

## Hamming-Codes

This programming assignment gives you the opportunity to implement a Hamming encoder and decoder. It counts 5% towards the final grade.

**Instructions:** You are allowed to work alone or in teams of two students. Submit an archive containing two files: (i) a file `HammingCode.java` which contains the Java code, and (ii) a file `Explanation.pdf` which contains a short description of your solution (1 page is sufficient).

**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 `String` and a filename. It breaks each character in the `String` into two 4-bit parts and encodes each 4-bit part using the (7,4)-Hamming code *discussed in the lecture*<sup>2</sup>. The encoded `String` 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 to store data in binary representation (more space efficient), or writing 0's and 1's as characters (probably easier for debugging).

Obviously, your program should be able to decode messages containing 1-bit errors.

Possibly useful functions:

- `Integer.toBinaryString(String.charAt(i))` – Translates a char into a binary string
- `Character.toChars(int i)` – Translates an integer into a char
- `Integer.parseInt(s, 2)` – Parses a binary `String` into an integer

**Submission:** Friday, 25<sup>th</sup> of March 2016, 10:00 via Moodle.

<sup>2</sup> The Hamming code in the lecture uses first four data bits ( $d_1 .. d_4$ ), then three parity bits ( $p_1 = d_1 \text{ xor } d_2 \text{ xor } d_3$ ,  $p_2 = d_2 \text{ xor } d_3 \text{ xor } d_4$ ,  $p_3 = d_3 \text{ xor } d_4 \text{ xor } d_1$ ). In the literature, parity bits and data bits are often mixed