Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
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_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. What will be the output for the following code?

```
import java.io.*;

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

class Test {
    public static void main(String[] args) {
        try {
            int temperature = 110;
            if (temperature > 100) {
                throw new TemperatureTooHighException("Temperature too
high");
```

```
        }
    } catch (TemperatureTooHighException e) {
        System.out.println(e.getMessage());
    }
  }
}
```

Answer

Temperature too high

Status : Correct ***Marks : 1/1***

2. what is the output of the following code?

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

class Test {
  public static void main(String[] args) {
    try {
      throw new MyException("Error occurred");
    } catch (MyException e) {
      System.out.println(e);
    }
  }
}
```

Answer

MyException: Error occurred

Status : Correct ***Marks : 1/1***

3. Which of the following is true about custom exceptions?

Answer

Custom exceptions must extend either Exception or RuntimeException

Status : Correct **Marks : 1/1**

4. What will be the output of the following code?

```
class MyException extends Exception {
    public MyException() {
        super("Default Exception Message");
    }
}

class Test {
    public static void main(String[] args) {
        try {
            throw new MyException();
        } catch (MyException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Answer

Default Exception Message

Status : Correct **Marks : 1/1**

5. What will be the output for the following code?

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

class Test {
    public static void main(String[] args) {
        try {
            String username = "abc";
            if (username.length() < 5) {
```

```
            throw new InvalidUsernameException("Username must be at
least 5 characters long");
        }
      } catch (InvalidUsernameException e) {
        System.out.println(e.getMessage());
      }
    }
}
```

Answer

Username must be at least 5 characters long

Status : Correct ***Marks : 1/1***

6. How do you create an unchecked custom exception?

Answer

By extending RuntimeException

Status : Correct ***Marks : 1/1***

7. What will be the output for the following code?

```
import java.io.*;

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

class Test {
    public static void main(String[] args) {
        try {
            int age = -5;
            if (age < 0) {
                throw new NegativeAgeException("Age cannot be negative");
            }
```

```
        } catch (NegativeAgeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Answer

Age cannot be negative

Status : Correct ***Marks : 1/1***

8. What is the purpose of a custom exception in Java?

Answer

To create user-defined exceptions for specific scenarios

Status : Correct ***Marks : 1/1***

9. What will be the output for the following code?

```
import java.io.*;

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

class Test {
    public static void main(String[] args) {
        try {
            int stock = 0;
            if (stock == 0) {
                throw new OutOfStockException("Item is out of stock");
            }
        } catch (OutOfStockException e) {
            System.out.println(e.getMessage());
        }
```

```
    }
}
```

Answer

Item is out of stock

Status : Correct ***Marks : 1/1***

10. What will be the output for the following code?

```
import java.io.*;

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

class Test {
    public static void main(String[] args) {
        try {
            int age = 17;
            if (age < 18) {
                throw new UnderageException("Underage, cannot proceed");
            }
        } catch (UnderageException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Answer

Underage, cannot proceed

Status : Correct ***Marks : 1/1***

11. What will be the output for the following code?

```
class NegativeBalanceException extends Exception {
```

```
    public NegativeBalanceException(String message) {
        super(message);
    }
}

class Test {
    public static void main(String[] args) {
        try {
            double balance = -500;
            if (balance < 0) {
                throw new NegativeBalanceException("Balance cannot be
negative");
            }
        } catch (NegativeBalanceException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Answer

Error: Balance cannot be negative

Status : Correct ***Marks : 1/1***

12. what is the output of the following code?

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

class Test {
    static void check() throws MyException {
        throw new MyException("Custom Exception Occurred");
    }

    public static void main(String[] args) {
        try {
```

```
            check();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Answer

