JAVA SCIENCE – UNIT 2: RECURSION - LANZ SIDNEY POVEY

A Java Purist

Professor Purist

Java Science 40S

May 23, 1995

Python, PHP, Perl, PL/C, Pascal, Pipelines! All the coding languages nowadays start with P, what we need is a little bit more of J! No, not Jython, Java and Javascript! All these dang langauges do is piss all over my work! Like seriously, don't do that it's disgusting! I mean, I go online, get some code, and it's NOT in Java! Like what??? There's only 1 coding language, JAVA, the rest of them just where cloaks of urine!

There's only 1 thing I hate in Java; JOptionPanes. The look, the smell… the TASTE. It's… repulsive truly. But everyone uses it! It's a sad sight, but you, I see hope in you. Come, will you aid a poor man like me?

The Law

1. Create a program that gathers user inputs without JOptionPanes.
2. Display said inputs without JOptionPanes.
3. Don't even mention its name, it invokes personal trauma.
4. If any input features one of the following words (Python, PHP, Perl, PL/C, Pascal, Pipelines and Not Java) into the command line arguments then tell the user via the system output, and I quote "your code reeks of urine". 10 times as well, so they don't forget. (Don't forget to account for the lowercase).
5. Error check to catch whether the user inputted something or not on the command prompt.

6. In case the user doesn't know what the aforementioned things are (and I'm being generous here), import a Scanner to get a System input, the opposite of a System output. (Don't worry, there's no word vendetta here).
7. Use Java's traditional (default) package.
8. There should only be 1 class, in 1 package. Anything else is social hierarchy.
9. Every method must be protected, no exceptions. Just so you won't feel the peer pressure to use more than 1 class/package.
10. The code must be organized via Java's traditional bracket formatting.
11. All single lined comments are banned. It just reeks of C#.
12. System outputs cannot be put into wrapper methods, it makes the code look to simple! If I wanted my code to look simple I'd use Scratch!
13. All system outputs must create a line break, because JAVA invented the console menu.
14. All line breaks must be labelled as a final static String under the name JAVA_CAME_FIRST because the whole world must know that JAVA was the first coding language.
15. Single if, else if, else and for statements must have 2 brackets.
16. No colors. You only need 1 color.
17. System output the entire English alphabet
18. System output the entire English alphabet, but backwards
19. Find how much letters are between 2 letters
20. Find how much ASCII letters are between 2 ASCII letters

Reward: A sense of pride in yourself.

```java
/** all single line comments are now multi line comments */
/** no package import, since it's the <default> package */

/** Traditional Input Parameters */
import java.util.List;
import java.util.ArrayList;
import java.util.Arrays;

/** More Modern Input Parameters */
import java.util.Scanner;
public class Encrypt {
    /** Variable initialization */
    static char letterStart;
    static char letterEnd;
    static int  count;
    static String JAVA_CAME_FIRST = "\n";
    static String HEADER = "==========";
    static String HEADER_X2_1 = "»»»»»»»»»»»»»»»≻≻≻≻≻≻≻≿≿≿";
    static String HEADER_X2_2 = "≾≾≾≺≺≺≺≺≺≺«««««««««««««««";

    public static void main(String[] args) {
        /** Words that reek of urine */
        String word1 = "python";
        String word2 = "Python";
        String word3 = "php";
        String word4 = "PHP";
        String word5 = "perl";
        String word6 = "Perl";
        String word7 = "pl/c";
        String word8 = "PL/C";
        String word9 = "pascal";
        String word10 = "Pascal";
        String word11 = "pipelines";
        String word12 = "Pipelines";
        String word13 = "not java";
        String word14 = "Not Java";

        boolean flag = false;
        int index = 0;

        /** array index finder test
        String[] yourArray = {"java", "also java", "c#"};
        for (int i = 0; i <= yourArray.length - 1; i++) {
            if (textInput.contains(yourArray[i])) {
                flag = true;
                index = i;
            }
        }
        if (flag) {
           System.out.println("found at index " + index);
```

```
{
else {
   System.out.println("not found ");
{
*/

/** Checks if the user actually inputed something or not */

/**   UNCOMMENT OUT IF YOU'RE USING COMMAND LINE ARGUMENTS
if(args==null || args.length<1) {
            System.out.println("Please enter some input in Input Arguments"
            + JAVA_CAME_FIRST + "Eg : one two three");
            return;
        }
*/ /**UNCOMMENT OUT IF YOU'RE USING COMMAND LINE ARGUMENTS */

        List<String> lst = new ArrayList<String>();
        lst.addAll(Arrays.asList(args));
        System.out.println("Input :" + lst);

        String[] array = lst.toArray(new String[0]);
        /** converts list<String> to String[]array */

        /** Checks whether or not the code reeks of urine */
        for (int i = 0; i <= array.length - 1; i++) {
    if (word1.contains(array[i]) || word2.contains(array[i])
     || word3.contains(array[i]) || word4.contains(array[i])
     || word5.contains(array[i]) || word6.contains(array[i])
     || word7.contains(array[i]) || word8.contains(array[i])
     || word9.contains(array[i])  || word10.contains(array[i])
     || word11.contains(array[i]) || word12.contains(array[i])
     || word12.contains(array[i]) || word13.contains(array[i])
     || word14.contains(array[i])) {
        flag = true;
        index = i;
    }
}
if (flag) {
   System.out.println("Found at index # " + index);
   System.out.println("your code reeks of urine" + JAVA_CAME_FIRST);
}

System.out.println("Who is starting the program?");
Scanner scanner = new Scanner(System.in);
String line = scanner.nextLine();
System.out.println(HEADER_X2_1 + " Welcome to the Encrypto-matic "
        + line + "! " + HEADER_X2_2);

char[] alphabet = new char[]{'a','b','c','d','e','f','g','h','i','j','k','l'
                             + 'm','n','o','p','q','r','s','t','u','v','w','x'
```

