Technical Report for Programming Portfolio

Text Localiser

Student ID: S208233

Date: 13/05/2020

Word Count: 2000

# Table of Contents

[Table of Contents 2](#_Toc37667893)

[List of Figures 2](#_Toc37667894)

[Introduction 2](#_Toc37667895)

[Requirements 3](#_Toc37667896)

[Design 5](#_Toc37667897)

[Type of design 5](#_Toc37667898)

[UML 5](#_Toc37667899)

[Flow Charts 6](#_Toc37667900)

[Pseudocode 10](#_Toc37667901)

[Implementation 10](#_Toc37667902)

[Methodology 10](#_Toc37667903)

[Development strategy 10](#_Toc37667904)

[Version control 10](#_Toc37667905)

[Test driven development 10](#_Toc37667906)

[Smoke testing 10](#_Toc37667907)

[Acceptance testing 10](#_Toc37667908)

[Unit testing 10](#_Toc37667909)

[Manual testing 10](#_Toc37667910)

[Conclusion 10](#_Toc37667911)

[References 10](#_Toc37667912)

[Bibliography 10](#_Toc37667913)

[Appendices 12](#_Toc37667914)

# List of Figures

|  |  |  |
| --- | --- | --- |
| Figure number: | Type of Figure: | Figure Description: |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

# Introduction

Briefly explain what this documentation is presenting ( that is, write the abstract)

# Requirements

According to the assessment specification there are two parts to this project.

Part one is to create a java library with specific methods that convert dates and currencies while implementing the interface given from the specification.

The methods include:

1. loadDateFormats method
2. localiseDate method
3. loadCurrencyFormats method
4. localiseCurrency method
5. localise method

Part two is the java application that runs, and that the user interacts with to choose certain requirements for the program.

Part two is to create a stand-alone application which uses the library from part one. This application allows the user to specify:

1. Input format
2. Input folder
3. Input file name
4. Output format
5. Output folder
6. Output file name

Using the parameter that the user specified that application used use the methods from the library created in part one to convert the input text file variables to the correct output format. Then create the output file and with the same content as the input file but with the localised variables.

# Design

## Type of design

Figure - User Interface Stepwise Refinement

This is a top down design where I have used stepwise refinement to break up the assignment smaller and easier to create, design and manage sections. The user interface is broken down into four parts. Get user input, read file where the file is designated by the user, use TextLocaliser() to convert all date and currency values in the file and create and write to a file with the same content as the input file but the date and currency values have been localised. The user input includes, input country, input file name, input file location, output country, output file name and output file location.

Figure - TextLocaliser Stepwise Refinement

The TextLocaliser() must implement an interface given by the assignment brief. The interface must include the following functions, Localise(), LocaliseDate(), LoadDateFormats(), LocaliseCurrency() and LoadCurrencyFormats(). The Localise() function looks through the input text and find the dates and currencies in the text and use LocaliseDate and LocaliseCurrency to convert them to localised values and return the values. The LocaliseDate() function uses LoadDateFormat() to load all the hard-coded elements into a map that is then used to reassign the date fields. It also must find the delimiter of the date. This function also must change the year value to two or four digits depending on the output country and finally this function must rearrange the input date fields into correct order. This can be done by assigning the input date field to the input date format and calling them in the order of the output date format. Then return the value to Localise(). The LocaliseCurrency() function uses LoadCurrencyFormat() to load all the hard-coded elements into a map that is then used to see if the output country need “,” or “.” in the currency and to see what symbol they use and what location the symbol is on the currency. This function also must find and multiply the input value by the correct exchange rate, add in the “,” or “.” Depending on the output country and add in the symbol in the correct location on the values. Then return the value to Localise(). LoadDateFormat() is passed a pre-created map then add the hard-coded values to the map and then returns it to LocaliseDate(). LoadCurrencyFormat() is passed a pre-created map then add the hard-coded values to the map and then returns it to LocaliseCurrency().

