Student: Alberto Asensio

TABA: Software Development

Student Number NCI: x23171782

URLGenerator

Index

IPO Diagram

Input	Process	Output
- Number of URLs - Company names - URLs to validate	- Generate URLs - Validate URLs	- Generated URLs - Validation results

Digits/Options developed (x23171782)

	a)	b)
Question 1	UR2	MCNA1
Question 2	Amazon URLs	-

Code explanation

Setting “protocol”

To change the protocol depending on the word “Meta” I used 2 methods:

- **.contains:** This method is the fastest and easiest way to detect the word “meta” in the companyName
- **.toLowerCase:** This method is widely used to skip upper cases and just focus on the word itself (in this case “meta”, “Meta”, “mEtA”, or whatever).

```
 if (companyName.toLowerCase().contains("meta")) {
            protocol = "coap";
        } else {
            protocol = "ftp";
        }
//Here we use 2 methods: .toLowerCase and .contains
```

Consonants Counter

The consonant counter is made before changing the companyName in the “Hostname Part 1”, so the consonants are counted with the full name.

To check the consonants without changing the original cases of the “comanyName” I decided to create the string “companyNameLowerCase” and use it only in this part and the “VowelPairs Counter”

```
//CONSONANTS COUNTER (before the name of the company changes to the
shortcuts Inc, Ltd or LLC)
        for(int f = 0; f < companyNameLowerCase.length(); f++){ //Here we
use the companyNameLowerCase variable to check the consonants

            // Veryfying the consonants to set the counter
            if (companyNameLowerCase.charAt(f) == 'b' ||
companyNameLowerCase.charAt(f) == 'c' || companyNameLowerCase.charAt(f)
== 'd' || companyNameLowerCase.charAt(f) == 'f' ||
companyNameLowerCase.charAt(f) == 'g' ||
            companyNameLowerCase.charAt(f) == 'h' ||
companyNameLowerCase.charAt(f) == 'j' || companyNameLowerCase.charAt(f)
== 'k' || companyNameLowerCase.charAt(f) == 'l' ||
companyNameLowerCase.charAt(f) == 'm' ||
            companyNameLowerCase.charAt(f) == 'n' ||
companyNameLowerCase.charAt(f) == 'p' || companyNameLowerCase.charAt(f)
```

```
== 'q' || companyNameLowerCase.charAt(f) == 'r' ||
companyNameLowerCase.charAt(f) == 's' ||
            companyNameLowerCase.charAt(f) == 't' ||
companyNameLowerCase.charAt(f) == 'v' || companyNameLowerCase.charAt(f)
== 'w' || companyNameLowerCase.charAt(f) == 'x' ||
companyNameLowerCase.charAt(f) == 'y' ||
            companyNameLowerCase.charAt(f) == 'z') {
            consonantsCount = consonantsCount + 1; //Counter
             }
         }
```

VowelPairs Counter

As the "Consonants Counter", here I used the String "companyNameLowerCase" to check the vowels.

```
//VOWEL COUNTER (before the name of the company changes to the shortcuts
Inc, Ltd or LLC)
        for(int g = 0; g < companyNameLowerCase.length(); g++){
            if(companyNameLowerCase.charAt(g) == 'a' ||
               companyNameLowerCase.charAt(g) == 'e' ||
               companyNameLowerCase.charAt(g) == 'i' ||
               companyNameLowerCase.charAt(g) == 'o' ||
               companyNameLowerCase.charAt(g) == 'u' &&
               companyNameLowerCase.charAt(g + 1) == 'a' ||
               companyNameLowerCase.charAt(g + 1) == 'e' ||
               companyNameLowerCase.charAt(g + 1) == 'i' ||
               companyNameLowerCase.charAt(g + 1) == 'o' ||
               companyNameLowerCase.charAt(g + 1) == 'u'){
            vowelPairsCount = vowelPairsCount + 1;
            }
        }
```

Hostname Part 1: Replacing 3 words

For replacing the words "Incorporated", "Limited" and "Limited Liability Company" I used the methods:

- **.contains:** If the word between the brackets is contained in the String, then the following is executed.
- **.replace:** This method has 2 spaces: in the first one is introduced the word that want to be replaced and in the second the new word.

