Rajalakshmi Engineering College

Name: Arilli Jagadeesh A
Email: 241901502@rajalakshmi.edu.in
Roll no: 241901502
Phone: 9361250488
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

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

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

```
import java.util.Scanner;

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

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int number = Integer.parseInt(scanner.nextLine().trim());

        try {
```

```
            if (number <= 0) {
                throw new InvalidPositiveNumberException("Invalid input. Please enter
a positive integer.");
            } else {
                System.out.println("Number " + number + " is positive.");
            }
        } catch (InvalidPositiveNumberException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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

2. 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

```
import java.util.*;
import java.text.*;

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

class DateValidator {
    public static void validateDate(String dateStr) throws
InvalidDateOfBirthException {
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        sdf.setLenient(false); // ensures strict date validation
        try {
            Date date = sdf.parse(dateStr);
        } catch (ParseException e) {
            throw new InvalidDateOfBirthException("Invalid date: " + dateStr);
        }
    }
}

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

        try {
            DateValidator.validateDate(dateStr);
            System.out.println(dateStr + " is a valid date of birth");
        } catch (InvalidDateOfBirthException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

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

3. Problem Statement

In an online shopping cart system, users can apply coupon codes during checkout to avail of discounts. However, to ensure the validity and security of coupon codes, the system enforces specific rules for their format. Your task is to implement a Java program named CouponCodeValidator that takes user input for a coupon code and validates it according to the specified rules.

Rules for Valid Coupon Code:

The coupon code must consist of exactly 10 characters.The coupon code must contain at least one alphabet (uppercase or lowercase) and at least one digit (0-9).Special characters are not allowed in the coupon code.

Implement a custom exception, InvalidCouponException, to handle cases where the entered coupon code does not meet the specified criteria.

Input Format

The input consists of a string s, representing the coupon code.

Output Format

The output is displayed in the following format:

If the entered coupon code meets the specified criteria, the program outputs

"Coupon code applied successfully!"

If the entered coupon code has less than or more than 10 characters it outputs

"Error: Invalid coupon code length. It must be exactly 10 characters."

If the entered coupon code contains only numeric or only alphabets it outputs

"Error: Invalid coupon code format. It must contain at least one alphabet and one digit."

If the entered coupon code contains special characters it outputs

"Error: Coupon code should not contain special characters."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: ABCD123456
Output: Coupon code applied successfully!

Answer

```
import java.util.*;

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

class CouponCodeValidator {
   public static void validateCoupon(String code) throws InvalidCouponException
{
      if (code.length() != 10) {
         throw new InvalidCouponException("Error: Invalid coupon code length. It
must be exactly 10 characters.");
      }

      boolean hasAlpha = false, hasDigit = false, hasSpecial = false;

      for (char ch : code.toCharArray()) {
         if (Character.isLetter(ch)) {
            hasAlpha = true;
         } else if (Character.isDigit(ch)) {
            hasDigit = true;
         } else {
```

```
            hasSpecial = true;
        }
    }

    if (hasSpecial) {
        throw new InvalidCouponException("Error: Coupon code should not
contain special characters.");
    }

    if (!(hasAlpha && hasDigit)) {
        throw new InvalidCouponException("Error: Invalid coupon code format. It
must contain at least one alphabet and one digit.");
    }
  }
}

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

    try {
        CouponCodeValidator.validateCoupon(code);
        System.out.println("Coupon code applied successfully!");
    } catch (InvalidCouponException e) {
        System.out.println(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

```
import java.util.*;

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

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

class PasswordValidator {
    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, hasLower = false, hasDigit = false;

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

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

```
        sc.close();

        try {
            PasswordValidator.validatePassword(password);
            System.out.println("Password changed successfully!");
        } catch (WeakPasswordException | MediumPasswordException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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