# OOP: Data Structures and Algorithms Assignment 1: Alphabet Speed Test

*Problem Statement with Analysis and Design Notes:*

- The aim of this assignment is to develop a program that will essentially run a speed test on the users input of typing the alphabet forwards or backwards depending on the input of the user. The program should assist the user in how to run the program successfully, indicate any errors during the test and how they went wrong while also prompting the user to continue if correct. The user must decide whether they want to type the alphabet forwards or backwards, then the timer will begin and end only when the sequence is correct.
- Within the program I must use scanners to allow the user to input into the terminal using system.in. For the timer I will use the Instant.now method to get the start and end time then minus start from end to get the elapsed time. We could also use the localTime method.
- We have to create an array that will store the values of the alphabet so we can increment/decrement through it.
- We need to create a function that will initialise the alphabet to forwards or backwards (initiate alphabet function) then a function that will read the user inputs and carry out the sequence in accordance to this (input function).
- The input should be in order and should only be one character in length each time
- We can use the toLowerCase() function to ensure all letters are taken the same.
- We can use the ASCII table to initiate the alphabet whether that be forwards (‘a’) or backwards (‘z’). We use the ASCII table to generate an alphabet by creating an array of characters and then assigning them the corresponding ASCII values.
- 'a' is ASCII value 97, so we start at 97 and add the value of i to get the next letter of the alphabet.

```
Code
import java.time.Instant;
import java.util.Scanner;

public class SpeedTest {
    public static void main(String [] args) {
        System.out.print("Type the alphabet in order (hit enter between
letters)\n");
        System.out.print("Forwards or Backwards (f/b)?");

        /*
         * we use the ASCII table to generate an alphabet by creating an
array of characters and
         * then assigning them the corresponding ASCII values.
         * First we initialise array of chars*/
        char alphabet[] = generateAlphabet();
        char in;

        System.out.printf("Type: %s\n", alphabet[0]);

        // time calculator using Instant.now().toEpochMilli() to get the
current time in milliseconds
```

```
        long startTime = Instant.now().toEpochMilli();

        /* we use this for loop to iterate through the array until we
complete it. The program will use orint statements
        to guide the user through the tasks i.e printing the next letter.
        if the user scans the wrong letter i will decrement back to the
letter ths user should have input until correct
        * */
        for (int i = 0; i < 26; i++) {
            in = getInput();

            if (in == alphabet[i] && i != 25) {
                System.out.printf("%s: Correct! Now Type: %s\n",
alphabet[i], alphabet[i + 1]);
            } else if (in == alphabet[i] && i == 25) {
                System.out.print("Correct! Well Done!\n");
            } else i--; // decrements back to the required letter after
incorrect input
        }

        // print out the time in seconds
        long endTime = Instant.now().toEpochMilli();
        double seconds = (endTime - startTime) / 1000.0;
        System.out.println("Time taken: " + seconds + "seconds");
    }

    private static char getInput() {
        // Scanner is used to get the input from the user
        Scanner scan = new Scanner(System.in); // system.in points to the
terminal
        String input = scan.nextLine();

        // Character input must be one character otherwise print error
        if (input.length() == 1) {
            return input.toLowerCase().charAt(0);
        }
        else {
            System.out.print("Error, input a single character\n");
            return getInput();
        }
    }

    private static char[] generateAlphabet() {
        char array[] = new char[26], in;
        int i = 0, y = 0;

        // f or b method
        in = getInput();

        /* Checking if input is f or b , then our array generates based on
the user input, we use the ASCII table to create
         * and initialise the alphabet array (i.e 97 == A). The array will
either begin with A or Z.
         * We call the function if the user doesn't input either f or b
until they do */
        if(in == 'f') {
            for(i = 97; i < 123; i++) { // 'a' is ASCII value 97, so we
start at 97 and add the value of i to get the next letter of the alphabet
                array[i - 97] = (char)i;
            }
        }
```