The variable “companyName” is changed here if some of the 3 words appears.

```
 // HOSTNAME part 1: Replacing 3 words with shortcuts
        if (companyNameLowerCase.contains("incorporated")) {
            companyName = companyName.replace("incorporated", "Inc");
        }
        else if (companyNameLowerCase.contains("limited liability
company")) {
            companyName = companyName.replace("limited liability
company", "LLC");
        }
        else if (companyNameLowerCase.contains("limited")) {
            companyName = companyName.replace("limited", "Ltd");
        }
```

Hostname Part 2: Replacing spaces

Here I used a “for” loop to check and replace character by character.

```
// HOSTNAME part 2: Replacing spaces for "_"
        for (int i = 0; i < companyName.length(); i++) {
            if (companyName.charAt(i) != ' ') {
                hostnameBuffer.append(companyName.charAt(i));
            } else {
                hostnameBuffer.append('_');
            }
        }
```

Hostname Part 3: Setting the “domain” with the consonant counter

Following the order, it’s time to set the domain depending on the consonants and checking if it’s odd or even.

```
  //HOSTNAME Part 3: Using the consonantCounter to find out the domain
(.org or .edu)
        if (consonantsCount % 2 == 1){
            domain = ".org";
        }else{
            domain = ".edu";
        }
```

Hostname Part 4: Setting the "domain" with the consonant counter

Now I set the "path" depending on the number of vowel pairs (done before).

"If" statement is used to check.

```
//HOSTNAME Part 4: Using the vowerPairsCount to define the path
        if (vowelPairsCount == 0){
            path = "bio";
        }
        else if(vowelPairsCount == 1 || vowelPairsCount == 2 ||
vowelPairsCount == 3){
            path = "FAQ";
        }
        else{
            path = "Glossary";
        }
```

Last Part: Appending all together

In this part I take all the results and append them in the final StringBuffer called "finalURLBuffer". Then I convert it into a String named "finalURLString" so the result can be returned with the getter method.

```
 //Appending all parts of the whole URL
        finalURLBuffer.append(protocol);
        finalURLBuffer.append("://");
        finalURLBuffer.append(hostnameString);
        finalURLBuffer.append(domain);
        finalURLBuffer.append("/");
        finalURLBuffer.append(path);

        //Creating the FINAL URL
        finalURLString = finalURLBuffer.toString();

    }
```

QUESTION 2: ComputeValidation

The interesting part of the question 2 falls in the compute method (called here computeValidation()). Here I built a "for" loop to check the different elements of the array and some methods as .contains or .length to check the conditions. Finally if all the

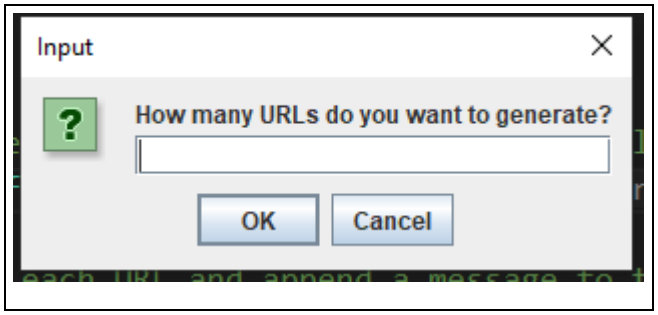
conditions are true or SOME of them are false, the urlBoolean array at that element is checked.