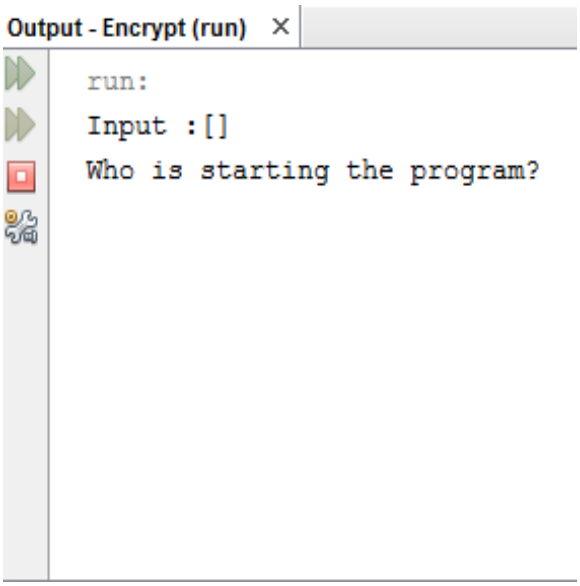
```
                + 'y','z'};
        char[] reverseAlphabet = new char[]{'z','y','x','w','v','u','t','s','r','q','p','o'
                + 'n','m','l','k','j','i','h','g','f','e','d','c'
                + 'b','a'};

        /** ==========     In Between Methods    ========== */
        reset();
        System.out.println(HEADER + " Between Methods " + HEADER);
        reset();
        characterMethod('e','l');
        reset();
        ASCIIMethod('e','l');

        /** ========== regularAlphabet (4 Loops) ========== */
        reset();
        System.out.println(HEADER + " Regular Alphabet " + HEADER);
        reset();
        alphabet(alphabet);
        doubleReset();
        alphabet4();
        doubleReset();
        alphabetR('a');

        /** ========== reverseAlphabet (4 Loops) ========== */
        doubleReset();
        System.out.println(HEADER + " Reverse Alphabet " + HEADER);
        reset();
        rAlphabet(reverseAlphabet);
        reset();
        rAlphabet4(alphabet);
        doubleReset();
        rAlphabetR(alphabet);
        doubleReset();

        System.out.println(HEADER_X2_1 + " Thank you for using the Encrypto-matic "
                + line + "! " + HEADER_X2_2);
    }

    /**
     * The intro and ending wrapper prompt for the between(char a, char b) method.
     * @param letterStart the starting letter (minimum range)
     * @param letterEnd the ending letter (maximum range)
     */
    protected static void characterMethod(char letterStart, char letterEnd) {
        System.out.println("-> Character Method <-");
        System.out.println("Choose which letter to start from...");
        letterStart = 'e';
        System.out.println(">Starting letter = "  + letterStart);
        System.out.println("Choose which letter to end from...");
        letterEnd = 'l';
```

