Rajalakshmi Engineering College

Name: SONASREE RP
Email: 240701521@rajalakshmi.edu.in
Roll no: 240701521
Phone: 7305340666
Branch: REC
Department: CSE - Section 10
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

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

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

   public static void checkPositive(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

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.*;

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 main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String password = sc.nextLine();
      try {
         validatePassword(password);
         System.out.println("Password changed successfully!");
      } catch (WeakPasswordException e) {
         System.out.println("Error: " + e.getMessage());
      } catch (MediumPasswordException e) {
         System.out.println("Error: " + e.getMessage());
      }
      sc.close();
   }

   public static void validatePassword(String password) throws
WeakPasswordException, MediumPasswordException {
      if (password.length() < 8)
         throw new WeakPasswordException("Weak password. It must be at least
8 characters long.");
      boolean hasUpper = false, hasLower = false, hasDigit = false;
      for (char c : password.toCharArray()) {
         if (Character.isUpperCase(c)) hasUpper = true;
         else if (Character.isLowerCase(c)) hasLower = true;
         else if (Character.isDigit(c)) hasDigit = true;
      }
      if (!(hasUpper && hasLower && hasDigit))
         throw new MediumPasswordException("Medium password. It must
include a mix of uppercase letters, lowercase letters, and digits.");
   }
}
```

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

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

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

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

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double balance = sc.nextDouble();
      double amount = sc.nextDouble();
      try {
         updateBalance(balance, amount);
      } catch (InvalidAmountException e) {
         System.out.println("Error: " + e.getMessage());
      } catch (InsufficientBalanceException e) {
         System.out.println("Error: " + e.getMessage());
      }
      sc.close();
   }

   public static void updateBalance(double balance, double amount) throws
InvalidAmountException, InsufficientBalanceException {
      if (balance < 0)
```

```
          throw new InvalidAmountException("Invalid amount. Please enter a
positive initial balance.");
        double newBalance = balance + amount;
        if (amount < 0 && newBalance < 0)
            throw new InsufficientBalanceException("Insufficient balance.");
        System.out.println("Account balance updated successfully! New balance: " +
newBalance);
    }
}
```

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

4. 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.*;

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 deposit = sc.nextDouble();
      double withdraw = sc.nextDouble();
      try {
         if (deposit <= 0)
            throw new InvalidAmountException(deposit + " is not valid");
         balance += deposit;
```

```
            if (withdraw > balance)
                throw new InsufficientFundsException("Insufficient funds");
            balance -= withdraw;
            System.out.printf("Amount Withdrawn: %.1f\n", withdraw);
            System.out.printf("Current Balance: %.1f", balance);
        } catch (InvalidAmountException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (InsufficientFundsException e) {
            System.out.println("Error: " + e.getMessage());
        }
        sc.close();
    }
}
```

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