Custom Exception Occurred

Status : Correct ***Marks : 1/1***

13. What will happen if a checked custom exception is thrown inside a method without being caught or declared?

Answer

Compilation Error

Status : Correct ***Marks : 1/1***

14. What will be the output for the following code?

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

class Test {
    public static void main(String[] args) {
        try {
            int age = 15;
            if (age < 18) {
                throw new InvalidVotingAgeException("You are not eligible to
vote");
            }
            System.out.println("Eligible to vote");
        } catch (InvalidVotingAgeException e) {
```

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

Answer

You are not eligible to vote

Status : Correct ***Marks : 1/1***

15. Which keyword is used to explicitly throw a custom exception?

Answer

throw

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Write a program to validate the email address and display suitable exceptions if there is any mistake.

Create 3 custom exception classes as below

DotExceptionAtTheRateExceptionDomainException

A typical email address should have a " . " character, and a "@" character, and also the domain name should be valid. Valid domain names for practice be 'in', 'com', 'net', or 'biz'.

Display Invalid Dot usage, Invalid @ usage, or Invalid Domain message based on email id.

Get the email address from the user, validate the email by checking the

above-mentioned criteria, and print the validity status of the input email address.

Input Format

The first line of input contains the email to be validated.

Output Format

The output prints a Valid email address or an Invalid email address along with the suitable exception

If email ends with . or contains not exactly one . after @, it throws:

DotException: Invalid Dot usage

Invalid email address

If @ appears not exactly once, it throws:

AtTheRateException: Invalid @ usage

Invalid email address

If the part after the last dot is not among accepted domains:

DomainException: Invalid Domain

Invalid email address

If all conditions satisfied then print:

Valid email address

Refer to the sample input and output for format specifications.

Sample Test Case

Input: sample@gmail.com

Output: Valid email address

Answer

```
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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

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

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

class EmailValidation {

   public static void validate(String email)
         throws AtTheRateException, DotException, DomainException {

      if (email == null || email.isEmpty()) {
         throw new IllegalArgumentException("Email cannot be empty.");
      }

      int atCount = 0;
```

```
        int atIndex = -1;
        for (int i = 0; i < email.length(); i++) {
            if (email.charAt(i) == '@') {
                atCount++;
                atIndex = i;
            }
        }

        if (atCount != 1) {
            throw new AtTheRateException("Invalid @ usage");
        }

        String domainPart = email.substring(atIndex + 1);

        int lastDotIndex = domainPart.lastIndexOf('.');
        int firstDotIndex = domainPart.indexOf('.');

        if (email.endsWith(".")) {
            throw new DotException("Invalid Dot usage");
        }

        if (lastDotIndex == -1) {
             throw new DotException("Invalid Dot usage");
        }

        if (firstDotIndex != lastDotIndex) {
            throw new DotException("Invalid Dot usage");
        }

        String domainExtension = domainPart.substring(lastDotIndex + 1);

        if (!domainExtension.equals("in") &&
            !domainExtension.equals("com") &&
            !domainExtension.equals("net") &&
            !domainExtension.equals("biz")) {
            throw new DomainException("Invalid Domain");
        }

    }
}

public class Main {
```

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

        try {
            email = scanner.nextLine().trim();

            EmailValidation.validate(email);

            System.out.println("Valid email address");

        } catch (AtTheRateException e) {
            System.out.println(e.getClass().getSimpleName() + ": " + e.getMessage());
            System.out.println("Invalid email address");
        } catch (DotException e) {
            System.out.println(e.getClass().getSimpleName() + ": " + e.getMessage());
            System.out.println("Invalid email address");
        } catch (DomainException e) {
            System.out.println(e.getClass().getSimpleName() + ": " + e.getMessage());
            System.out.println("Invalid email address");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Elsa, a busy professional, is using a scheduling application to plan her meetings efficiently. The application requires users to input meeting durations in minutes, ensuring that the duration is a positive integer and does not exceed 240 minutes (4 hours). Elsa needs a program to assist her in scheduling meetings securely with proper exception handling.

Create a Java class named ElsaMeetingScheduler.Implement a custom exception:InvalidDurationException for invalid meeting duration entries.Implement the main method to interactively take user input for a meeting duration.Implement the validateMeetingDuration method to validate the meeting duration based on the specified rules and throw a custom exception if the validation fails.Print appropriate success or error messages based on the meeting duration.

Implement a custom exception, InvalidDurationException, to handle cases where the entered meeting duration does not meet the specified criteria.

Input Format

The input consists of an integer value 'n', representing the meeting duration.

Output Format

The output is displayed in the following format:

If the entered meeting duration meets the specified criteria, the program outputs

"Meeting scheduled successfully!"

If the entered meeting duration is invalid, the program outputs an error message indicating the issue.

"Error: Invalid meeting duration. Please enter a positive integer not exceeding 240 minutes (4 hours)."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 120
Output: Meeting scheduled successfully!

Answer

