask user to select the file	
5	System	Open file browser	
6	User	Selects file	
7	System	Translate file	
8	System	Write translation to another text file	A
9	System	Calculate and displays words per second (translation speed)	

Data Dictionary: User wants to translate X language text file to Y language			**Alternatives**
1	User	Starts the program	
2	System	Opens start menu	
3	User	Selects translate X language to Y language via text file	
4	System	Ask user to select the file	
5	System	Open file browser	
6	User	Selects file	
7	System	Translate file	
8	System	Write translation to another text file	A
9	System	Calculate and displays words per second (translation speed)	

A		
A1	System	Informs user of any unfound words
A2	System	Ask user if they would like to add unfound words to dictionary
A3	User	Has the option to add words
A4	System	Adds user inputs to dictionary
A5	System	Continues from step 8

Summary & Self-evaluation:

As part of this group project, our group decided to use Binary trees as the main data structure for the dictionary. The main language we have chosen for the translation is English to French and we have created binary trees for all the languages we have used i.e.: English to French, French to English, Italian to English and English to Italian. But we have also used hash code which provides a unique ID to all the words which makes it easier especially when we are balancing the tree and search a particular node (word).

Furthermore, we have gotten all the words from an online source mention (http://www.gwicks.net/dictionaries.htm) and created our own individual text files translated into the chosen language and also with the help of hashing providing a unique ID number to individual words.

Overall, I believe we have covered all the mandatory parts of the project and everything works very well as expected. Additionally, we have also covered many aspects of the optional parts i.e. Adding an additional language and use of grammar. We also worked on GUI but due to an unforeseen circumstance, we didn't manage to implement with the project. Additionally, to the GUI we also added Text-to-speech (tts) to make it more convenient for people with difficulties or those who don't know that particular language.

Problem:

Main problems:

1. Translating phrases: we weren't able to implement the full phrases. As they would require the insight of the foreign language. We had prototype running that allow us to translate singular phrases in the desired language. However, that wasn't suitable for larger text file or paragraphs.
2. Translated formatted text: Due to us using a basic input method we run into issues when working with formatted text. Due to this we had to reconstruct the input method by making them more suitable for formatted text file. (this didn't resolve the issues fully as we used standard JAVA libraries without working the file formatted text)
3. Hashing: Since we are working with string, it was quite difficult to search and locate a specific string or word when needed. To approach with this problem, it was necessary to create ID for individual words so Taha created a method for hashing which will give ID to all the words. But then we faced another problem with the length of the ID which needed to constraint. So, we decided to use the default hash code method which is available within the string library. Instead of creating our own hash method.
4. We didn't manage to **'perfect'** translating for foreign language to English as we didn't full overcome the grammatical difficulties. But that was mainly caused because of our attempt to make the program universal as it would be capable to any foreign languages. We could have resolved this issue if there was more time available to deeper our research into the grammatical sides of those foreign languages.
5. One of the other major issue we faced at the last minute was in relation to our GUI which one of the responsibilities of mine (Mohammed). We fully managed to make the GUI work along with text to speech feature. But due to an unforeseen circumstance, he somehow deleted entire GUI and since that happened at a very last minute, we didn't have a chance to write the GUI program again.

Minor problems:

1. Managing a large amount of references.
2. Translating from a foreign language to English, as our foreign language dictionary weren't rich as we hoped for. However, we did resolve this issue in our final design.

User Manual:

1. As soon as user runs the program, they will be asked to select an option provided in the program.
2. Once the option is selected, it will run certain parts of the code.
    a. If user inputs 1 or chooses the 1st option, they will be asked to choose the input and output language followed by allowing the user to input the words they wish to translate.
    b. Option 2 is chosen; it will allow the user to once again choose the input and output language followed by allowing the user to input the name of the text file. If the text file is found, it will be translated to the chosen output language.
    c. Option 3 is chosen; it will allow the user to add a new word to the dictionary if the word is not found in the dictionary. User will be asked whether they wish to add the word to the dictionary or not.
    d. Option 4 is chosen; the word will be added straight to the dictionary.
    e. Option 5 is chosen; it will allow the user to delete a word from the dictionary. Once the word is deleted, it will be also deleted from the other languages.
    f. Option 6 is chosen; it will print the data dictionary with the chosen language along with the assigned hash code.
    g. Option 7 is chosen; it will allow the user to save the chosen language dictionary to a text file.
3. Finally, ask the user to close the program.