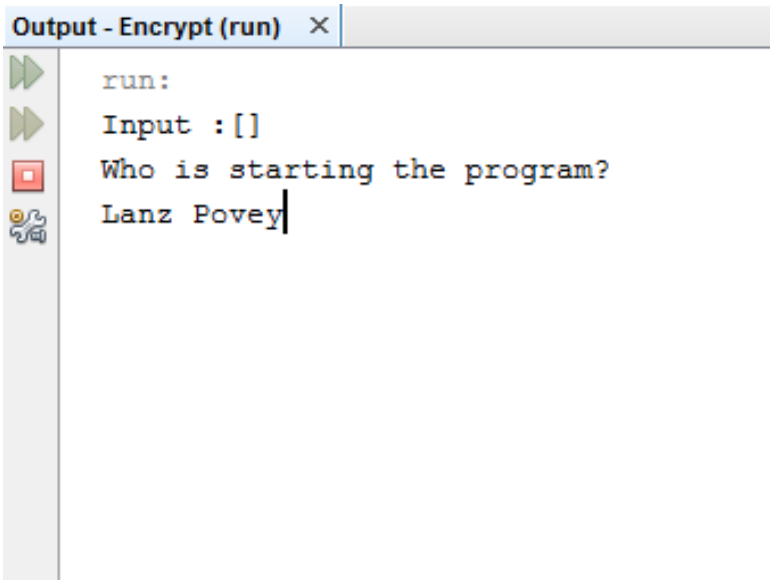
```
    letterEnd   = 'I';
    System.out.println(">Ending letter   = "  + letterEnd);
    between(letterStart,letterEnd);
    System.out.println("There are " + count   + " letters in between of "
                       + letterStart  + " and " + letterEnd);
}

/**
 * The intro and ending wrapper prompt for the betweenASCII(char a, char b) method.
 * @param letterStart the starting letter (minimum range)
 * @param letterEnd the ending letter (maximum range)
 */
protected static void ASCIIMethod(char letterStart, char letterEnd) {
    System.out.println("-> ASCII Method <-");
    System.out.println(">Starting letter = " + letterStart);
    System.out.println(">Ending letter   = " + letterEnd);
    betweenASCII(letterStart,letterEnd);
    System.out.println("There are " + count   + " letters in between of "
                       + letterStart  + " and " + letterEnd);
}

/**
 * Displays the characters that are between char a and char b.
 * (Uses count to keep track of how many recursions take place)
 * (A.K.A: The number of characters between char a and char b)
 * @param a the starting letter (minimum range)
 * @param b the ending letter (maximum range)
 * same as characterMethod parameters
*/
protected static void between(char a, char b) {
    count = count + 1; // count++;
    System.out.println(a);
    if (a < b) {
        between((char) (a+1), b);
    } /** recursive case */
    // base case
}

/**
 * Displays the characters that are between char a and char b
 * (based on their # position on the ASCII table)
 * (Uses count to keep track of how many recursions take place)
 * (A.K.A: The number of characters between char a and char b)
 * @param a the starting letter (minimum range)
 * @param b the ending letter (maximum range)
 * same as ASCIIMethod parameters
*/
protected static void betweenASCII(char a, char b) {
    count = count + 1; /** count++; */
    if    (a==b) {
```

```
            System.out.println(b+1);
        }
        else {
            System.out.println(a+1);
            betweenASCII((char)(a+1), b);
        }
        /** base case */
    }

    /** ==================== regularAlphabet ====================

    /**
     * Display the entire English alphabet from an array to a String via the
     * system output. (Also used for verification measures, since I don't want
     * to sing the alphabet over and over again in my head while
     * working on this project...
     * @param alphabet the entire English alphabet (26 letters)
     */
    protected static void alphabet(char[] alphabet)
    {
        System.out.println("-> Regular Alphabet (Array.toString) <-");
        System.out.print(Arrays.toString(alphabet));
    }

    /**
     * Travels through the ASCII chart (int) and displays said character codes
     * as actual chars via the system output. It's why this method doesn't
     * need a char[] alphabet parameter (or any at all).
     */
    protected static void alphabet4() {
        System.out.println("-> Regular Alphabet (For Loop) <-");
        char seed = 'a';
        char[] letters = new char[26];
        for (char i = 0; i < 26 ; i++) {
            letters[i] = (char) (seed + i);
            System.out.print(letters[i]);
        }
    }

    /**
     * A wrapper method for the true recursive alphabet. The title and the
     * variables are created and initiated here as to not have those trapped
     * in the recursion.
     * @param current the current letter of the English alphabet that the
     *                recursive method is on
     */
    protected static void alphabetR(char current)
    {
        System.out.println("-> Regular Alphabet (Recursion) <-");
        char seed = 'a';
```

```
    char[] letters = new char[26];
    //count = alphabet.length + 1;
    System.out.print('a');
    alphabetR1('a');
}

/**
 * The true recursive alphabet method. Similar to the alphabet for loop method
 * it travels through the ASCII chart (int) and displays the codes as casted
 * char values.
 * @param current the current letter that the recursion is on
 */
protected static void alphabetR1(char current)
{

    if (current != 'z') {
        System.out.print((char) (current + 1));
        alphabetR1((char) (current + 1));
    }
}

/** ==================== reverseAlphabet ====================
 /**
 * Display the entire English alphabet from an array to a String via the
 * system output, except it's backwards.
 * (Also used for verification measures, since... seriously there's no song
 * for the reversed English alphabet)
 * @param reverseAlphabet the entire English alphabet (26 letters), but
 *                        reversed
 */
protected static void rAlphabet(char[] reverseAlphabet)
{
    System.out.print("-> Reverse Alphabet (Array.toString) <-");
    reset();
    System.out.println(Arrays.toString(reverseAlphabet));
}

/**
 * Unlike the alphabet for loop method, this isn't simply that but reversed.
 * Rather, it uses a whole new method altogether, where it travels the char
 * array of un-reversed alphabet in reverse. Sure, I could have just traveled
 * forward in the reversed alphabet, but I want those style points yo.
 * @param alphabet the entire English alphabet (26 letters)
 */
protected static void rAlphabet4(char alphabet[]) {
    System.out.println("-> Reverse Alphabet (For Loop) <-");
    for (int i = alphabet.length-1; i >= 0 ; i--) {
        System.out.print(alphabet[i]);
    }
}
```

