Assignment 2 Simon Razniewski

Hamming-Codes

This programming assignment gives you the opportunity to implement a Hamming encoder and decoder.

Instructions: You are allowed to work alone or in teams of two students. Submit a single file HammingCode.java which contains (i) an explanation of your solution to the problem as comment, (ii) the Java code.

Problem Statement: Implement two static functions void *encode(String message, String filename)* and *String decode(String filename)* as follows:

The function *encode* takes as parameters a text and a filename. It breaks each character in the text into two 4-bit parts and encodes each 4-bit part using the (7,4)-Hamming code *discussed in the lecture*[2]. The encoded text is then stored under the specified filename.

The function *decode* analogously decodes files encoded in (7,4)-Hamming code.

For the encoded version, you may choose binary representation (more space efficient), or writing 0's and 1's as characters (probably easier for debugging).

Your program should be able to decode messages containing small errors.

Possibly useful functions:

- Integer.toBinaryString(String.charAt(i))
- Character.toChars(int i)

Submission: Friday, 20th of March 2015, 10:00 via the Moodle, or if it does not work, by email.

[2] The Hamming code in the lecture uses first four data bits (d_1 .. d_4), then three parity bits (p_1 = d_1 xor d_2 xor d_3, p_2 = d_2 xor d_3 xor d_4, p_3 = d_3 xor d_4 xor d_1) . In the literature, the parity bits are commonly mixed with the data bits