**National College of Ireland**

**Higher Diploma in Computing Software Development, HDSDEV\_SPOL23\_Y2**

Miguel Angel Vinas

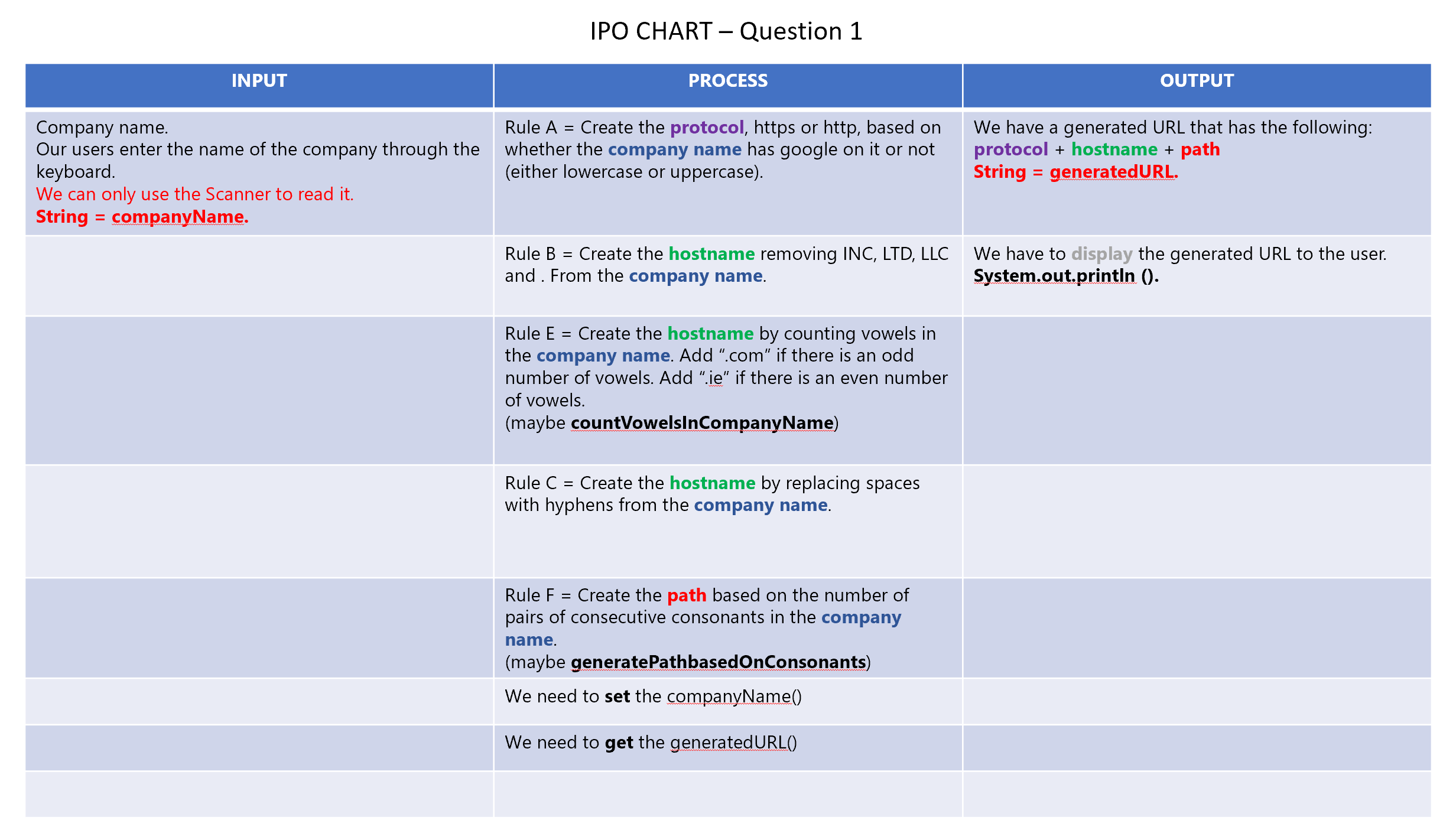
Student Number: [**x22116133@student.ncirl.ie**](mailto:x22116133@student.ncirl.ie)

