Rajalakshmi Engineering College

Name: Anbarasu V
Email: 241501018@rajalakshmi.edu.in
Roll no: 241501018
Phone: 9488440199
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

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

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);
   }
}

class EmailValidator {
   public static void validate(String email) throws InvalidEmailException {
      if (email == null || email.trim().isEmpty() || !email.equals(email.trim())) {
         throw new InvalidEmailException("Error: Invalid email format.");
      }
```

```
        int atCount = 0;
        for (char c : email.toCharArray()) {
            if (c == '@') atCount++;
        }

        if (atCount != 1) {
            throw new InvalidEmailException("Error: Invalid email format.");
        }

        String[] parts = email.split("@");
        if (parts.length != 2 || parts[0].isEmpty() || parts[1].isEmpty() || !
parts[1].contains(".")) {
            throw new InvalidEmailException("Error: Invalid email format.");
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String email = sc.nextLine();
        try {
            EmailValidator.validate(email);
            System.out.println("Email address is valid!");
        } catch (InvalidEmailException e) {
            System.out.println(e.getMessage());
        }
        sc.close();
    }
}
```

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

2. Problem Statement

Hemanth is designing a banking system for XYZ Bank. The system should allow customers to perform deposit, withdrawal, and balance inquiry operations. Implement exception handling for scenarios involving invalid transaction amounts or insufficient funds.

Create two custom exception classes, InvalidAmountException and InsufficientFundsException, both extending the Exception class.Throw an InvalidAmountException with a message if the deposit amount is less than or equal to zero.Throw an InsufficientFundsException if the withdrawal amount is greater than the available balance.Deduct the withdrawal amount from the balance if the withdrawal is successful.

Assist Hemanth in designing the program.

Input Format

The first line of input consists of a double value B, representing the initial balance.

The second line consists of a double value D, representing the deposit amount.

The third line consists of a double value W, representing the withdrawal amount.

Output Format

If the withdrawal is successful, print the amount withdrawn and the current balance, rounded off to one decimal place.

If an InvalidAmountException occurs, print "Error: [D] is not valid".

If an InsufficientFundsException occurs, print "Error: Insufficient funds".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1050.1
270.2
150.3

Output: Amount Withdrawn: 150.3
Current Balance: 1170.0

Answer

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

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

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

class BankAccount {
   private double balance;

   public BankAccount(double balance) {
      this.balance = balance;
   }

   public void deposit(double amount) throws InvalidAmountException {
      if (amount <= 0) {
         throw new InvalidAmountException("Error: " + amount + " is not valid");
      }
      balance += amount;
   }

   public void withdraw(double amount) throws InsufficientFundsException {
      if (amount > balance) {
         throw new InsufficientFundsException("Error: Insufficient funds");
      }
      balance -= amount;
      System.out.printf("Amount Withdrawn: %.1f Current Balance: %.1f\n",
amount, balance);
   }
}

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

```
        double W = sc.nextDouble();
        sc.close();

        BankAccount account = new BankAccount(B);

        try {
            account.deposit(D);
            account.withdraw(W);
        } catch (InvalidAmountException | InsufficientFundsException 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

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

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

public class Main {
   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 hasAlphabet = false;
        boolean hasDigit = false;

        for (int i = 0; i < code.length(); i++) {
            char ch = code.charAt(i);
            if (Character.isLetter(ch)) {
                hasAlphabet = true;
            } else if (Character.isDigit(ch)) {
                hasDigit = true;
            } else {
                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 sc = new Scanner(System.in);
        String code = sc.next();
        sc.close();

        try {
            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

```
// 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 checkPassword(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.");
```

```
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String password = sc.next();
        sc.close();
        try {
            checkPassword(password);
            System.out.println("Password changed successfully!");
        } catch (WeakPasswordException | MediumPasswordException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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