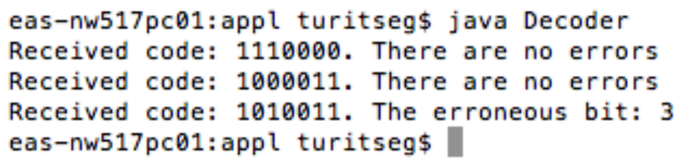
Java Programming Coursework

To be submitted by 23:59 on Tuesday, 25 November 2014

Aim of this coursework is to demonstrate all your knowledge gained in this module.

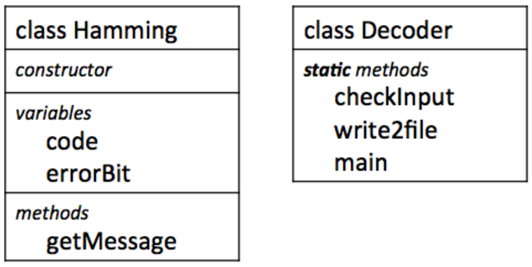
You are expected to implement 2 Java classes to perform decoding of codes that were encoded using Hamming (7,4) coding. One program (Decoder class) should open a file "Codes.txt" with the stored data of 7-bits binary codes on each line, (The file is provided). You have to read the codes from the file (line by line), verify that each of them is 7 bits long; that it contains only '1's and '0's. Then the other program (Hamming class) checks if an error occurred during transmission, corrects it (if there is any), displays the received code and the result of the check on screen. For example, like this:



```
eas-nw517pc01:appl turitseg$ java Decoder
Received code: 1110000. There are no errors
Received code: 1000011. There are no errors
Received code: 1010011. The erroneous bit: 3
eas-nw517pc01:appl turitseg$
```

After that all the results of the decoding should be written into another file "Codes_decoded.txt" line by line. (Here we assume that only 1 error occurred during transmission.)

You have to implement 2 classes – **Hamming** and **Decoder** with the following structure:



Here are the detailed instructions for implementation of the classes and the marking scheme.

<u>Class Hamming.</u>

1. In class **Hamming** declare a regular array **code** to hold 7 integer values and an integer variable **errorBit** for storing the value of an erroneous bit. Declare all variables **private**. **[2]**
2. Implement a *constructor*, which accepts a String variable as a parameter. Inside the constructor:
    a. Convert that string into an array of integers and store the obtained values in array **code**. **[3]**
    b. After that you have to perform 3 parity checks to find the erroneous bit. The obtained value should be stored in variable **errorBit**. If there are no errors the **errorBit** value remains 0. **[5]**
    c. Display the received String input and value of the erroneous bit. If there are no errors, display "There are no errors" message (as above) **[2]**
3. Implement a method **getMessage()** of type String. In this method:
    a. Check if the value of **errorBit** is 0.

b. Otherwise, correct the corresponding digit in the array **code.** **[5]**
c. Assign values to a String variable **message** using the corresponding values of array **code** and **return** it. The returned String value will be used to write into the file. **[5]**

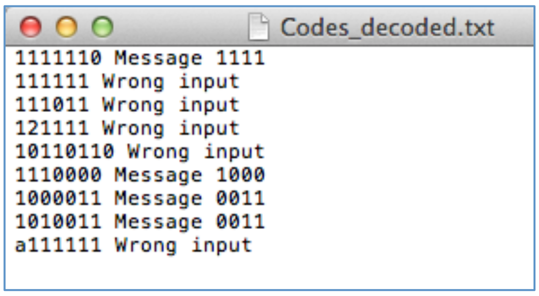
<u>Class Decoder</u>

1. In the **main** method you have to open a provided file "Codes.txt" and read the lines (received codes). Each code should be send as an argument to the method **checkInput** for verification. **[3]**
   a. If the returned value is **true**, then the received code is appropriate for decoding. In this case an object of class **Hamming** is created using the class constructor.
   b. After that the method **getMessage** is applied to the object; **[5]**
   c. The received message should be sent to the method **write2file** together with the original code. **[2]**
   d. If the returned value of the method call **checkInput** is **false**, the received code and the text "Wrong input" should be sent to the method **write2file**. **[3]**

2. A **static boolean** method **checkInput** accepts a String parameter **input**. It performs two checks:
   1) If the string **input** is 7 characters long, and
   2) It contains only '1's and '0's.

   If both of the checks pass, it returns **true**; and if any of the checks fails, it returns **false**. **[10]**

3. A static method **write2file** accepts 2 String parameters – a received code and a decoded message. It appends both of them to the file "Codes_decoded.txt". (Create a file if it doesn't exists). The resulting file should look like this:



```
1111110 Message 1111
111111 Wrong input
111011 Wrong input
121111 Wrong input
10110110 Wrong input
1110000 Message 1000
1000011 Message 0011
1010011 Message 0011
a111111 Wrong input
```

[5]

Make sure your programs contain lots of comments, are well designed and are easy to follow.

If your submitted program fails to compile, you get 5 marks off immediately.

Submit your codes Hamming.java and Decoder.java electronically via Blackboard.

If you struggle to complete everything, complete some parts of the coursework. Be ready to answer any questions I may ask regarding your codes. Incorrect answers will cause deductions of marks.

Good luck!

Appendix.

Some commands and Java functions you might use:

Class `String`, method `charAt(index)` – returns a character of a given string at the particular index.

Method `Character.getNumericValue(char)` accepts a char variable and converts it into an integer.

For modulo 2 addition one can use a `%2` operation.

Hamming decoding

The easiest way to detect a single error and find the erroneous bit in a received 7-bits binary code is to perform 3 parity checks as described by the Venn diagram:

Check 1 involves bits in the positions 1, 3, 5, 7. If it fails, the errorBit value increases by 1.

Check 2 involves bits 2, 3, 6, 7. If it fails, the errorBit value increases by 2.

Check 3 involves the bits 4, 5, 6, 7. If it fails, the errorBit value increases by 4.

As soon as you have found in which of the bits an error occurred, you can correct it.

Remember that the original 4 bits of transmitted information (the ones you have to recover) are stored in the positions 3, 5, 6, 7 of the encoded message.