Rajalakshmi Engineering College

Name: Darshan S
Email: 241801040@rajalakshmi.edu.in
Roll no: 241801040
Phone: 7305911089
Branch: REC
Department: AI & DS - Section 4
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Write a program to validate the email address and display suitable exceptions if there is any mistake.

Create 3 custom exception classes as below

DotExceptionAtTheRateExceptionDomainException

A typical email address should have a " . " character, and a "@" character, and also the domain name should be valid. Valid domain names for practice be 'in', 'com', 'net', or 'biz'.

Display Invalid Dot usage, Invalid @ usage, or Invalid Domain message based on email id.

Get the email address from the user, validate the email by checking the above-mentioned criteria, and print the validity status of the input email address.

Input Format

The first line of input contains the email to be validated.

Output Format

The output prints a Valid email address or an Invalid email address along with the suitable exception

If email ends with . or contains not exactly one . after @, it throws:

DotException: Invalid Dot usage

Invalid email address

If @ appears not exactly once, it throws:

AtTheRateException: Invalid @ usage

Invalid email address

If the part after the last dot is not among accepted domains:

DomainException: Invalid Domain

Invalid email address

If all conditions satisfied then print:

Valid email address

Refer to the sample input and output for format specifications.

Sample Test Case

Input: sample@gmail.com
Output: Valid email address

Answer

```
// You are using Java
import java.util.Scanner;

// Custom exceptions
class DotException extends Exception {
   public DotException(String message) {
      super(message);
   }
}

class AtTheRateException extends Exception {
   public AtTheRateException(String message) {
      super(message);
   }
}

class DomainException extends Exception {
   public DomainException(String message) {
      super(message);
   }
}

class EmailValidator {

   public static void validateEmail(String email) throws DotException,
AtTheRateException, DomainException {
      // Check that email length between 5 and 50 (given in constraints)
      if (email.length() < 5 || email.length() > 50) {
         // Not explicitly asked to handle this, but assuming valid input length as
per problem
      }
```

```
        // Check that email does not start or end with '.' or '@'
        if (email.startsWith(".") || email.startsWith("@") || email.endsWith(".") ||
email.endsWith("@")) {
            throw new DotException("DotException: Invalid Dot usage");
        }

        // Check exactly one '@'
        int atCount = 0;
        for (char c : email.toCharArray()) {
            if (c == '@') atCount++;
        }
        if (atCount != 1) {
            throw new AtTheRateException("AtTheRateException: Invalid @ usage");
        }

        // Check consecutive '.' or '@' not allowed
        for (int i = 1; i < email.length(); i++) {
            char curr = email.charAt(i);
            char prev = email.charAt(i - 1);
            if ((curr == '.' && prev == '.') || (curr == '@' && prev == '@')) {
                throw new DotException("DotException: Invalid Dot usage");
            }
        }

        // Get parts around '@'
        int atIndex = email.indexOf('@');
        String localPart = email.substring(0, atIndex);
        String domainPart = email.substring(atIndex + 1);

        // domainPart must have at least one '.' (after @)
        int dotAfterAt = domainPart.indexOf('.');
        if (dotAfterAt == -1) {
            throw new DotException("DotException: Invalid Dot usage");
        }

        // After '@', check the number of '.'s (should be exactly one dot? The problem
says "not exactly one . after @")

        int dotCountAfterAt = 0;
        for (char c : domainPart.toCharArray()) {
            if (c == '.') dotCountAfterAt++;
        }
        if (dotCountAfterAt != 1) {
            throw new DotException("DotException: Invalid Dot usage");
        }

        // Extract domain extension after last dot
        int lastDotIndex = domainPart.lastIndexOf('.');
        if (lastDotIndex == domainPart.length() - 1) {
            // ends with dot (handled above), but just in case
            throw new DotException("DotException: Invalid Dot usage");
        }
        String domainExtension = domainPart.substring(lastDotIndex + 1);

        // Check domain extension
        if (!(domainExtension.equals("in") || domainExtension.equals("com") ||
domainExtension.equals("net") || domainExtension.equals("biz"))) {
            throw new DomainException("DomainException: Invalid Domain");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String email = scanner.nextLine();

        try {
            validateEmail(email);
            System.out.println("Valid email address");
        } catch (DotException | AtTheRateException | DomainException e) {
            System.out.println(e.getMessage());
            System.out.println("Invalid email address");
        }

        scanner.close();
    }
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Darshan S
Email: 241801040@rajalakshmi.edu.in
Roll no: 241801040
Phone: 7305911089
Branch: REC
Department: AI & DS - Section 4
Batch: 2028
Degree: B.E - AI & DS

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Elsa, a busy professional, is using a scheduling application to plan her meetings efficiently. The application requires users to input meeting durations in minutes, ensuring that the duration is a positive integer and does not exceed 240 minutes (4 hours). Elsa needs a program to assist her in scheduling meetings securely with proper exception handling.

Create a Java class named ElsaMeetingScheduler.Implement a custom exception:InvalidDurationException for invalid meeting duration entries.Implement the main method to interactively take user input for a meeting duration.Implement the validateMeetingDuration method to validate the meeting duration based on the specified rules and throw a custom exception if the validation fails.Print appropriate success or error messages based on the meeting duration.

Implement a custom exception, InvalidDurationException, to handle cases where the entered meeting duration does not meet the specified criteria.

Input Format

The input consists of an integer value 'n', representing the meeting duration.

Output Format

The output is displayed in the following format:

If the entered meeting duration meets the specified criteria, the program outputs

"Meeting scheduled successfully!"

If the entered meeting duration is invalid, the program outputs an error message indicating the issue.

"Error: Invalid meeting duration. Please enter a positive integer not exceeding 240 minutes (4 hours)."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 120
Output: Meeting scheduled successfully!

Answer

```
import java.util.Scanner;

// Custom Exception class extending Exception
class InvalidDurationException extends Exception {
    public InvalidDurationException(String message) {
        super(message);
    }
}

class ElsaMeetingScheduler {

    // Whitelist validation method: only allow positive integers <= 240
    public static void validateMeetingDuration(int duration) throws
InvalidDurationException {
        if (duration <= 0 || duration > 240) {
            throw new InvalidDurationException(
                "Error: Invalid meeting duration. Please enter a positive integer not
exceeding 240 minutes (4 hours)."
            );
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            int duration = scanner.nextInt();
            validateMeetingDuration(duration);
            System.out.println("Meeting scheduled successfully!");
        } catch (InvalidDurationException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {

            System.out.println("Error: Invalid input. Please enter a valid integer.");
        } finally {
            scanner.close();
        }
    }
}
```

Status : Correct ***Marks : 10/10***