# Rajalakshmi Engineering College

Name: Jesvanth Sabarish V K
Email: 240701214@rajalakshmi.edu.in
Roll no: 240701214
Phone: 9080128264
Branch: REC
Department: CSE - Section 6
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

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;

// Custom Exception
class InvalidCreditCardException extends Exception {
  public InvalidCreditCardException(String message) {
      super(message);
```

```
    }
}

class CreditCardValidator {

    public static void validateCard(String card) throws InvalidCreditCardException
{

        // Rule 1: Length must be exactly 16
        if (card.length() != 16) {
            throw new InvalidCreditCardException("Invalid credit card number
length.");
        }

        // Rule 2: Must be numeric only
        if (!card.matches("[0-9]+")) {
            throw new InvalidCreditCardException("Invalid credit card number
format.");
        }
    }

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

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

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

2. 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.*;

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

class CouponCodeValidator {

   public static void validateCoupon(String code) throws InvalidCouponException
{

      // Rule 1: Length must be exactly 10
      if (code.length() != 10) {
         throw new InvalidCouponException(
            "Error: Invalid coupon code length. It must be exactly 10 characters."
         );
      }

      // Check for special characters (only letters + digits allowed)
      if (!code.matches("[A-Za-z0-9]+")) {
         throw new InvalidCouponException(
            "Error: Coupon code should not contain special characters."
         );
      }

      // Must contain at least 1 alphabet and 1 digit
      boolean hasAlpha = code.matches(".*[A-Za-z].*");
      boolean hasDigit = code.matches(".*[0-9].*");

      if (!hasAlpha || !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 code = scanner.nextLine();

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

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

public class Main {

   public static void validate(int n) throws InvalidPositiveNumberException {
      if (n <= 0) {
         throw new InvalidPositiveNumberException(
            "Error: Invalid input. Please enter a positive integer."
         );
      }
   }

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

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

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

4. 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.*;

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

public class Main {

   public static void validateEmail(String email) throws InvalidEmailException {

      // Rule: no leading/trailing spaces
      if (!email.equals(email.trim())) {
         throw new InvalidEmailException("Error: Invalid email format.");
      }

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

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

      // Must have non-empty username and domain
      if (parts.length != 2 || parts[0].isEmpty() || parts[1].isEmpty()) {
         throw new InvalidEmailException("Error: Invalid email format.");
```

```
        }

        String domain = parts[1];

        // Domain must contain at least one '.'
        if (!domain.contains(".")) {
            throw new InvalidEmailException("Error: Invalid email format.");
        }
    }

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

        try {
            validateEmail(email);
            System.out.println("Email address is valid!");
        } catch (InvalidEmailException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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