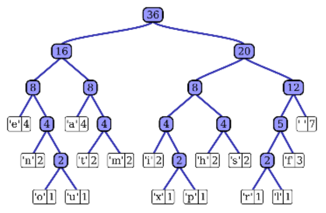
P1: HUFFMAN



Revised: 2022-01-15

BACKGROUND

A Huffman code is a type of prefix code that is used for lossless data compression. The algorithm, developed by David A. Huffman, creates an encoding scheme based on the frequency of symbols so that the most frequently occurring symbols are represented with the fewest number of bits.

ASSIGNMENT

Create a Huffman encoder/decoder class, named **`Huffman`**, based on the algorithm described in chapter 16.3 of Cormen (2009).

You will be given an `edu.metrostate.ics340.p1.TreeNode.java` interface. Do not modify it or move it from its package. Also do not place your code in its package.

Feature	Specification	Information
Constructors	N/A	Make all constructors private. Class will use a factory method for instance creation
Methods	`static Huffman build(String filePath)`	Returns a Huffman coder based on the frequency of characters contained in the specified file. Precondition: filepath must exist and be accessible.
	`String encode(String text)`	Returns an encoded string of '0's and '1's with the Huffman encoding of the given text Precondition: all symbols of the given text must have an encoding from the reference file, otherwise method throws an IllegalArgumenException that displays the unrecognized character.

Copyright © 2022 Ralph A. Foy. All Rights Reserved.

Feature	Specification	Information
	`String decode(String code)`	Returns the decoded text of the given code. Precondition: all codes must be recognized by the decoder otherwise method throws an IllegalArgumentException that displays the unrecognized code.
	`Map <Character, String> getEncodingMap()`	Returns a map of symbols and their Huffman code
	`TreeNode<Character> getDecodingTree()`	Returns the root of the decoding tree for this Huffman coder. Note: the `TreeNode::getValue` method should return the Character value of the node
Input	`String filepath`	An absolute path to a file of text characters which will include punctuation and non-printing characters (e.g. spaces, newlines). While the file may have characters of mixed case, the encoder will be **case-insensitive**. Note: you cannot assume that the program will be run on the same operating system that you develop on.
Output		None, besides method return values

Copyright © 2022 Ralph A. Foy. All Rights Reserved.

PROJECT REQUIREMENTS

	Requirements
Submission	Your submission shall be an exported Eclipse project zip file • **ZIP Filename**: P1_*AAnnnn*.zip • **Project type**: Eclipse Maven Project • **Project Name**: P1_*AAnnnn*_Huffman where • *AAnnnn* is your "student identifier" where ○ *AA:*your initials ○ *nnnn:* the 4 digits embedded in your StarID The zip file must contain your Java sources and any data files for your test cases Your classes must be in a package named with the following prefix: `edu.metrostate.ics340.p1.`*`aannnn`* where *aannnn* is your student identifier as described above
Code	The Constructors and methods defined in the capabilities table will be public, spelled as specified, and with the return type as specified (void otherwise) Your submissions will be tested with an automated testing framework and therefore must adhere to the specification. Also ensure the Eclipse project is running at code Java code level 14. Your submission **must** be free of compile-time errors. Your code must comply with Java coding conventions. You are encouraged to develop any helper methods or classes as you deem fit. It is generally advisable to make them non-public—only specified methods should be public.

Copyright © 2022 Ralph A. Foy. All Rights Reserved.

	Requirements
	You must also provide a JUnit test class, separate from the specified class, showing your test cases. Your tests should demonstrate how you ensure your work meets the specification. The tests should validate preconditions and boundary cases.

Copyright © 2022 Ralph A. Foy. All Rights Reserved.