# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

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;

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 {
        String[] validDomains = {"in", "com", "net", "biz"};

        if (email.length() < 5 || email.length() > 50) {
            throw new IllegalArgumentException("Email length out of bounds");
        }

        if (email.indexOf('@') == -1 || email.indexOf('@') != email.lastIndexOf('@') ||
            email.startsWith("@") || email.endsWith("@") || email.contains("@@")) {
            throw new AtTheRateException("AtTheRateException: Invalid @ usage");
        }
```

```
    if (email.endsWith(".") || email.startsWith(".") || email.contains("..")) {
        throw new DotException("DotException: Invalid Dot usage");
    }

    String[] parts = email.split("@");
    String domainPart = parts[1];

    if (!domainPart.contains(".")) {
        throw new DotException("DotException: Invalid Dot usage");
    }

    String[] domainSegments = domainPart.split("\\.");
    if (domainSegments.length < 2) {
        throw new DotException("DotException: Invalid Dot usage");
    }

    String domainExtension = domainSegments[domainSegments.length - 1];
    boolean isValidDomain = false;
    for (String valid : validDomains) {
        if (domainExtension.equals(valid)) {
            isValidDomain = true;
            break;
        }
    }

    if (!isValidDomain) {
        throw new DomainException("DomainException: Invalid Domain");
    }

    System.out.println("Valid email address");
}

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

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

```
        }
    }
}
```

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

# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

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;

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

public class Main {
    public static void validateMeetingDuration(int n) throws
InvalidDurationException {
```

```
        if (n <= 0 || n > 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 n = sc.nextInt();
        try {
            validateMeetingDuration(n);
            System.out.println("Meeting scheduled successfully!");
        } catch (InvalidDurationException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

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

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

```
            System.out.println("Username is valid: " + username);
        }
    }

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

        try {
            validateUsername(username);
        } catch (InvalidUsernameException e) {
            System.out.println(e.getMessage());
        }

        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

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***

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

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

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

        try {
            int age = scanner.nextInt();

            if (age < 18) {
                throw new InvalidAgeException("Age is not valid to vote");
            }

            System.out.println("Eligible to vote");
```

```
        } catch (InputMismatchException e) {
            System.out.println("An error occurred: " + e);
        } catch (InvalidAgeException e) {
            System.out.println("Exception occurred: " + e.getClass().getSimpleName()
+ ": " + e.getMessage());
        }
    }
}
```

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

# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

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;

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

public class Main {

    public static void validateFileName(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 {
            validateFileName(fileName);
            System.out.println("Valid file name");
        } catch (InvalidFileNameException e) {
            System.out.println(e.getMessage());
        }

        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

### REC_2028_OOPS using Java_Week 8_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

### Section 1 : Coding

1. Problem Statement

You are tasked to create a program that defines a custom exception GradeException. The program should include a Student class with fields for the student's name, age, and grade. Implement a method in the Student class that checks the grade, and if the grade is below 40, it should throw a GradeException. Otherwise, it should display the student's details.

***Input Format***

The input consists of three parameters in separate lines:

1. A string representing the student's name.
2. An integer representing the student's age.
3. An integer representing the student's grade.

***Output Format***

The output will display the student's details if the grade is valid.

If the grade is below 40, the program will display an error message "Grade is below 40".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: Alice
20
85

Output: Name: Alice
Age: 20
Grade: 85

***Answer***

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

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

class Student {
    String name;
    int age;
    int grade;

    public Student(String name, int age, int grade) {
        this.name = name;
        this.age = age;
        this.grade = grade;
    }

    public void validateGrade() throws GradeException {
        if (grade < 40) {
            throw new GradeException("Grade is below 40");
```

```
        } else {
            System.out.println("Name: " + name);
            System.out.println("Age: " + age);
            System.out.println("Grade: " + grade);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String name = sc.nextLine();
        int age = sc.nextInt();
        int grade = sc.nextInt();

        Student student = new Student(name, age, grade);

        try {
            student.validateGrade();
        } catch (GradeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

2. Problem Statement

Enigma is developing a simple web application that takes a user-input URL, validates it, and throws a custom exception InvalidURLFormatException if the URL does not start with "http://" or "https://".

The main method prompts the user for input, validates the URL, and prints whether it is valid or not.

***Input Format***

The input consists of a string, representing the URL entered by the user.