**TABA Report for Software Development YEAR 1**

**Index**

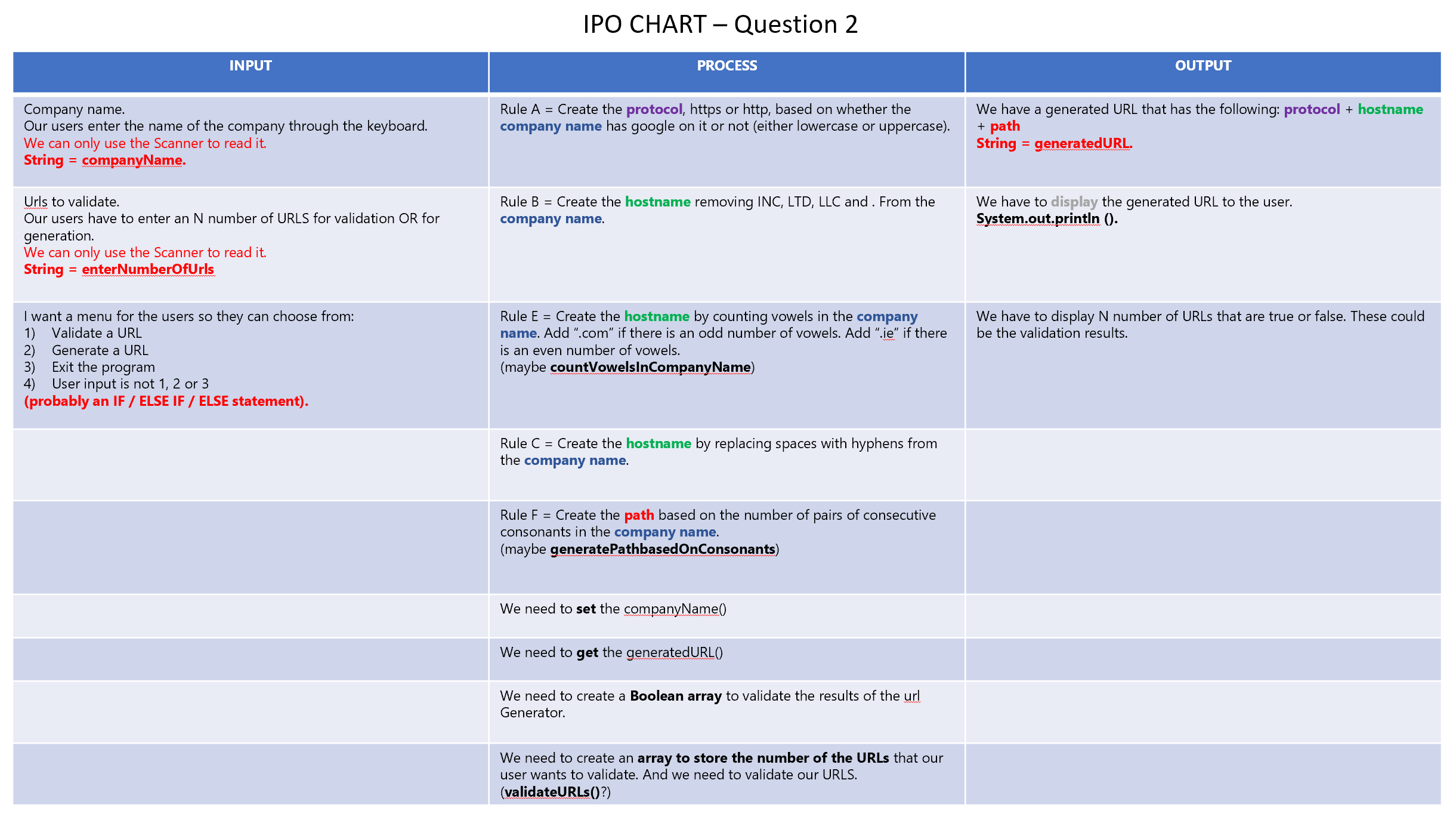
1. Input, main processing, and output (IPO) charts for Question 1 and Question 2. Included in the Zip File as well.
2. Mapping.
3. Java Source Code for Question 1. Main Class and Instantiable Class.
4. Java Source Code for Question 2. Main Class and Instantiable Class.

