**Lab Assignment 5:**

**SMS Language & PigLatin**

|  |
| --- |
| **Due Date**  **(a one-week LA)** |
| **3/23/16 11:59pm – For Wednesday Lab Sections**  **3/24/16 11:59pm – For Thursday Lab Section** |

**Objectives**

* To practice manipulating strings and chars using String methods in Java
* To practice writing code with good exception handling using try-catch blocks and throw statements.
* To practice writing and using an Exception class.

**Problem Specification**

Short Message Service (SMS) is a communication service that allows sending text messages of 160 or fewer characters between mobile phones. Since the length of the SMS message is limited, SMS Language – abbreviations of common words and phrases in mobile text messages is often used. For example –“in my opinion” is “IMO” in SMS language. For further information, you can (if you choose) research SMS language online at (<http://www.lingo2word.com/>.)

Pig-Latin is a form of a coded language often used for amusement. Many variations exist to form pig Latin words. For simplicity, use the following algorithm:

- to translate each English word into a Pig-Latin word, place the first letter of the English word at the end of the word and add the letters “ay”. Thus the word “Hello” becomes “ellohay” and “the” becomes “hetay”.

In this assignment, your task is to write a Java program to convert SMS to text and vice versa, and also to convert English words to their Pig-Latin equivalent.

The application should exhibit the following functionality:

* Prompt user for input depending on what user wants to do (see options in program output)
  + Use try-catch blocks to enforce correct input. Catch the “InputMismatchException”, and throw and catch the IllegalArgumentException”.
* Allow the user to choose from a menu of options:
  + Text to SMS conversion
    - Convert from normal text to SMS-text.
  + SMS to Text conversion
    - Convert from SMS-text to normal text.
  + Text to PigLatin conversion
    - Convert an English word to PigLatin
  + Quit
    - Exit the application
* Allow the user to continue making requests until s/he selects the Quit option.

**Example Program Output:**

What would you like to do?

Choose from the following menu:

1 - Text to SMS conversion

2 - SMS to Text conversion

3 - Text to PigLatin conversion

0 - Quit

what?

Invalid option. Input must be a number.

Please enter: 0, 1, 2 or 3.

5

Invalid number! Option must be 0, 1, 2 or 3.

1

Input text to be converted to SMS:

what a great game!

Your SMS is:

w@ a gr8 gme!

What would you like to do?

Choose from the following menu:

1 - Text to SMS conversion

2 - SMS to Text conversion

3 - Text to PigLatin conversion

0 - Quit

2

Input SMS to be converted to text:

Error: Input cannot be an empty string.

Input SMS to be converted to text:

ncredbl! uk had d a1 team in d wrld cup.

Your text is:

incredible! England had the best team in the world cup.

What would you like to do?

Choose from the following menu:

1 - Text to SMS conversion

2 - SMS to Text conversion

3 - Text to PigLatin conversion

0 - Quit

3

Input text to be converted to pigLatin:

Error: Input cannot be an empty string.

Input text to be converted to pigLatin:

computer

Your text in pigLatin is:

omputercay

What would you like to do?

Choose from the following menu:

1 - Text to SMS conversion

2 - SMS to Text conversion

3 - Text to PigLatin conversion

0 - Quit

3

Input text to be converted to pigLatin:

java

Your text in pigLatin is:

avajay

What would you like to do?

Choose from the following menu:

1 - Text to SMS conversion

2 - SMS to Text conversion

3 - Text to PigLatin conversion

0 - Quit

0

Program terminating ...

You may use the following for the Text to SMS conversion (feel free to be creative and perform the conversion more efficiently but that’s not a requirement of the assignment).

// “input” is the user input. For example, if user inputs the string “What a great game by England” then input = “What a great game by England”

NOTE: The “replace” method is case-sensitive.

input = input.replace("what", "w@");

input = input.replace("incredible", "NcreDbl");

input = input.replace("best", "A1");

input = input.replace("the", "d");

input = input.replace("england", "UK");

input = input.replace("world", "wrld");

input = input.replace("and", "n");

input = input.replace("stuff", "stuf");

input = input.replace("that", "dat");

input = input.replace("game", "gme");

input = input.replace("has", "hs");

input = input.replace("twisted", "twistD");

input = input.replace("one", "1");

input = input.replace("ten", "10");

input = input.replace("another", "nothA");

input = input.replace("again", "agn");

input = input.replace("before", "B4");

input = input.replace("up", "^");

input = input.replace("even", "evn");

input = input.replace("great", "gr8");

input = input.replace("have", "av");

input = input.replace("night", "nite");

input = input.replace("good", "gud");

The conversions for SMS to Text are given below:

input = input.replace("w@", "what");

input = input.replace("ncredbl", "incredible");

input = input.replace("a1", "best");

input = input.replace(" d ", " the ");

input = input.replace("uk", "England");

input = input.replace("wrld", "world");

input = input.replace(" n ", " and ");

input = input.replace("stuf ", "stuff");

input = input.replace(" dat ", " that ");

input = input.replace("gme", "game");

input = input.replace(" hs ", " has ");

input = input.replace("twistd", "twisted");

input = input.replace("1", "one");

input = input.replace("notha", "another");

input = input.replace("agn", "again");

input = input.replace("b4", "before");

input = input.replace("^", "up");

input = input.replace("evn", "even");

input = input.replace("gr8", "great");

input = input.replace("av", "have");

input = input.replace("nite", "night");

input = input.replace("have", "av");

input = input.replace("gud", "good");