***Output Format***

The output displays one of the following results:

If the entered URL is valid according to the specified format, the program prints:

"[URL] is a valid URL"

If the entered URL is not valid according to the specified format, the program prints:

"Invalid URL format: [URL]"

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: http://www.example.com
Output: http://www.example.com is a valid URL

***Answer***

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

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

public class Main {
    public static void validateURL(String url) throws InvalidURLFormatException {
        if (url.startsWith("http://") || url.startsWith("https://")) {
            System.out.println(url + " is a valid URL");
        } else {
            throw new InvalidURLFormatException("Invalid URL format: " + url);
        }
    }
```

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

        try {
            validateURL(url);
        } catch (InvalidURLFormatException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

3. Problem Statement

Daniel is developing a program to verify the age of users. He wants to ensure that the entered age is within a valid range. Write a program to help Daniel implement this age-checking feature using custom exceptions.

Daniel needs a program that takes an integer input representing a person's age.If the age is between 0 and 150 (inclusive), the program should print "Age is valid!".If the age is less than 0 or greater than 150, the program should throw a custom exception (InvalidAgeException) with the message "Invalid age. Please enter an age between 0 and 150."

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

***Input Format***

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

***Output Format***

The output is displayed in the following format:

If the age is valid (between 0 and 150, inclusive), print

"Age is valid!".

If the age is invalid, print

"Error: Invalid age. Please enter an age between 0 and 150."

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 45

Output: Age is valid!

***Answer***

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

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

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

        int age = scanner.nextInt();

        try {
            validateAge(age);
            System.out.println("Age is valid!");
        } catch (InvalidAgeException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void validateAge(int age) throws InvalidAgeException {
        if (age < 0 || age > 150) {
            throw new InvalidAgeException("Invalid age. Please enter an age between
0 and 150.");
        }
    }
```

}

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

4. Problem Statement

An HR software system is being developed to process employee payrolls. During payroll processing, the system must ensure that no employee has a negative salary and that no employee's salary exceeds 2,00,000. If either condition occurs, the system should throw a custom exception.

Create a custom exception InvalidSalaryException and a class Employee that processes salary according to the following rules:

If salary < 0, throw InvalidSalaryException with the message: "Salary cannot be negative". If salary > 200000, throw InvalidSalaryException with the message: "Salary exceeds threshold limit". Otherwise, display: "Salary processed successfully for <empName>: <salary>".

The payroll processing should always display: "Payroll process completed" at the end, regardless of whether an exception occurs.

***Input Format***

The first line of input contains an integer representing the employee ID.

The second line contains a string representing the employee's name.

The third line contains a floating-point number representing the salary of the employee.

***Output Format***

If the salary is valid: "Salary processed successfully for <empName>: <salary>"

"Payroll process completed"

If the salary is invalid: "<Exception Message>"

"Payroll process completed"

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 101
Rahul
150000.0
Output: Salary processed successfully for Rahul: 150000.0
Payroll process completed

***Answer***

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

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

class Employee {
    private int empId;
    private String empName;
    private double salary;

    public Employee(int empId, String empName, double salary) {
        this.empId = empId;
        this.empName = empName;
        this.salary = salary;
    }

    public void processSalary() throws InvalidSalaryException {
        if (salary < 0) {
            throw new InvalidSalaryException("Salary cannot be negative");
        } else if (salary > 200000) {
            throw new InvalidSalaryException("Salary exceeds threshold limit");
        } else {
            System.out.println("Salary processed successfully for " + empName + ": "
+ salary);
        }
    }
```

```
}

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

        int empId = Integer.parseInt(scanner.nextLine());
        String empName = scanner.nextLine();
        double salary = Double.parseDouble(scanner.nextLine());

        Employee employee = new Employee(empId, empName, salary);

