HDWD_SEP23OL

SOFTWARE DEVELOPMENT TABA

Luis Martins - 23194456

National College of Ireland

Table of Contents

Introduction

- This report presents an application developed to address two distinct functionalities related to website URLs. The first function, corresponding to Question 1 (UR1), involves generating a unique URL for a company based on its provided name. The second function, as per Question 2 (MCNA2), focuses on validating user-input URLs. Both functionalities operate within a user-interactive interface built using the JOptionPane library.

- This report details the input, processing, and output (IPO) for each function, along with the specific digits and requirements from the assignment that were implemented. Additionally, it provides the test plan and the complete Java source code for both functionalities that are part of the TABA.

Question 1

IPO Chart

Input	Processing	Output
String CompanyName	• protocol = determineProtocol(companyName);" • hostnameWithoutSuffix = removeSuffixes(companyName); • hostname = replaceSpacesWithHyphens(hostnameWithoutSuffix);	String GeneratedURL
	IF (hostname.isEmpty()) { • DISPLAY "Invalid company name";	

	}	
	• hostname += determineHostnameEnding(hostnameWithoutSuffix); • path = determinePath(hostnameWithoutSuffix); • generatedURL = protocol + "://" + hostname + path; RETURN generatedURL;	

	determineProtocol(companyName) {
	
	• If companyName contains "google" RETURN "https"
	• Else RETURN "http"
	
	}	

	removeSuffixes(companyName) {
	
	• FOR (String suffix in" Inc.", " Ltd.", " LLC") {
	
	• IF (companyName.endsWith(suffix)) {
	• RETURN companyName substring suffix);
	
	}
	
	RETURN companyName;	

	replaceSpacesWithHyphens(hostnameWithoutSuffix) {
	
	• RETURN hostnameWithoutSuffix.replaceAll(" ", "-");
	
	}	

	determineHostnameEnding(hostnameWithoutSuffix) {
	
	• int vowelCount = 0;
	• FOREACH char in hostnameWithoutSuffix {
	    o IF (isVowel(char)) => vowelCount++;
	
	    }
	
	}	

	determinePath(hostnameWithoutSuffix) {	

	• int consonantPairCount = 0; • FOREACH char in hostnameWithoutSuffic { • IF char is not vowel && nextChar is not vowel- consonantPairCount++; } • RETURN (consonantPairCount == 0) => "/index" (consonantPairCount < 3) => "/contactDetails" else => "/basket";	
	isVowel(char){ • RETURN (char in "aeiou"); }	

Specific Digits and Requirements:

The URL Rules followed for question 1 were “UR1” and “MCNA2” as the **penultimate** digit of my student id is **5** and the last is **6**:

UR1

“Given a company name in full i.e., Amazon Inc.
Note: Any letter can be in either upper case or lower case in the provided company name.

To select the appropriate URL protocol:

- If the company name contains the word “Meta” regardless of case, use “coap”. Otherwise use “ftp”

The URL hostname is generated from the given company name as follows:

- Replace any instances of incorporated, limited, or limited liability company with Inc, Ltd, LLC respectively for the company name.

Then the rest of the URL hostname is generated from the given company name as follows:

- The spaces are replaced by the underscore symbol '_'
- Apart from the above-mentioned letters and characters, all the other letters should be copied unchanged to the URL hostname from the given company name.

- If the company name contains an odd number of consonants add ".org" to the end of the hostname. If it contains an even number of consonants add ".edu" to the end of the hostname.

To set the URL's path:

- Count the number of pairs of vowels next to each other in the company name. If there are no pairs the path name should be "bio", 1-3 vowel pairs contain "FAQ", 3+ pairs "Glossary""

MCNA2

"Ask the user to provide a company name and after the corresponding URL is generated and displayed on the screen, ask the user if they would like to generate another URL. As long as the user enters "yes" the application should work as described in the previous sentence. When the user enters anything other than "yes", no other items are created."