**1.  
Question 1)**

****

*Figure 1 – IPO Chart showing the input, processes and outputs of the program in Question 1.*

**Question 2)**

****

*Figure 2 – IPO Chart showing the input, processes and outputs of the program in Question 2.*

**2.**Last year we did a lot of charts and diagrams with Chetna in OOS, like Use Cases, etc but I am not familiar with the following:   
*The specific digits, mapped to the options/requirements that you have used to develop your applications.*

**3.**

**A) URLGenerator.java class (Instantiable class)**

/\*  
Title: URLGenerator.java  
Author: Miguel Angel Vinas  
Date: 5th January, 2024  
Purpose: Software Development TABA for National College Of Ireland  
\*/  
  
//Instantiable Class  
  
public class URLGenerator  
{  
 //These are the variables that we are going to use  
 private String companyName;  
 private String generatedURL;  
  
 //Here is our constructor. We are going to use to initialize this class.  
 public URLGenerator()  
 {  
 companyName = "";  
 generatedURL = "";  
 }  
  
 //Setter method to generate the company name.  
 public void setCompanyName(String companyName)  
 {  
 this.companyName = companyName;  
 }  
  
 //Getter method to return the generated URL.  
 public String getGeneratedURL()  
 {  
 return generatedURL;  
 }  
  
 //This is our compute method to generate and create the URL.  
 //We have several rules that we have to apply to generate and create the URL.  
 //So we are going to break the compute method in the different rules before we create the URL.  
 public void compute()  
 {  
 //Rule A = If the company name contains the word “Google” regardless of case, use “https”. Otherwise use “http”  
 //We are going to use a ternary operator (? :) to determine the protocol.  
 //We are converting the companyName string to a lowercase and checking if the companyName contains a substring "google"  
 //By using the ternary operator (? :), if the companyName contains google then we are setting https, else we are using http.  
 //In other words, our condition to be true is that the companyName contains google, if that's true we assign https, else we assign http.  
 String protocol = (companyName.toLowerCase().contains("google")) ? "https" : "http";  
  
 //Rule B = Remove any instances of Inc., Ltd., or LLC, from the company name.  
 //We are going to generate the hostname by calling the generateHostname() method.  
 //This method is going to be responsible for generating the hostname based on the rule above.  
 //The result is stored in the hostname variable so we can use it later.  
 String hostname = generateHostname();  
  
 //Rule F = Count the number of pairs of consonants next to each other in the company name.  
 //If there are no pairs the path name should be “index”, 1-3 consonant pairs contain “contactDetails”, 3+ pairs “basket”  
 //We are going to generate the path by calling the generatePath() method.  
 //This method is going to be responsible for generating the path based on the rule above.  
 //The result is stored in the path variable so we can use it later.  
 String path = generatePath();  
  
 //We are going to construct the URL now, based on the code and rules above.  
 generatedURL = protocol + "://" + hostname + "/" + path;  
 }  
  
 //Method to generate the hostname.  
  
  
 private String generateHostname()  
 {  
 //Rule C = Eliminate all spaces between the words in the company name and replace them with hyphens e.g., ‘-’.  
 String processedName = companyName.replaceAll("\\s+", "-");  
  
 //Rule B = Remove any instances of Inc., Ltd., or LLC, from the company name.  
 processedName = processedName.replaceAll("(?i)\\b(?:Inc\\.|Ltd\\.|LLC)\\b", "");  
  
 //Count the vowels and append either .com or .ie based on the number of vowels in the name.  
 int vowelCount = countVowels (processedName);  
 processedName += (vowelCount % 2 == 0) ? ".com" : ".ie";  
  
 return processedName;  
 }  
  
 //Method to count vowels.  
 private int countVowels (String input)  
 {  
 //This is our string where we put all the vowels in lowercase and uppercase.  
 String vowels = "aeiouAEIOU";  
 //And we initialize a counter to count them.  
 int count = 0;  
 //We are getting the length of the string that has been inputted.  
 int length = input.length();  
  
 //And we are iterating through each character of the string  
 for (int i = 0; i < length; i++)  
 {  
 //We are checking if the current character is a vowel by checking if it is in the input substring.  
 if (vowels.contains (input.substring(i, i +1)))  
 {  
 //If it is, we increment the count.  
 count++;  
 }  
 }  
 //And we return the count of the vowels in the string inputted.  
 return count;  
 }  
 //Method to generate the path  
 private String generatePath()  
 {  
 //Count the pairs of consecutive consonants to determine the path.  
 int pairCount = countConsonantPairs();  
  
 if (pairCount == 0)  
 {  
 return "index";  
 }  
 else if (pairCount <= 3)  
 {  
 return "contactDetails";  
 }  
 else  
 {  
 return "basket";  
 }  
 }  
  
 //Method to count pairs of consecutive consonants.  
 //It is similar to the countvowels method.  
 private int countConsonantPairs()  
 {  
 //We are defining a string that contains all the consonants in lowercase and uppercase.  
 String consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ";  
  
