24.03.2022

VALIDATION OF AN EMAIL ID PROJECT

Prepared by: PRIYANKA DAS

Project objective:

As a developer, write a program to search a string entered by a user from the array of strings.

Background of the problem statement:

You have an array of email IDs of employees. As a programmer, write a program to search the email ID entered by a user.

We must use the following:

Eclipse/IntelliJ: An IDE to code the application

Java: A programming language

Git: To connect and push files from the local system to GitHub

GitHub: To store the application code and track its versions

Core Java concepts: Methods, collections, and strings

Following requirements should be met:

The versions of the code should be tracked on GitHub repositories

The code should be able to search the required string from the array of strings

About Project

Email validation is pivotal in ascertaining the quality and accuracy of email addresses. Keeping non-existent, inactive, or mistyped email accounts can unnecessarily fill your contact lists, leading to reduced deliverability and hurting your promotion efforts.

A valid email address is comprised of three sections: a local part, the at-sign (@), and a domain name. If this format is not followed, then the address becomes invalid.

According to the specifications of most Internet mail systems, the local part may use any of the following ASCII standard characters:

- digits—0 to 9
- lowercase and uppercase Latin letters—a to z and A to Z
- printable characters—!#$%&'*+-/=?^_`{|}~
- dot—., as long as it is not the initial or final character, or is not used consecutively

About Project(contd)

Furthermore, the domain name section of the email address may consist of the following characters:

- digits—0 to 9
- lowercase and uppercase Latin letters—a to z and A to Z
- hyphen or dot — - or . , as long as they are not the initial or final characters.
-

Here are some examples of valid email addresses:

- alice@example.com
- alice.bob@example.com
- alice@example.me.org
-

On the other hand, here are some examples of invalid email addresses:

- alice.example.com (no @ character)
- alice..bob@example.com (two consecutive dots not permitted)
- alice@.example.com (domain cannot start with a dot)

About Project Code

Here the project is been done by me in two ways.
--> **Using Regex**

In Java, **email validation** is performed by using the regular expression.

Here I used the type of regex is :
---> Regex to restrict leading, trailing, or consecutive dots in emails;
The regular expression "^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+(?:\\.[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+)*@[a-zA-Z0-9-]+(?:\\.[a-zA-Z0-9-]+)*$" restrict us to add consecutive dots, trailing, and leading. An email can contain more than one dot in both the local part and the domain name, but consecutive dots are allowed. Our email can also not be started or ended with a dot. The regex validates the email based on these three conditions too.

--> **Without Using Regex**
By using String Manipulation along with array implementation.

Using Regex

```
 import java.util.regex.*;
import java.util.*;
public class EmailValidationRegex {
 public static void main(String args[]){
 ArrayList<String> emails = new ArrayList<String>();

 emails.add("Priyanka@domain.co.in");
 emails.add("priYanka@domain.com");
 emails.add("priyaNka.name@domain.com");
 emails.add("pRiyanka#@domain.co.in");
 emails.add("priyankat@domain.com");
 emails.add(".priyankat@yahoo.com");
 emails.add("priyanka@domain.com.");
 emails.add("priyanka#domain.com");
 emails.add("priyanka@domain..com");
 //Regular Expression
 String regex = "^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+(?:\\.[a-
zA-Z0-9_!#$%&'*+/=?`{|}~^-]+)*@[a-zA-Z0-9-]+(?:\\.[a-zA-
Z0-9-]+)*$";
 //Compile regular expression to get the pattern
 Pattern pattern = Pattern.compile(regex);
 //Iterate emails array list
 for(String email : emails){
 //Create instance of matcher
 Matcher matcher = pattern.matcher(email);
 System.out.println(email +" : "+
matcher.matches()+"\n");
 }
 }
}
```

Using String

```
import java.util.Arrays;

import java.util.Scanner;

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

 String[] emailList = new String[10];

 emailList[0] = "BTS@0000.com";
 emailList[1] = "RM@s1111.com";
 emailList[2] = "Jimin@2222.com";
 emailList[3] = "Jin@3333.com";
 emailList[4] = "Jungkook@4444.com";
 emailList[5] = "Suga@5555.com";
 emailList[6] = "Jhope@6666.com";
 emailList[7] = "Taehyung@7777.com";
 emailList[8] = "Dynamite@8888.com";
 emailList[9] = "Butter@9999.com";

 Scanner sc = new Scanner(System.in);
 System.out.println("Email addresses in current Array: " +
Arrays.toString(emailList));
```

Using String(contd)

```
 System.out.println("\nEnter an email address OR email
id to check if there is an associated email address in "+
 "the Array.");
 System.out.println("Note: email addresses and ids are
case sensitive!");

 String searchStr = sc.nextLine();

 System.out.println(checkIfEmailExists(searchStr,
emailList));
 }
 // Method is only used for the Array implementation
 private static String checkIfEmailExists(String
searchStr, String[] emailList){
 for (String s : emailList) {
 if (s.toLowerCase().equals(searchStr.toLowerCase()) ||
 searchStr.toLowerCase().equals(s.substring(0,
s.indexOf('@')).toLowerCase()))
 return "\n\"" + searchStr + "\"" + " exists in the Array!\n" +
"Matched email: " + s;
 }
 return "\""+searchStr+"\""+" does not exist in the Array.";
 }


}
}
```

.....THE END.....