# Rajalakshmi Engineering College

Name: RamRohith Muthusamy
Email: 240701419@rajalakshmi.edu.in
Roll no: 240701419
Phone: 8248706370
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

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);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double initialBalance = scanner.nextDouble();
        double updateAmount = scanner.nextDouble();

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

            double newBalance = initialBalance + updateAmount;

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

            System.out.println("Account balance updated successfully! New balance:
" + newBalance);
        } catch (InvalidAmountException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (InsufficientBalanceException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

***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*

```
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 validatePassword(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 scanner = new Scanner(System.in);
        String password = scanner.nextLine();

        try {
            validatePassword(password);
            System.out.println("Password changed successfully!");
        } catch (WeakPasswordException | MediumPasswordException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

***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*

```
import java.util.Scanner;
import java.text.SimpleDateFormat;
import java.text.ParseException;

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

public class Main {
    public static void validateDate(String date) throws 
InvalidDateOfBirthException {
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        sdf.setLenient(false);
        try {
            sdf.parse(date);
        } catch (ParseException e) {
            throw new InvalidDateOfBirthException("Invalid date: " + date);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String date = scanner.nextLine();
        try {
            validateDate(date);
            System.out.println(date + " is a valid date of birth");
        } catch (InvalidDateOfBirthException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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

4. 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);
   }
}

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 hasAlpha = false;
      boolean hasDigit = false;
```

```
        for (char ch : code.toCharArray()) {
            if (Character.isLetter(ch)) {
                hasAlpha = true;
            } else if (Character.isDigit(ch)) {
                hasDigit = true;
            } else {
                throw new InvalidCouponException("Error: Coupon code should not
contain special characters.");
            }
        }

        if (!hasAlpha || !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 scanner = new Scanner(System.in);
        String couponCode = scanner.nextLine();

        try {
            validateCoupon(couponCode);
            System.out.println("Coupon code applied successfully!");
        } catch (InvalidCouponException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

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