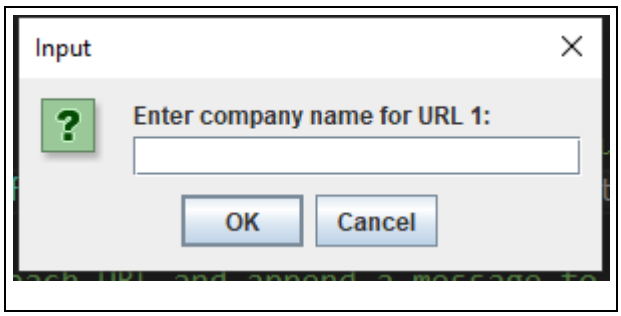
```
public void computeValidation() {
        // Validating each URL in the urlValidation array
        for (int i = 0; i < urlValidation.length; i++) {
            // Convert URL to lowercase for the comparison
            String currentURL = urlValidation[i].toLowerCase();
            // Check conditions for a valid Amazon URL
            boolean containsAmazon = currentURL.contains("amazon");
            boolean containsAWS = currentURL.contains("aws");
            boolean lengthInRange = currentURL.length() >= 5 &&
currentURL.length() <= 16;
            boolean containsValidCharacters = currentURL.matches("[a-z0-
9_/\\.]+");

            // Validate the URL based on conditions and update urlBoolean
array
            urlBoolean[i] = (containsAmazon || containsAWS) &&
lengthInRange && containsValidCharacters;
        }
    }
```

