

1 Introduction

This assignment is to help familiarize yourselves with Eiffel, its libraries, and BON. In particular, you should focus on understanding key data structures, like *STRING* and *ARRAY*, which will prove vital in carrying out the project later. You should also focus on understanding the three main relationships in BON: inheritance, aggregation, and association, and when to use them. You may not require all three relationships in solving this assignment, but you should be able to justify the use of any relationships that you apply.

The general problem you will solve is to design and implement a simple pattern-based language translator. The problem is not complicated, and you should be able to produce a small design, written in BON, and a small Eiffel program, to satisfy all the requirements.

2 The Problem

The *Swedish Chef* is a character from the old Muppet Show that was broadcast in the 1970s. To find out more about who the Swedish Chef really is, check out www.muppets.com/chef/chef.htm. The Chef speaks his own language (an amalgam of Swedish, English, and who knows what else), and your task is to design and implement an Eiffel program to translate English sentences into the Chef's language.

In general, language translation (particularly, natural language translation) is very difficult to automate. You might investigate Altavista's Babelfish translator to see how difficult it can be! Translating from English to Swedish Chef-speak is a much simpler task, as it is generally based on replacing a phrase, word fragment, or letter in an English sentence by another phrase, fragment, or letter. Effectively, then, you will be manipulating strings representing English sentences, and transforming them into strings representing Chef sentences.

3 The Translation Rules

The basic rules for translating English to Chef are as follows. Your program should implement all of these rules correctly. Some examples follow.

1. All sentences or paragraphs that are entered should end in **Bork bork bork**.
2. All occurrences of **an** in a sentence must be replaced by **un**.
3. Replace all occurrences of **au** in a sentence with **oo**
4. In any word:
 - replace all **a**'s not followed by whitespace with the letter **e**. Thus, the word **a** by itself would not be translated to **e**
 - replace all **o**'s in a word by **u**. Example: **Doh** becomes **Duh**.
 - replace the first occurrence of **i** in a word with **ee**. Example: **This** becomes **Thees**.
 - replace all **en**'s not followed by another letter with **ee**. So **en** at the end of a word would be translated to **ee**
 - replace **f** by **ff**
 - replace **ir** by **ur**
 - replace **ow** by **oo**
5. If there is an **e** at the end of a word, replace it by **e-a**.
6. Replace **E** at the beginning of a word by **I**.
7. Replace all occurrences of **the** by **zee**

8. Replace all occurrences of **u** by **oo**
9. Replace all occurrences of **v** by **f** and all occurrences of **w** by **v**
10. Replace all occurrences of the word fragment **tion** by **shun**.

Here are some examples of translations. Your program should generate exactly the translations shown for the given inputs.

```
% This is a test.  
> Thees is a test.  
  Bork bork bork!  
% Do you know the way to San Jose?  
> Du yuoo knoo zee vey tu Sun Juse-a?  
  Bork bork bork!
```

Your translator should be able to translate English text from standard input (i.e., from the keyboard) as well as ASCII text files. All output should be to standard output (i.e., the screen).

4 What To Do?

This assignment has two parts. The first part is to construct a software design for your Enchefalizer. The design is to be presented in the BON modelling language. Effectively, your design will diagrammatically show the classes that you will need to solve the problem, and the relationships needed between the classes. The second part of the assignment is to use your design to implement your Eiffel program.

You must produce your BON design first. Do not start the assignment by writing code: write BON models, then produce code. If it becomes apparent, during marking, that you have merely reverse engineered a BON model from a program, then you will be awarded a lower grade!

Your BON design will consist of several diagrams. The first diagram will be a high-level description of the architecture of your translator: the classes that you will need (drawn as ellipses), and their relationships. You need show *no* class interface details at this stage. Separate diagrams, one for each class, must also be provided showing interface details. Thus, if your translator requires five classes, you will have six diagrams. Note that you do not need to present interface details for library classes, e.g., *STRING* and *ARRAY*.

A proper class interface diagram includes: features, information hiding details, pre- and postconditions for features, and class invariant details. Please use meaningful names for all features of classes. Note that a feature called *i* is *not* well-named!

Your design is not complete without discussion. In presenting your design, you should answer the following questions:

1. Why are the classes that we have used really classes?
2. Why have we chosen to relate classes in the way shown?
3. Why is our design superior to any other?

The point of this discussion is to sell your design to the marker and reader, and to make you to think about different possibilities for structuring and representing your translator.

5 What to Hand In?

You should hand in, on paper, a copy of your design (all diagrams, plus discussion). You should also hand in, on paper, a copy of your Eiffel source code and a brief user manual. Your source code should be commented properly, and should make good use of require and ensure clauses as well as class invariants. Electronically submit (see the web page for submission instructions) a zipped tar-file with the name `a1.tar.zip` containing: your source code, your user manual, and appropriate testing. Testing consists of both test cases (with explanations of what you are testing) and the output of testing. Please speak to the instructor about different ways of presenting the results of testing. Your testing should demonstrate the correctness of your translator.