 //And we are initializing our counter to count the pairs of consecutive consonants.  
 int pairCount = 0;  
  
 //We are getting the length of the string.  
 int length = companyName.length();  
  
 //And we are iterating through each character.  
 for (int i = 0; i < length -1; i++)  
 {  
 //We are extracting the current character and the next character from the string  
 String currentChar = companyName.substring(i, i +1);  
 String nextChar = companyName.substring(i + 1, i +2);  
  
 //And we are checking if both (current and next characters) are consonants by checking our "consonants" string.  
 if (consonants.contains(currentChar) && consonants.contains(nextChar))  
 {  
 //If both are, we are incrementing the count of pairs of consonants.  
 pairCount++;  
 }  
 }  
 //And we are returning the total count of pairs of consecutive consonants.  
 return pairCount;  
 }  
  
}

**B) URLGeneratorApp.java class (Main class)**

/\*  
Title: URLGeneratorApp.java  
Author: Miguel Angel Vinas  
Date: 5th January, 2024  
Purpose: Software Development TABA for National College Of Ireland  
\*/  
  
import java.util.Scanner;  
  
//App Class  
public class URLGeneratorApp  
{  
 public static void main (String[] args)  
 {  
 //We are creating an instance of our URLGenerator class.  
 URLGenerator urlGenerator = new URLGenerator();  
  
 //We are going to get the company name from the user through the Scanner.  
 //(I would have loved to use the JPane though!)  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println ("Please, enter the company name that you want to generate a URL for: ");  
 String companyName = scanner.nextLine();  
  
 //We are going to set the company name.  
 urlGenerator.setCompanyName(companyName);  
  
 //We are going to generate the URL.  
 urlGenerator.compute();  
  
 //We are going to display the generated URL.  
 System.*out*.println ("Here is your generated URL: " + urlGenerator.getGeneratedURL());  
 System.*out*.println ("Thank you very much for using: THE GENERATOR!");  
 System.*out*.println (" ");  
  
 //We are closing the scanner because it is good practice. Thanks Francis!!  
 scanner.close();  
 }  
  
}

4.  
**A) URLGenerator.java class (Instantiable class)**

/\*  
Title: URLGenerator.java  
Author: Miguel Angel Vinas  
Date: 5th January, 2024  
Purpose: Software Development TABA for National College Of Ireland  
\*/  
  
//Instantiable Class  
  
import org.jetbrains.annotations.NotNull;  
  
public class URLGenerator  
{  
 //These are the variables that we are going to use  
 private String companyName;  
 private String generatedURL;  
  
 //Here is our constructor. We are going to use to initialize this class.  
 public URLGenerator()  
 {  
 companyName = "";  
 generatedURL = "";  
 }  
  
 //Setter method to generate the company name.  
 public void setCompanyName(String companyName)  
 {  
 this.companyName = companyName;  
 }  
  
 //Getter method to return the generated URL.  
 public String getGeneratedURL()  
 {  
 return generatedURL;  
 }  
  
 //This is our compute method to generate and create the URL.  
 //We have several rules that we have to apply to generate and create the URL.  
 //So we are going to break the compute method in the different rules before we create the URL.  
 public void compute()  
 {  
 //Rule A = If the company name contains the word “Google” regardless of case, use “https”. Otherwise use “http”  
 //We are going to use a ternary operator (? :) to determine the protocol.  
 //We are converting the companyName string to a lowercase and checking if the companyName contains a substring "google"  
 //By using the ternary operator (? :), if the companyName contains google then we are setting https, else we are using http.  
 //In other words, our condition to be true is that the companyName contains google, if that's true we assign https, else we assign http.  
 String protocol = (companyName.toLowerCase().contains("google")) ? "https" : "http";  
  
 //Rule B = Remove any instances of Inc., Ltd., or LLC, from the company name.  
 //We are going to generate the hostname by calling the generateHostname() method.  
 //This method is going to be responsible for generating the hostname based on the rule above.  
 //The result is stored in the hostname variable so we can use it later.  
 String hostname = generateHostname();  
  
 //Rule F = Count the number of pairs of consonants next to each other in the company name.  
 //If there are no pairs the path name should be “index”, 1-3 consonant pairs contain “contactDetails”, 3+ pairs “basket”  
 //We are going to generate the path by calling the generatePath() method.  
 //This method is going to be responsible for generating the path based on the rule above.  
 //The result is stored in the path variable so we can use it later.  
 String path = generatePath();  
  
 //We are going to construct the URL now, based on the code and rules above.  
 generatedURL = protocol + "://" + hostname + "/" + path;  
 }  
  
 //Method to generate the hostname.  
  