Pseudocode:

```
Public void leafFinder(Node temp)
{
Pass in node called temp
Check if the left of temp is null temp.left !=null
If it is the run the method again with the left of temp leafFinder(temp.left);
If it isn't then check if the left of temp is null temp.right !=null
If it is the run the method again with the left of temp leafFinder(temp.right);
}

public void depth(Node temp) {
the left depth is 0 dl =0;
the right depth is 0 dr =0;
check the arraylist of the leafs called leafs and run it through the searchnodeleft(leafs.get(i), temp); method .
check the arraylist of the leafs called leafs and run it through the searchnoderight(leafs.get(i), temp); method .

public void searchnodeleft(Node k, Node m) {
the node k is the current node
the node m is the previous node
if the dl is less than one the the left depth is equal to 0 dl is increased by one and l is false 7
if the current node is less than the previous one the increase the left depth by one and run the method again.
```

if the current node is less than the previous one the increase the right depth by one and run the method again.
If l equals false and the id of node k and m is equal to each other then increase the depth by one and
Store it in the array depthcountL[0].
If this leaf is larger than what is in depth count then replace depthcountL[0].
If this leaf is not larger than what is in depth count then do nothing.
}
public void searchnoderight(Node p, Node f) {
the node p is the current node
the node f is the previous node
if the dr is less than one the the right depth is equal to 0 dr is increased by one and l is false 7
if the current node is less than the previous one the increase the right depth by one and run the method again.
if the current node is less than the previous one the increase the left depth by one and run the method again.
If l equals false and the id of node p and m is equal then increase the depth by one and
Store it in the array depthcountR[0].
If this leaf is larger than what is in depth count then replace depthcountR[0].
If this leaf is not larger than what is in depth count then do nothing.
}
public Node deleteentireTree(BinaryTree bt,Node node) {

this method is the method that deletes the entire tree
creates a node object n = null;
check if it is null if it is return null

set the left side of the node to null and run it through the method
set the right side of the node to null and run it through the method
set the root to null
return the current node
}

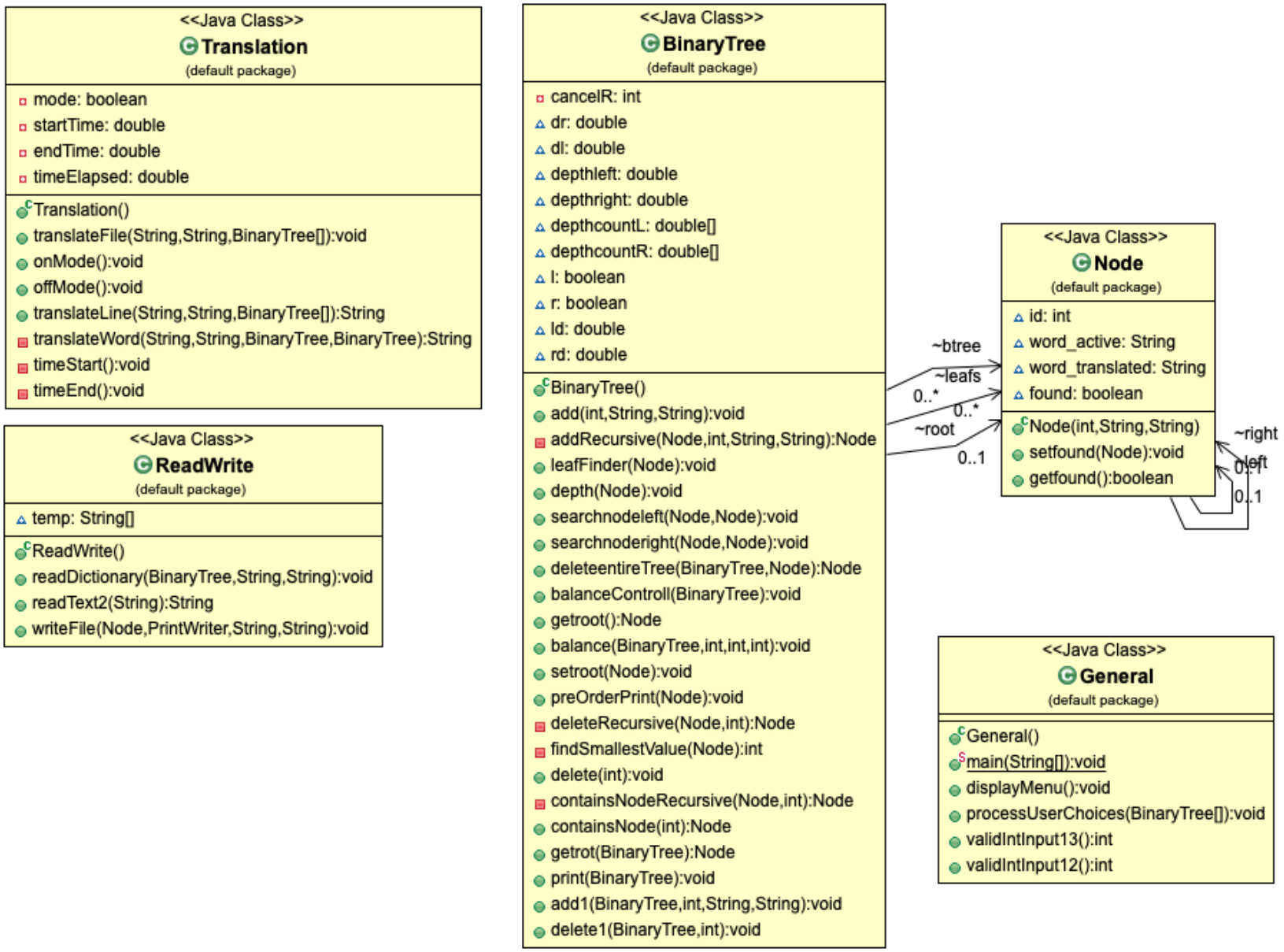
Public void balanceControll(BinaryTree bt)
{
Check if the array of right leafs and left leafs if they differ by one if they do run the method
Start is the beginning of the array list
End is the end of the array list
Middle is start plus end divided by 2
Get middle put it in the tree
End equals middle
Start equals 0
Middle is start plus end divided by 2
Run the recursive balance method with putting it in bt, start, middle and end balance(bt,start, middle, end);
Start equals end
End stays the same

