Rajalakshmi Engineering College

Name: Aldrine Linjoe.s
Email: 240701033@rajalakshmi.edu.in
Roll no: 240701033
Phone: 7092049029
Branch: REC
Department: CSE - Section 10
Batch: 2028
Degree: B.E - CSE

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

```java
import java.util.*;
class InvalidCouponException extends Exception{
    public InvalidCouponException(String m){
        super(m);
    }
}
class Main{
    public static void main(String[] args){
```

```
Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
try{
    int c=0,p=0,h=0;
    for(int i=0;i<s.length();i++){
        if(Character.isLetter(s.charAt(i))){
            c++;
        }else if(Character.isDigit(s.charAt(i))){
            p++;
        }else{
            h++;
        }
    }
    if(s.length()!=10){
        throw new InvalidCouponException("Error: Invalid coupon code length.it
must be exactly 10 characters.");
    }else if(h!=0){
        throw new InvalidCouponException("Error: Coupon code should not
contain special characters.");
    }else if(!(c>=1 && p>=1)){
        throw new InvalidCouponException("Error: Invalid coupon code format.
It must contain at least one alphabet and one digit.");
    }else{
        System.out.print("Coupon code applied successfully!");
    }
}catch(InvalidCouponException e){
    System.out.println(e.getMessage());
}
}
}
```

Status : Correct   *Marks : 10/10*

2. Problem Statement

Theo is trying to update his payment information on a subscription-based
streaming service. To proceed, the system requires Theo to provide a valid
credit card number consisting of 16 digits. However, Theo wants to make
sure that the credit card number he enters meets the specified criteria with
proper exception handling.

The credit card number must consist of exactly 16 digits.If the entered credit card number does not meet the specified criteria, the program should throw a custom exception, InvalidCreditCardException, and provide Theo with specific error messages:If the length of the credit card number is not 16 digits, the exception message should be: "Invalid credit card number length."If the credit card number contains non-numeric characters, the exception message should be: "Invalid credit card number format."

Implement a custom exception, InvalidCreditCardException, to fulfill Theo's requirements and keep his payment information secure.

Input Format

The input consists of a string value 's', consisting of the 16-digit credit card number.

Output Format

The output is displayed in the following format:

If the entered credit card number is valid, the program should output a success message:

"Payment information updated successfully!"

If the entered credit card has more than 16 digits or less than 16 digits it displays

"Error: Invalid credit card number length."

If the entered 16-digit credit card has non-integers it displays

"Error: Invalid credit card number format."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1234567890123456
Output: Payment information updated successfully!

Answer

```java
import java.util.*;
class InvalidCreditCardException extends Exception{
    public InvalidCreditCardException(String m){
        super(m);
    }
}
class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        String n=sc.nextLine();
        try{
            int c=0;
            for(int i=0;i<n.length();i++){
                if(!Character.isDigit(n.charAt(i))){
                    c++;
                }
            }
            if(n.length()!=16){
                throw new InvalidCreditCardException("Error: Invalid credit card number length.");
            }else if(c!=0){
                throw new InvalidCreditCardException("Error: Invalid credit card number format.");
            }else{
                System.out.println("Payment information updated successfully!");
            }
        }catch(InvalidCreditCardException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Status : Correct   *Marks : 10/10*

3. 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

```java
import java.util.*;
class WeakPasswordException extends Exception{
  public WeakPasswordException(String m){
    super(m);
  }
}
class MediumPasswordException extends Exception{
  public MediumPasswordException(String m){
    super(m);
  }
}
class Main{
  public static void main(String[] args){
    Scanner sc=new Scanner(System.in);
    String s=sc.nextLine();
    try{
      int u=0,l=0,d=0;
      for(int i=0;i<s.length();i++){
        if(Character.isUpperCase(s.charAt(i))){
          u++;
        }else if(Character.isLowerCase(s.charAt(i))){
          l++;
        }else if(Character.isDigit(s.charAt(i))){
          d++;
        }
      }
      if(s.length()<8){
        throw new WeakPasswordException("Error:Weak password. It must be
at least 8 characters long.");
      }else if(!(u>=1 && l>=1 && d>=1)){
        throw new MediumPasswordException("Error: Medium password. It
must include a mix of uppercase letters, lowercase letters, and digits.");
      }else{
        System.out.println("Password changed successfully!");
      }
    }catch(WeakPasswordException e){
      System.out.println(e.getMessage());
    }catch(MediumPasswordException e){
```

```
        System.out.println(e.getMessage());
      }
    }
}
```

Status : Correct   *Marks : 10/10*

4. 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

```java
import java.util.*;
class InvalidDateOfBirthException extends Exception{
    public InvalidDateOfBirthException(String m){
        super(m);
    }
}
class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        try{
            int r=0,j=0,y=0;
            if(s.charAt(2)=='-' && s.charAt(5)=='-'){
                String[] k=s.split("-");
                r=Integer.valueOf(k[0]);
                j=Integer.valueOf(k[1]);
                y=Integer.valueOf(k[2]);
            }
            if(s.charAt(2)!='-' || s.charAt(5)!='-'){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==2 && r>29 && y%4==0){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==2 && r>28 && y%4!=0){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }
            else if(j>12 || j<1){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==1 && r>31){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==3 && r>31){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==4 && r>30){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==5 && r>31){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==6 && r>30){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
            }else if(j==7 && r>31){
                throw new InvalidDateOfBirthException("Invalid date: "+s);
```

```java
    }else if(j==8 && r>31){
        throw new InvalidDateOfBirthException("Invalid date: "+s);
    }else if(j==9 && r>30){
        throw new InvalidDateOfBirthException("Invalid date: "+s);
    }else if(j==10 && r>31){
        throw new InvalidDateOfBirthException("Invalid date: "+s);
    }else if(j==11 && r>30){
        throw new InvalidDateOfBirthException("Invalid date: "+s);
    }else if(j==12 && r>31){
        throw new InvalidDateOfBirthException("Invalid date: "+s);
    }else{
        System.out.println(s+" is a valid date of birth");
    }
    }catch(InvalidDateOfBirthException e){
        System.out.println(e.getMessage());
    }catch(Exception e){
        System.out.println("Invalid date: "+s);
    }
    }
    }
}
```

Status : Correct   *Marks : 10/10*