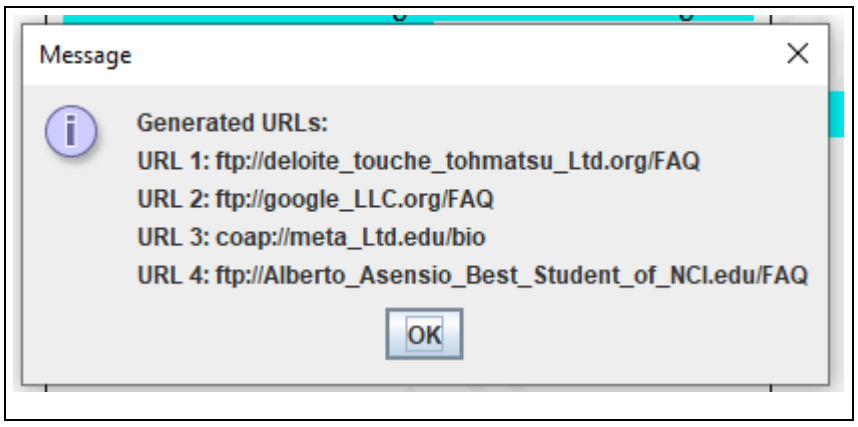
Screenshots

QUESTION 1: Input

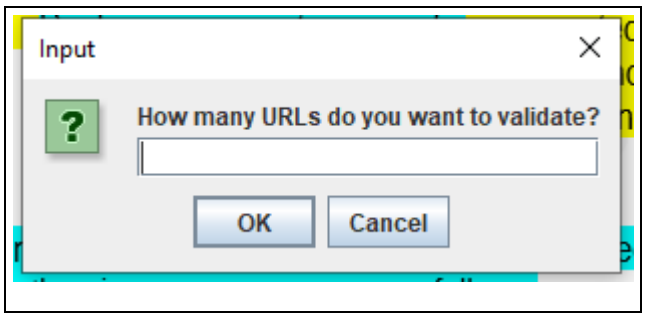


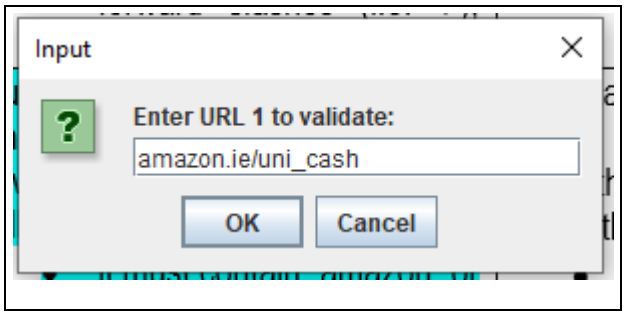


Question 1: Output

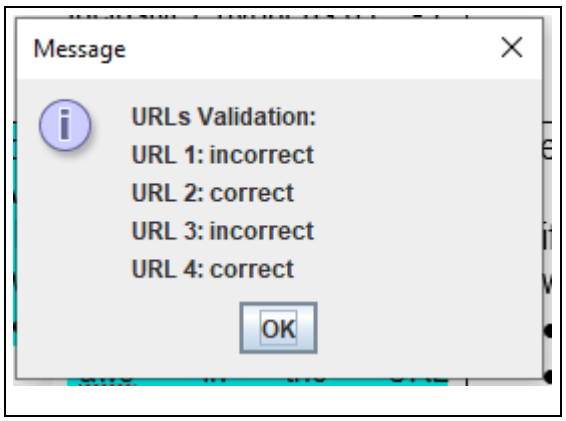


Question 2: Input





Question 2: Output



Appendix: Source Code

URLGeneratorApp

```
/* URLGeneratorApp
 * @author Alberto Asensio
 * 05/01/2024
 * studentID: x23171782
 * Approach ID: MCNA1
 */



import javax.swing.JOptionPane;

public class URLGeneratorApp {
   public static void main(String[] args) {

      //Declare variables
      String companyName;
      int numURLs;

      // Ask the user how many URLs they want to generate
      numURLs = Integer.parseInt(JOptionPane.showInputDialog(null, "How many URLs do you want to generate?"));

      // Create an array to store generated URLs. The capacity of the array will be the input of the user
      String[] generatedURLs = new String[numURLs];




      // Ask for company namepas and generate URLs
```

```
        for (int i = 0; i < numURLs; i++) {
            companyName = JOptionPane.showInputDialog(null, "Enter company name for URL " + (i + 1) + ":");

            //Create the object. This allow to use the methods of the instantiable class
            URLGenerator urlGen = new URLGenerator();

            //Using the setter to pass the information to the instantiable class
            urlGen.setCompanyName(companyName);

            //Using the "compute" method to do the calculations. This is better explained in the instantiable class
            urlGen.compute();

            //Store the generated URL into the array
            generatedURLs[i] = urlGen.getURL();
        }

        // Display generated URLs using JOptionPane
        StringBuilder urlsMessage = new StringBuilder("Generated URLs:\n");
        for (int i = 0; i < numURLs; i++) {
            urlsMessage.append("URL ").append(i + 1).append(": ").append(generatedURLs[i]).append("\n");
        }
        JOptionPane.showMessageDialog(null, urlsMessage.toString());



        /**********************QUESTION 2**********************/

        //Declare new variables
        int urlValidationNums;
```

```
        String urlToValidate;
        String status;

        //INPUT: Ask the user how many URLs they want to validate
        urlValidationNums = Integer.parseInt(JOptionPane.showInputDialog(null, "How many URLs do you want to validate?"));

        //OBJECT: I can create a new object to use separately in this question 2
        URLGenerator urlValidation = new URLGenerator();

        //SETTER the num so the APP will ask that amount of validations to the user
        urlValidation.setURLValidationNums(urlValidationNums);

        //Initialize the Array to store the computed results in this class
        boolean[] urlBooleanResults = new boolean[urlValidationNums];

        // Prompt the user X times for the URLs to validate and store them in the URLGenerator object
        for (int i = 0; i < urlValidationNums; i++) {
            urlToValidate = JOptionPane.showInputDialog(null, "Enter URL " + (i + 1) + " to validate:");
            urlValidation.setURLValidation(i, urlToValidate);
        }

        //COMPUTE the validations
        urlValidation.computeValidation();

        // Store the results of validation
        for (int i = 0; i < urlValidationNums; i++) {
            urlBooleanResults[i] = urlValidation.getURLBoolean()[i];
        }
```

```
        // Create a StringBuffer to store the results
        StringBuffer message = new StringBuffer("URLs Validation:\n");

        // Check each URL and append a message to the StringBuffer
        for (int j = 0; j < urlBooleanResults.length; j++) {
            int urlIndex = j + 1;
            if (urlBooleanResults[j]) {
                status = "correct";
            }else{
                status = "incorrect";
            }
            message.append("URL                                    ").append(urlIndex).append(":
").append(status).append("\n");
        }

        // Show the message by converting the StringBuffer to String using JOptionPane
        JOptionPane.showMessageDialog(null, message.toString());




    }
}
```

