Binary Code Translator

CSE 174 – Fall 2021

Lab9: 20 points – Due Saturday, Oct 23, 2021 by 11:59pm

Background:

A **binary code** represents text, computer processor instructions, or any other data using "0" and "1". The binary code assigns a pattern of binary digits, also known as bits, to each character, instruction, etc. For example, a binary string of eight bits can represent any of 256 possible values and can, therefore, represent a wide variety of different items (binary code Wiki).

The binary coding system is a collection of systems, which gives a fixed binary number to all letters of the alphabet and symbols. For example:
01000001 is a 8 bit binary code that represents number 65 which is the ascii code of character 'A'. In other words we can say "01000001" represents the character 'A'.
Looking at the ASCII table you can see that each character/symbol has a corresponding ascii code. Also, each ascii code can be written as a binary code. For instance:

Character	ASCII Code	Binary Code
'a'	97	01100001
'b'	98	01100010
'c'	99	01100011
'd'	100	01100100

Computers don't understand words or numbers the way humans do. But it can understand binary electrical signals like 0 and 1 (representing on or off). To make sense of complicated data, your computer has to encode it in binary.

Each binary code can be converted into a ASCII code first, then each ASCII code can be converted into a character. For instance:
If you used the binary code 01001111, you need to square all the numbers from the right to left using the digit 2 ^ n where n is [0-7] (0 for the first digits on the right, 7 for the last digit on the left). Let's make the calculation starting with the correspondence of numbers with the value:

1 = 2 ^ 0
1 = 2 ^ 1
1 = 2 ^ 2
1 = 2 ^ 3
0 = 2 ^ 4
0 = 2 ^ 5
1 = 2 ^ 6
0 = 2 ^ 7

Continue on the Next Page!

Let's make the calculation:
1 = (1 x 2 ^ 0) = 1
1 = (1 x 2 ^ 1) = 2
1 = (1 x 2 ^ 2) = 4
1 = (1 x 2 ^ 3) = 8
0 = Since zeros are off, calculations always equal zero.
0 = zero here too
1 = (1 x 2 ^ 6) = 64
0 = 0
If you add 64, 8, 4, 2 and 1 together, then you would get 79. On the ASCII table, it resembles the letter 'O'.

Part1 - Study the Lab9 Class:

- Download the Lab9.java file.
- Study the class and **do not change anything inside the main method**. You need to understand what is happening inside the given code, so if you have problems understanding any part of the code for any reason, please ask your TA or instructor to explain it to you.
- If you try to compile the code, you will get an error message regarding line 26 saying it can not find the symbol getLetter. The reason for that is the **getLetter** method is a method that you need to write and complete along with another method called **getWord** method.

Part2 - Adding Methods to the Lab9 Class:

Implement the following two methods with the given details. You should not change anything inside the main method.

Return Type	Description	Inputs
char	**getLetter** method gets a binary code as a string, and returns the corresponding character. (you can always assume the given String is holding 8 digit binary code)	String
String	**getWord** method gets binary codes as a string, and returns the representing word. (You can always assume the given string is holding a combination of 8-digit binary codes)	String

Continue on the Next Page!

- **getLetter:**
  You can always use the debugger to follow the value of variables to make sure your code is working as expected.
  Study the Math.pow() method for squaring a value or raising a value to a specific power.
  You can also test your code by calling your method on Interaction Pane as the following:

  Lab9.getLetter("01101101") ---> you should see 'm' as the answer (don't use semicolon at the end)
  Lab9.getLetter("01111010") ----> 'z'
  Lab9.getLetter("01001010") ----> 'J'

- **getWord:**
  You can separate 8 characters at the time and call the getLetter method to figure out a character that represents that binary code, and repeat the same process for the rest of the characters.

  You can also test your code by calling your method on Interaction Pane as the following:

  Lab9.getWord("01001000") ---> you should see "H" as the answer (don't use semicolon at the end)
  Lab9.getWord("0100100001001001") ----> "HI"
  Lab9.getWord("01001010011101010111001101110100") ----> "Just"
  Lab9.getWord("010101000110010101110011011101000110100101101110011001111") ----> "Testing"

Continue on the Next Page!

Part3- Sample Runs:

If you completed and tested the methods with no issues, you can run your code as the following sample runs to make sure your program is generating the correct results.

```
**Binary Code Translator v 1.0**
1. Translate a binary code into a letter
2. Translate binary codes into a word
3. Exit
Enter a number [1-3]: 1
Enter a single binary code: 01100111
The letter is: g

**Binary Code Translator v 1.0**
1. Translate a binary code into a letter
2. Translate binary codes into a word
3. Exit
Enter a number [1-3]: 1
Enter a single binary code: 01001010
The letter is: J

**Binary Code Translator v 1.0**
1. Translate a binary code into a letter
2. Translate binary codes into a word
3. Exit
Enter a number [1-3]: 2
Enter binary codes: 010000110101001001000101
The word is: CRE

**Binary Code Translator v 1.0**
1. Translate a binary code into a letter
2. Translate binary codes into a word
3. Exit
Enter a number [1-3]: 2
Enter binary codes: 0101010001100101011100110111010001101001011011100110 0111
The word is: Testing

**Binary Code Translator v 1.0**
1. Translate a binary code into a letter
2. Translate binary codes into a word
3. Exit
Enter a number [1-3]: 3
End!
----jGRASP wedge: exit code for process is 0.
----jGRASP: operation complete.
```

Continue on the Next Page!

Submission:

After testing your code thoroughly, when you believe that you have correctly solved the problem, and that your code follows the style guidelines, submit it on canvas. If the autograder rejects your solution, fix it and resubmit it again.

The code plugging only tests your two methods, not the whole program. So if you didn't define your methods exactly as explained in part2, or if you changed the class name from Lab9 to something else, your code won't pass the tests.

Scoring Rubric (Full Rubric on Canvas):

- **Successful Submission via CODE (8 points):** You will receive full credit for this if your code passes all the tests. Otherwise you will get zero or partial credit.
- **Hidden Test (4 points):** You will receive full credit for this if your code passes all the additional tests. Otherwise you will get zero or partial credit.
- **Correct Style (4 points):** You will receive full credit if your code has 0 style errors. Also your code has comments at the top at least with your name and description, and some comments inside the code to explain the code.
- **Comments for the methods (2 points):** You will receive full credit if you have comments before each method explaining the purpose of the method and parameter variables and the return value using @param and @return clause.
- **Proper Programming Practices (2 Points):** If your program is written applying all of the programming practices **covered in class:**
    - Variable naming
    - Proper logic
    - Using proper loops and conditions