```
import java.util.Scanner;
import java.util.InputMismatchException;

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

public class Main {

    private static final String ERROR_MESSAGE =
```

```
    "Error: Invalid meeting duration. Please enter a positive integer not
exceeding 240 minutes (4 hours).";

   public static void validateMeetingDuration(int durationMinutes) throws
InvalidDurationException {
      if (durationMinutes <= 0) {
         throw new InvalidDurationException(ERROR_MESSAGE);
      }

      if (durationMinutes > 240) {
         throw new InvalidDurationException(ERROR_MESSAGE);
      }
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int duration = 0;

      try {
         if (scanner.hasNextInt()) {
            duration = scanner.nextInt();

            validateMeetingDuration(duration);

            System.out.println("Meeting scheduled successfully!");
         } else {
            System.out.println(ERROR_MESSAGE);
         }

      } catch (InvalidDurationException e) {
         System.out.println(e.getMessage());
      } catch (InputMismatchException e) {
         System.out.println(ERROR_MESSAGE);
      } catch (Exception e) {
         System.err.println("An unexpected error occurred.");
      } finally {
         if (scanner != null) {
            scanner.close();
         }
      }
   }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a user registration system, there is a requirement to implement a username validation module. Users attempting to register must adhere to specific criteria for their usernames to be considered valid.

Your task is to develop a program that takes user input for a desired username and validates it according to the following rules:

The username must not contain any spaces.The username must be at least 5 characters long.

Implement a custom exception, InvalidUsernameException, to handle cases where the entered username does not meet the specified criteria.

Input Format

The input consists of a string S, representing the desired username.

Output Format

If the username is valid, print "Username is valid: [S]".

If the username is invalid:

1. If the username is short, print "Invalid Username: Username must be at least 5 characters long"
2. If the username contains spaces, print "Invalid Username: Username cannot contain spaces"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: John
Output: Invalid Username: Username must be at least 5 characters long

Answer

```
import java.util.Scanner;

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

public class Main {

    private static final String ERROR_SHORT =
        "Invalid Username: Username must be at least 5 characters long";
    private static final String ERROR_SPACES =
        "Invalid Username: Username cannot contain spaces";

    public static void validateUsername(String username) throws
InvalidUsernameException {
        if (username.contains(" ")) {
            throw new InvalidUsernameException(ERROR_SPACES);
        }
```

```
        if (username.length() < 5) {
            throw new InvalidUsernameException(ERROR_SHORT);
        }
    }

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

        try {
            if (scanner.hasNextLine()) {
                username = scanner.nextLine();

                validateUsername(username);

                System.out.println("Username is valid: " + username);
            }

        } catch (InvalidUsernameException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.err.println("An unexpected error occurred during processing.");
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

A local municipality is implementing an online voting system for a community event and wants to ensure that only eligible voters (those aged 18 or older) can participate.

Your task is to develop a program that validates the age of individuals attempting to vote online. If the user's age is below 18, the program should throw a custom exception, InvalidAgeException, preventing them from casting their vote. If the input is invalid, catch the appropriate InputMismatchException and print the in-built exception message.

Input Format

The input consists of an integer representing the age.

Output Format

If the age is 18 or older, print "Eligible to vote"

If the age is below 18, print "Exception occurred: InvalidAgeException: Age is not valid to vote"

If there is any other type of exception, print "An error occurred: " followed by the in-built exception message.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20
Output: Eligible to vote

Answer

```
import java.util.Scanner;
import java.util.InputMismatchException;

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

public class Main {

    private static final int MINIMUM_VOTING_AGE = 18;

    public static void validateAge(int age) throws InvalidAgeException {
        if (age < MINIMUM_VOTING_AGE) {
            throw new InvalidAgeException("Age is not valid to vote");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int age = 0;

        try {
```

```
            if (scanner.hasNextInt()) {
                age = scanner.nextInt();

                validateAge(age);

                System.out.println("Eligible to vote");
            } else {
                throw new InputMismatchException("java.util.Input");
            }

        } catch (InvalidAgeException e) {
            System.out.println("Exception occurred: InvalidAgeException: " +
e.getMessage());
        } catch (InputMismatchException e) {
            System.out.println("An error occurred: " + e.getMessage() +
"MismatchException");
        } catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 8_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a file management system, users are required to provide a valid file name when creating new files. The system enforces specific rules for file names to maintain consistency and avoid potential issues. Your task is to implement a Java program named FileNameValidator that takes user input for a file name and validates it according to the specified rules.

