Assignment 3 – Spring 2021
Due Date: by Friday March 26, 2021 11:59PM
How to submit: upload JAVA files to Blackboard

Please note:
- ✓ This is an individual assignment; please do your own work. Sharing and/or copying code in part or whole with/from others will result in a grade of 0 and disciplinary actions for all involved parties. If you run into problems and have done your best to solve them, please talk to me during office hours or by e-mail.
- ✓ There is a 25% grade deduction for every day the assignment is late unless prior permission is granted.

Preamble

Morse code is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called dots and dashes or dits and dahs. Morse code is named for Samuel Morse, an inventor of the telegraph. For further details, I refer to Wikipedia (https://en.wikipedia.org/wiki/Morse_code)

For this assignment, you will write a Java program to convert text (of selected characters) to Morse code using a sequence of dots and dashes. The program consists of the two classes shown in Figure 1. Please note the following:

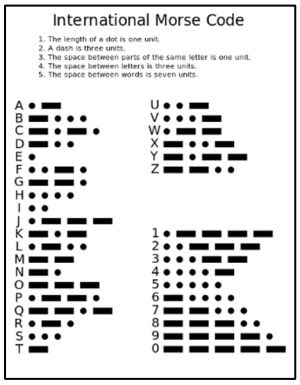

Figure 1[i]

- ✓ Your code MUST compile and run with the test code provided with the assignment (*TestMorseConverter.java*). Please do not modify OR submit the test code.
- ✓ Properly comment your code. Comments should precede variables, methods, and major steps in your code.
- ✓ Follow the Java naming conventions and use the class and method names as shown in the class diagram.
- ✓ File Input: The *java.util.Scanner* class is a multi-purpose class. It can be used to read input from the command line or from a text file. For instance, the statement *new Scanner*(*System.in*) creates an instance for reading input from the *command line*. The statement *new Scanner*(*new File*("file.dat")) creates an instance for reading input from a text file called file.dat. After the instance is created, the behavior is the same as with command-line input. Use *next*( ) to read one *String* entry from the command line or the file; use *nexInt*( ) to read one Integer entry …
- ✓ Submit two *Java* source files – do not submit the test class *TestMorseConverter.java*:
  1. *MorseCode.java*
  2. *MorseCodeConvert.java*

Running the accompanying JAR:

1. Create a directory on your file system (e.g. under *A3* under *Desktop*)
2. Place the *JAR* and both text files in the directory just create.
3. Open a command prompt console and CD to the directory created in step 1 (e.g. *cd Desktop/A3*)
4. Type the following command:
   java -jar Assignment3.jar

Note that the path to *java* must be already added to your system's path, otherwise you must provide the full path to the java executable

Class' Members:

I. ***MorseCode***:

- **Non-default constructor:** accepts character values of ASCII codes between 32 and 90 and an encoding value that cannot be NULL and of length at least one. If either criteria is NOT met, the constructor will throw an Exception with the message: *"The character ≪ the invalid character ≫ is not a supported Morse character"*
- **Setters and Getters:** for *character* and *encoding*.

II. ***MorseCodeConvert***:

- **listCodes**: an *ArrayList* to hold objects of type *MorseCode*. All valid encodings are held as in this list. Each object in the list represented a single conversion.
- **Non-default constructor:**
  Accepts one parameter (the file to be read). The constructor attempts to open the file, if it fails, an exception is thrown with the message *Failed to open file*: ≪ *file name* ≫
  Read all lines from the file; add each valid line to the *ArrayList*. Each line in the input file consists of two columns separated by a tab (i.e. \t). Some lines in the file are corrupt and should be skipped. Make sure to, also, empty lines and lines that do not have exactly two entries. Your program should NOT halt on input errors; read the entire file.
  If an exception from *MorseCode* is encountered, print the exception message, do not add the character to the *ArrayList*, and continue to the next line in the input file.
  For skipped lines, print the message *Invalid line*: ≪ *the skipped line* ≫ and continue with the next line.
- **printEncodingList:** Prints the entire content of the *ArrayList*.
- **encodeString:** Accepts a string parameter and prints the corresponding Morse code for that string.
- **encodeFile:**
  Accepts a file name as input and prints the corresponding Morse code for the entire file's content.
  Attempt to open the file, if it fails, an exception is thrown.
  Read the contents of the file and print the corresponding Morse code.
  If an invalid character is encountered, print `'?'` for its conversion.