## UML

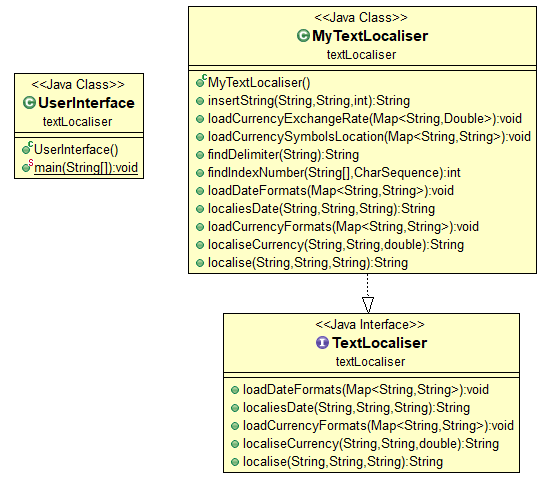


Figure 3 - UML Diagram of Program

Unified modelling language (UML) is used in object-oriented programming (OOP) to visualise classes. The diagram above represents the program created for this assignment in UML.

## Flow Charts

Figure 4 - User Interface Flowchart Part 1

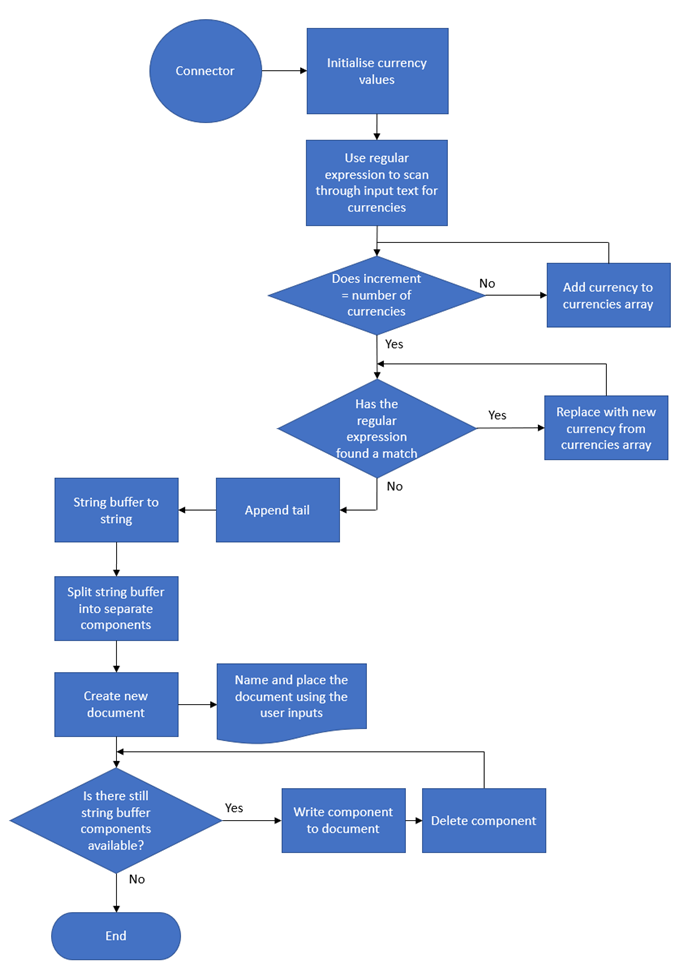


Figure 5 - User Interface Flowchart Part 2

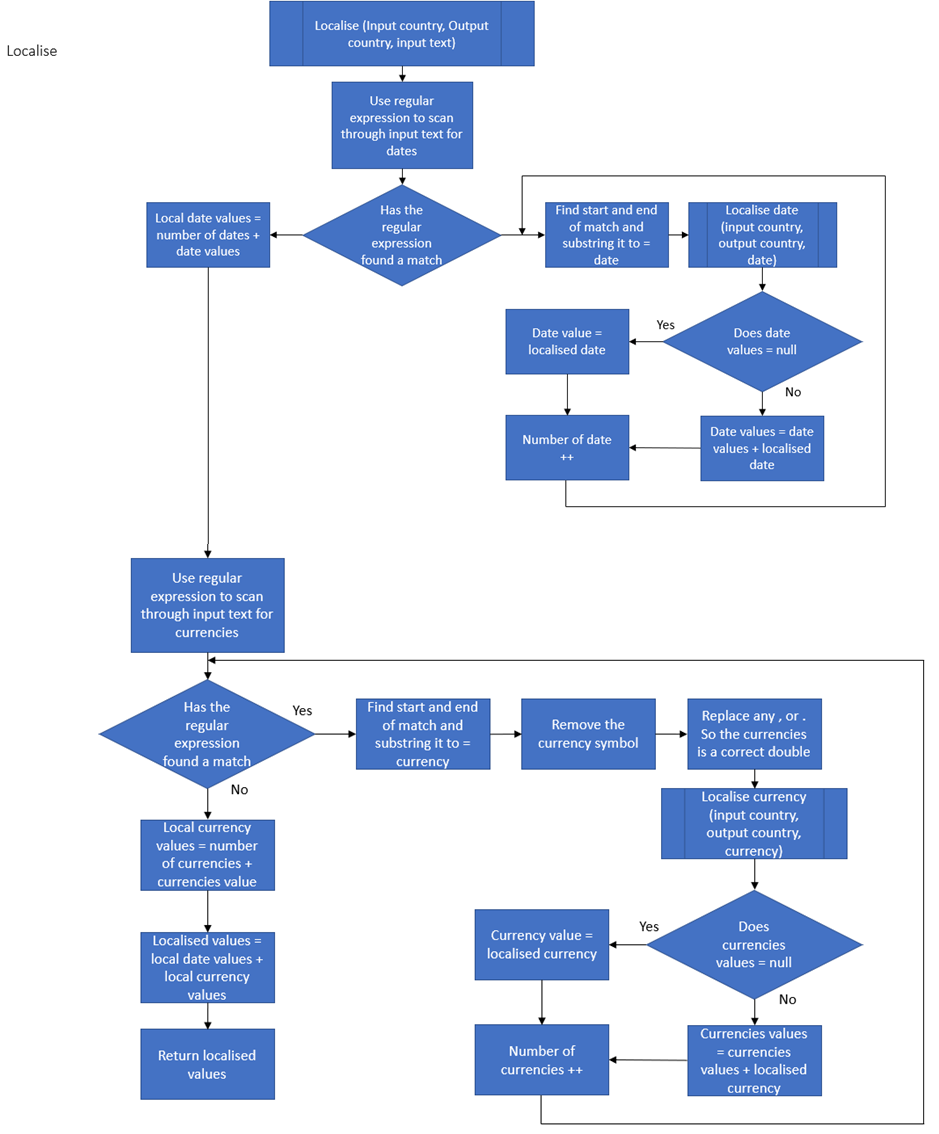


Figure 6 - Localise Flowchart

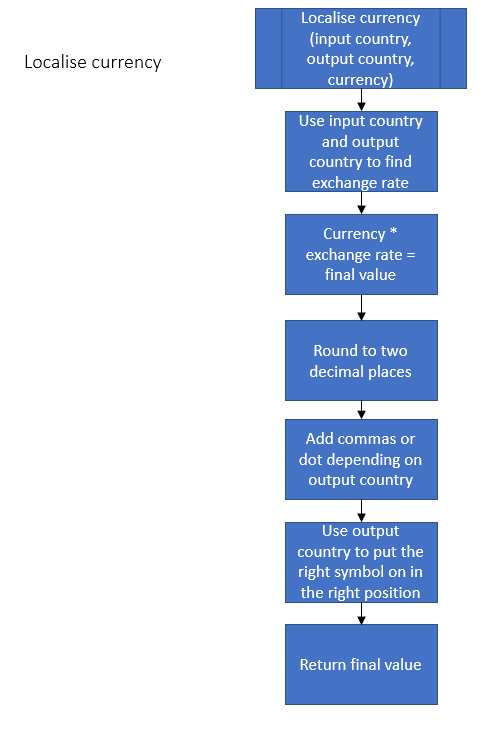
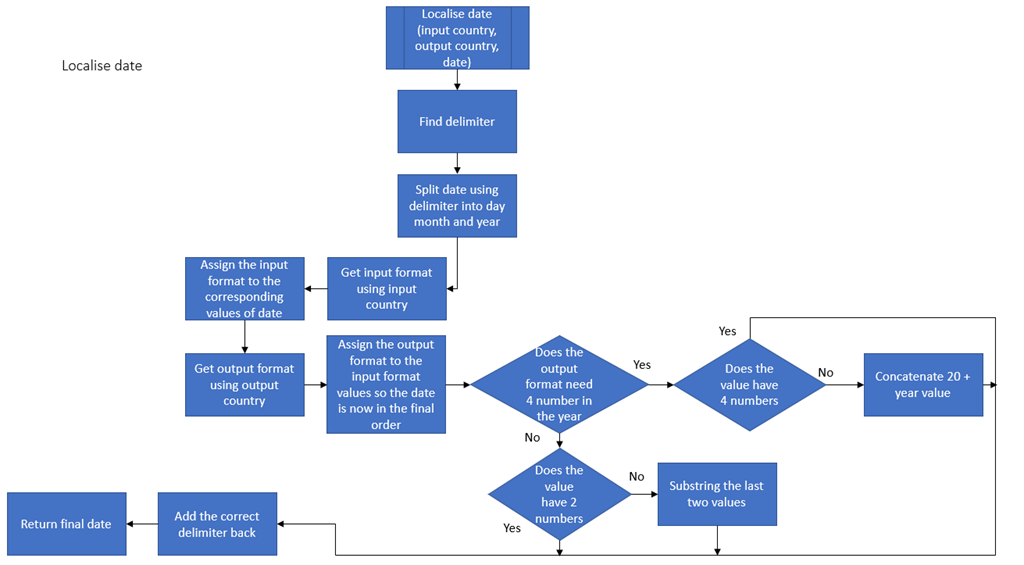


Figure 7 - Localise Date Flowchart

Figure 8 - Localise Currency Flowchart

These are the flowcharts that represent the basic concept of what will happen during the duration of the program. Some aspects of the flowcharts will need to change as the flowcharts are only theoretical and may include logic errors.

## Pseudocode

Using the flow charts above pseudocode will be developed to aid the design of the assignment.

User Interface:

Main function User Interface () {

