Computing Project

Student-devised Programming Project – Toki Pona spell checker

FINAL REPORT

Abstract

I made a spell checker and predictive text system for the Toki Pona language, developed using the Java programming language and BlueJ. The system includes a separate list of valid words, names, and functions to check whether words are spelled correctly, suggest words based on prefixes, and check sentence grammar against Toki Pona's basic grammar rules. This has been thoroughly tested and improved based on teacher/student feedback. This can be used in communication tools, translation software and language learning applications, allowing users to communicate more effectively using Toki Pona and making languages easier to use.

Contents

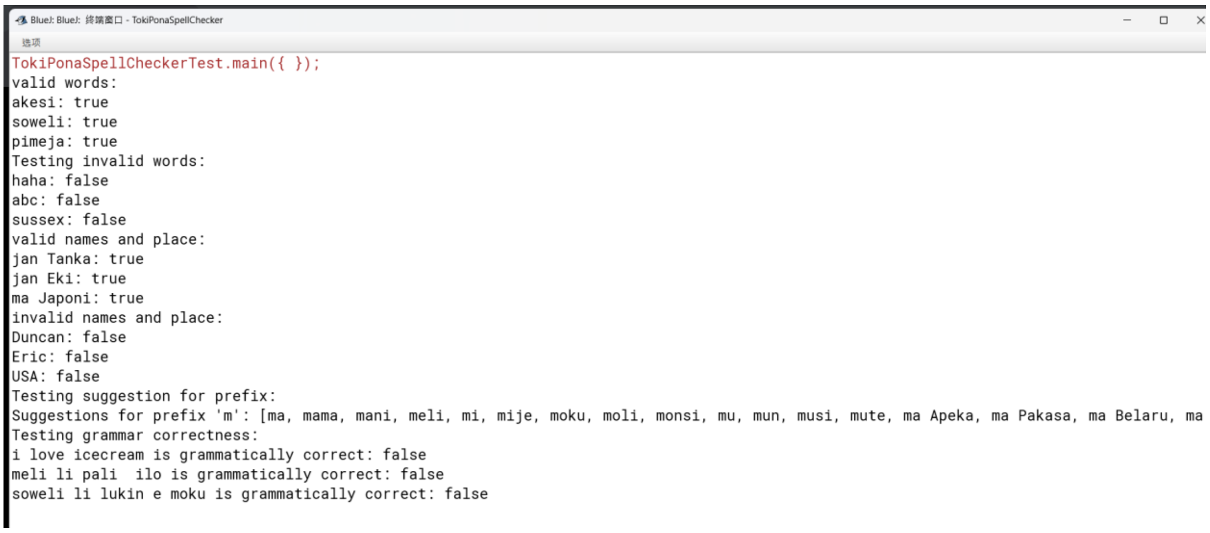
Abstract 2
1. Outline of the work done 3
  1.1 Description 3
  1.2 Screen shots 3
2. Evaluation of the product 4
3. Evaluation of the process 5
Appendix A : Code listings 6
  Main code 6
  Test code 14
  Reference 15

1. Outline of the work done

1.1 Description

I developed a spell checker and predictive text system for Toki Pona language using Java programming language and BlueJ. This includes functions for checking whether words are spelled correctly, suggesting words based on prefixes and checking sentence grammar against Toki Pona's basic grammar rules. It was done in 6 days, which is better than the time I set before. I've made a powerful, effective spell checker and predictive text system for Toki Pona language that can be used in language learning applications.

1.2 Screen shots



```
TokiPonaSpellCheckerTest.main({ });
valid words:
akesi: true
soweli: true
pimeja: true
Testing invalid words:
haha: false
abc: false
sussex: false
valid names and place:
jan Tanka: true
jan Eki: true
ma Japoni: true
invalid names and place:
Duncan: false
Eric: false
USA: false
Testing suggestion for prefix:
Suggestions for prefix 'm': [ma, mama, mani, meli, mi, mije, moku, moli, monsi, mu, mun, musi, mute, ma Apeka, ma Pakasa, ma Belaru, ma
Testing grammar correctness:
i love icecream is grammatically correct: false
meli li pali  ilo is grammatically correct: false
soweli li lukin e moku is grammatically correct: false
```

2. Evaluation of the product

Requirements satisfied.

The project brief for a spell checker and predictive text system for Toki Pona language, which is exactly what I have developed. Includes functions to check for correct spelling, suggest words based on prefixes, check sentence grammar against its basic grammar rules. I have used a separate list of valid words, names to ensure that it is accurate. I have tested and improved it based on feedback, which demonstrates a commitment to meet the requirements of the project brief.

Best features

Suggestion algorithms: The system includes suggestion algorithms that suggest appropriate corrections for misspelled words and provide predictive text suggestions based on a prefix. These features can help users write efficiently and effectively in Toki Pona.

Possible improvements

Multilingual support: Expanded to support multiple languages, allowing users to check spelling and grammar in different languages. I think this feature could be useful for translation software and language learning applications.

Advanced grammar checking: Grammar checking algorithm expanded to include more complex sentence structures and grammatical rules, allowing the system to identify and suggest corrections for more advanced grammar errors.

1. The code does not explicitly check for common modifiers like ni, suli, ante, etc. But allowing unfamiliar words is enough.
2. My code focuses on subject-verb-object word order. I know Toki Pona allows verb-subject-object, a rule for that could be added. But subject-verb-object is by far the most common.

Possible extensions

Machine learning-based suggestion algorithms: Suggestion algorithms can improve by incorporating machine learning techniques to provide more personalized, relevant suggestions based on user behaviour and preferences.

Voice recognition: The system could add voice recognition capabilities. Users speak their words into the system to examined for spelling and grammar mistakes.

Create a Graphical User Interface: The system currently runs on a BlueJ terminal, which is cumbersome for users. Creating a GUI makes my work accessible to a wider audience, including those unfamiliar with using command line interfaces.

