Rajalakshmi Engineering College

Name: naveenraj s
Email: 240701350@rajalakshmi.edu.in
Roll no: 240701350
Phone: 6379711376
Branch: REC
Department: CSE - Section 3
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

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If the withdrawal is successful, print the amount withdrawn and the current balance, rounded off to one decimal place.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Output: Amount Withdrawn: 150.3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

```
import java.util.Scanner;

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

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

```
}

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

        double balance = sc.nextDouble();
        double deposit = sc.nextDouble();
        double withdraw = sc.nextDouble();
        sc.close();

        try {
            // Deposit validation
            if (deposit <= 0) {
                throw new InvalidAmountException(deposit + " is not valid");
            }
            balance += deposit;

            // Withdrawal validation
            if (withdraw > balance) {
                throw new InsufficientFundsException("Insufficient funds");
            }

            // Successful withdrawal
            balance -= withdraw;
            System.out.printf("Amount Withdrawn: %.1f Current Balance: %.1f",
withdraw, balance);

        } catch (InvalidAmountException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (InsufficientFundsException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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

2. 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;

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

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

        try {
            validateEmail(email);
            System.out.println("Email address is valid!");
        } catch (InvalidEmailException e) {
            System.out.println("Error: Invalid email format.");
        }
    }

    public static void validateEmail(String email) throws InvalidEmailException {
        // Check leading/trailing spaces
        if (!email.equals(email.trim())) {
            throw new InvalidEmailException("Invalid format");
        }

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

        // Split username and domain
        int atIndex = email.indexOf('@');
        String username = email.substring(0, atIndex);
        String domain = email.substring(atIndex + 1);

        // Username and domain must be non-empty
        if (username.isEmpty() || domain.isEmpty()) {
```

```
            throw new InvalidEmailException("Invalid format");
        }

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

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;

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

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

        try {
            validateDOB(dob);
            System.out.println(dob + " is a valid date of birth");
        } catch (InvalidDateOfBirthException e) {
            System.out.println("Invalid date: " + dob);
        }
    }

    public static void validateDOB(String dob) throws InvalidDateOfBirthException
{
        // Define date format
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        sdf.setLenient(false); // Strict parsing to reject invalid dates

        try {
            sdf.parse(dob);
```

```
        } catch (ParseException e) {
            throw new InvalidDateOfBirthException("Invalid date format");
        }
    }
}
```

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

4. 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

```
// You are using Java
import java.util.Scanner;
class InvalidAmountException extends Exception{
    InvalidAmountException(){
        super("Error:Invalid amount. Please enter a positive initial balance.");
    }
}
class InsufficientBalanceException extends Exception{
    InsufficientBalanceException(){
        super("Error:Insufficient balance.");
    }
}
class main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        double d,d1;
        try{
```

```
            d=sc.nextDouble();
            d1=sc.nextDouble();
            double b = d+d1;
            if(d<0)
             throw new InvalidAmountException();
            else if(b<0)
             throw new InsufficientBalanceException();
            else
             System.out.printf("Account balance updated successfully! new balance:
%.1f",b);
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

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