Rules for Valid File Name:

The file name must consist of alphanumeric characters (letters and digits) only.The file name must have a minimum length of 3 characters.

Implement a custom exception, FileNameValidator, to handle cases where the entered filename does not meet the specified criteria.

Input Format

The input consists of a string S, representing the desired filename.

Output Format

The output is displayed in the following format:

If the entered file name meets the specified criteria, the program outputs

"Valid file name"

If the entered file name does not meet the criteria and triggers the InvalidFileNameException, the program outputs

"Error: Invalid file name. It must be alphanumeric and have a minimum length of 3 characters."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: myfile123
Output: Valid file name

Answer

```
import java.util.Scanner;

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

public class Main {

    private static final String ERROR_MESSAGE = "Error: Invalid file name. It must be alphanumeric and have a minimum length of 3 characters.";

    public static void validateFileName(String fileName) throws InvalidFileNameException {
        if (fileName.length() < 3) {
            throw new InvalidFileNameException(ERROR_MESSAGE);
```

```
        }

        if (!fileName.matches("[a-zA-Z0-9]+")) {
            throw new InvalidFileNameException(ERROR_MESSAGE);
        }
    }

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

        try {
            if (scanner.hasNextLine()) {
                fileName = scanner.nextLine();

                validateFileName(fileName);

                System.out.println("Valid file name");
            }
        } catch (InvalidFileNameException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
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_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Daniel is developing a program to verify the age of users. He wants to ensure that the entered age is within a valid range. Write a program to help Daniel implement this age-checking feature using custom exceptions.

Daniel needs a program that takes an integer input representing a person's age.If the age is between 0 and 150 (inclusive), the program should print "Age is valid!".If the age is less than 0 or greater than 150, the program should throw a custom exception (InvalidAgeException) with the message "Invalid age. Please enter an age between 0 and 150."

Implement a custom exception, InvalidAgeException, to handle cases where the entered age does not meet the specified criteria.

Input Format

The input consists of an integer value 'n', representing the age.

Output Format

The output is displayed in the following format:

If the age is valid (between 0 and 150, inclusive), print

"Age is valid!".

If the age is invalid, print

"Error: Invalid age. Please enter an age between 0 and 150."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 45
Output: Age is valid!

Answer

```
import java.util.Scanner;
import java.util.InputMismatchException;

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

public class Main {

    private static final String ERROR_MESSAGE = "Error: Invalid age. Please enter
an age between 0 and 150.";
    private static final int MIN_AGE = 0;
    private static final int MAX_AGE = 150;

    public static void validateAge(int age) throws InvalidAgeException {
        if (age < MIN_AGE || age > MAX_AGE) {
            throw new InvalidAgeException(ERROR_MESSAGE);
```

```
    }
}

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int age = 0;

        try {
            if (scanner.hasNextInt()) {
                age = scanner.nextInt();

                validateAge(age);

                System.out.println("Age is valid!");
            } else {
                System.err.println("Input is not an integer.");
            }
        } catch (InvalidAgeException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

2. Problem Statement

An HR software system is being developed to process employee payrolls. During payroll processing, the system must ensure that no employee has a negative salary and that no employee's salary exceeds ₹2,00,000. If either condition occurs, the system should throw a custom exception.

Create a custom exception InvalidSalaryException and a class Employee

that processes salary according to the following rules:

If salary < 0, throw InvalidSalaryException with the message: "Salary cannot be negative". If salary > 200000, throw InvalidSalaryException with the message: "Salary exceeds threshold limit". Otherwise, display: "Salary processed successfully for <empName>: <salary>".

The payroll processing should always display: "Payroll process completed" at the end, regardless of whether an exception occurs.

Input Format

The first line of input contains an integer representing the employee ID.

The second line contains a string representing the employee's name.

The third line contains a floating-point number representing the salary of the employee.

