08

**Fall**

**AC 12001**

**Group**

05

Members

Mckenzie Darren

Jeffries Shaun

Joyce Daniel

Dwyer Stephen

Kirvesmäki Mira

Design document for Bliss project

[Design document for Bliss project 2](#_Toc288837663)

[Introduction to the project 2](#_Toc288837664)

[The Requirements 3](#_Toc288837665)

[The Requirements tackled 4](#_Toc288837666)

[Initial classes 5](#_Toc288837667)

[Pseudo Code 5](#_Toc288837668)

[Data structures 6](#_Toc288837669)

[Class diagram 7](#_Toc288837670)

[Use cases 8](#_Toc288837671)

[Flow chart 8](#_Toc288837672)

[Test plans 9](#_Toc288837673)

[Menu: 9](#_Toc288837674)

[English to Bliss: 10](#_Toc288837675)

[Bliss to English: 11](#_Toc288837676)

[Evaluation 12](#_Toc288837677)

[Management.doc 13](#_Toc288837678)

# Introduction to the project

The aim of this project is to create a program that will let the user translate BLISS – words to English words and vice versa using data structures that we have learned in this module. The program will use numbers to represent the Bliss words. It will be a very useful translation tool.

Bliss symbols were first invented after world war two to allow communication between people who could not speak the same language but it became very useful for people with cerebal palcy or other disorders because it was very easy to learn.

The Bliss symbols offered people with communication difficulties an easy and intuitive language which they could use to communicate.

Our program will benefit these people greatly as it will be a very useful tool for communicating between everyone.

# The Requirements

The requirements needed for the bliss project fall in to three categories, technical, compulsory user requirements and optional user requirements.

1. **List of Technical Requirements:**

The program shall

1.1 use recursion.

1.2 use Binary Tree's.

1.3 use Queues using linked lists.

1. **List of Compulsory User Requirements:**

The program shall:

* 1. Store the code of both bliss and English characters as text files.

2.2 Load the text files and store each in their respective binary trees

2.3 Allow the user to input a sentence to be translated

2.4 Validate the user input

2.5 Split the user sentence into word that can be located in the binary trees

2.6 Translate from Bliss symbols into English.

2.7 Translate English into Bliss symbols.

2.8 Allow the user to enter a new sentence to be translated

1. **List of Optional User Requirements:**

The program may:

* 1. be able to translate Bliss-words which consist of multiple characters.

3.2 allow the user to add new Bliss words (using the existing list of Bliss-characters

)

3.3 be able to re-write to a text file.

3.4 be able to handle English synonyms (two ways of saying the same thing) i.e. happy & glad would translate to the same Bliss-word.

3.5 use the prototype semantic3 Bliss-font.

3.6 load and translate simple text files into Blisssymbolics.

3.7 load and translate simple text files of Blisssymbols into English.

# The Requirements tackled

**Technical Requirements:**

For this project we managed to tackle all of the technical requirements.

**Compulsory User Requirements:**

For this project we managed to tackle all of the compulsory user requirements.

**Optional User Requirements:**

For this project we managed to tackle the following points of the optional user requirements:

* 1. Being able to translate Bliss-words which consist of multiple characters.

3.6 Loading and translating simple text files into Blisssymbolics.

3.7 Loading and translating simple text files of Blisssymbols into English.

# 

# Initial classes

To start off our project our team went through a list of all the possible candidate classes that our program would contain. We decided to use the following

* Menu
* Translator
* Tree
* Tree node
* Queue
* List
* List node

# Pseudo Code

Our Menu class contained all the essential code for the user to be able to interact with our program. It prints a menu with different options available to the user. It contained methods such as:

displayMainMenu() – This just printed the menu to the user and allows the user to see his options.

runMenu() – This called all the options in the menu and also from other classes if necessary.

promptChoice() – This got what option the user wanted from the menu once it was displayed.

getEnglishWord() – If the user had selected from the menu options to translate English to Bliss this method then would take in a sentence from the user that they wanted translated. It would then run methods from our other classes to translate it.

getBlissNumber() – If the user had selected from the menu options to translate Bliss to English this method then would take in a number from the user. It would then run methods from other classes to translate it.

validationSplit() – This method was a way to get the number that was associated with the Bliss character.

Starting() –

getToMenu () – this method allowed the user to look at the printed information for as long as they needed and being able to return to the menu by simply typing in -1.

exit() – This was a method to exit the program completely.

The translator class contained all of our code that was needed to translate Bliss to English and vice versa. This included methods such as:

loadBlissTree () Loads the file and passes the details to be added to the Bliss binary tree

loadEnglishTree () –Loads the file and passes the details to be added to the English Binary tree.

searchForBliss () – Performs a binary search in the bliss tree trying to find the words the user has entered to be translated

transBlissToEnglish() –Takes the user input, splits it to separate words and translates each word into English. It first checks if there are “special words” that contain more than one bliss symbol.

transEnglishToBliss() – Splits the user input into words, checks if they are “special” two part words (for example belongs to) and then translates it into bliss.

printTranslation() – Prints the translated words for the user to see.

