Rajalakshmi Engineering College

Name: Deepashri Murali
Email: 240701100@rajalakshmi.edu.in
Roll no: 240701100
Phone: 9566041751
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

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

```
import java.util.*;

class DotException extends Exception { DotException(String m){super(m);} }
class AtTheRateException extends Exception { AtTheRateException(String m)
{super(m);} }
class DomainException extends Exception { DomainException(String m)
{super(m);} }

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

   static void validate(String e) throws DotException, AtTheRateException,
DomainException {
      long atCount = e.chars().filter(ch -> ch == '@').count();
      if (atCount != 1)
         throw new AtTheRateException("Invalid @ usage");
```

```
        if (e.startsWith(".") || e.startsWith("@") || e.endsWith(".") || e.endsWith("@") ||
            e.contains("..") || e.contains("@@"))
            throw new DotException("Invalid Dot usage");

        String[] parts = e.split("@");
        if (parts.length != 2 || !parts[1].contains("."))
            throw new DotException("Invalid Dot usage");

        String domain = parts[1].substring(parts[1].lastIndexOf('.') + 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: Deepashri Murali
Email: 240701100@rajalakshmi.edu.in
Roll no: 240701100
Phone: 9566041751
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

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 {
   InvalidDurationException(String msg) { super(msg); }
}

class ElsaMeetingScheduler {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      try {
         validateMeetingDuration(n);
```

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

    static void validateMeetingDuration(int n) throws InvalidDurationException {
        if (n <= 0 || n > 240)
            throw new InvalidDurationException("Invalid meeting duration. Please
enter a positive integer not exceeding 240 minutes (4 hours).");
    }
}
```

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

Rajalakshmi Engineering College

Name: Deepashri Murali
Email: 240701100@rajalakshmi.edu.in
Roll no: 240701100
Phone: 9566041751
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

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 {
   InvalidUsernameException(String msg) { super(msg); }
}

class UsernameValidator {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      String u = s.nextLine();
      try {
         checkUsername(u);
         System.out.println("Username is valid: " + u);
      } catch (InvalidUsernameException e) {
         System.out.println(e.getMessage());
      }
   }

   static void checkUsername(String u) throws InvalidUsernameException {
      if (u.contains(" "))
```

```
        throw new InvalidUsernameException("Invalid Username: Username
cannot contain spaces");
      if (u.length() < 5)
         throw new InvalidUsernameException("Invalid Username: Username must
be at least 5 characters long");
   }
}
```

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

Rajalakshmi Engineering College

Name: Deepashri Murali
Email: 240701100@rajalakshmi.edu.in
Roll no: 240701100
Phone: 9566041751
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

A local municipality is implementing an online voting system for a community event and wants to ensure that only eligible voters (those aged 18 or older) can participate.

Your task is to develop a program that validates the age of individuals attempting to vote online. If the user's age is below 18, the program should throw a custom exception, InvalidAgeException, preventing them from casting their vote. If the input is invalid, catch the appropriate InputMismatchException and print the in-built exception message.

Input Format

The input consists of an integer representing the age.

Output Format

If the age is 18 or older, print "Eligible to vote"

If the age is below 18, print "Exception occurred: InvalidAgeException: Age is not valid to vote"

If there is any other type of exception, print "An error occurred: " followed by the in-built exception message.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20
Output: Eligible to vote

Answer

```
import java.util.*;

class InvalidAgeException extends Exception {
   InvalidAgeException(String msg) { super(msg); }
}

class VotingSystem {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      try {
         int age = s.nextInt();
         checkAge(age);
         System.out.println("Eligible to vote");
      } catch (InvalidAgeException e) {
         System.out.println("Exception occurred: InvalidAgeException: " +
e.getMessage());
      } catch (InputMismatchException e) {
         System.out.println("An error occurred: " + e);
      }
   }

   static void checkAge(int age) throws InvalidAgeException {
      if (age < 18)
         throw new InvalidAgeException("Age is not valid to vote");
```

```
    }
}
```

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

Rajalakshmi Engineering College

Name: Deepashri Murali
Email: 240701100@rajalakshmi.edu.in
Roll no: 240701100
Phone: 9566041751
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

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

```
import java.util.Scanner;

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

class FileNameValidator {
    public static void validate(String fileName) throws InvalidFileNameException {
        if (!fileName.matches("[A-Za-z0-9]+") || fileName.length() < 3) {
            throw new InvalidFileNameException("Error: Invalid file name. It must be
alphanumeric and have a minimum length of 3 characters.");
        }
    }

    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        String fileName = sc.nextLine();
        try {
            validate(fileName);
            System.out.println("Valid file name");
        } catch (InvalidFileNameException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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