 @NotNull  
 private String generateHostname()  
 {  
 //Rule C = Eliminate all spaces between the words in the company name and replace them with hyphens e.g., ‘-’.  
 String processedName = companyName.replaceAll("\\s+", "-");  
  
 //Rule B = Remove any instances of Inc., Ltd., or LLC, from the company name.  
 processedName = processedName.replaceAll("(?i)\\b(?:Inc\\.|Ltd\\.|LLC)\\b", "");  
  
 //Count the vowels and append either .com or .ie based on the number of vowels in the name.  
 int vowelCount = countVowels (processedName);  
 processedName += (vowelCount % 2 == 0) ? ".com" : ".ie";  
  
 return processedName;  
 }  
  
 //Method to count vowels.  
 private int countVowels (String input)  
 {  
 //This is our string where we put all the vowels in lowercase and uppercase.  
 String vowels = "aeiouAEIOU";  
 //And we initialize a counter to count them.  
 int count = 0;  
 //We are getting the length of the string that has been inputted.  
 int length = input.length();  
  
 //And we are iterating through each character of the string  
 for (int i = 0; i < length; i++)  
 {  
 //We are checking if the current character is a vowel by checking if it is in the input substring.  
 if (vowels.contains (input.substring(i, i +1)))  
 {  
 //If it is, we increment the count.  
 count++;  
 }  
 }  
 //And we return the count of the vowels in the string inputted.  
 return count;  
 }  
 //Method to generate the path  
 private String generatePath()  
 {  
 //Count the pairs of consecutive consonants to determine the path.  
 int pairCount = countConsonantPairs();  
  
 if (pairCount == 0)  
 {  
 return "index";  
 }  
 else if (pairCount <= 3)  
 {  
 return "contactDetails";  
 }  
 else  
 {  
 return "basket";  
 }  
 }  
  
 //Method to count pairs of consecutive consonants.  
 //It is similar to the countvowels method.  
 private int countConsonantPairs()  
 {  
 //We are defining a string that contains all the consonants in lowercase and uppercase.  
 String consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ";  
  
 //And we are initializing our counter to count the pairs of consecutive consonants.  
 int pairCount = 0;  
  
 //We are getting the length of the string.  
 int length = companyName.length();  
  
 //And we are iterating through each character.  
 for (int i = 0; i < length -1; i++)  
 {  
 //We are extracting the current character and the next character from the string  
 String currentChar = companyName.substring(i, i +1);  
 String nextChar = companyName.substring(i + 1, i +2);  
  
 //And we are checking if both (current and next characters) are consonants by checking our "consonants" string.  
 if (consonants.contains(currentChar) && consonants.contains(nextChar))  
 {  
 //If both are, we are incrementing the count of pairs of consonants.  
 pairCount++;  
 }  
 }  
 //And we are returning the total count of pairs of consecutive consonants.  
 return pairCount;  
 }  
  
 //Here is the method to validate Google URLs  
 public boolean[] validateGoogleURLs(String[] urls)  
 {  
 boolean[] validationResults = new boolean[urls.length];  
  
 for (int i = 0; i < urls.length; i++)  
 {  
 validationResults[i] = validateGoogleURL(urls[i]);  
 }  
  
 return validationResults;  
 }  
  
 // Method to validate a single Google URL  
 private boolean validateGoogleURL(String url)  
 {  
 //The rules for validating the Google URLs are:  
 //1) It must contain "Google" in the hostname.  
 //2) The hostname can only contain the extension ".ie" or ".com".  
 //3) It cannot be shorter than 6 characters.  
 //4) It can only contain letters (a-z, A-Z), digits (0-9), hyphens (-), forward slashes (/), and periods (.).  
  
