Rajalakshmi Engineering College

Name: Punith Kumar.S
Email: 241501155@rajalakshmi.edu.in
Roll no: 241501155
Phone: 9600149411
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

```
import java.util.*;

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

class UsernameValidator {
    public void validate(String username) throws InvalidUsernameException {
        if (username.contains(" ")) {
            throw new InvalidUsernameException("Invalid Username: Username
cannot contain spaces");
        }
        if (username.length() < 5) {
            throw new InvalidUsernameException("Invalid Username: Username must
be at least 5 characters long");
        }
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String username = sc.nextLine();
        UsernameValidator validator = new UsernameValidator();

        try {
            validator.validate(username);
            System.out.println("Username is valid: " + username);
        } catch (InvalidUsernameException e) {
            System.out.println(e.getMessage());
        }

        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Punith Kumar.S
Email: 241501155@rajalakshmi.edu.in
Roll no: 241501155
Phone: 9600149411
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

```
import java.util.*;

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

class ElsaMeetingScheduler {
    public 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 sc = new Scanner(System.in);
        int duration = sc.nextInt();
        ElsaMeetingScheduler scheduler = new ElsaMeetingScheduler();

        try {
            scheduler.validateMeetingDuration(duration);
            System.out.println("Meeting scheduled successfully!");
        } catch (InvalidDurationException e) {
            System.out.println(e.getMessage());
        }

        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Punith Kumar.S
Email: 241501155@rajalakshmi.edu.in
Roll no: 241501155
Phone: 9600149411
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 : 7.5

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

```
import java.util.*;

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 validate(String email) throws DotException,
AtTheRateException, DomainException {
        if (email.startsWith(".") || email.endsWith(".") || email.startsWith("@") ||
email.endsWith("@") || email.contains("..") || email.contains("@@")) {
            throw new DotException("Invalid Dot usage");
        }

        int atCount = email.length() - email.replace("@", "").length();
        if (atCount != 1) {
            throw new AtTheRateException("Invalid @ usage");
        }
```

```
        int atIndex = email.indexOf("@");
        String domainPart = email.substring(atIndex + 1);
        int lastDotIndex = domainPart.lastIndexOf(".");
        if (lastDotIndex == -1 || lastDotIndex == domainPart.length() - 1) {
            throw new DotException("Invalid Dot usage");
        }

        String domain = domainPart.substring(lastDotIndex + 1);
        List<String> validDomains = Arrays.asList("in", "com", "net", "biz");
        if (!validDomains.contains(domain)) {
            throw new DomainException("Invalid Domain");
        }
    }
}

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

        try {
            EmailValidator.validate(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");
        }
        sc.close();
    }
}
```

Status : Partially correct ***Marks : 7.5/10***