**Exception Handling**

Your application ***must*** contain good exception handling in the form of *try-catch* blocks and *throw* statements. If an exception occurs, the application should indicate the problem and attempt to recover gracefully. You should handle the following exceptions:

* InputMismatchException
* IllegalArgumentException

When reading in the input strings/words for the different conversions, you should throw and catch an EmptyStringException. This is not one of the Java-defined exceptions, so you will need to write this class (which extends the Exception class). This exception should be thrown if the user hits the return key without having entered an input. An appropriate message should be displayed to the user to make it clear what the problem is (see example program output).

Your program must not “crash” no matter what input is entered by the user; if it does, not all exceptions have been properly handled in the program.

**Design** **Requirements**

***You are required to finish the design report before you leave the lab.***

*Note: Lab reports**will be worth* ***40%*** *of the total LA5 grade.*

Your lab report should contain the following two parts.

1. **Basic Structure**

This section of your report should be done using UML diagrams. Your diagrams should show the relationships between classes (classes implementing interfaces, association between classes (if any).

Your project should have the following classes and interfaces:

* The **main** class which is provided for you and should not be modified.
* The **Controller** class which implements the **IController** interface.
* The **Sms** class which implements the **ISms** interface.
* The **PigLatin** class which implements the **IPigLatin** interface
* The **EmptyStringException** class which extends the **Exception** class (which is defined by Java and does not need to be included in your poject).

The interfaces and the main class are provided below. These are NOT to be modified.

**package** edu.wmich.cs1120.LA5;

**public** **interface** IController {

/\*\*

\* Sets the SMS object.

\* **@param** mySms - the object used to set the SMS object.

\*/

**void** setSmsObject(ISMS mySms);

/\*\*

\* Sets the PigLatin object.

\* **@param** myPigLatin - the object used to set the PigLatin object.

\*/

**void** setPigLatinObject(IPigLatin myPigLatin);

/\*\*

\* This method displays the menu to the user, uses a try-catch block to read in the

\* input from the user and, based on the user's choice, calls one of the three

\* "convert..." methods in this class. This method should keep prompting the user

\* for input until the user wants to quit.

\*/

**void** runProgram();

/\*\*

\* This method prompts the user for text to be converted into SMS-text. It uses the

\* SMS object in this class to call the appropriate method in the SMS class to do

\* the conversion from text to SMS-text.

\*/

**void** convertTextToSMS();

/\*\*

\* This method prompts the user for SMS-text to be converted into normal text. It

\* uses the SMS object in this class to call the appropriate method in the SMS class

\* to do the conversion from SMS-text to normal text.

\*/

**void** convertSMSToText();

/\*\*

\* This method prompts the user for a word to be converted to PigLatin. It uses the

\* PigLatin object to call the appropriate method in the PigLatin class to do the

\* required conversion. To convert to PigLatin, remove the first letter of the word

\* and append to what is left of the word, then append "ay". So, "hello" becomes

\* "ellohay".

\*/

**void** convertTextToPigLatin();

}

**package** edu.wmich.cs1120.LA5;

**public** **interface** ISms {

/\*\*

\* Sets the input field to userInput (myInput).

\* **@param** myInput - user's input.

\*/

**void** setInput(String myInput);

/\*\*

\* Converts text from user to SMS-text and returns it.

\* **@return** - SMS-text form of user's input.

\*/

String textToSMS();

/\*\*

\* Converts SMS-text to normal text and returns it.

\* **@return** - normal text form of user's input.

\*/

String smsToText();

}

**package** edu.wmich.cs1120.LA5;

**public** **interface** IPigLatin {

/\*\*

\* Sets the input field to userInput (myInput).

\* **@param** myInput - user's input.

\*/

**void** setInput(String myInput);

/\*\*

\* Converts user input to PigLatin format.

\* **@return** - PigLatin form of user input.

\*/

String textToPigLatin();

}

**package** edu.wmich.cs1120.LA5;

**public** **class** LA5Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ISMS sms = **new** SMS();

IPigLatin pigLatin = **new** PigLatin();

IController control = **new** Controller(sms, pigLatin);

control.runProgram();

}

}

1. **Pseudocode**

Write pseudocode for all methods, listing them by class. Pseudocode is not required for the interfaces; only for the classes implementing them.

**Additional Requirements**

**Coding Standards**

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

**Javadoc**

You must include **Javadoc** comments and generate a Javadoc document for your project. Points will be deducted in the following cases:

* Javadoc comments are included but the Javadoc document is not generated.
* No Javadoc comments are included at all in the project.

**Testing**

Test your application with different kinds of wrong input to be sure these will not cause your program to crash. Also test your program thoroughly to ensure that the different kinds of conversion are done correctly.

**Assignment Submission**

* Generate a .zip file that contains all your files, including:
  + Program Files
  + Any input or output files
* Submit the .zip file to the appropriate folder in elearning.

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