It seems that the possible extensions for the project are good, but as I did not learn them before, I hope to attend university to gain the necessary knowledge.

3. Evaluation of the process

Work that went well:

- Planning and Organization: My project plan is well-structured, well-organized with clear goals, tasks, and deadlines. Weekly progress checks keep projects on track and ensure tasks are completed on time.

Difficulties:

- Limited vocabulary: Toki Pona language has a limited vocabulary, made it challenging to develop a suggestion algorithm that provided relevant and useful suggestions. I used a prefix-based search algorithm, for which words, names beginning with same prefix.

- Grammar checking algorithm: Developing an accurate and reliable grammar-checking algorithm for Toki Pona was challenging due to the language's basic grammar rules and simple sentence structures. So, a rule-based approach was used, which number of words considered the in the sentence and their forms.

- Limited resources: There were limited resources available for the Toki Pona language, which made it challenging to gather all necessary information, materials for the project. To overcome this difficulty, research was conducted using online resources, such as Toki Pona official site.

changes I make:

Improving suggestion algorithm: After receiving feedback from teacher/classmate, the suggestion algorithm was improved by incorporating a prefix-based search algorithm, which searched for words, names beginning with the same prefix. This change provided more relevant and useful suggestions for users.

Learned:

- Improved programming skills: This project improves my programming skills, especially Java and Bluej programming languages. It will help me in future programming projects, improve efficiency and effectiveness of my solution.

do differently:

- Incorporating machine learning techniques: In the future, I'll investigate how to enhance precision, efficiency of spell checkers and predictive text systems using machine learning techniques like neural networks and deep learning algorithms.

- If I were to do this project again, I would concentrate on implementing a graphical user interface (GUI) instead of relying on the BlueJ terminal. This makes the system more user-friendly and accessible.

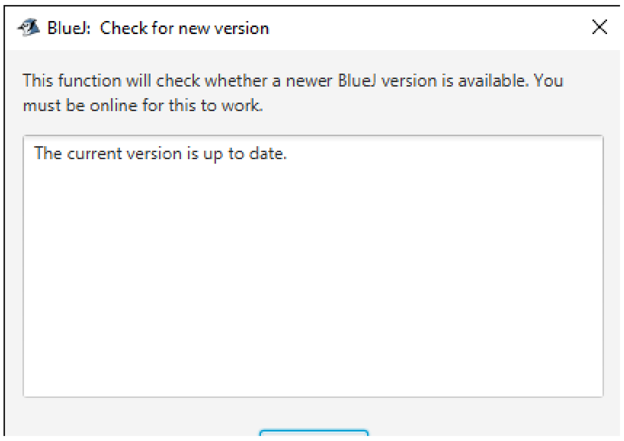
Appendix A:

I use two lists to store valid words, names, places.

The isSpelledCorrectly method checks whether a word is spelled correctly, isGrammarCorrect method checks whether a given sentence is grammatically correct.

The suggestWords method suggests a list of valid words start with a given prefix.

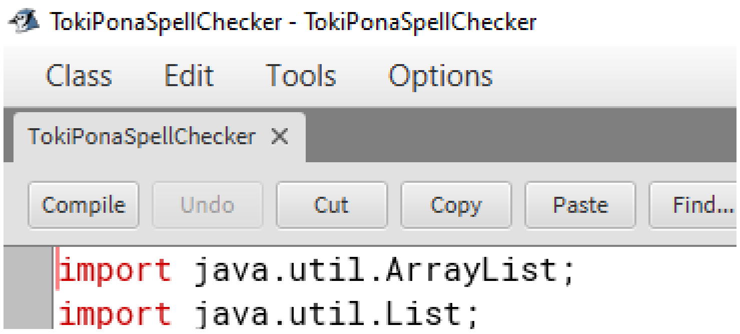
The program can be improved by adding more words to validWords list, by adding more rules to the isGrammarCorrect method.


BlueJ: Check for new version

This function will check whether a newer BlueJ version is available. You must be online for this to work.

The current version is up to date.

At the beginning of my programming, I checked bluej is the newest version.

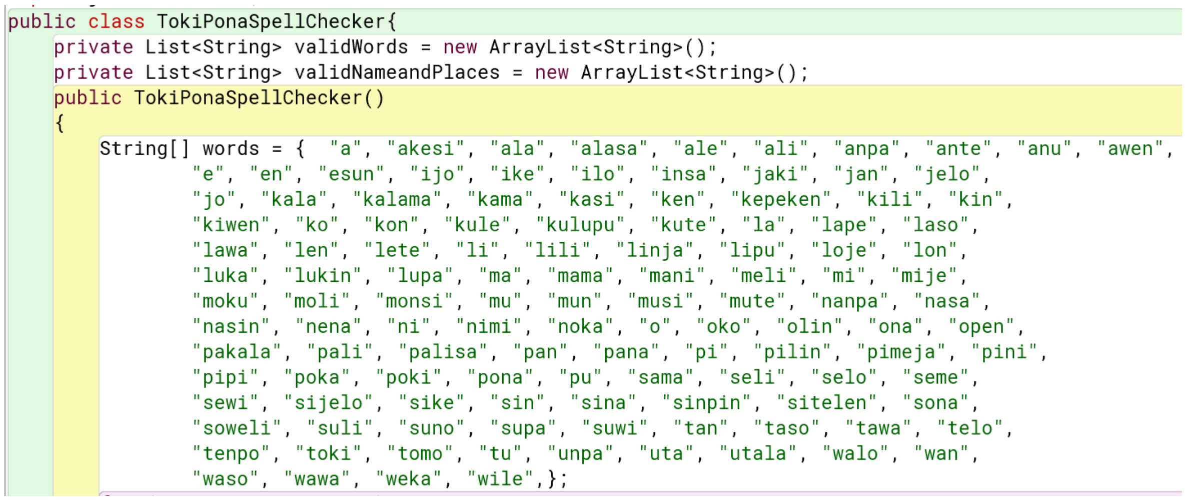


```
import java.util.ArrayList;
import java.util.List;
```

I import two classes from the Java Utility Pack, Array List and List.