        try {
            employee.processSalary();
        } catch (InvalidSalaryException e) {
            System.out.println(e.getMessage());
        } finally {
            System.out.println("Payroll process completed");
        }
    }
}
```

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

# Rajalakshmi Engineering College

Name: sravanthika s
Email: 241001258@rajalakshmi.edu.in
Roll no:
Phone: 7010331084
Branch: REC
Department: IT - Section 4
Batch: 2028
Degree: B.E - IT

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

## REC_2028_OOPS using Java_Week 8_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

## Section 1 : Coding

1. Problem Statement

Tim was tasked with creating a user profile system that validates the user's date of birth input. The system should throw a custom exception, InvalidDateOfBirthException, if the date is not in the specified format "dd-mm-yyyy" or if it represents an invalid calendar date.

The main method takes user input, validates the date of birth, and prints whether it is valid or not.

***Input Format***

The input consists of a string, representing the date of birth of the user.

***Output Format***

The output displays one of the following results:

If the entered date of birth is valid according to the specified format, the program prints:

"[Date] is a valid date of birth"

If the entered date of birth is not valid according to the specified format, the program prints:

"Invalid date: [Date]"

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 01-01-2000

Output: 01-01-2000 is a valid date of birth

***Answer***

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

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

public class Main {
    public static void validateDateOfBirth(String dob) throws
InvalidDateOfBirthException {
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        sdf.setLenient(false); // Strict parsing

        try {
            sdf.parse(dob);
            System.out.println(dob + " is a valid date of birth");
```

```
        } catch (ParseException e) {
            throw new InvalidDateOfBirthException("Invalid date: " + dob);
        }
    }

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

        try {
            validateDateOfBirth(dob);
        } catch (InvalidDateOfBirthException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

2. Problem Statement

A company is developing a user registration system that requires users to provide valid email addresses. The development team is implementing an EmailValidator program to ensure that the entered email addresses meet certain criteria using exception handling.

The email address must contain the "@" symbol.The email address must consist of a non-empty username(before "@" symbol) and a non-empty domain(after "'@" symbol).The domain part of the email address must contain at least one period (".").The email address must not contain leading or trailing spaces.

Implement a custom exception, InvalidEmailException, to fulfill the company's requirements and validate it according to the specified rules.

***Input Format***

The input consists of a string value 's', which represents the email address.

***Output Format***

The output is displayed in the following format:

If the entered email address is valid according to the specified rules, the program prints:

"Email address is valid!"

If the entered email address misses the username or domain part or misses "@" symbol or has two or more "@" symbols or misses '.' in the domain part it outputs:

"Error: Invalid email format."

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: johndoe@example.com

Output: Email address is valid!

***Answer***

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

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

public class Main {
    public static void validateEmail(String email) throws InvalidEmailException {
        email = email.trim();

        int atCount = email.length() - email.replace("@", "").length();
        if (atCount != 1) {
            throw new InvalidEmailException("Error: Invalid email format.");
        }

        String[] parts = email.split("@");
        if (parts.length != 2 || parts[0].isEmpty() || parts[1].isEmpty()) {
```

```
            throw new InvalidEmailException("Error: Invalid email format.");
        }

        if (!parts[1].contains(".")) {
            throw new InvalidEmailException("Error: Invalid email format.");
        }

        System.out.println("Email address is valid!");
    }

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

        try {
            validateEmail(email);
        } catch (InvalidEmailException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

3. Problem Statement

Alice is designing a program that requires users to enter positive numbers. She wants to implement a solution that validates whether the entered number is positive. In case the input is not a positive number, she wants to throw a custom exception.

The number should be a positive integer.If this condition is violated, the program should throw a custom exception:InvalidPositiveNumberException with the message "Invalid input. Please enter a positive integer."

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

***Input Format***

The input consists of an integer value 'n', representing the entered number.

***Output Format***

The output is displayed in the following format:

If the validation passes, print

"Number {number} is positive."

The {number} represents the entered positive integer.

If the entered number is negative then it displays

"Error: Invalid input. Please enter a positive integer."

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 100
Output: Number 100 is positive.

***Answer***

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

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

public class Main {
    public static void validateNumber(int number) throws
InvalidPositiveNumberException {
        if (number <= 0) {
            throw new InvalidPositiveNumberException("Invalid input. Please enter a
positive integer.");
```

```
        } else {
            System.out.println("Number " + number + " is positive.");
        }
    }

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

        try {
            validateNumber(number);
        } catch (InvalidPositiveNumberException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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

4. Problem Statement

Camila, a user of a social media platform, is looking to change her password to enhance account security. The platform enforces specific rules for password strength to ensure the safety of user accounts. Camila needs a program that prompts her to enter a new password and throws custom exceptions based on the strength of the password.

Password Strength Criteria:

Weak Password:

Length less than 8 characters.Medium Password:

Length 8 or more characters.Missing a mix of uppercase letters, lowercase letters, and digits.

Implement a custom exception, to assist Camila in changing her password securely. The program should interactively take user input for a new password, categorize its strength, and handle custom exceptions (WeakPasswordException and MediumPasswordException) if the password fails to meet the specified criteria.

***Input Format***

The input consists of a string s, representing the new password.

***Output Format***

The output is displayed in the following format:

If the entered password meets the strength criteria, the program outputs

"Password changed successfully!"

If the entered password is weak, the program outputs

"Error: Weak password. It must be at least 8 characters long."

If the entered password is of medium strength, the program outputs

"Error: Medium password. It must include a mix of uppercase letters, lowercase letters, and digits."

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: ComplexP@ss1
Output: Password changed successfully!

***Answer***

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

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

class MediumPasswordException extends Exception {
    public MediumPasswordException(String message) {
        super(message);
```

```
    }
}

public class Main {

    public static void validatePassword(String password) throws
WeakPasswordException, MediumPasswordException {
        if (password.length() < 8) {
            throw new WeakPasswordException("Error: Weak password. It must be at
least 8 characters long.");
        }

        boolean hasUpper = false;
        boolean hasLower = false;
        boolean hasDigit = false;

        for (char ch : password.toCharArray()) {
            if (Character.isUpperCase(ch)) hasUpper = true;
            else if (Character.isLowerCase(ch)) hasLower = true;
            else if (Character.isDigit(ch)) hasDigit = true;
        }

        if (!(hasUpper && hasLower && hasDigit)) {
            throw new MediumPasswordException("Error: Medium password. It must
include a mix of uppercase letters, lowercase letters, and digits.");
        }

        System.out.println("Password changed successfully!");
    }

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

        try {
            validatePassword(password);
        } catch (WeakPasswordException | MediumPasswordException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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