Testing

Test Case ID	Company Name	Expected URL	Actual URL	Pass (Yes/No)
TC01	Deloitte Touche Tohmatsu Ltd.	http://Deloitte-Touche-Tohmatsu.com/basket	http://Deloitte-Touche-Tohmatsu.com/basket	yes
TC02	Google LLC	https://Google.ie/contactDetails	https://Google.ie/contactDetails	yes
TC03	Monsters Inc.	http://Monsters.com/basket	http://Monsters.com/basket	yes
TC04	Meta Ltd.	http://Meta.com/index	http://Meta.com/index	yes

Question 2

IPO Chart

Input	Processing	Output

String[] urls	FOREACH url IN urls {
	• lowerUrl = url.toLower();
	• isValid = validateURL(lowerUrl);
	• Validations.append(isValid);
	• RETURN validations;
	}
		bool[] validations

	validateURL(lowerUrl) {
	• containsAmazonOrAws = ("amazon" in lowerUrl or "aws" in lowerUrl);
	• validLength = (5 <= len(lowerUrl) <= 16);
	• validChars = True;
	FOREACH char IN lowerUrl {
	IF char is not number or character and char not in "_/." {
	• validChars = False;
	• break;
	}
	• hasDot = "." in lowerUrl;
	• RETURN containsAmazonOrAws and validLength and validChars and hasDot;
	}
	}	

Specific Digits and Requirements

The rules followed in this were “Amazon URLs” as the **penultimate** digit of my student id is **5:**

Amazon URLs

“Functionality: Check the validity of an Amazon URL
A valid Amazon URL satisfies all the following rules:

- it must contain “amazon” or “aws” in the URL hostname.
- It cannot be shorter than 5 characters.
- It cannot be longer than 16 characters.

  It can only contain letters (i.e. a – z inclusive, A – Z inclusive), digits (i.e. 0 – 9 inclusive), and underscores (i.e. ‘ _ ’), forward slashes (i.e. ‘/’), and periods (i.e. ‘.’)”

Testing

Test Case ID	URL	Expected Result	Actual Result	Pass (Yes/No)
TC01	amazon.ie/uni_cash	false	false	yes
TC02	aws.ie/servers	true	true	yes
TC03	amazon	false	false	yes
TC04	aws.com/%s	false	false	yes
TC05	aws.com/hosts12	true	true	

Java Source Code

App Class – URLGeneratorApp

```
import javax.swing.JOptionPane;


public class URLGeneratorApp {


    public static void main(String[] args) {
        URLGenerator urlGenerator = new URLGenerator();
        String answer = "yes";
```

```java
        while (answer.equalsIgnoreCase("yes")) {
            // Prompt the user to choose functionality
            String choice = JOptionPane.showInputDialog("Choose a functionality:\n1. Generate URL\n2. Validate
URLs");

            if (choice.equals("1")) {
                urlGenerator = new URLGenerator(); // reset in case it's second+ time user generates a URL
                // Generate URL
                String companyName = urlGenerator.promptCompanyName();
                urlGenerator.setCompanyName(companyName);
                urlGenerator.computeURL();
                String generatedURL = urlGenerator.getGeneratedURL();
                JOptionPane.showMessageDialog(null, generatedURL);
            } else if (choice.equals("2")) {
                // Validate URLs
                int numberOfURLs = Integer.parseInt(JOptionPane.showInputDialog("Enter the number of URLs to
validate:"));
                String[] urls = new String[numberOfURLs];

                for (int i = 0; i < numberOfURLs; i++) {
                    urls[i] = JOptionPane.showInputDialog("Enter URL " + (i + 1) + ":");
                }

                boolean[] validations = urlGenerator.validateURLs(urls);

                for (int i = 0; i < numberOfURLs; i++) {
                    String message = urls[i] + " is ";
                    if(validations[i] == true) {
                        message += "valid";
                    }
                    else {
                        message += "invalid";
                    }
                    JOptionPane.showMessageDialog(null, message);
                }
            } else {
                JOptionPane.showMessageDialog(null, "Invalid choice. Please choose 1 or 2.");
            }

            // MCNA2 - Ask if the user wants to continue, accommodated to work with both methods.
            answer = JOptionPane.showInputDialog("Would you like to perform another action? (yes/no)");
        }
    }
}
```