URLGenerator
/* URLGenerator * @author Alberto Asensio * 05/01/2024

```
 * studentID: x23171782
 * URL Rules ID: UR2
 */

public class URLGenerator {
    private String companyName;
    private StringBuffer finalURLBuffer;
    private String finalURLString;

    // Other variables used for computation
    private String protocol;
    private String companyNameLowerCase; //To check the number of consonants, vowels and also the Hostname part 1
    private int consonantsCount;
    private int vowelPairsCount;
    private StringBuffer hostnameBuffer;
    private String hostnameString;
    private String domain;
    private String path;




    public URLGenerator() {
        this.companyName = "";
        this.finalURLBuffer = new StringBuffer();
        // Initializing other variables
        this.companyNameLowerCase = "";
        this.protocol = "";
        this.hostnameBuffer = new StringBuffer();
        this.hostnameString = "";
```

```
        this.vowelPairsCount = 0;
        this.consonantsCount = 0;
        this.domain = "";
        this.path = "";
        this.finalURLString = "";
    }


    // Setter method intruduced in URLGeneratorApp
    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }


    public void compute() {

        companyNameLowerCase = companyName.toLowerCase(); //To check the number of consonants, vowels and also the Hostname part 1

        //Setting the PROTOCOL
        if (companyName.toLowerCase().contains("meta")) { //Here we use 2 methods: .toLowerCase and .contains
            protocol = "coap";
        } else {
            protocol = "ftp";
        }

        //CONSONANTS COUNTER (before the name of the company changes to the shortcuts Inc, Ltd or LLC)
        for(int f = 0; f < companyNameLowerCase.length(); f++){ //Here we use the companyNameLowerCase variable to check the consonants
```

```
        // Veryfying the consonants to set the counter
        if (companyNameLowerCase.charAt(f) == 'b' ||
companyNameLowerCase.charAt(f) == 'c' || companyNameLowerCase.charAt(f) ==
'd' || companyNameLowerCase.charAt(f) == 'f' || companyNameLowerCase.charAt(f)
== 'g' ||
        companyNameLowerCase.charAt(f) == 'h' ||
companyNameLowerCase.charAt(f) == 'j' || companyNameLowerCase.charAt(f) == 'k'
|| companyNameLowerCase.charAt(f) == 'l' || companyNameLowerCase.charAt(f) ==
'm' ||
        companyNameLowerCase.charAt(f) == 'n' ||
companyNameLowerCase.charAt(f) == 'p' || companyNameLowerCase.charAt(f) ==
'q' || companyNameLowerCase.charAt(f) == 'r' || companyNameLowerCase.charAt(f)
== 's' ||
        companyNameLowerCase.charAt(f) == 't' ||
companyNameLowerCase.charAt(f) == 'v' || companyNameLowerCase.charAt(f) ==
'w' || companyNameLowerCase.charAt(f) == 'x' ||
companyNameLowerCase.charAt(f) == 'y' ||
        companyNameLowerCase.charAt(f) == 'z') {
        consonantsCount = consonantsCount + 1; //Counter
         }
      }

    //VOWEL COUNTER (before the name of the company changes to the shortcuts
Inc, Ltd or LLC)
    for (int g = 0; g < companyNameLowerCase.length() - 1; g++) {
       if ((companyNameLowerCase.charAt(g) == 'a' ||
           companyNameLowerCase.charAt(g) == 'e' ||
           companyNameLowerCase.charAt(g) == 'i' ||
           companyNameLowerCase.charAt(g) == 'o' ||
           companyNameLowerCase.charAt(g) == 'u') &&
```

```
            (companyNameLowerCase.charAt(g + 1) == 'a' ||
             companyNameLowerCase.charAt(g + 1) == 'e' ||
             companyNameLowerCase.charAt(g + 1) == 'i' ||
             companyNameLowerCase.charAt(g + 1) == 'o' ||
             companyNameLowerCase.charAt(g + 1) == 'u')) {

            vowelPairsCount++;

            // If a pair is found, the g increases automatically 1 more to avoid counting 3 letters together as an extra pair