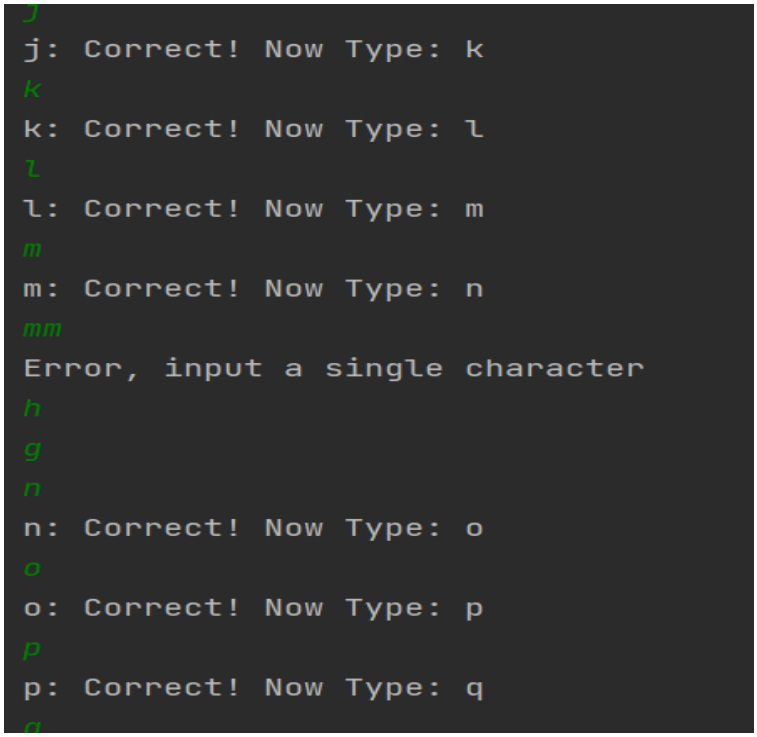
```
        else if(in == 'b') {
            for(i = 122; i >= 97; i--) {
                array[y] = (char)i;
                y++;
            }
        }
        else {
            return generateAlphabet(); // if there is an invalid input
        }
        return array;
    }
}
```

*Testing:*

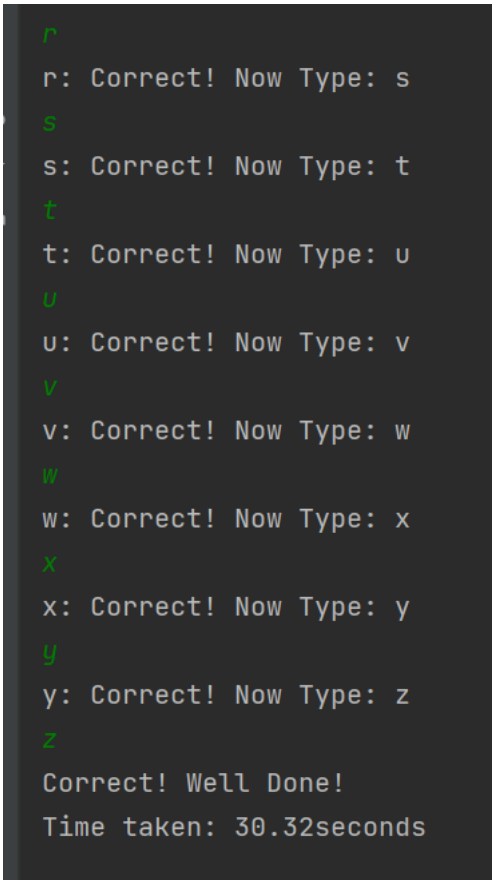
1. *This is the begging of our code where we show the instructions and ask for the user inputs. We have the full functionality from the test.*

```
Type the alphabet in order (hit enter between letters)
Forwards or Backwards (f/b)?f
Type: a
a
a: Correct! Now Type: b
b
b: Correct! Now Type: c
c
c: Correct! Now Type: d
d
d: Correct! Now Type: e
e
e: Correct! Now Type: f
f
f: Correct! Now Type: g
g
g: Correct! Now Type: h
h
h: Correct! Now Type: i
i
i: Correct! Now Type: j
j
```

- Here we test the single letter case and indicate what happens when typing the incorrect letter.
- As we see, we cant input more than 1 letter at a time and we cant move on till the correct letter is placed in the terminal.




- Finally we see our terminal once the program has been complete where the user is told the program has ended and they receive their time taken to complete the program.




2. We will test the backwards functionality of the test.

```
Type the alphabet in order
Forwards or Backwards (f/b)
Type: z
z
z: Correct! Now Type: y
y
y: Correct! Now Type: x
x
x: Correct! Now Type: w
w
w: Correct! Now Type: v
v
v: Correct! Now Type: u
u
u: Correct! Now Type: t
t
t: Correct! Now Type: s
s
s: Correct! Now Type: r
r
r: Correct! Now Type: q
q
```

```
i
i: Correct! Now Type: h
h
h: Correct! Now Type: g
g
g: Correct! Now Type: f
f
f: Correct! Now Type: e
e
e: Correct! Now Type: d
d
d: Correct! Now Type: c
c
c: Correct! Now Type: b
b
b: Correct! Now Type: a
a
Correct! Well Done!
Time taken: 31.161seconds
```