Lab Assignment 6
Literal Translator

Due Date

One-week assignment

Lab Time	Due Date
Wednesday Lab	**04/05/2017 11:59 PM**
Thursday Lab	**04/06/2017 11:59 PM**

Concepts

- Processing strings and chars
- Handling exceptions

Background

One simple technique for machine translation involves going word by word in the source text, and replacing it with a corresponding word in the target language. Such literal translation (cf. http://en.wikipedia.org/wiki/Literal_translation) usually results in a close but inaccurate translation, primarily because the rules for word order vary from language to language.

However, this literal translator actually has some good uses. First, it's not always necessary to have a perfect translation, especially if it is much quicker and easier not to. Second, other methods use such a translation to generate a set of words, and then statistically find the most likely ordering for those words. For example, try putting the following words in the correct order (you probably can do it easily because you know that certain words generally follow other words):

> have programming a seen never I language better

Problem Specification

Write a program that provides a literal translation for text entered by a user. To determine how to translate the words, you will use something called a *parallel corpus.* A parallel corpus is a large collection of text in two or more languages.

For this assignment the corpus will be much simpler than most corpora, in that each line contains only a single word in each language, separated by a comma. All you will need to do is find the word in the first language, and replace it with its partner in the second. If the corpus does not contain a particular word, you should leave it untranslated.

The following is an example of a parallel corpus text file (French-English). The first line indicates the number of entries (you may assume this number is correct). Accents on letters are ignored (for easy keyboard input). Sometimes two different words might be translated to the same word ("le" / "la" both translate to "the"), but you may assume each word on the left-hand side appears only once.

le,the
la,the
meduse,jellyfish
garcon,boy
danse,dances
avec,with
paresseux,lazy
puante,smelly

Using this corpus, a literal translator can translate:

Le garcon paresseux danse avec la meduse puante!

into:

The boy lazy dances with the jellyfish smelly!

For this assignment, provide two corpus files: one to convert from English to French and another to convert from French to English. Select the appropriate corpus file based on the user's option.

Design Phase

Your program should:

- Allow the user to select an option (1 or 2) for what translation is to be done (English to French OR French to English). This determines the file name of the corpus to be used for the translation (this way your program could be used for multiple languages).
    - Display a message indicating the corpus file was successfully read.
- Allow the user to enter some source text that is to be translated.
- When the user commits the input by pressing on the "enter" key, translate the text and display the result.
    - The source text may contain capital letters and/or punctuation. You should convert it to lowercase and ignore punctuation, except:
        - if the first character in the source is capitalized, capitalize it in the result as well.
        - if the last character in the source is punctuation (*not* a letter or digit), include it in the result as well.
    - If a source word cannot be found in the corpus, just leave it untranslated (this is good for things like proper names).
- Your program should loop continuously till the user indicates that s/he wants to quit.
    - With each loop, prompt the user to re-enter an option to determine the filename of the corpus to be used, as well as a new source text to be translated.
- If any exceptions are encountered, display a message to the user indicating the problem.

Your application **must** perform exception handling using *try-catch* blocks and *throws* statements where appropriate. You **must** handle the following exceptions, but you may include more if you feel it necessary. For the following exceptions, you should display an appropriate message to the user and prompt the user to re-enter the information.

- **FileNotFoundException**
    - in case a user tries to open a corpus file that does not exist
- **InputMismatchException** OR **NumberFormatException**
    - the specific Exception depends on what methods you use to read/parse the first line of input
        - if the first line of input is not a number, handle the appropriate exception
        - if the input is a number but not any of the valid options, throw and catch an InvalidInputException and display an appropriate message.
- **EmptyStringException** (this is a class you will create; it should extend Exception class)

- in case the user inputs nothing but presses down the "Enter" key.

- **InvalidInputException** (this is another class you will create; it should also extend the Exception class)
  - in case the user inputs an option that is not one of the valid options (e.g. 3, instead of 1 or 2).

