**CSCI 232 Lab 4**

Due Thursday June 1st @ 11:59 PM. Please submit this assignment (.java files) to the appropriate dropbox on D2L

**Background and Instructions**

In this lab, you will use Java’s HashMap library to build a program that will convert Emoji Codes (:GrinningFace:) to Emojis 😀. A user will be able to supply a sentence to your program with one-to-many Emoji Codes, and the program will display the same sentence, but with the Emojis substituted properly.

You will use Lab4Demo.java as a starting point, which can be found in the “Starting Code” section.

You will need to define the EmojiTranslator class, which **must** use a HashMap to hold mappings of emoji codes (Strings) to emojis (Strings). Then you must define the following methods:

1. **public HashMap<String,String> loadHashTable()**. This method reads in from a text file (emojis.txt), and fills the HashMap with the proper Key Value pairs. The keys of your HashMap will be the Emoji Codes (:GrinningFace:), and the value linked to the key will be the actual Emoji (😀). This method needs to return the filled HashMap. You can call this method from the constructor of the EmojiConverterClass.

2. **public String convert(String sentence)**. This method takes in a sentence and will convert any word that is an emoji code to the actual emoji using the HashMap you built earlier. You can assume an emoji code will always be a word that begins and ends with a colon. This method needs to return the new sentence with the proper emojis. For example,

**"Go Cats Go! :CatFaceWithWrySmile: Go Cats Go! :CatFaceWithWrySmile: :FlexedBiceps: :GrinningFace:"**

Gets converted to:

Go Cats Go! 😼 Go Cats Go! 😼 💪 😀

3. **public String getEmoji(String code)**. This method takes in a String and will attempt to pull the Emoji who’s Emoji Code matches the argument. When **getEmoji(“Taco”)**, is called, it will attempt to find the :Taco: emoji and then return the Taco Emoji (if it exists) 🌮. If the Emoji does not exist, then a “Emoji not found” error should be printed out (see sample output)

Test your code to make sure it works how you want it to before you turn it in.

**Sample Output**

When you run your program, it should look exactly like the screenshot below. Once you have verified the correctness of your output, you can try your own sentences/emojis

A picture containing text, screenshot, font

Description automatically generated

**Restrictions**

You are not allowed to use any data structure that is **not** a HashMap. The one exception is that you are allowed to use an Array while reading in from emojis.txt when using the .split() method.

**Starting Code**

* **Lab4Demo:** [**https://www.cs.montana.edu/pearsall/classes/summer2023/232/labs/Lab4Demo.java**](https://www.cs.montana.edu/pearsall/classes/summer2023/232/labs/Lab4Demo.java)
* **Emojis.txt (input file you are reading from in part 1)** [**https://www.cs.montana.edu/pearsall/classes/summer2023/232/labs/emojis.txt**](https://www.cs.montana.edu/pearsall/classes/summer2023/232/labs/emojis.txt)

**Grading**

Grading will be done as follows:

• The **loadHashTable** () method is correct - 3 points

• The **convert** () method is correct - 4 points

• The **getEmoji** () method is correct - 3 points

NOTE: If your code does not compile, correctness cannot be verified, and you won’t receive any points for your code. Turn in code that compiles!

**Hints**

* You can use the .charAt() method to check the character at the beginning and end of a word.
* You may find the official HashMap documentation to be helpful: [https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html](https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html%20)