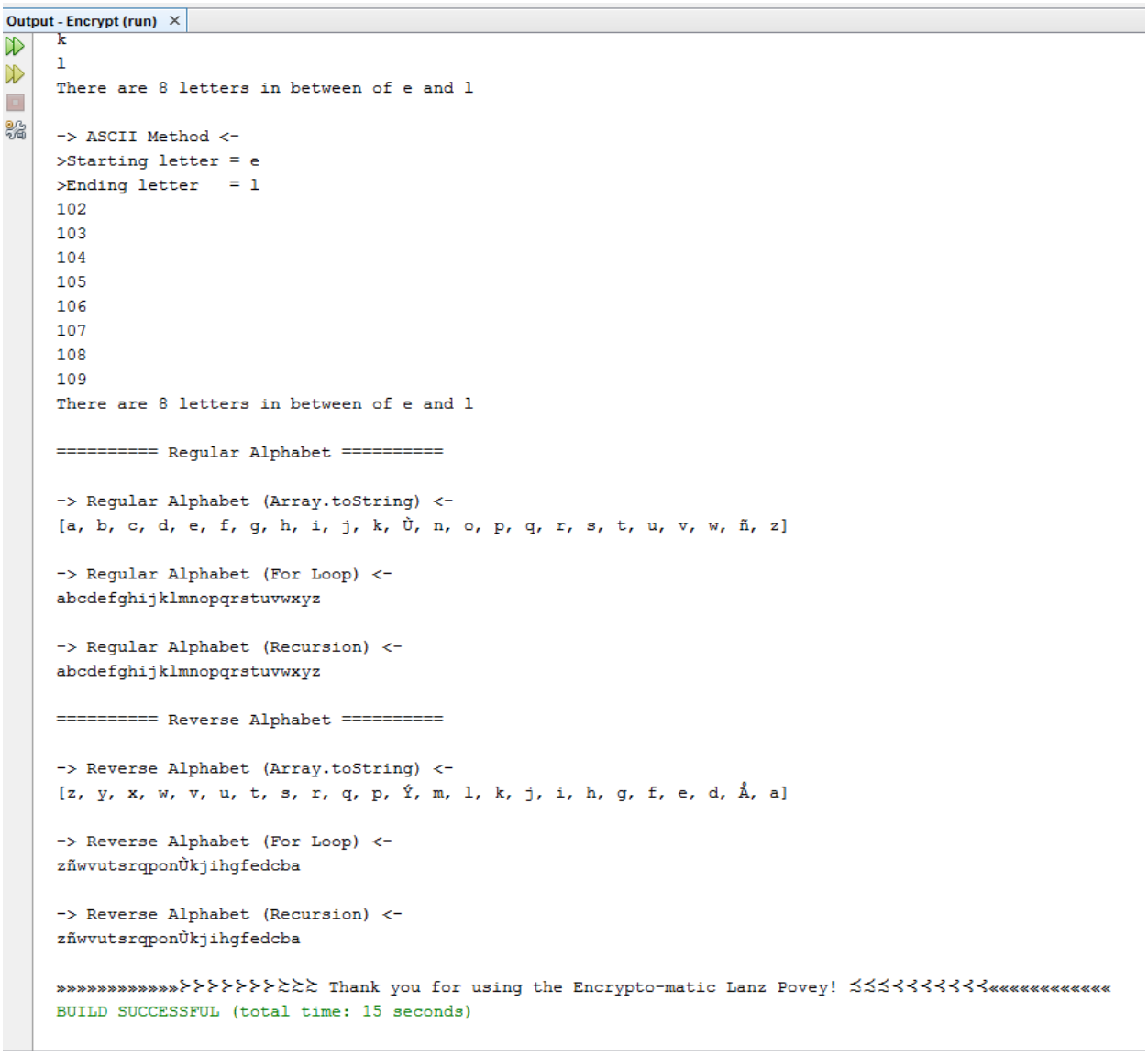
```java
    /**
     * A wrapper method for the true recursive alphabet. The title's created here
     * just so that it isn't recursively printed to the system's output 26 times.
     * @param alphabet the entire English alphabet (26 letters) [not reversed]
     */
    protected static void rAlphabetR(char alphabet[]){
        System.out.println("-> Reverse Alphabet (Recursion) <-");
        rAlphabetR1(alphabet);
    }

    /**
     * A recursive method for traveling through a char array of the entire
     * English alphabet (un-reversed) except in reverse so that it displays said
     * characters in reverse as well.
     * @param alphabet the entire English alphabet (26 letters) [not reversed]
     */
    protected static void rAlphabetR1(char alphabet[]) {
        count = alphabet.length + 1;
        for (int i = alphabet.length-1; i >= 0 ; i--) {
            if (count >=0) {
                System.out.print(alphabet[i]);
            }
            else {
                rAlphabetR1(alphabet);
                System.out.print(alphabet[i]);
            }
        }
    }

    /** ==================== Formatting Options ==================== */
    /**
     * A method that resets the recursion count
     * + adds a line break to the System.output.
     */
    protected static void reset() {
        count = 0;
        System.out.println("");
    }

    /**
     * Literally just the reset() method except it's used two times.
     */
    protected static void doubleReset(){
        for (int i = 0; i <= 1 ; i++) {
            reset();
        }
    }
}
```