Basic Structure

Your program should have a user interface class **UserInterface** which implements the interface **IUserInterface** (given below). This class interfaces with the user: requests input, reads in user input and uses the appropriate classes to perform the required tasks.

```
public interface IUserInterface {
        // Calls the necessary methods to: display a greeting, ask the user what s/he wants to do; update the
        // courpus file name and the text to be translated based on the user's input and uses these to initialize
        // a Translator object; loops until the user no longer wants to translate any sentences.
        void runProgram() throws FileNotFoundException;

        // Displays a greeting to the user and indicates briefly what the program does.
        void greeting();

        // Displays the translation options to the user and, after reading in the option selected by the user, sets
        // the name of the corpus file.
        // Uses try-catch block(s) to enforce correct input.
        void getCorpusFileName();

        // Requests the text the user wants to translate (i.e. the source text) and sets the name of the source text.
        void getSourceText();

        // Uses the Translator object to translate the source text and displays the translated text to the user.
        void translate() throws FileNotFoundException;
}
```

You should also include a Translator class in your project. This implements the ITranslator interface which is given below.

```
public interface ITranslator {
        // Reads in the data from the input file and stores the data in a two dimensional array for easy retrieval
        // during the translation process.
        void readCorpus() throws FileNotFoundException;

        // Calls the necessary methods to perform the translation and returns the translated text.
        String translate();

        // Uses a private method: "private String lookup(String key);" to look up an English word in the source
        // text, and sets the attribute for the translated word (in French).
        void englishToFrench();

        // Uses a private method: "private String lookup(String key);" to look up a French word in the source text,
        // and sets the attribute for the translated word (in English).
        void frenchToEnglish();
}
```

Your project should also have a main class (**LA6Main**) with the following main method which **should NOT be modified**.

```
public static void main(String[] args) {
        // TODO Auto-generated method stub
        IUserInterface uI = new UserInterface();
        try {
                uI.runProgram();
        }
        catch (FileNotFoundException e) {
                System.out.println("FIle not found!");
        }
}
```

Pseudocode
Write pseudocode for all the methods in your classes.

The UML and Pseudocode are due in class.

- ✓ Submit a printed copy to the Lab instructor.

Implementation Phase

Create a project in Eclipse named LA6*yourLastName*. In this project, create a package named edu.wmich.cs1120.yourName.LA6, and the classes from the Design Phase above, along with attributes and constructors if necessary. Add all of the methods designed earlier, translating your above pseudocode into Java code. Include additional methods and/or attributes if necessary, but do not by-pass the ones already given in the interface or referred to earlier in this document.

An example output is provided below. Your program should print out a similar output.

Testing Phase

Example Output:

```
Welcome to the Translator! Bienvenue!!
You can translate sentences from English to French and vice versa!
So, what would you like to do? Please select from one of the following:
>> From English to French: (input '1')
>> From French to English: (input '2')
k
Invalid input! Input must be an integer! Try again:
2.3
Invalid input! Input must be an integer! Try again:
5
Invalid input! Input should be 1 or 2. Please try again:
1
Input the sentence for translation:
The lazy boy dances with the smelly jellyfish!
Your sentence translated is:
 >> Le paresseux garcon danse avec le puante meduse!
Another translation? (y/n)
2
Invalid input! Please enter 'y' or 'n':
Y
So, what would you like to do? Please select from one of the following:
>> From English to French: (input '1')
>> From French to English: (input '2')
2
```

Input the sentence for translation:

Invalid (empty) string!
Input the sentence for translation:
Le meduse puante Chacha.
Your sentence translated is:
>> The jellyfish smelly chacha.
Another translation? (y/n)
n

Assignment Submission

Coding Standards

You must adhere to all conventions in the CS 1120 Java coding standards. This includes the use of white space for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods.

Assignment Submission

- Generate **Javadoc** in Eclipse for your program. This should include descriptive comments for each class, method, parameter and return value.
- Generate a .zip file that contains all your files (as shown in class), including:
  - ✓ Program Files
  - ✓ Any input or output files
- Submit the .zip file to the appropriate folder (LA6HomePortion) in eLearning.

NOTE: The eLearning folder will be inaccessible after the due date/time.