Grading:

Item	Points
Comments (Javadoc and major steps)	10
Class MorseCode	
Non-default constructor (char and encoding check)	15
Exception	5
Setters/Getters	10
Class MorseCodeConvert	
Non-default constructor: Exception File read Line skip Populate list	20
printEncodingList()	10
encodeString()	10
encodeFile()	20
	100

Figures:

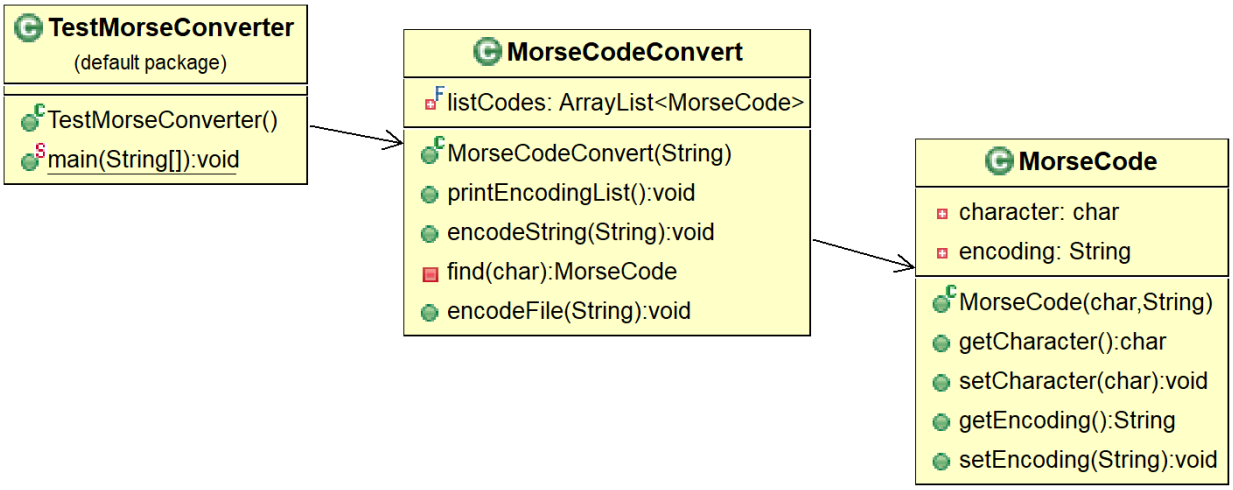

Figure 2: UML Class Diagram

```
Invalid line: ~
Invalid line: GG
java.lang.Exception: The character \ is not a supported Morse character
java.lang.Exception: The character ^ is not a supported Morse character
java.lang.Exception: The character ` is not a supported Morse character
Select one of the following choices:
  1. Print conversion codes.
  2. Convert string.
  3. Convert file.
  4. Exit.
Choice (1-4):
```

Figure 3: Program start output messages.
The message "java.lang.Exception: The character …" is from MorseCode Constructor
The message "Invalid line …" is from MorseCodeConvert Constructor

```
Select one of the following choices:
  1. Print conversion codes.
  2. Convert string.
  3. Convert file.
  4. Exit.
Choice (1-4): 1
('!', -.-.--)
('"', .-..-.)
('#', -.-..-)
('$', ---..-)
('%', .--..-)
('&', .-...)
(''', .-....)
('(', -.--.)
(')', -.--.-)
…
```

Figure 4: Option 1 prints the ArrayList objects

```
Select one of the following choices:
  1. Print conversion codes.
  2. Convert string.
  3. Convert file.
  4. Exit.
Choice (1-4): 2

Enter line to convert: Hello Java
.... . .-.. .-.. --- .--- .- ...- .-
```

Figure 5: Option 2 prompts for a line and prints it's MorseCode equivalence

[i] https://en.wikipedia.org/wiki/File:International_Morse_Code.svg