 String lowercaseUrl = url.toLowerCase();  
  
 boolean containsGoogle = lowercaseUrl.contains("google");  
 boolean validHostname = lowercaseUrl.matches(".\*\\bgoogle\\b.\*\\.(com|ie).\*");  
 boolean validLength = url.length() >= 6;  
 boolean validCharacters = url.matches("[a-zA-Z0-9/\\-\\.]+");  
  
 return containsGoogle && validHostname && validLength && validCharacters;  
  
 //I cannot make this method work, same as with the previous one in Question 1 regarding the LLD and etc!.  
 }  
  
}

**B) URLGeneratorApp.java class (Main class)**

/\*  
Title: URLGeneratorApp.java  
Author: Miguel Angel Vinas  
Date: 5th January, 2024  
Purpose: Software Development TABA for National College Of Ireland  
\*/  
  
import java.util.Scanner;  
  
//App Class  
public class URLGeneratorApp  
{  
 public static void main (String[] args)  
 {  
 //We are creating an instance of our URLGenerator class.  
 URLGenerator urlGenerator = new URLGenerator();  
  
 //We are going to get the company name from the user through the Scanner.  
 //(I would have loved to use the JPane though!)  
 Scanner scanner = new Scanner(System.*in*);  
  
 //We want for the user to choose what he / she wants to do, either generate a URL or validate a URL.  
 //And we want to create an infinite loop so the user HAS to make a valid choice!.  
 //In order to do so we are going to use the while (true) construct.  
 while (true)  
 {  
 //Here we are displaying the menu.  
 System.*out*.println("Please choose an option.");  
 System.*out*.println("1: Generate URLs.");  
 System.*out*.println("2: Validate URLs.");  
 System.*out*.println("3: Exit the program.");  
 System.*out*.println("Enter either 1, 2 or 3, depending on what you want to do.");  
 System.*out*.println("Thanks.");  
  
 //We are storing the user's choice in an integer.  
 int choice = scanner.nextInt();  
 scanner.nextLine();  
  
 //And we are going to perform the choice.  
 if (choice == 1)  
 {  
 //We are going to execute the generateURLs method.  
 *generateURLs*(urlGenerator, scanner);  
 }  
 else if (choice == 2)  
 {  
 //We are going to execute the validateURLs method.  
 *validateURLs*(urlGenerator, scanner);  
 }  
 else if (choice == 3)  
 {  
 //We are going to exit the loop and end the program  
 System.*out*.println ("Thank you very much for using Miguel Angel Vinas' program: THE VALIDATOR!");  
 System.*out*.println ("See you soon!");  
 System.*out*.println (" ");  
 break;  
 }  
 else  
 {  
 //We are going to tell the user that the choice is not valid... and ask to enter a valid choice!.  
 System.*out*.println ("It seems like you have tried to enter a choice that is not 1, 2 or 3.");  
 System.*out*.println ("Please make a choice again.");  
 System.*out*.println (" ");  
 }  
 }  
  
 //We are closing the scanner because it is good practice. Thanks Francis!!  
 scanner.close();  
 }  
  
 //This is the method to generate URLs.  
 private static void generateURLs(URLGenerator urlGenerator, Scanner scanner)  
 {  
 //We are asking the user for the company name  
 System.*out*.println ("Please, enter the company name that you want to generate a URL for: ");  
 String companyName = scanner.nextLine();  
  
 //We are going to set the company name.  
 urlGenerator.setCompanyName(companyName);  
  
 //We are going to generate the URL.  
 urlGenerator.compute();  
  
 //We are going to display the generated URL.  
 System.*out*.println ("Here is your generated URL: " + urlGenerator.getGeneratedURL());  
 System.*out*.println ("Thank you very much for using: THE GENERATOR!");  
 System.*out*.println (" ");  
 }  
  
 //This is the method validate URLs  
 //We are passing the URLGenerator and the scanner as parameters.  
 private static void validateURLs (URLGenerator urlGenerator, Scanner scanner)  
 {  
 //We are asking the user for the number of URLs to validate.  
 System.*out*.print("Enter the number of URLs to validate: ");  
 int numberOfURLs = scanner.nextInt();  
 scanner.nextLine();  
  
 //We are creating an array to store the number of URLs that the user wants to validate.  
 String[] urls = new String[numberOfURLs];  
  
 //We are asking the user to input each URL.  
 for (int i = 0; i < numberOfURLs; i++)  
 {  
 System.*out*.print("Enter URL " + (i + 1) + ": ");  
 urls[i] = scanner.nextLine();  
 }  
  
 //And we are validating the URLs.  
 boolean[] validationResults = urlGenerator.validateGoogleURLs(urls);  
  
 //And after that validating them, we are displaying the validation results.  
 System.*out*.println("Validation results:");  
 for (int i = 0; i < numberOfURLs; i++)  
 {  
 System.*out*.println("URL " + (i + 1) + ": " + validationResults[i]);  
 System.*out*.println (" ");  
 }  
 }  
}