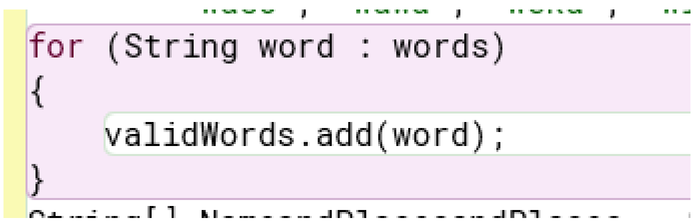
The Array List class is an extensible array implementation of the List interface. Allows dynamic addition and removal of items, provides methods for accessing, manipulating items in the list.

The List interface is a sub interface of Collection interface and represent an ordered sequence of items. Defines methods for adding, removing, and accessing items in the list.



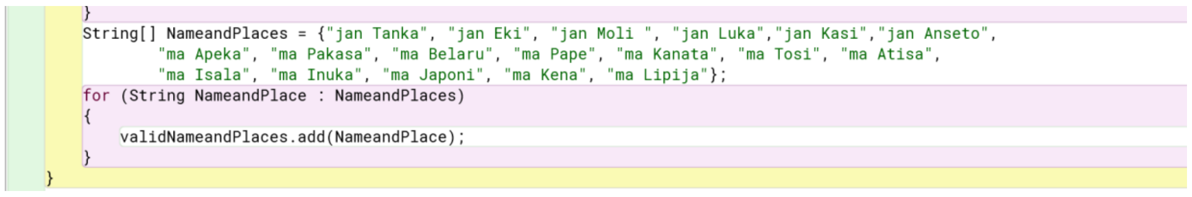
```
public class TokiPonaSpellChecker{
    private List<String> validWords = new ArrayList<String>();
    private List<String> validNameandPlaces = new ArrayList<String>();
    public TokiPonaSpellChecker()
    {
        String[] words = {  "a", "akesi", "ala", "alasa", "ale", "ali", "anpa", "ante", "anu", "awen",
                "e", "en", "esun", "ijo", "ike", "ilo", "insa", "jaki", "jan", "jelo",
                "jo", "kala", "kalama", "kama", "kasi", "ken", "kepeken", "kili", "kin",
                "kiwen", "ko", "kon", "kule", "kulupu", "kute", "la", "lape", "laso",
                "lawa", "len", "lete", "li", "lili", "linja", "lipu", "loje", "lon",
                "luka", "lukin", "lupa", "ma", "mama", "mani", "meli", "mi", "mije",
                "moku", "moli", "monsi", "mu", "mun", "musi", "mute", "nanpa", "nasa",
                "nasin", "nena", "ni", "nimi", "noka", "o", "oko", "olin", "ona", "open",
                "pakala", "pali", "palisa", "pan", "pana", "pi", "pilin", "pimeja", "pini",
                "pipi", "poka", "poki", "pona", "pu", "sama", "seli", "selo", "seme",
                "sewi", "sijelo", "sike", "sin", "sina", "sinpin", "sitelen", "sona",
                "soweli", "suli", "suno", "supa", "suwi", "tan", "taso", "tawa", "telo",
                "tenpo", "toki", "tomo", "tu", "unpa", "uta", "utala", "walo", "wan",
                "waso", "wawa", "weka", "wile",};
```

I add a public class "TokiPonaSpellChecker" with two private lists of strings. The first list, "validWords," contains all valid words in Toki Pona language. The list, "validNameandPlaces," contains valid names and places in the language.



```
for (String word : words)
{
    validWords.add(word);
}
```

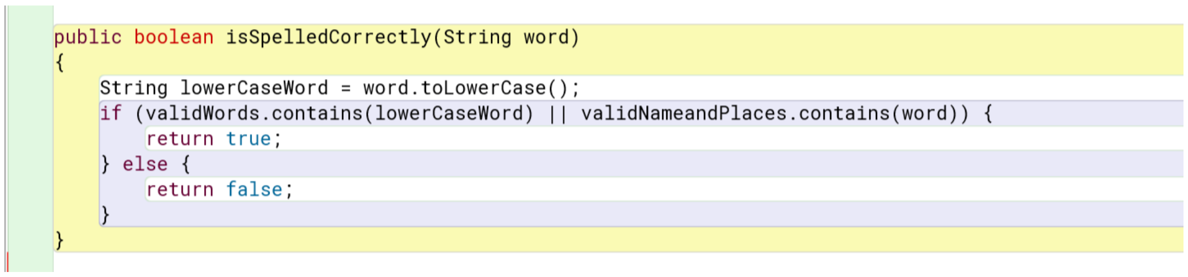
I add a foreach loop that iterates over an array of strings "words". this loop takes each string in the "words" array and adds it to the "validWords" list. This is an efficient way to initialize a list with a predetermined set of values.



```
        }
        String[] NameandPlaces = {"jan Tanka", "jan Eki", "jan Moli ", "jan Luka","jan Kasi","jan Anseto",
                "ma Apeka", "ma Pakasa", "ma Belaru", "ma Pape", "ma Kanata", "ma Tosi", "ma Atisa",
                "ma Isala", "ma Inuka", "ma Japoni", "ma Kena", "ma Lipija"};
        for (String NameandPlace : NameandPlaces)
        {
            validNameandPlaces.add(NameandPlace);
        }
    }
```

I add an initializes an array of strings " NameandPlaces" with a list of Toki Pona names and places.

I add a loop takes each name or place in the "NameandPlaces" array and adds it to the "validNameandPlaces" list.



