Rajalakshmi Engineering College

Name: Lakshitha K
Email: 241801132@rajalakshmi.edu.in
Roll no: 241801132
Phone: 6381920328
Branch: REC
Department: AI & DS - Section 3
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_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a file management system, users are required to provide a valid file name when creating new files. The system enforces specific rules for file names to maintain consistency and avoid potential issues. Your task is to implement a Java program named FileNameValidator that takes user input for a file name and validates it according to the specified rules.

Rules for Valid File Name:

The file name must consist of alphanumeric characters (letters and digits) only.The file name must have a minimum length of 3 characters.

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

Input Format

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

Output Format

The output is displayed in the following format:

If the entered file name meets the specified criteria, the program outputs

"Valid file name"

If the entered file name does not meet the criteria and triggers the InvalidFileNameException, the program outputs

"Error: Invalid file name. It must be alphanumeric and have a minimum length of 3 characters."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: myfile123

Output: Valid file name

Answer

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

// Custom exception class
class InvalidFileNameException extends Exception {
    public InvalidFileNameException(String message) {
        super(message);
    }
}

public class Main {

    // Method to validate the file name
    public static void validateFileName(String fileName) throws
InvalidFileNameException {
        if (fileName.length() < 3 || !fileName.matches("[a-zA-Z0-9]+")) {
            throw new InvalidFileNameException(
```

```
            "Invalid file name. It must be alphanumeric and have a minimum length
of 3 characters."
            );
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String fileName = scanner.nextLine();
        try {
            validateFileName(fileName);
            System.out.println("Valid file name");
        } catch (InvalidFileNameException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Lakshitha K
Email: 241801132@rajalakshmi.edu.in
Roll no: 241801132
Phone: 6381920328
Branch: REC
Department: AI & DS - Section 3
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 exception for invalid dot usage
class DotException extends Exception {
    public DotException(String message) {
        super(message);
    }
}

// Custom exception for invalid '@' usage
class AtTheRateException extends Exception {
    public AtTheRateException(String message) {
        super(message);
    }
}

// Custom exception for invalid domain
class DomainException extends Exception {
    public DomainException(String message) {
        super(message);
    }
}

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

        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");
        }
    }

    public static void validateEmail(String email)
            throws DotException, AtTheRateException, DomainException {

        // Check for exactly one '@'
        int atCount = email.length() - email.replace("@", "").length();
        if (atCount != 1) {
            throw new AtTheRateException("Invalid @ usage");
        }

        // Email should not start or end with '.' or '@'
        if (email.startsWith(".") || email.endsWith(".") ||
            email.startsWith("@") || email.endsWith("@")) {
            throw new DotException("Invalid Dot usage");
        }

        // Consecutive '.' or '@' are not allowed
        if (email.contains("..") || email.contains("@@")) {
            throw new DotException("Invalid Dot usage");
        }

        // There must be a '.' after the '@'
        int atIndex = email.indexOf('@');
        int dotAfterAt = email.indexOf('.', atIndex);
        if (dotAfterAt == -1) {
            throw new DotException("Invalid Dot usage");
        }
```

```
        // Email should not end with '.'
        if (email.endsWith(".")) {
            throw new DotException("Invalid Dot usage");
        }

        // Validate domain extension (part after last '.')
        int lastDot = email.lastIndexOf('.');
        if (lastDot == -1 || lastDot == email.length() - 1) {
            throw new DomainException("Invalid Domain");
        }
        String domain = email.substring(lastDot + 1);
        if (!(domain.equals("in") || domain.equals("com") ||
              domain.equals("net") || domain.equals("biz"))) {
            throw new DomainException("Invalid Domain");
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Lakshitha K
Email: 241801132@rajalakshmi.edu.in
Roll no: 241801132
Phone: 6381920328
Branch: REC
Department: AI & DS - Section 3
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_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 class
class InvalidDurationException extends Exception {
    public InvalidDurationException(String message) {
        super(message);
    }
}

public class Main {
```

```
    // Method to validate the meeting duration
    public static void validateMeetingDuration(int duration) throws
InvalidDurationException {
        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);
        int duration = scanner.nextInt();
        scanner.close();

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

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

Rajalakshmi Engineering College

Name: Lakshitha K
Email: 241801132@rajalakshmi.edu.in
Roll no: 241801132
Phone: 6381920328
Branch: REC
Department: AI & DS - Section 3
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_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 class
class InvalidDurationException extends Exception {
    public InvalidDurationException(String message) {
        super(message);
    }
}

public class Main {
```

```
    // Method to validate the meeting duration
    public static void validateMeetingDuration(int duration) throws
InvalidDurationException {
        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);
        int duration = scanner.nextInt();
        scanner.close();

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

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