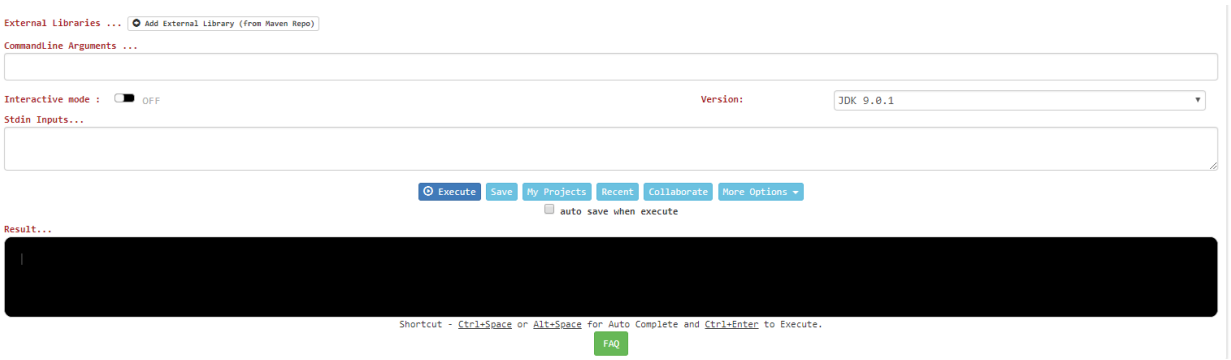

Output - Encrypt (run)
run:
Input :[]
Who is starting the program?