Instantiable class – URLGenerator

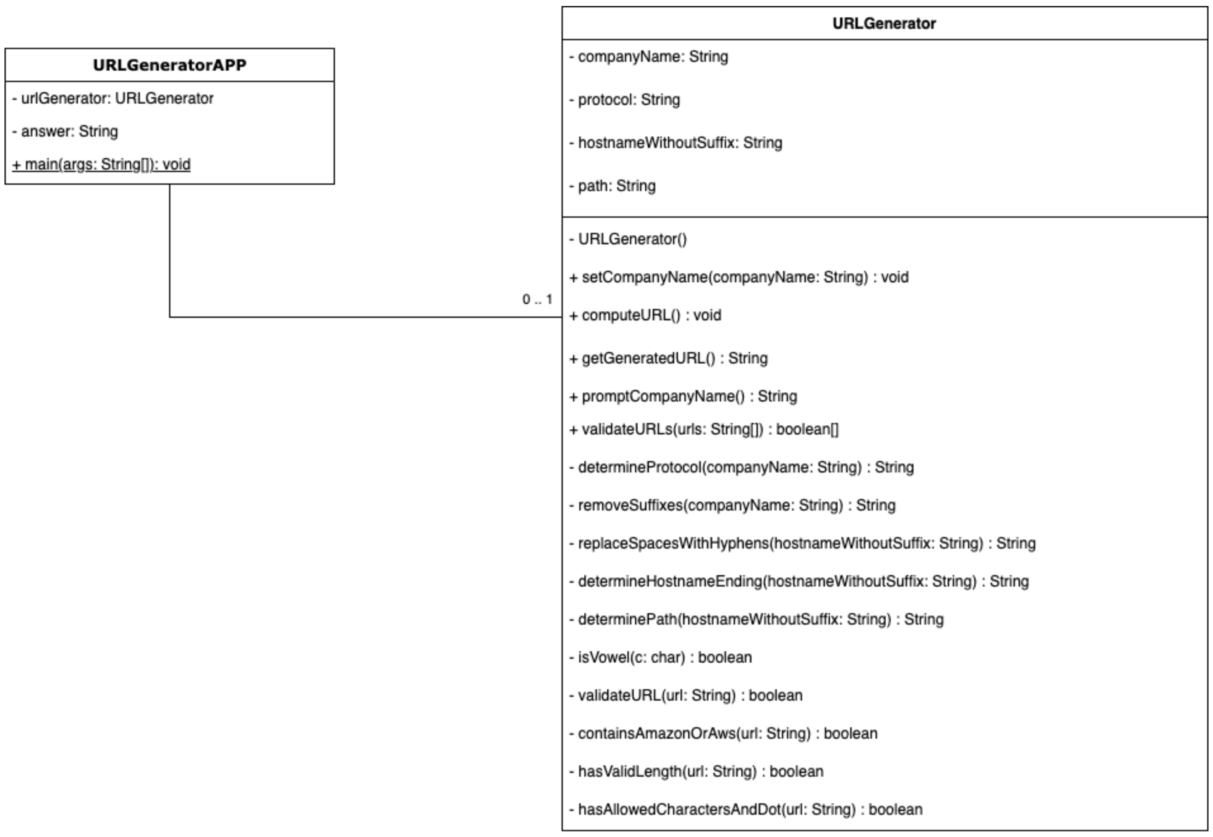
```java
import javax.swing.JOptionPane;

public class URLGenerator {

    // Data members (instance variables) to store URL components
    private String companyName;
    private String protocol;
    private String hostname;
    private String hostnameWithoutSuffix;
    private String path;

    // Constructor to initialize data members
    public URLGenerator() {
        protocol = "";
        hostname = "";
        path = "";
        hostnameWithoutSuffix = "";
    }

    // Question 1 - UR1

    // Setter method to set the company name
    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }
    // Main compute method for Question 1
    public void computeURL() {
        protocol = determineProtocol(companyName);
        hostnameWithoutSuffix = removeSuffixes(companyName);
        hostname = replaceSpacesWithHyphens(hostnameWithoutSuffix);

        if (hostname.isEmpty()) {
```

```java
        JOptionPane.showMessageDialog(null, "Invalid company name");
        return;
    }

    hostname += determineHostnameEnding(hostnameWithoutSuffix);
    path = determinePath(hostnameWithoutSuffix);
}

// a) Determine protocol based on company name "google" regardless of case
private String determineProtocol(String companyName) {
    return companyName.toLowerCase().contains("google") ? "https" : "http";
}

// b) Remove specified suffixes from company name
private String removeSuffixes(String companyName) {
    String[] suffixes = {" Inc.", " Ltd.", " LLC"};
    for (String suffix : suffixes) {
        if (companyName.endsWith(suffix)) {
            return companyName.substring(0, companyName.length() - suffix.length());
        }
    }
    return companyName;
}

// c) Replace spaces with hyphens
// d) Apart from the above-mentioned letters and characters,
// all the other letters should be copied unchanged to the URL hostname from the given company name.
private String replaceSpacesWithHyphens(String hostnameWithoutSuffix) {
    return hostnameWithoutSuffix.replaceAll(" ", "-");
}

// e) Determine hostname ending based on vowel count
private String determineHostnameEnding(String hostnameWithoutSuffix) {
    int vowelCount = 0;
    for (char c : hostnameWithoutSuffix.toLowerCase().toCharArray()) {
        if (isVowel(c)) {
            vowelCount++;
        }
    }
    return vowelCount % 2 == 0 ? ".com" : ".ie";
```

```java
}

// f) Determine path based on consonant pair count
private String determinePath(String hostnameWithoutSuffix) {
    int consonantPairCount = 0;
    for (int i = 0; i < hostnameWithoutSuffix.length() - 1; i++) {
        char c1 = hostnameWithoutSuffix.toLowerCase().charAt(i);
        char c2 = hostnameWithoutSuffix.toLowerCase().charAt(i + 1);
        if (!isVowel(c1) && !isVowel(c2)) {
            consonantPairCount++;
        }
    }
    return consonantPairCount == 0 ? "/index" : consonantPairCount < 3 ? "/contactDetails" : "/basket";
}


private boolean isVowel(char c) {
    return "aeiou".indexOf(c) != -1;
}

// Getter method to return the generated URL
public String getGeneratedURL() {
    return protocol + "://" + hostname + path;
}

// Method to prompt the user for company name using JOptionPane
public String promptCompanyName() {
    return JOptionPane.showInputDialog("Enter the full company name:");
}
// Question 2 - Amazon URLs
public boolean[] validateURLs(String[] urls) {
    boolean[] validations = new boolean[urls.length];

    for (int i = 0; i < urls.length; i++) {
        validations[i] = validateURL(urls[i]);
    }

    return validations;
}
```

```java
private boolean validateURL(String url) {
    url = url.toLowerCase(); // Convert URL to lowercase

    if (!containsAmazonOrAws(url)) {
        return false;
    }

    if (!hasValidLength(url)) {
        return false;
    }

    return hasAllowedCharactersAndDot(url);
}

// It must contain “amazon” or “aws” in the URL hostname.
private boolean containsAmazonOrAws(String url) {
    return url.contains("amazon") || url.contains("aws");
}

// It cannot be shorter than 5 characters.
// It cannot be longer than 16 characters.
private boolean hasValidLength(String url) {

    return url.length() >= 5 && url.length() <= 16;
}

// It can only contain letters (i.e. a – z inclusive, A – Z inclusive),
// digits (i.e. 0 – 9 inclusive), and underscores (i.e. ‘_’), forward slashes (i.e. ‘/’), and periods (i.e. ‘.’)
private boolean hasAllowedCharactersAndDot(String url) {
    // Check allowed characters
    boolean isValid = true;
    boolean dotFound = false;
    for (char c : url.toCharArray()) {
        if (!(Character.isLetterOrDigit(c) || c == '_' || c == '/' || c == '.')) {
            isValid = false;
            break;
        }
        if (c == '.') {
            dotFound = true;
        }
    }
    return isValid && dotFound;
  }



}
```