            //Hard to understand, but it works
            g++;

        }
    }

    // HOSTNAME part 1: Replacing 3 words with shortcuts
    if (companyNameLowerCase.contains("incorporated")) {
        companyName = companyNameLowerCase.replace("incorporated", "Inc");
    }
    else if (companyNameLowerCase.contains("limited liability company")) {
        companyName = companyNameLowerCase.replace("limited liability company", "LLC");
    }
    else if (companyNameLowerCase.contains("limited")) {
        companyName = companyNameLowerCase.replace("limited", "Ltd");
    }

    // HOSTNAME part 2: Replacing spaces for "_"
    for (int i = 0; i < companyName.length(); i++) {
        if (companyName.charAt(i) != ' ') {
```

```
            hostnameBuffer.append(companyName.charAt(i));
        } else {
            hostnameBuffer.append('_');
        }
    }

    //Creating the String for the hostname (so we have a String instead of a StringBuffer)
    hostnameString = hostnameBuffer.toString();

    //HOSTNAME Part 3: Using the consonantCounter to find out the domain (.org or .edu)
    if (consonantsCount % 2 == 1){
        domain = ".org";
    }else{
        domain = ".edu";
    }

    //HOSTNAME Part 4: Using the vowerPairsCount to define the path
    if (vowelPairsCount == 0){
        path = "bio";
    }
    else if(vowelPairsCount == 1 || vowelPairsCount == 2 || vowelPairsCount == 3){
        path = "FAQ";
    }
    else{
        path = "Glossary";
    }

    //Appending all parts of the whole URL
    finalURLBuffer.append(protocol);
```

```
        finalURLBuffer.append("://");
        finalURLBuffer.append(hostnameString);
        finalURLBuffer.append(domain);
        finalURLBuffer.append("/");
        finalURLBuffer.append(path);

        //Creating the FINAL URL
        finalURLString = finalURLBuffer.toString();

    }

    // Getter method to return the generated URL
    public String getURL() {
        return finalURLString;
    }



    /**********************QUESTION
2************************************/

    //Declaring variables QUESTION 2
    int urlValidationNums;
    String[] urlValidation;
    boolean[] urlBoolean;

    //SETTER and INITIALIZE the arrays witht he given number
    public void setURLValidationNums(int urlValidationNums) {
        this.urlValidationNums = urlValidationNums;
        this.urlValidation = new String[urlValidationNums]; // Initialize the array with
the given size
```

```
        this.urlBoolean = new boolean[urlValidationNums]; // Initialize the boolean array with the given size
        }

   //Setting the URLs inserted by the user into the String array
   public void setURLValidation(int index, String urlToValidate) {
      urlValidation[index] = urlToValidate; // Set individual URLs in the array
   }

   public void computeValidation() {
      // Validating each URL in the urlValidation array
      for (int i = 0; i < urlValidation.length; i++) {
         // Convert URL to lowercase for the comparison
         String currentURL = urlValidation[i].toLowerCase();
         // Check conditions for a valid Amazon URL
         boolean containsAmazon = currentURL.contains("amazon");
         boolean containsAWS = currentURL.contains("aws");
         boolean lengthInRange = currentURL.length() >= 5 && currentURL.length() <= 16;
         boolean containsValidCharacters = currentURL.matches("[a-z0-9_/\\.]+");

         // Validate the URL based on conditions and update urlBoolean array
         urlBoolean[i] = (containsAmazon || containsAWS) && lengthInRange && containsValidCharacters;
      }
   }


   public boolean[] getURLBoolean() {
      return urlBoolean;
   }
```

```
}
```