Rajalakshmi Engineering College

Name: Saranya Vimalanathan
Email: 240701620@rajalakshmi.edu.in
Roll no: 240701620
Phone: 9789689339
Branch: REC
Department: CSE - Section 9
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

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;

// Custom exception class
class InvalidCouponException extends Exception {
    public InvalidCouponException(String message) {
        super(message);
    }
```

```
}

public class Main  {

    // Method to validate the coupon code
    public static void validateCouponCode(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 (char ch : code.toCharArray()) {
            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.");
        }

        System.out.println("Coupon code applied successfully!");
    }

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

        try {
            validateCouponCode(couponCode);
        } catch (InvalidCouponException e) {
            System.out.println(e.getMessage());
```

```
    }
  }
}
```

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

2. 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
// You are using Java
import java.util.Scanner;
import java.util.InputMismatchException;

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);
 String errorMessage = "Error: Invalid input. Please enter a positive integer.";
 try {
 int n = sc.nextInt();
 if (n <= 0) {
 throw new InvalidPositiveNumberException(errorMessage);
 }
 System.out.println("Number " + n + " is positive.");
 } catch (InvalidPositiveNumberException e) {
 System.out.println(e.getMessage());
 } catch (InputMismatchException e) {
 System.out.println(errorMessage);
 }
 }
}
```

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

3. 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
// 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);

 }
}
public class Main {
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 double balance = sc.nextDouble();
 double depositAmount = sc.nextDouble();
 double withdrawalAmount = sc.nextDouble();
 try {
 if (depositAmount <= 0) {
 throw new InvalidAmountException(depositAmount + " is not valid");
 }
 balance += depositAmount;
 if (withdrawalAmount > balance) {
 throw new InsufficientFundsException("Insufficient funds");
 }
 balance -= withdrawalAmount;
 System.out.println("Amount Withdrawn: " + withdrawalAmount);
 System.out.printf("Current Balance: %.1f\n", balance);
 } catch (InvalidAmountException e) {
 System.out.println("Error: " + e.getMessage());
 } catch (InsufficientFundsException e) {
 System.out.println("Error: " + e.getMessage());
 }
 }
}
```

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;

// Custom exception for invalid initial balance
class InvalidAmountException extends Exception {
    public InvalidAmountException(String message) {
        super(message);
    }
}

// Custom exception for insufficient balance during withdrawal
class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String message) {
        super(message);
    }
}

public class Main {

    public static void updateBalance(double initialBalance, double updateAmount)
            throws InvalidAmountException, InsufficientBalanceException {

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

```
        double newBalance;

        if (updateAmount < 0) {
            if (initialBalance + updateAmount < 0) {
                throw new InsufficientBalanceException("Error: Insufficient balance.");
            } else {
                newBalance = initialBalance + updateAmount;
            }
        } else {
            newBalance = initialBalance + updateAmount;
        }

        System.out.println("Account balance updated successfully! New balance: " +
newBalance);
    }

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

        double initialBalance = scanner.nextDouble();
        double updateAmount = scanner.nextDouble();

        try {
            updateBalance(initialBalance, updateAmount);
        } catch (InvalidAmountException | InsufficientBalanceException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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