Output Format

If the salary is valid: "Salary processed successfully for <empName>: <salary>"

"Payroll process completed"

If the salary is invalid: "<Exception Message>"

"Payroll process completed"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 101
Rahul
150000.0
Output: Salary processed successfully for Rahul: 150000.0
Payroll process completed

Answer

```
import java.util.Scanner;
```

```
import java.util.InputMismatchException;

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

class Employee {
    private int empId;
    private String empName;
    private double salary;

    public Employee(int empId, String empName, double salary) {
        this.empId = empId;
        this.empName = empName;
        this.salary = salary;
    }

    public void processSalary() throws InvalidSalaryException {
        if (salary < 0) {
            throw new InvalidSalaryException("Salary cannot be negative");
        }

        if (salary > 200000) {
            throw new InvalidSalaryException("Salary exceeds threshold limit");
        }

        System.out.println("Salary processed successfully for " + empName + ": " +
salary);
    }
}

public class Main {

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

        try {
            int empId = scanner.nextInt();
            scanner.nextLine();
```

```
            String empName = scanner.nextLine();

            double salary = scanner.nextDouble();

            Employee employee = new Employee(empId, empName, salary);

            try {
                employee.processSalary();
            } catch (InvalidSalaryException e) {
                System.out.println(e.getMessage());
            }

        } catch (InputMismatchException e) {
            System.out.println("Input format error: Please ensure ID is integer, Name
is string, and Salary is a number.");
        } catch (Exception e) {
            System.out.println("An unexpected error occurred during input: " +
e.getMessage());
        } finally {
            System.out.println("Payroll process completed");
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

3. Problem Statement

Enigma is developing a simple web application that takes a user-input URL, validates it, and throws a custom exception InvalidURLFormatException if the URL does not start with "http://" or "https://".

The main method prompts the user for input, validates the URL, and prints whether it is valid or not.

Input Format

The input consists of a string, representing the URL entered by the user.

Output Format

The output displays one of the following results:

If the entered URL is valid according to the specified format, the program prints:

"[URL] is a valid URL"

If the entered URL is not valid according to the specified format, the program prints:

"Invalid URL format: [URL]"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: http://www.example.com
Output: http://www.example.com is a valid URL

Answer

```
import java.util.Scanner;

class InvalidURLFormatException extends Exception {
    public InvalidURLFormatException(String url) {
        super("Invalid URL format: " + url);
    }
}

public class Main {

    public static void validateURL(String url) throws InvalidURLFormatException {
        if (!url.startsWith("http://") && !url.startsWith("https://")) {
            throw new InvalidURLFormatException(url);
        }
    }
```

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

        try {
            if (scanner.hasNextLine()) {
                url = scanner.nextLine();
            }

            validateURL(url);

            System.out.println(url + " is a valid URL");

        } catch (InvalidURLFormatException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

4. Problem Statement

You are tasked to create a program that defines a custom exception GradeException. The program should include a Student class with fields for the student's name, age, and grade. Implement a method in the Student class that checks the grade, and if the grade is below 40, it should throw a GradeException. Otherwise, it should display the student's details.

Input Format

The input consists of three parameters in separate lines:

1. A string representing the student's name.

2. An integer representing the student's age.
3. An integer representing the student's grade.

Output Format

The output will display the student's details if the grade is valid.

If the grade is below 40, the program will display an error message "Grade is below 40".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Alice
20
85

Output: Name: Alice
Age: 20
Grade: 85

Answer

```
import java.util.Scanner;
import java.util.InputMismatchException;

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

class Student {
    private String name;
    private int age;
    private int grade;

    public Student(String name, int age, int grade) {
        this.name = name;
        this.age = age;
        this.grade = grade;
    }
```

```
    public void validateAndDisplayDetails() throws GradeException {
        if (this.grade < 40) {
            throw new GradeException("Grade is below 40");
        } else {
            System.out.println("Name: " + this.name);
            System.out.println("Age: " + this.age);
            System.out.println("Grade: " + this.grade);
        }
    }
}

public class Main {

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

        try {
            String name = scanner.nextLine();

            int age = scanner.nextInt();

            int grade = scanner.nextInt();

            Student student = new Student(name, age, grade);

            student.validateAndDisplayDetails();

        } catch (GradeException e) {
            System.out.println(e.getMessage());
        } catch (InputMismatchException e) {
            System.err.println("Input error: Please ensure age and grade are
integers.");
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mohamed Humaid Zahir Hussain
Email: 240701322@rajalakshmi.edu.in
Roll no: 2116240701322
Phone: 6382261139
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