The other classes were our data structures which organised and stored our data that we needed for our translator to work.

# 

# Data structures

For the purpose of the Bliss translator two binary trees were used to store the words and symbols. One that was arranged in alphabetical order using the English words and the other that was ordered in numerical order using the bliss symbols. These can then be searched with efficiency and would accommodate possible even a large scale translator in the future. Queues using linked lists were implemented to store the translated words and symbols all of the words were translated. Recursion was used for printing the dictionary.

# Class diagram

|  |
| --- |
| Translator |
| Tree theTree = new Tree();  -Queue myQueue = new Queue();  -Queue notFoundQueue = new Queue();  -Queue specialEnglish = new Queue();  - String foundWords;  -String foundNumbers;  - String wordOne;  - String wordTwo; |
| +isItNumberBliss()  +isItNumberEnglish()  +loadBlissTree()  +loadEnglishTree()  +printTranslation()  +printTree()  +transBlissToEnglish()  +tranEnglishToBliss()  +transSpecial()  +transSpecialEnglish() |

|  |
| --- |
| Menu |
| Translator theTranslator = new Translator ()  -int choice  - BufferedReader in  -String read |
| + Tester()  + static void main (String [] args)  + void displayMainMenu ()  + void runMenu ()  + void promptChoice()  + void getEnglishWord()  + void getBlissNumber()  + boolean validationSplit(String)  +void starting ()  + void getToMenu()  + void instructions()  + void exit() |

|  |
| --- |
| Tree |
| + TreeNode rootBliss;  + TreeNode rootEnglish;  + String foundInfo; |
| +addToBlissTree()  +addToEnglishTree()  +callingTraverse()  +searchInBlissTree()  +searchInEnglishTree()  +traverseTree() |

|  |
| --- |
| Queue |
| List aList = new List(); |
| +checkQueueEmpty()  +initialiseQueue()  +pop()  +popNoPrint()  +printer()  +push() |

|  |
| --- |
| Treenode |
| - int bliss;  - String english;  - String specialBliss;  - TreeNode right;  - TreeNode left; |
| + getBliss()  +getEnglish()  +getInfo()  +getLeft()  +getRight()  +printInfoBliss()  +printInfoEnglish()  +setLeft()  +setRight() |

|  |
| --- |
| List |
| - ListNode head;  - ListNode tail;  - int queueSize; |
| +addToTail()  +deleteFromList()  +deleteNode()  +getHead()  +getTail()  +isEmpty()  + printList()  +setHead()  +setTail() |

|  |
| --- |
| ListNode |
| + String information;  - ListNode next; |
| +formatInfo()  +getInformation()  +getNext()  +setInformation()  +setNext() |

# Use cases

**1. User:** Opens the program

**2. System:** Displays a greeting message and displays a menu with three options (1.Bliss-English, 2.English-Bliss and 3.Exit)

**3. User:** Selects either option 1 or 2

**4. System**: Displays a command line where the user can enter the word/symbol to be translated

**5. User:** Enters the word/symbol to be translated

**6. System:** Displays the translation or error message if the word was not found

**7. System:** Gives the user the option to enter another word/symbol or get back to main menu

**8.User:** Selects to get back to main menu

**9. System:** Displays the main menu with three options (1.Bliss-English, 2.English-Bliss and 3.Exit)

**10.User:** Selects the third (Exit) option

**11. System:** An farewell message will be printed and the system will qui

# Flow chart

# Test plans

### Menu:

|  |  |  |  |
| --- | --- | --- | --- |
| **What is tested** | **Input** | **Expected result** | **Result** |
| Menu is displayed correctly | n/a | Menu choices are displayed for the user to see at the appropriate times | worked |
| Prompting for user choice | Correct input (numbers 1-5) | The user is prompted for a selection and the system will call appropriate methods | worked |
| Prompting for user choice | Incorrect input | The user is notified that the input was incorrect and asked for a new selection | worked |
| Menu selections | Number 1 | The user will be directed to the English to Bliss translation section | worked |
| Menu selections | Number 2 | The user will be directed to the Bliss to English translation section | worked |
| Menu selections | Number 3 | The user will be shown the instructions on how to use the program and asked to enter -1 to get back to main menu | worked |
| Menu selections | Number 4 | The whole current dictionary will be printed for the user to see and the user will be asked to enter -1 to get back to the menu | worked |
| Menu selections | Number 5 | The system will display an exit message and quit the program | worked |
| Getting back to menu | Correct input (number -1) | The user will be asked to enter -1 to get back to menu and will then display the menu selections for the user | worked |
| Getting back to menu | Incorrect input | An error message will appear and the user will be prompted to enter their selection again | worked |
| Interface | n/a | The screen will flush at appropriate times keeping the interface clean and readable | worked |

