C212 Lab 3

Intro to Software Systems

Objectives:

- Develop ability to read given instructions (or requirements), and design the software before coding (writing pseudocode or algorithm)
- Implementing static methods
- Reading values from keyboard (Using Scanner class)
- Working with strings, Casting
- Using a simple loop, simple conditions
- JUnit Testing (with **version 5**)

Lab instructions

1. Create a class named Lab3Exercises and Implement the following static methods (we will not use this class to *instantiate objects*) and write **JUnit tests** when asked:

   a) *public static String pigLatinEncoder (String engInput)*
      i. Implement a function that receives an String of one or many English words from user (in the main method) and encodes the input in Pig Latin. The general rules for translating English to Pig Latin are as follows:
         1. When a word begins with one or more consonants, move the consonant(s) to the end of the word preceded by a "-", then add "ay" to the end of that.
         2. When a word begins with a vowel, simply add "-way" to the end of the word.
         3. An exception to the rule is with words containing a 'y'.
            a. If 'y' is the first letter, treat it as a consonant using Rule 1.
            b. If 'y' comes after a group of consonants, treat it as a vowel (i.e. do not move it).

Examples: *"horse" -> "orse-hay", "programmer" -> "ogrammer-pray", "Eclipse" -> "Eclipse-way", "yesterday" -> "esterday-yay", "rhythm" -> "ythm-rhay", "Yggdrasil" -> "asil-Yggdray"*... you can find more online but remember to stick to these particular rules, as other examples may not. Every word in a String is translated: "*You must love using Java" -> "You-way ust-may ove-lay using-way ava-Jay"*
You can assume that all Strings will use no punctuation or extraneous characters like numbers for this method, but you should expect capitalized letters to appear.

      ii. Use the *Scanner* class to get input from the user (in main method) and pass that as a parameter to the method above. Get the returned string and print it along with the user input to show what was translated.
      iii. After thorough testing, Write appropriate **JUnit tests.**

Hint: You can work with one word at a time in reading the English and writing the Pig Latin. All words are, of course, separated by the space character. The substring method may come in handy.

b) *public static String pigLatinDecoder (String pigInput)*

i. Write a static Java method that takes one or many words in Pig Latin (as described above) as input and decodes the string back into English words. You may notice that there could be multiple ways to decode a Pig Latin word back into English, such as with "e-Hay as-way e-thay inner-way". If the proper translation is ambiguous, your translation should include both possible decodings, e.g. "He (was/as) the (winner/ inner)".

Notice that, assuming the English string that was translated used proper capitalization, you should be able to mostly ignore capitalization and still have the decoded string be correct. However, when considering the exception mentioned above, "-way" and "-Way" signify different original English words. Your code should reflect this difference and not print double translations for "-Way". Ex: "We are winners" -> "e-Way are-way inners-way" -> "We (ware/are) (winners/inners)".

ii. Use the **Scanner** class to get the input. You may use a loop if needed.

iii. Write appropriate **JUnit tests**, including tests for going back and forth between English and Pig Latin using the two methods you've made.

c) *public static String randomCarsGenerator (int numCars)*

i. Write a method that accepts the number of cars (as parameter) and then will generate a comma separated String of random cars passing by. For example: if user wants 5 cars, the program could return "Red truck, Purple SUV, Silver sedan, White Jeep, Ambulance" which were generated randomly. Your program must include at least 4 general car types, 4 colors, and two "special" car types that do not have a color, such as police car, news van, etc. **Hint**: You can use a Math.random() or Random class to generate random numbers which can be hardcoded to correspond to different colors and car types, or a special car type. However, do not hardcode *combinations* of Strings like "White Jeep" -- generate them separately and combine.

ii. Test your method for different inputs by printing returned Strings in a main method.

d) *public static String hexToIntNBin (String hexInput)*

i. Implement a method that converts a hexadecimal input into their binary and decimal equivalents. Your Hexadecimal input number must have 0x preceding the hex digits (*hence the type string*) and must have minimum of 4 hex digits. It returns a combined String in the following format. "*Your number is XXXX (in decimal) and YYYY (in binary)*" where XXXX is the positive integer/decimal value and YYYY is the binary equivalent number. For example: for input 0x000A, the result is:
Your number is 10 (in decimal) and 1010 (in binary)
***Note**: You may use intToBinary.* HERE are two very good explanations on converting a hex value to decimal integer & binary.

ii. Test your program properly and Write appropriate **JUnit tests.**

e) Three java source files are provided on Canvas (DebugProgramOne, DebugProgramTwo, and DebugProgramThree). These files have some errors. Help us fix them and make sure that they produce the expected results. Each file includes appropriate instructions. *To compile each file, you can create a new Java Project and then add each file to the src.*
***Note**: In case you have any trouble, I suggest using online resources (like YouTube) which can guide you through the steps. It only takes a few seconds to create a new project and*

add your existing code to it. ***Example****: For Eclipse: Create new Project -> Import Code from the File System*

3. Answer the following questions as comments below your class file or submit a text file:
   a. Give the type and value for each of the following expressions:
      i. 7.8 == 78
      ii. 2.3+5*7.01
      iii. N*4 (treat C as character or 'N')
      iv. 9+8+"21"
   b. Without compiling/running the code or looking at any documentation, predict the value printed by each of the following code fragments. Now compile/run (*you may have to write that in your main method*) and see if your answer was correct. Report the output and whether you were initially correct. If incorrect, explain in one sentence what was the error you missed:

      i. See the following separate code fragments (assume a main if not present):

```
public class Main {
    public static void main (String[] args) {
            double num = 9.78;
            int newNum = (int) num;
            String newStr = String.valueOf(newNum);
            System.out.println(num);
            System.out.println(newNum);
            System.out.println(newStr);
            System.out.println(num+newNum);
            System.out.println(newStr+num);
    }
}
```

      ii. *System.out.printf("%03d",7);*

      iii.
```
String s = "Bye ByeBye";
s = String.replace('y', 'e');
System.out.println(s);
```

   c. *How are Strings immutable in Java? Explain in a few sentences. What alternatives do you have for manipulating Strings?*