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

```
import java.util.Scanner;
import java.util.InputMismatchException;

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

```
}

public class Main {

    public static void validateCreditCard(String cardNumber) throws
InvalidCreditCardException {

        if (cardNumber.length() != 16) {
            throw new InvalidCreditCardException("Error: Invalid credit card number
length.");
        }

        try {
            Long.parseLong(cardNumber);
        } catch (NumberFormatException e) {
            throw new InvalidCreditCardException("Error: Invalid credit card number
format.");
        }
    }

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

        try {
            String cardNumber = scanner.nextLine();

            validateCreditCard(cardNumber);

            System.out.println("Payment information updated successfully!");

        } catch (InvalidCreditCardException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.err.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }
}
```

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

2. 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.util.InputMismatchException;

class InvalidDateOfBirthException extends Exception {
    public InvalidDateOfBirthException(String dateStr) {
        super("Invalid date provided: " + dateStr);
    }
}

public class Main {

    public static boolean isLeapYear(int year) {
        return (year % 4 == 0) && (year % 100 != 0) || (year % 400 == 0);
    }

    public static void validateDate(String dateStr) throws
InvalidDateOfBirthException {
        if (dateStr.length() != 10 || dateStr.charAt(2) != '-' || dateStr.charAt(5) != '-') {
            throw new InvalidDateOfBirthException(dateStr);
        }

        int day, month, year;

        try {
            day = Integer.parseInt(dateStr.substring(0, 2));
            month = Integer.parseInt(dateStr.substring(3, 5));
            year = Integer.parseInt(dateStr.substring(6, 10));
        } catch (NumberFormatException e) {
            throw new InvalidDateOfBirthException(dateStr);
        }

        if (year < 1 || month < 1 || month > 12 || day < 1 || day > 31) {
            throw new InvalidDateOfBirthException(dateStr);
        }

        int daysInMonth;

        switch (month) {
            case 4:
            case 6:
            case 9:
            case 11:
                daysInMonth = 30;
```

```
            break;

        case 2:
            daysInMonth = isLeapYear(year) ? 29 : 28;
            break;

        default:
            daysInMonth = 31;
            break;
    }

    if (day > daysInMonth) {
        throw new InvalidDateOfBirthException(dateStr);
    }
}

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

    try {
        dateStr = sc.nextLine();

        validateDate(dateStr);

        System.out.println(dateStr + " is a valid date of birth");

    } catch (InvalidDateOfBirthException e) {
        System.out.println("Invalid date: " + dateStr);

    } catch (Exception e) {
        System.out.println("Invalid date: " + dateStr);

    } finally {
        if (sc != null) {
            sc.close();
        }
    }
}
}
```

Status : Correct *Marks : 10/10*

3. 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

```
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 scanner = new Scanner(System.in);

        try {
            int n = scanner.nextInt();

            if (n <= 0) {
                String errorMessage = "Invalid input. Please enter a positive integer.";
                throw new InvalidPositiveNumberException(errorMessage);
            }

            System.out.println("Number " + n + " is positive.");

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

        } catch (InputMismatchException e) {
            System.out.println("Error: Input is not a valid integer.");

        } finally {
            scanner.close();
        }
    }
}
```

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 validateCouponCode(String couponCode) throws
InvalidCouponException {
      if (couponCode.length() != 10) {
         throw new InvalidCouponException("Invalid coupon code length. It must
be exactly 10 characters.");
      }

      boolean hasAlphabet = false;
      boolean hasDigit = false;

      for (char c : couponCode.toCharArray()) {
         if (!Character.isLetterOrDigit(c)) {
            throw new InvalidCouponException("Coupon code should not contain
special characters.");
         }

         if (Character.isLetter(c)) {
            hasAlphabet = true;
         } else if (Character.isDigit(c)) {
            hasDigit = true;
         }
      }
```

```
        if (!hasAlphabet || !hasDigit) {
            throw new InvalidCouponException("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 = "";
        if (scanner.hasNextLine()) {
            couponCode = scanner.nextLine();
        }

        try {
            validateCouponCode(couponCode);
        } catch (InvalidCouponException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

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