Rajalakshmi Engineering College

Name: UMAR FAROOK
Email: 240701573@rajalakshmi.edu.in
Roll no: 240701573
Phone: 9791730398
Branch: REC
Department: CSE - Section 5
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 {
            validatePositiveNumber(n);
```

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

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

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

```
import java.util.Scanner;

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

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

```
            validateCoupon(s);
            System.out.println("Coupon code applied successfully!");
        } catch (InvalidCouponException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void validateCoupon(String s) throws InvalidCouponException {
        if (s.length() != 10) {
            throw new InvalidCouponException("Invalid coupon code length. It must
be exactly 10 characters.");
        }
        boolean hasLetter = false;
        boolean hasDigit = false;
        boolean hasSpecial = false;
        for (char c : s.toCharArray()) {
            if (Character.isLetter(c)) hasLetter = true;
            else if (Character.isDigit(c)) hasDigit = true;
            else {
                hasSpecial = true;
                break;
            }
        }
        if (hasSpecial) {
            throw new InvalidCouponException("Coupon code should not contain
special characters.");
        } else if (!hasLetter || !hasDigit) {
            throw new InvalidCouponException("Invalid coupon code format. It must
contain at least one alphabet and one digit.");
        }
    }
}
```

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

3. Problem Statement

Faustus is managing his bank account and wants to create a program to update his account balance based on certain conditions. However, he needs to handle specific scenarios related to invalid inputs and insufficient balances. Faustus wants to update his account balance. He inputs the

current balance and the amount to be updated.

The initial account balance should be positive. If Faustus enters a negative initial balance, the program should throw an InvalidAmountException with the message "Invalid amount. Please enter a positive initial balance."If the amount to be updated is negative, the program should check if the subtraction results in a negative balance. If so, it should throw an InsufficientBalanceException with the message "Insufficient balance."If the amount to be updated is positive, it should be added to the current balance, and the new balance should be printed.

Implement a custom exception, InvalidAmountException, and InsufficientBalanceException, to manage his bank account.

Input Format

The first line of input consists of a double value 'd', representing the initial account balance.

The second line of input consists of a double value 'd1', representing the amount to be updated.

Output Format

The output is displayed in the following format:

If the validation passes, print

"Account balance updated successfully! New balance: {new_balance}"

where {new_balance} is the updated account balance.

If the initial bank amount is negative it displays

"Error: Invalid amount. Please enter a positive initial balance."

If the updated amount exceeds the initial account balance in withdrawal it displays

"Error: Insufficient balance."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1000
500
Output: Account balance updated successfully! New balance: 1500.0

Answer

```
import java.util.Scanner;

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

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

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

        try {
            if (balance < 0) {
                throw new InvalidAmountException("Invalid amount. Please enter a
positive initial balance.");
            }

            if (amount < 0 && balance + amount < 0) {
                throw new InsufficientBalanceException("Insufficient balance.");
            }

            balance += amount;
            System.out.println("Account balance updated successfully! New balance:
" + balance);
```

```
        } catch (InvalidAmountException | InsufficientBalanceException e) {
            System.out.println("Error: " + 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

```
import java.util.Scanner;

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

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

        try {
            if (email.startsWith(" ") || email.endsWith(" "))
                throw new InvalidEmailException("Invalid email format.");

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

            if (parts.length != 2 || parts[0].isEmpty() || parts[1].isEmpty())
                throw new InvalidEmailException("Invalid email format.");

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

            System.out.println("Email address is valid!");
        } catch (InvalidEmailException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
```

}

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