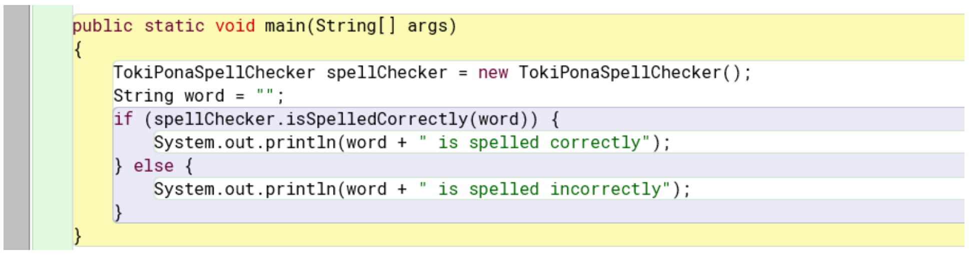
```
    public boolean isSpelledCorrectly(String word)
    {
        String lowerCaseWord = word.toLowerCase();
        if (validWords.contains(lowerCaseWord) || validNameandPlaces.contains(word)) {
            return true;
        } else {
            return false;
        }
    }
```

I add a defines a public method "isSpelledCorrectly" takes a string parameter "word" and returns a boolean value.

It converts "word" to lowercase using "toLowerCase" method to ensure the case of input string does not affect the comparison with valid words, names/places in "validWords" and "validNameandPlaces" lists.

Checks if "lowerCaseWord" is contained in either the "validWords" list or the "validNameandPlaces" list using the "contains" method.

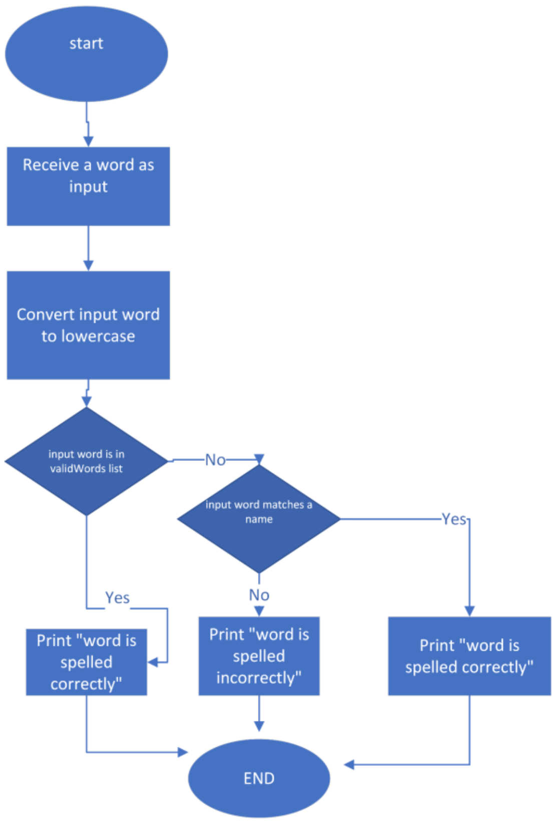
If "lowerCaseWord" is found in either list, it returns true, indicating that the input string is spelled correctly. Otherwise, it turns false, indicating the input string is not spelled correctly according to Toki Pona language.

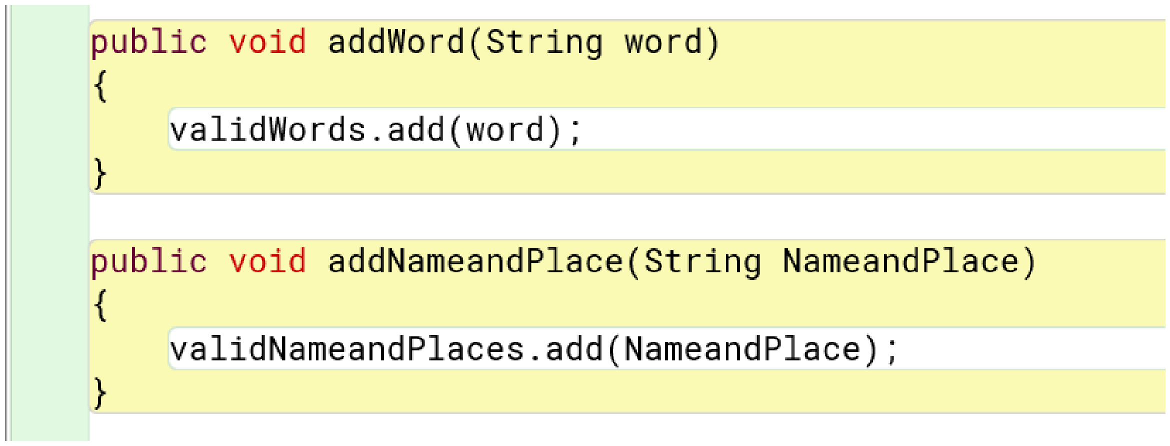


```java
public static void main(String[] args)
{
    TokiPonaSpellChecker spellChecker = new TokiPonaSpellChecker();
    String word = "";
    if (spellChecker.isSpelledCorrectly(word)) {
        System.out.println(word + " is spelled correctly");
    } else {
        System.out.println(word + " is spelled incorrectly");
    }
}
```

I add a defines a "main" method, which is the entry point for Java application.