### English to Bliss:

|  |  |  |  |
| --- | --- | --- | --- |
| **What is tested** | **Input** | **Expected result** | **Result** |
| Choice between file and manual input | Number 1 | The user will be asked to enter the name of the file they wish to translate | worked |
| Choice between file and manual input | Number 2 | The user will be asked to write a sentence they wish to translate | worked |
| Choice between file and manual input | Invalid input | An error message will be displayed and the user will be taken back to main menu | worked |
| Entering file name | Valid file name | The content of the file will be translated to bliss and the user will be asked to enter -1 to get back to menu | worked |
| Entering file name | Invalid file name | An error message will appear and the user will be taken back to main menu | worked |
| Entering a sentence manually | One word (valid) | The program will translate the word and show the user the bliss translation | worked |
| Entering a sentence manually | One word (invalid) | The program will tell the user that it was not able to translate the given word | worked |
| Entering a sentence manually | Multiple words, all valid | The program will translate all the words and show their bliss translations | worked |
| Entering a sentence manually | Multiple words, valid and invalid mixed | The program will translate the valid words and tell the user which word it was not able to translate | worked |
| Entering a sentence manually | Multiple words, all invalid | The program will inform the user that it was not able to translate the entered words | worked |

### Bliss to English:

|  |  |  |  |
| --- | --- | --- | --- |
| **What is tested** | **Input** | **Expected result** | **Result** |
| Choice between file and manual input | Number 1 | The user will be asked to enter the name of the file they wish to translate | worked |
| Choice between file and manual input | Number 2 | The user will be asked to write a sentence they wish to translate | worked |
| Choice between file and manual input | Invalid input | An error message will be displayed and the user will be taken back to main menu | worked |
| Entering file name | Valid file name | The content of the file will be translated to bliss and the user will be asked to enter -1 to get back to menu | worked |
| Entering file name | Invalid file name | An error message will appear and the user will be taken back to main menu | worked |
| Entering a sentence manually | One symbol (valid) | The translation for that symbol will be shown to the user | worked |
| Entering a sentence manually | One symbol (invalid) | An error message will appear and the user will be prompted to enter a new input(file or manual) | worked |
| Entering a sentence manually | Multiple symbols, all valid, all separate words | A translation for all the words will be shown to the user | worked |
| Entering a sentence manually | Multiple words, valid and invalid mixed, all separate words | An error message will appear and the user will be prompted to enter a new input(file or manual) | worked |
| Entering a sentence manually | Multiple words, all invalid, all separate words | An error message will appear and the user will be prompted to enter a new input(file or manual) | worked |
| Entering a sentence manually | A word containing more than one bliss symbols | A translation for the word will be shown to the user | worked |
| Entering a sentence manually | A word containing more than one bliss symbolic combined with individual bliss symbols | A translation for all the words will be shown to the user | worked |
| Entering a sentence manually | Multiple bliss words that contain multiple bliss symbols | A translation for all the words will be shown to the user | worked |

# Evaluation

As a group we were successful in getting our program to work. It can translate more than one bliss symbol to English and can translate English sentences to Bliss. The team worked well together and there were few problems.

One of the hardest parts were dividing the tasks so that everyone was in charge of a logical part of the program that could then be combined with the other parts with ease. For this to fully work we had to work partly in shifts working on the same piece of code for an allocated time and passing it to others then to be completed.

As to the coding some of the biggest difficulties we had were trying to include the “special” words into the translation system. Examples of these include bliss words that use more than one bliss symbol and English words that have two parts.

# Management.doc

Tuesday 8.3.

9-10 am

Mira, Stephen & Darren

Initial decisions/ideas, dividing the report

Thursday 10.3.

6.30-8 pm

Darren, Mira, Stephen, Shaun, Daniel

Going through the report, data structures and pseudo-code

Friday 11.3.

9-10

Darren, Daniel & Mira

Final adjustments to the report, class diagram

Monday 14.3.

9-10

Darren, Daniel, Mira, Stephen

Feedback on report, what next

Tuesday 15.3.

9-10

Darren, Daniel, Mira, Stephen

Going through the report with the feedback, dividing code

Thursday 17.3

5-6

Darren, Mira, Stephen, Shaun, Daniel

Came together to just speaking of progress, help and other things.

Monday 21.3

9-12

Darren, Mira, Stephen, Shaun, Daniel

planned for what needs done next, starting to make final preparations

Tuesday 22.3

Darren, Mira, Stephen, Daniel

came together for work.

Wednesday 23.3

Darren, Mira, Stephen, Daniel

more work.

Thursday 24.3

Darren, Mira, Stephen, Daniel

working

Friday 25.3

Darren, Mira, Stephen, Daniel

Finished everything.