Output - Encrypt (run)
run:
Input :[]
Who is starting the program?
Lanz Povey

Output - Encrypt (run)

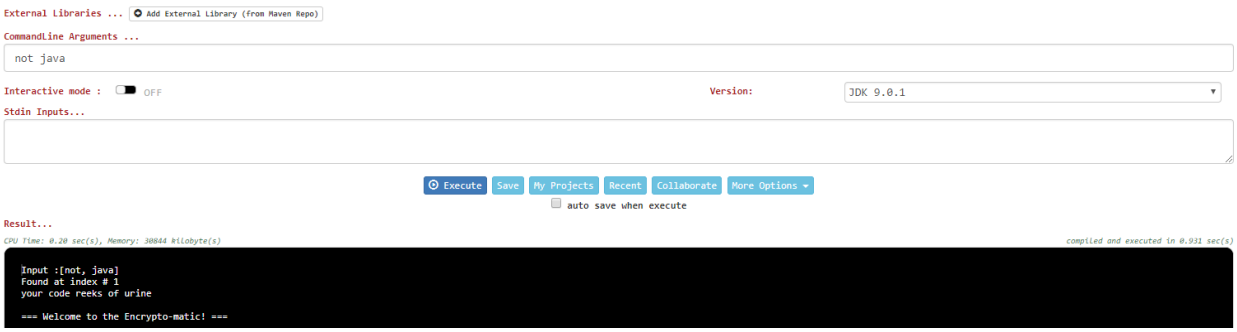
```
run:
Input :[]
Who is starting the program?
Lanz Povey
»»»»»»»»»»»»»≻≻≻≻≻≻≻≿≿≿ Welcome to the Encrypto-matic Lanz Povey! ≾≾≾≺≺≺≺≺≺««««««««««««

========== Between Methods ==========

-> Character Method <-
Choose which letter to start from...
>Starting letter = e
Choose which letter to end from...
>Ending letter   = l
e
f
g
h
i
j
k
l
There are 8 letters in between of e and l

-> ASCII Method <-
>Starting letter = e
>Ending letter   = l
102
103
104
105
106
107
108
109
There are 8 letters in between of e and l

========== Regular Alphabet ==========

-> Regular Alphabet (Array.toString) <-
[a, b, c, d, e, f, g, h, i, j, k, Ù, n, o, p, q, r, s, t, u, v, w, ñ, z]

-> Regular Alphabet (For Loop) <-
```


Output - Encrypt (run)
k
l
There are 8 letters in between of e and l
-> ASCII Method <-
>Starting letter = e
>Ending letter   = l
102
103
104
105
106
107
108
109
There are 8 letters in between of e and l
========== Regular Alphabet ==========
-> Regular Alphabet (Array.toString) <-
[a, b, c, d, e, f, g, h, i, j, k, Ù, n, o, p, q, r, s, t, u, v, w, ñ, z]
-> Regular Alphabet (For Loop) <-
abcdefghijklmnopqrstuvwxyz
-> Regular Alphabet (Recursion) <-
abcdefghijklmnopqrstuvwxyz
========== Reverse Alphabet ==========
-> Reverse Alphabet (Array.toString) <-
[z, y, x, w, v, u, t, s, r, q, p, Ý, m, l, k, j, i, h, g, f, e, d, Å, a]
-> Reverse Alphabet (For Loop) <-
zñwvutsrqponÙkjihgfedcba
-> Reverse Alphabet (Recursion) <-
zñwvutsrqponÙkjihgfedcba
Thank you for using the Encrypto-matic Lanz Povey!
BUILD SUCCESSFUL (total time: 15 seconds)



When the user inputs nothing.

When the user inputs a String from the list of words that “reek of urine”.



When the user inputs a String that’s not from the list of words that “reek of urine”.