Get user input for input country, output country, output file name, output file location, input file name and input file location

Create input file and output file by concatenating corresponding file location and file name.

Scan input file

Create an array list called file lines

While input file has a new line {

Add line into file lines array list

}

While file lines array list has another value {

Add all the file line values to one string input text while spacing them apart with a common delimiter

}

Assign returned values to the return of the function Localise (input country, output country, input text)

Split returned values into array called Localised values using the common delimiter

Find the number of dates value

Add all the date values into an array called date values

Scan through the input text with a date regular expression

While regular expression finds a match {

Replace all old values with new values from date values array

}

Add the tail to the end of the string buffer

Convert string buffer to new input text string

Find the number of currencies value

Add all the currency values into an array called currency values

Scan through the input text with a currency regular expression

While regular expression finds a match {

Replace all old values with new values from currency values array

}

Add the tail to the end of the string buffer

Convert string buffer to final text

Split final text using common delimiter into an array called output text

Create file using output file variable

While output text has a value {

Write each line into output file

}

}

Localise:

Function Localise (input country, output country and input text) {

Scan through the input text with a date regular expression

While regular expression finds a match {

Find the date and set as a string called date text

Assign local date to the return value of the function Localise date (input country, output country, date text)

Increment number of dates by one

Add local date to localised date value spaced with common delimiter

}

Add number of dates to front of localised date value spaced with common delimiter