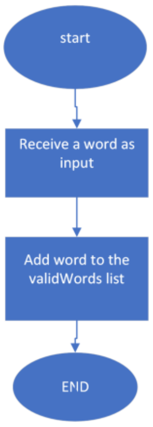
I create an instance of "TokiPonaSpellChecker" class "spellChecker". This object will be used to check the spelling of Toki Pona words and names/places, The next line initializes a string variable "word" with an empty string. Uses an "if-else" statement to check if the empty string is spelled correctly using "isSpelledCorrectly" method of "spellChecker" object. If the empty string is spelled correctly, it will print" is spelled correctly". Otherwise, the program will print" is spelled incorrectly".


start
Receive a word as input
Convert input word to lowercase
input word is in validWords list
No
input word matches a name
Yes
Yes
No
Print "word is spelled correctly"
Print "word is spelled incorrectly"
Print "word is spelled correctly"
END

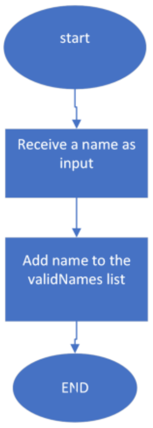


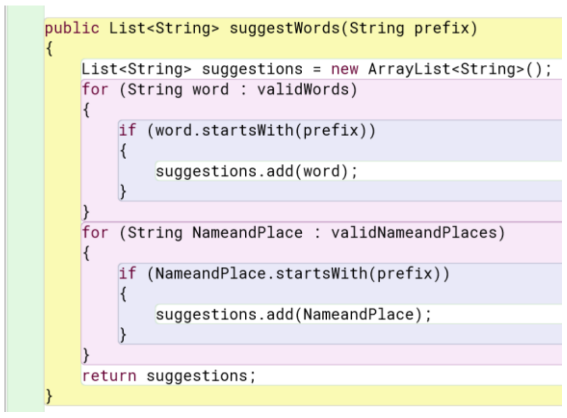
```
public void addWord(String word)
{
    validWords.add(word);
}

public void addNameandPlace(String NameandPlace)
{
    validNameandPlaces.add(NameandPlace);
}
```

I add a method "addWord," takes a string parameter "word" and adds it to "validWords" list using "add" method. It adds new words to the list of valid Toki Pona words.



"addNameandPlace," takes a string parameter "NameandPlace" and adds it to the "validNameandPlaces" list using "add" method. It adds new names or places to the list of valid Toki Pona names/places.



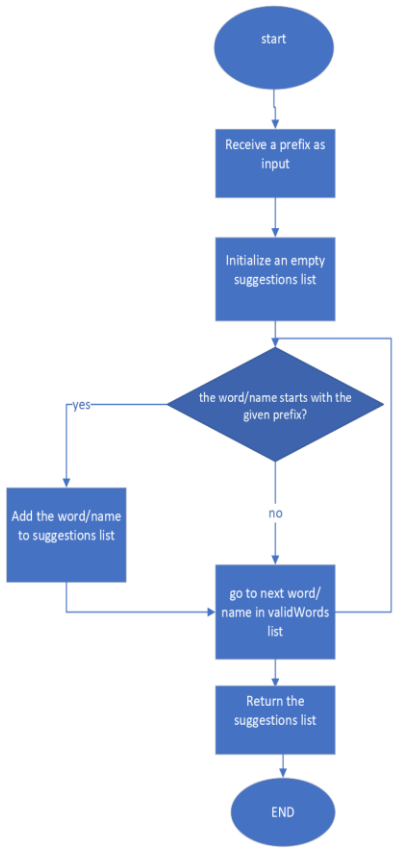

```
public List<String> suggestWords(String prefix)
{
    List<String> suggestions = new ArrayList<String>();
    for (String word : validWords)
    {
        if (word.startsWith(prefix))
        {
            suggestions.add(word);
        }
    }
    for (String NameandPlace : validNameandPlaces)
    {
        if (NameandPlace.startsWith(prefix))
        {
            suggestions.add(NameandPlace);
        }
    }
    return suggestions;
}
```

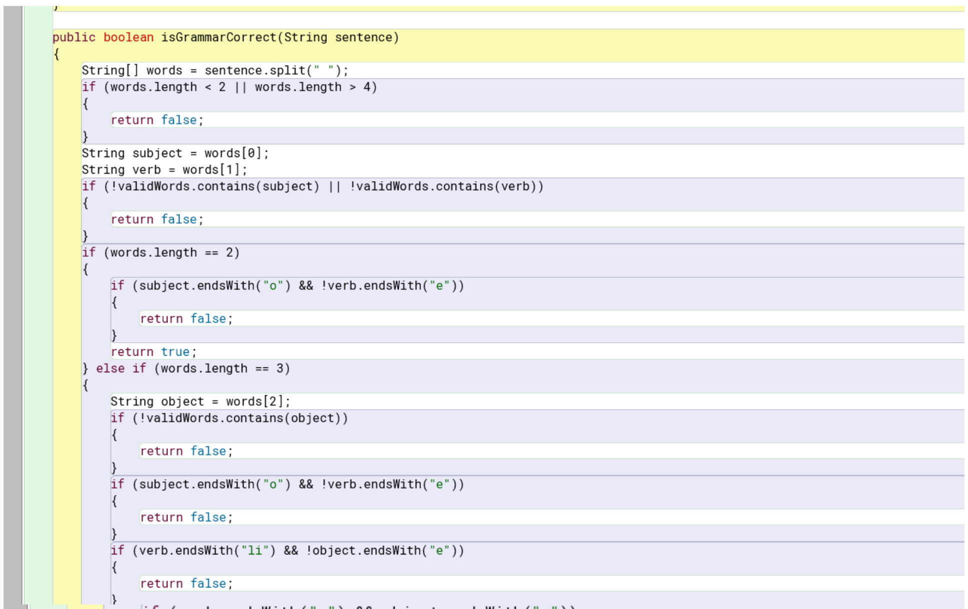
I add a defines public method "suggestWords" takes a string parameter "prefix" and returns a list of strings. It creates a new empty list of strings "suggestions" using "ArrayList" class.

I use a foreach loop to iterate over each string in "validWords" list. For each string in the list, the loop checks if it starts with "prefix" string using "startsWith" method. If the string does start with prefix, it is added to "suggestions" list using "add" method.

