Rajalakshmi Engineering College

Name: Shreenidhi T
Email: 240701503@rajalakshmi.edu.in
Roll no: 240701503
Phone: 9150942326
Branch: REC
Department: CSE - Section 10
Batch: 2028
Degree: B.E - CSE

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

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

        int n = sc.nextInt();

        try {
            if (n <= 0) {
                throw new InvalidPositiveNumberException("Invalid input. Please enter
a positive integer.");
            }


            System.out.println("Number " + n + " is positive.");
        }
        catch (InvalidPositiveNumberException e) {

            System.out.println("Error: " + e.getMessage());
        }

        sc.close();
    }
}
```

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

2. Problem Statement

Theo is trying to update his payment information on a subscription-based streaming service. To proceed, the system requires Theo to provide a valid credit card number consisting of 16 digits. However, Theo wants to make sure that the credit card number he enters meets the specified criteria with proper exception handling.

The credit card number must consist of exactly 16 digits.If the entered credit card number does not meet the specified criteria, the program should throw a custom exception, InvalidCreditCardException, and provide Theo with specific error messages:If the length of the credit card number is not 16 digits, the exception message should be: "Invalid credit card number length."If the credit card number contains non-numeric characters, the exception message should be: "Invalid credit card number format."

Implement a custom exception, InvalidCreditCardException, to fulfill Theo's requirements and keep his payment information secure.

Input Format

The input consists of a string value 's', consisting of the 16-digit credit card number.

Output Format

The output is displayed in the following format:

If the entered credit card number is valid, the program should output a success message:

"Payment information updated successfully!"

If the entered credit card has more than 16 digits or less than 16 digits it displays

"Error: Invalid credit card number length."

If the entered 16-digit credit card has non-integers it displays

"Error: Invalid credit card number format."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1234567890123456

Output: Payment information updated successfully!

Answer

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

class InvalidCreditCardException extends Exception {
   public InvalidCreditCardException(String message) {
      super(message);
   }
}
 class CreditCardValidator {
   public static void validateCreditCardNumber(String cardNumber) throws
InvalidCreditCardException {
      if (cardNumber.length() != 16) {
          throw new InvalidCreditCardException("Error: Invalid credit card number
length.");
      }
      if (!cardNumber.matches("\\d{16}")) {
          throw new InvalidCreditCardException("Error: Invalid credit card number
format.");
      }
   }

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

      try {
          validateCreditCardNumber(cardNumber);
          System.out.println("Payment information updated successfully!");
      } catch (InvalidCreditCardException e) {
          System.out.println(e.getMessage());
      }
   }
}
```

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

3. 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.util.Scanner;
import java.text.SimpleDateFormat;
import java.text.ParseException;

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

 class DateOfBirthValidator {
   public static void validateDateOfBirth(String dob) throws
InvalidDateOfBirthException {
       if (dob.length() != 10) {
           throw new InvalidDateOfBirthException("Invalid date: " + dob);
       }

       SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
       sdf.setLenient(false);
       try {
           sdf.parse(dob);
       } 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();
       scanner.close();

       try {
           validateDateOfBirth(dob);
           System.out.println(dob + " is a valid date of birth");
       } catch (InvalidDateOfBirthException e) {
           System.out.println(e.getMessage());
       }
   }
}
```

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

4. 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

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

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

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

        boolean hasAlphabet = coupon.matches(".*[A-Za-z].*");
        boolean hasDigit = coupon.matches(".*\\d.*");
        boolean hasSpecialChar = !coupon.matches("[A-Za-z0-9]+");

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

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

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

        try {
            validateCouponCode(coupon);
            System.out.println("Coupon code applied successfully!");
        } catch (InvalidCouponException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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