Scan through the input text with a currency regular expression

While regular expression finds a match {

Find the currency and set as a string called currency text

Remove any characters to make a double value

Assign local currency to the return value of the function Localise currency (input country, output country, currency text)

Add one to number of currencies

Add local currency to localised currency value spaced with common delimiter

}

Add number of currencies to front of localised currency value spaced with common delimiter

Create localised values by concatenating both localised date and localised currency spaced by a common delimiter

Return localised values

}

LocaliseDate:

Function Localise Date (input format, output format, input text) {

Find delimiter of the input text

Split the input text using the delimiter into an array called date values

Load date format into a map

Retrieve input and output country date format using the load date format map

Find the delimiter of input format fields and split input format fields into an array called input format fields using the delimiters

Find the delimiter of output format fields and split output format fields into an array called output format fields using the delimiters

Create a new map called conversion and assign the corresponding values from date values array to the input format fields

If (output format [year value] = input format [year value]) {}

Else If input format [year value] == “yy” {

Add “20” to the begin of the year value

}

Else {

Substring the year value for the last 2 digits

}

Concatenate all the fields together using the output format fields order as the key calling from the conversion map while putting the correct delimiter between the values.

Return localised date

}

LocaliseCurrency:

Function Localise Currency (input format, output format, input text) {

Load exchange rate into a map

Load location of symbol into a map

Load type of symbol into a map

Create a key by concatenating input format + “-“ + output format

Get exchange rate from map using key

Assign final value to exchange rate \* input text

Round final value to 2 decimal places

Convert final value to a string

Add in the comma or dots in the right locations by inserting them into the final value string

Add the correct symbol in the correct location using the symbol and location maps and the output format to the final value string

Return final value

}

# Implementation

## Methodology

## Development strategy

## Version control

# Test driven development

## Smoke testing

## Acceptance testing

## Unit testing

## Manual testing

# Conclusion

# References

# Bibliography

URLs:

1. <https://docs.oracle.com/javase/8/docs/api/java/util/Map.html>
2. <https://www.geeksforgeeks.org/java-util-dictionary-class-java/>
3. <https://stackoverflow.com/questions/13543457/how-do-you-create-a-dictionary-in-java>
4. <https://lh.2xlibre.net/locale/en_GB/>
5. <https://stackoverflow.com/questions/22622676/java-variabledeclaratorid-expected-after-this-token>
6. <https://coderanch.com/t/554767/java/Syntax-error-token-list-VariableDeclaratorId>
7. <http://doc.gold.ac.uk/~mas01sd/classes/JavaAbstractSyntax/1documentation/JavaAbstractSyntax/VariableDeclaratorId.html>
8. <https://www.quora.com/How-do-I-parse-a-string-in-Java>
9. <https://stackoverflow.com/questions/8248277/how-to-determine-if-a-string-has-non-alphanumeric-characters>
10. <https://www.guru99.com/string-length-method-java.html>
11. <https://www.w3schools.com/java/java_while_loop.asp>
12. <https://www.guru99.com/string-contains-method-java.html>
13. <https://www.w3schools.com/java/java_hashmap.asp>
14. <https://www.w3schools.com/java/java_for_loop.asp>
15. <https://www.w3schools.com/java/java_arrays.asp>
16. <https://www.tutorialspoint.com/Java-String-substring-Method-example>
17. <https://beginnersbook.com/2013/12/java-string-substring-method-example/>
18. <https://www.javatpoint.com/substring>
19. <https://www.geeksforgeeks.org/hashmap-containskey-method-in-java/>
20. <https://www.geeksforgeeks.org/check-if-a-value-is-present-in-an-array-in-java/>
21. <https://stackoverflow.com/questions/2131802/java-arraylist-how-can-i-check-if-an-index-exists>
22. <https://stackoverflow.com/questions/12897615/java-check-if-array-item-exists/12897640>
23. <https://mkyong.com/java/java-check-if-array-contains-a-certain-value/>
24. <https://stackoverflow.com/questions/12897615/java-check-if-array-item-exists>
25. <https://stackoverflow.com/questions/1128723/how-do-i-determine-whether-an-array-contains-a-particular-value-in-java>
26. <https://www.geeksforgeeks.org/map-get-method-in-java-with-examples/>
27. <https://www.geeksforgeeks.org/hashmap-get-method-in-java/>
28. <https://www.google.com/search?client=firefox-b-d&q=how+to+convert+double+into+string+in+java>
29. <https://lh.2xlibre.net/locale/en_US/>
30. <https://lh.2xlibre.net/locale/de_DE/>
31. <https://www.w3schools.com/java/java_files.asp>
32. <https://www.w3schools.com/java/java_files_create.asp>
33. <https://www.w3schools.com/java/java_files_read.asp>
34. <https://stackoverflow.com/questions/3886797/how-to-map-a-mapstring-double>
35. <https://www.w3schools.com/java/java_arraylist.asp>
36. <http://www.java2s.com/Code/Java/Collections-Data-Structure/HashMapStringDouble.htm>
37. <https://stackoverflow.com/questions/8224240/issue-with-using-double-as-value-in-hashmap>
38. <https://stackoverflow.com/questions/787735/what-is-parse-parsing>
39. <https://docs.oracle.com/javase/8/docs/api/org/xml/sax/Parser.html>
40. <https://docs.oracle.com/javase/8/docs/api/java/text/ParsePosition.html>
41. <https://www.geeksforgeeks.org/period-parse-method-in-java-with-examples/>
42. [https://stackoverflow.com/questions/24760984/java-file-parsing-based-on-position#](https://stackoverflow.com/questions/24760984/java-file-parsing-based-on-position)
43. <http://pages.cs.wisc.edu/~hasti/cs302/examples/Parsing/parseString.html>
44. <https://www.oreilly.com/library/view/learning-java-4th/9781449372477/ch10s04.html>
45. <https://codingbat.com/doc/java-string-indexof-parsing.html>
46. <https://www.java67.com/2015/06/how-to-convert-string-to-double-java-example.html>
47. <https://www.journaldev.com/18392/java-convert-string-to-double>
48. <https://docs.oracle.com/javase/8/docs/api/index.html?java/util/regex/package-summary.html>
49. <https://www.tutorialspoint.com/java/java_regular_expressions.htm>
50. <https://www.geeksforgeeks.org/how-to-add-an-element-to-an-array-in-java/>
51. <https://www.journaldev.com/763/java-array-add-elements>
52. <https://stackoverflow.com/questions/2843366/how-to-add-new-elements-to-an-array>
53. <https://stackoverflow.com/questions/14098032/add-string-to-string-array>
54. <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>
55. <https://stackoverflow.com/questions/17646724/how-to-retrieve-array-values-in-java>
56. <https://stackoverflow.com/questions/15775534/extracting-value-from-an-array-in-java>
57. <https://www.geeksforgeeks.org/matcher-appendtailstringbuffer-method-in-java-with-examples/>
58. <https://www.w3schools.com/java/java_abstract.asp>
59. <https://www.w3schools.com/java/java_user_input.asp>
60. <https://www.baeldung.com/java-date-regular-expressions>
61. <https://mkyong.com/java/java-convert-string-to-int/>
62. <https://stackoverflow.com/questions/8491687/write-newline-into-a-file>
63. <https://www.tutorialspoint.com/java/java_regular_expressions.htm>
64. <https://stackoverflow.com/questions/43246115/java-regular-expression-matching-square-brackets>
65. <https://www.freeformatter.com/java-regex-tester.html#ad-output>
66. <https://ux.stackexchange.com/questions/9105/international-currency-formatting-guidelines-currency-codes>
67. <https://mkyong.com/java/how-to-round-double-float-value-to-2-decimal-points-in-java/>
68. <https://www.geeksforgeeks.org/insert-a-string-into-another-string-in-java/>
69. <https://www.tutorialspoint.com/javaregex/javaregex_matcher_appendreplacement.htm>

# Appendices