Rajalakshmi Engineering College

Name: Mohamed Fadil
Email: 241501114@rajalakshmi.edu.in
Roll no: 241501114
Phone: null
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

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

241501114

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 Exception for Dot validation
class DotException extends Exception {
    public DotException(String message) {
        super(message);
    }
}

// Custom Exception for @ validation
class AtTheRateException extends Exception {
    public AtTheRateException(String message) {
        super(message);
    }
}

// Custom Exception for Domain validation
class DomainException extends Exception {
    public DomainException(String message) {
        super(message);
    }
}

public class Main {

    public static void validateEmail(String email) throws DotException,
AtTheRateException, DomainException {
        // Check for exactly one @ character
        int atCount = 0;
        int atPosition = -1;
        for (int i = 0; i < email.length(); i++) {
```

```
        if (email.charAt(i) == '@') {
            atCount++;
            atPosition = i;
        }
    }

    if (atCount != 1) {
        throw new AtTheRateException("Invalid @ usage");
    }

    // Check if email ends with dot
    if (email.endsWith(".")) {
        throw new DotException("Invalid Dot usage");
    }

    // Check if email starts with dot or @
    if (email.startsWith(".") || email.startsWith("@")) {
        throw new DotException("Invalid Dot usage");
    }

    // Get the part after @
    String afterAt = email.substring(atPosition + 1);

    // Check if there's at least one dot after @
    if (!afterAt.contains(".")) {
        throw new DotException("Invalid Dot usage");
    }

    // Check for consecutive dots
    if (email.contains("..")) {
        throw new DotException("Invalid Dot usage");
    }

    // Check for consecutive @ symbols
    if (email.contains("@@")) {
        throw new AtTheRateException("Invalid @ usage");
    }

    // Check if dot appears immediately after @
    if (afterAt.startsWith(".")) {
        throw new DotException("Invalid Dot usage");
    }
```

```
        // Extract domain (part after last dot)
        int lastDotPosition = email.lastIndexOf('.');
        String domain = email.substring(lastDotPosition + 1);

        // Validate domain
        if (!domain.equals("in") && !domain.equals("com") && !domain.equals("net")
&& !domain.equals("biz")) {
            throw new DomainException("Invalid Domain");
        }

        // Check if there's content between @ and the first dot after @
        int firstDotAfterAt = afterAt.indexOf('.');
        if (firstDotAfterAt == 0) {
            throw new DotException("Invalid Dot usage");
        }

        // Check if there's content between last dot and end (domain should not be
empty)
        if (domain.isEmpty()) {
            throw new 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 e) {
            System.out.println("DotException: " + e.getMessage());
            System.out.println("Invalid email address");
        } catch (AtTheRateException e) {
            System.out.println("AtTheRateException: " + e.getMessage());
            System.out.println("Invalid email address");
        } catch (DomainException e) {
            System.out.println("DomainException: " + e.getMessage());
            System.out.println("Invalid email address");
        } finally {
            scanner.close();
```

2415011114

```
        }
    }
}
```

Status : Correct *Marks : 10/10*

241501114

Rajalakshmi Engineering College

Name: Mohamed Fadil
Email: 241501114@rajalakshmi.edu.in
Roll no: 241501114
Phone: null
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

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

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

// Custom Exception for Invalid Duration
class InvalidDurationException extends Exception {
    public InvalidDurationException(String message) {
        super(message);
    }
}

public class Main {
```

2415011114

```
    // Method to validate meeting duration
    public static void validateMeetingDuration(int duration) throws
InvalidDurationException {
        // Check if duration is positive and does not exceed 240 minutes
        if (duration <= 0 || duration > 240) {
            throw new InvalidDurationException("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);

        // Read meeting duration from user
        int meetingDuration = scanner.nextInt();

        try {
            // Validate the meeting duration
            validateMeetingDuration(meetingDuration);

            // If validation passes, print success message
            System.out.println("Meeting scheduled successfully!");

        } catch (InvalidDurationException e) {
            // If validation fails, print error message
            System.out.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

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

2415011114

Rajalakshmi Engineering College

Name: Mohamed Fadil
Email: 241501114@rajalakshmi.edu.in
Roll no: 241501114
Phone: null
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a user registration system, there is a requirement to implement a username validation module. Users attempting to register must adhere to specific criteria for their usernames to be considered valid.

Your task is to develop a program that takes user input for a desired username and validates it according to the following rules:

The username must not contain any spaces.The username must be at least 5 characters long.

Implement a custom exception, InvalidUsernameException, to handle cases where the entered username does not meet the specified criteria.

Input Format

The input consists of a string S, representing the desired username.

Output Format

If the username is valid, print "Username is valid: [S]".

If the username is invalid:

1. If the username is short, print "Invalid Username: Username must be at least 5 characters long"
2. If the username contains spaces, print "Invalid Username: Username cannot contain spaces"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: John
Output: Invalid Username: Username must be at least 5 characters long

Answer

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

// Custom Exception for Invalid Username
class InvalidUsernameException extends Exception {
    public InvalidUsernameException(String message) {
        super(message);
    }
}

public class Main {

    // Method to validate username
    public static void validateUsername(String username) throws
InvalidUsernameException {
        // Check if username contains spaces
        if (username.contains(" ")) {
            throw new InvalidUsernameException("Username cannot contain
spaces");
        }
```

```
        // Check if username is at least 5 characters long
        if (username.length() < 5) {
            throw new InvalidUsernameException("Username must be at least 5
characters long");
        }
    }

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

        // Read username from user
        String username = scanner.nextLine();

        try {
            // Validate the username
            validateUsername(username);

            // If validation passes, print success message
            System.out.println("Username is valid: " + username);

        } catch (InvalidUsernameException e) {
            // If validation fails, print error message
            System.out.println("Invalid Username: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

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

2415011114