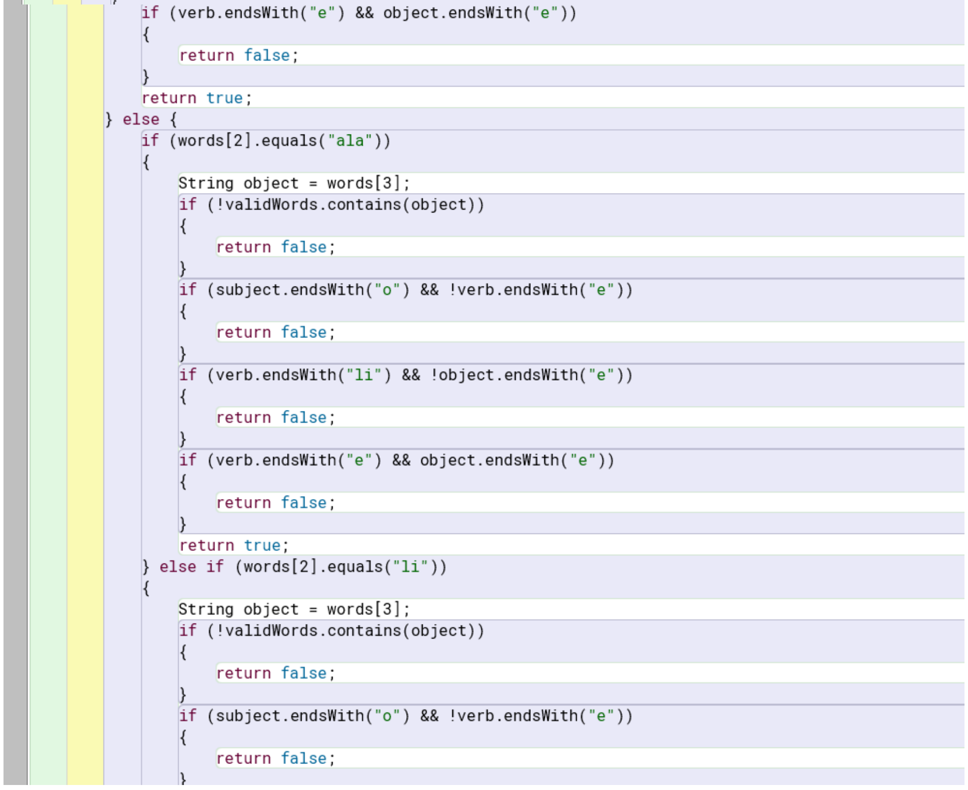
I used another foreach loop to iterate over each string in "validNameandPlaces" list. For each string in the list, loop checks if it starts with "prefix" string using "startsWith" method. If the string does start with prefix, it is added to "suggestions" list using "add" method. It will return "suggestions" list, contains all the words, names/places start with specified prefix.

This useful for providing autocomplete or suggestion functionality in a Toki Pona text editor or other application that uses TokiPonaSpellChecker class.

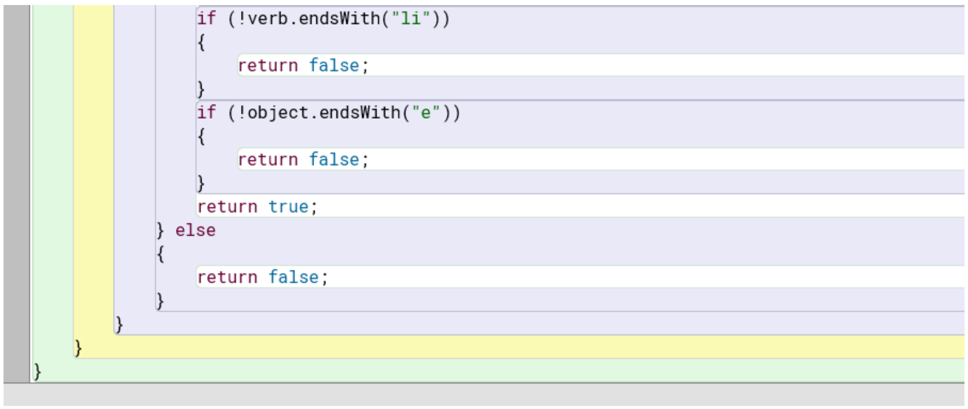

start
Receive a prefix as input
Initialize an empty suggestions list
the word/name starts with the given prefix?
yes
no
Add the word/name to suggestions list
go to next word/ name in validWords list
Return the suggestions list
END



```
    public boolean isGrammarCorrect(String sentence)
    {
        String[] words = sentence.split(" ");
        if (words.length < 2 || words.length > 4)
        {
            return false;
        }
        String subject = words[0];
        String verb = words[1];
        if (!validWords.contains(subject) || !validWords.contains(verb))
        {
            return false;
        }
        if (words.length == 2)
        {
            if (subject.endsWith("o") && !verb.endsWith("e"))
            {
                return false;
            }
            return true;
        } else if (words.length == 3)
        {
            String object = words[2];
            if (!validWords.contains(object))
            {
                return false;
            }
            if (subject.endsWith("o") && !verb.endsWith("e"))
            {
                return false;
            }
            if (verb.endsWith("li") && !object.endsWith("e"))
            {
                return false;
            }
```



```
            if (verb.endsWith("e") && object.endsWith("e"))
            {
                return false;
            }
            return true;
        } else {
            if (words[2].equals("ala"))
            {
                String object = words[3];
                if (!validWords.contains(object))
                {
                    return false;
                }
                if (subject.endsWith("o") && !verb.endsWith("e"))
                {
                    return false;
                }
                if (verb.endsWith("li") && !object.endsWith("e"))
                {
                    return false;
                }
                if (verb.endsWith("e") && object.endsWith("e"))
                {
                    return false;
                }
                return true;
            } else if (words[2].equals("li"))
            {
                String object = words[3];
                if (!validWords.contains(object))
                {
                    return false;
                }
                if (subject.endsWith("o") && !verb.endsWith("e"))
                {
                    return false;
                }
```



```
                if (!verb.endsWith("li"))
                {
                    return false;
                }
                if (!object.endsWith("e"))
                {
                    return false;
                }
                return true;
            } else
            {
                return false;
            }
        }
    }
}
```

I add a public method "isGrammarCorrect" that takes a string parameter "sentence" and returns a Boolean value indicating whether the sentence is grammatically correct according to Toki Pona's rules.