Middle is start plus end divided by 2
Run balance with new inputs balance(bt,start, middle, end);
public void balance(BinaryTree bt,int start, int middle, int end)
{
Bt binary tree object
Start is the start of the array
End is the end of the array
Middle is start plus end divided by 2
End1 equals end
If found for p is false then add p to the tree
Then set the found of p to true
}
}

Task Allocation:

Team Members	Tasks
Taha Kashaf	Find and create the data sets that will be used in the program. Including work on a hashing algorithm to store the strings with unique integer values as ID's
Ayodeji Shote	Work on the actual Binary Tree we will be using as our data structure, most importantly a balancing algorithm to make sure our tree data structure is balanced and efficient to search through.
Preslav Chonev	Leading the work to use both Ayodeji and Taha's code to construct the basic functionality for translation.
Mohammed Tokaria	Work to make sure all lose ends of the program are tied up and do significant work on "finishing touches" including attempts to implement the "optional" extras. Furthermore, work as the secretary to record all the meeting dates and work on the final report.

Class Diagram:



Test Plan:

Name : Mohammed Tokaria, Ayodeji Shote, Preslav Chonev & Taha Kashaf

Matric number : 180008589, 180004145, 180007405 & 180020889

Lab Title : AC12001 Assignment 4: Group Project

Date/version : 27/03/2019

Test Notes: Tests run on submitted assignment

Due to the fact that the program was coded across many different systems, and that certain large blocks of code that were necessary to establish a foundation for the rest of the of the program to build onto, and that these large blocks of code to not appear in the final program although they were vigorously tested. Has resulted in us not having a consolidated test class. However, many different tests were conducted during the programming, and these will be brought up during the presentation, even if a singular Test Class does not exist.

Meetings and Updates

When	Where	Discussed	All members attended
20/03/19	QMB	After the initial submission of the report, this was our first meeting where we just wanted to gather a bit of intel on how everyone is progressing to their part of the project and if we had to make any changes to the project.	Yes
21/03/19	QMB	After facing a several issues in relation to searching a word using the ID created with the help of our hashing method, we decided to change the method and use default Java hash code instead of our own method.	Yes
25/03/19	Library	After making the necessary changes to our initial plan of the project, we worked together in the library to make sure we manage to accomplish the words translated to the chosen foreign language with all prefix & suffix.	Yes
26/03/19	Online	Discussed about what additional language we would like to add as part of the optional requirements and implement it with our current code.	Yes
27/03/19	Library	Finalising the project and work on the report.	Yes