Class Diagram


URLGeneratorAPP
- urlGenerator: URLGenerator
- answer: String
+ main(args: String[]): void
0 .. 1
URLGenerator
- companyName: String
- protocol: String
- hostnameWithoutSuffix: String
- path: String
- URLGenerator()
+ setCompanyName(companyName: String) : void
+ computeURL() : void
+ getGeneratedURL() : String
+ promptCompanyName() : String
+ validateURLs(urls: String[]) : boolean[]
- determineProtocol(companyName: String) : String
- removeSuffixes(companyName: String) : String
- replaceSpacesWithHyphens(hostnameWithoutSuffix: String) : String
- determineHostnameEnding(hostnameWithoutSuffix: String) : String
- determinePath(hostnameWithoutSuffix: String) : String
- isVowel(c: char) : boolean
- validateURL(url: String) : boolean
- containsAmazonOrAws(url: String) : boolean
- hasValidLength(url: String) : boolean
- hasAllowedCharactersAndDot(url: String) : boolean

Conclusion

- This application effectively addresses two distinct URL-related functionalities: generating unique URLs based on company names and validating Amazon URLs.

- The development of this application adhered closely to the TABA specifications, integrating key functionalities for URL generation and validation. It demonstrates practical application of knowledge gained throughout the course, including string manipulation techniques for URL construction and modification, loop structures for character counting and array operations, conditional logic for applying validation rules, and the effective use of JOptionPane dialogs to create a user-friendly interface that guides users through input and displays results clearly.