I add this method splits the input sentence into an array of individual words using the "split" method with a space as the delimiter. Then checks the length of the words array. If it is less than 2 or greater than 4, the method returns false, as Toki Pona sentences are typically composed of 2-4 words.

After that it will extracts the subject and verb words from the words array and checks if they are valid words using "validWords" list. If either subject or verb is not valid, it returns false.

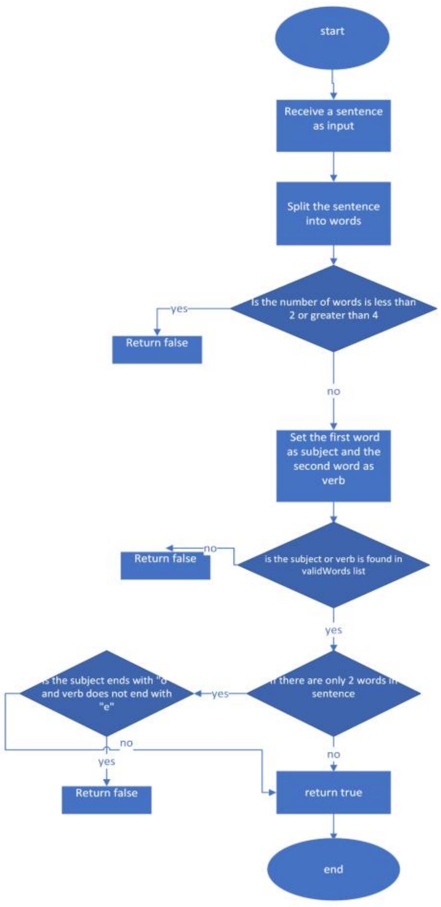
It performs additional checks based on the length of words array:

- If the sentence has only two words, it checks if the subject ends with "o" and verb ends with "e". If so, it returns false.

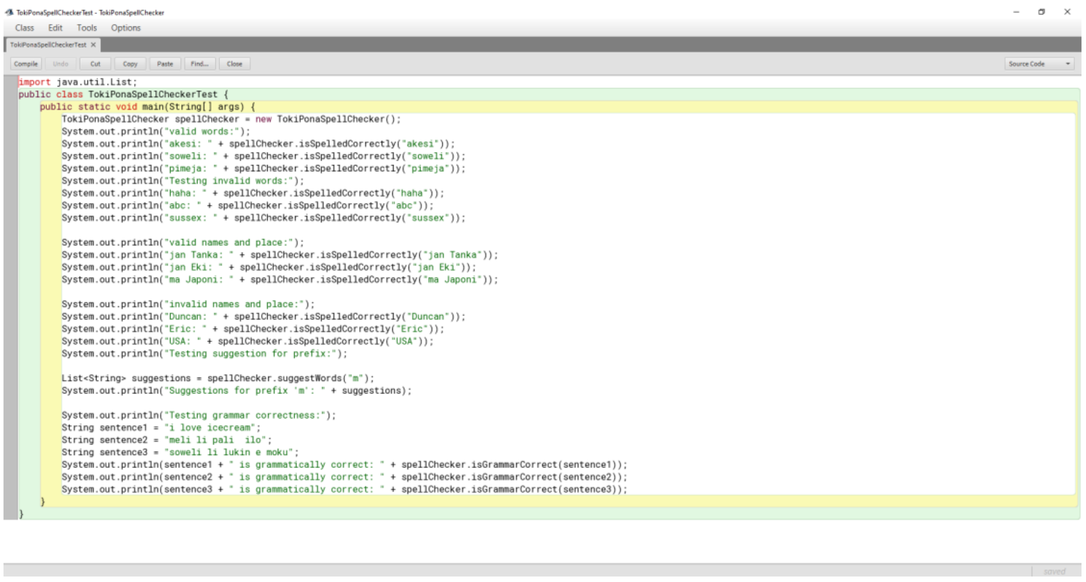
- If the sentence has three words, method extracts object word and checks if it is a valid word. The method performs additional checks to ensure the sentence structure is correct according to Toki Pona grammar rules.

- if the sentence has four words, it checks the third word in the array to determine whether it is "ala" or "li". Depending on the value of the third word, it extracts the object word and performs additional checks to ensure that the sentence structure is correct according to Toki Pona grammar rules.

If all the checks pass, it returns true, Otherwise, it turns false.


start
Receive a sentence as input
Split the sentence into words
Is the number of words is less than 2 or greater than 4
yes
Return false
no
Set the first word as subject and the second word as verb
is the subject or verb is found in validWords list
no
Return false
yes
there are only 2 words in sentence
yes
s the subject ends with "o" and verb does not end with "e"
no
yes
Return false
no
return true
end

Test code:

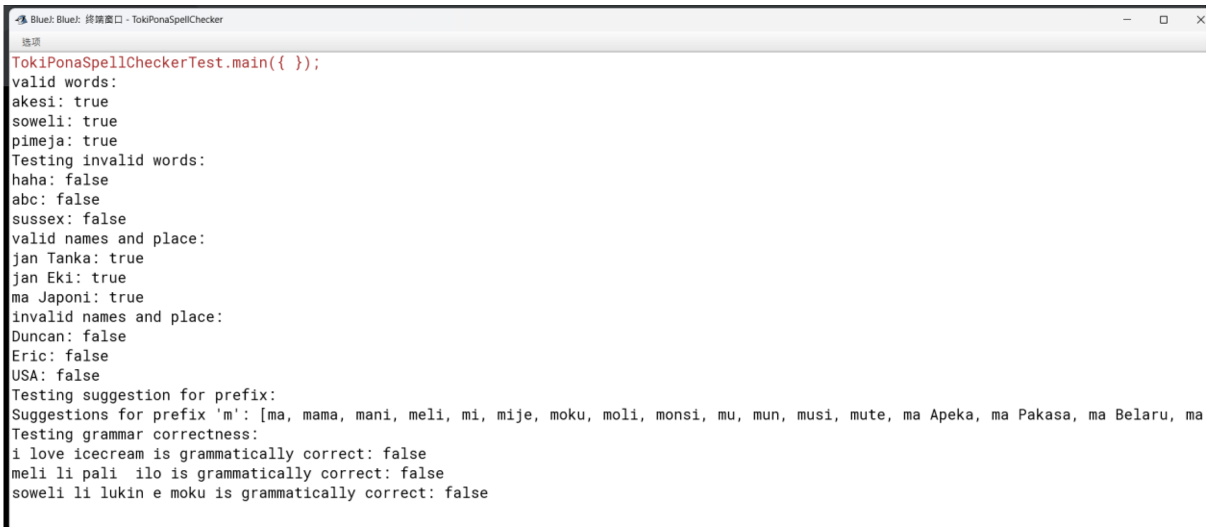


```java
import java.util.List;
public class TokiPonaSpellCheckerTest {
    public static void main(String[] args) {
        TokiPonaSpellChecker spellChecker = new TokiPonaSpellChecker();
        System.out.println("valid words:");
        System.out.println("akesi: " + spellChecker.isSpelledCorrectly("akesi"));
        System.out.println("soweli: " + spellChecker.isSpelledCorrectly("soweli"));
        System.out.println("pimeja: " + spellChecker.isSpelledCorrectly("pimeja"));
        System.out.println("Testing invalid words:");
        System.out.println("haha: " + spellChecker.isSpelledCorrectly("haha"));
        System.out.println("abc: " + spellChecker.isSpelledCorrectly("abc"));
        System.out.println("sussex: " + spellChecker.isSpelledCorrectly("sussex"));

        System.out.println("valid names and place:");
        System.out.println("jan Tanka: " + spellChecker.isSpelledCorrectly("jan Tanka"));
        System.out.println("jan Eki: " + spellChecker.isSpelledCorrectly("jan Eki"));
        System.out.println("ma Japoni: " + spellChecker.isSpelledCorrectly("ma Japoni"));

        System.out.println("invalid names and place:");
        System.out.println("Duncan: " + spellChecker.isSpelledCorrectly("Duncan"));
        System.out.println("Eric: " + spellChecker.isSpelledCorrectly("Eric"));
        System.out.println("USA: " + spellChecker.isSpelledCorrectly("USA"));
        System.out.println("Testing suggestion for prefix:");

        List<String> suggestions = spellChecker.suggestWords("m");
        System.out.println("Suggestions for prefix 'm': " + suggestions);

        System.out.println("Testing grammar correctness:");
        String sentence1 = "i love icecream";
        String sentence2 = "meli li pali  ilo";
        String sentence3 = "soweli li lukin e moku";
        System.out.println(sentence1 + " is grammatically correct: " + spellChecker.isGrammarCorrect(sentence1));
        System.out.println(sentence2 + " is grammatically correct: " + spellChecker.isGrammarCorrect(sentence2));
        System.out.println(sentence3 + " is grammatically correct: " + spellChecker.isGrammarCorrect(sentence3));
    }
}
```

I use defined main method to class "TokiPonaSpellCheckerTest" that tests the methods of the "TokiPonaSpellChecker" class.

The main method creates a new instance of TokiPonaSpellChecker class and uses it to test the isSpelledCorrectly method by checking the spelling of valid, invalid words and names/places. It prints the results of these tests to the console. It tests the SuggestWords method by calling it with "m" prefix and printing the resulting list of suggestions to the console. Also a method tests isGrammarCorrect by checking grammar of three example sentences and printing the results to the console. Add this test code to verify that the methods of TokiPonaSpellChecker class work correctly and return the expected results.

Screenshot of the output in blue j:



```
TokiPonaSpellCheckerTest.main({ });
valid words:
akesi: true
soweli: true
pimeja: true
Testing invalid words:
haha: false
abc: false
sussex: false
valid names and place:
jan Tanka: true
jan Eki: true
ma Japoni: true
invalid names and place:
Duncan: false
Eric: false
USA: false
Testing suggestion for prefix:
Suggestions for prefix 'm': [ma, mama, mani, meli, mi, mije, moku, moli, monsi, mu, mun, musi, mute, ma Apeka, ma Pakasa, ma Belaru, ma
Testing grammar correctness:
i love icecream is grammatically correct: false
meli li pali  ilo is grammatically correct: false
soweli li lukin e moku is grammatically correct: false
```

Reference:

W3school for java coding: https://www.w3schools.com/java/default.asp

Toki Pona (official site): https://tokipona.org/

Toki Pona Word list: https://en.wikibooks.org/wiki/Toki_Pona/Word_list

Toki Pona place list : https://jan-ne.github.io/tp/names

What happened when I tried to learn Toki Pona in 48 hours using memes: https://www.theguardian.com/education/2015/jan/08/toki-pona-invented-language-memrise

Grade: 90% (50 pts possible) Excellent work. Well done.

Student: Duncan			
Score	Category	Possible	Comment
18	Outline	20	Good
27	Quality	30	Missing test logs.
7	Product evaluation	10	Good
8	PM Evaluation	10	Good
17	Project Management	20	Missing documents

8	Report - Organisation	10	Good
85		100	
43			
	Outline - Description of the work done, from section 1 : 20%		
	Quality - The quality of the work done, based on Section 1, and Appendix A : 30%		
	Product evaluation - Evaluation of the project output and how it can be improved : 10%		
	Project Management Evaluation – Student's evaluation of how they carried out the project, based on section 3 : 10%		
	Project Management - Our assessment of how well the project was managed. Based on the whole report : 20%		
	Report - Organisation of the report, including structure, presentation and use of language : 10%		