

# Rajalakshmi Engineering College

Name: Adithya B

Email: 240701018@rajalakshmi.edu.in

Roll no: 240701018

Phone: 9444117405

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\_PAH

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### **Section 1 : Coding**

#### **1. 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**

```
// You are using Java
class GradeException extends Exception {
    public GradeException(String msg) {
        super(msg);
    }
}

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

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

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

```

        System.out.println("Age: " + age);
        System.out.println("Grade: " + grade);
    }
}
}

public class Main {
    public static void main(String[] args) {
        try {
            java.util.Scanner sc = new java.util.Scanner(System.in);
            String name = sc.nextLine();
            int age = sc.nextInt();
            int grade = sc.nextInt();

            Student s = new Student(name, age, grade);
            s.checkGrade();
        } catch (GradeException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

**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***

```
// You are using Java
class InvalidSalaryException extends Exception {
    public InvalidSalaryException(String msg) {
        super(msg);
    }
}
class Employee {
    int empld;
```

```

String empName;
double salary;

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

public void processSalary() throws InvalidSalaryException {
    if (salary < 0) {
        throw new InvalidSalaryException("Salary cannot be negative");
    } else if (salary > 200000) {
        throw new InvalidSalaryException("Salary exceeds threshold limit");
    } else {
        System.out.println("Salary processed successfully for " + empName + ":" +
+ salary);
    }
}
}

public class Main {
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);

        int id = Integer.parseInt(sc.nextLine());
        String name = sc.nextLine();
        double sal = Double.parseDouble(sc.nextLine());

        Employee e = new Employee(id, name, sal);

        try {
            e.processSalary();
        } catch (InvalidSalaryException ex) {
            System.out.println(ex.getMessage());
        } finally {
            System.out.println("Payroll process completed");
        }
    }
}

```

**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 `InvalidURLException` 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***

```
// You are using Java
class InvalidURLException extends Exception {
    public InvalidURLException(String msg) {
```

```

        super(msg);
    }

class URLValidator {
    public void validate(String url) throws InvalidURLFormatException {
        if (!(url.startsWith("http://") || url.startsWith("https://"))) {
            throw new InvalidURLFormatException("Invalid URL format: " + url);
        } else {
            System.out.println(url + " is a valid URL");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        String url = sc.nextLine();

        URLValidator validator = new URLValidator();

        try {
            validator.validate(url);
        } catch (InvalidURLFormatException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

**Status : Correct**

**Marks : 10/10**

#### 4. 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***

```
// You are using Java
class InvalidAgeException extends Exception {
    public InvalidAgeException(String msg) {
        super(msg);
    }
}

public class Main {
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        int age = sc.nextInt();
```

```
try {
    checkAge(age);
    System.out.println("Age is valid!");
} catch (InvalidAgeException e) {
    System.out.println("Error: " + e.getMessage());
}

public static void checkAge(int age) throws InvalidAgeException {
    if (age < 0 || age > 150) {
        throw new InvalidAgeException("Invalid age. Please enter an age between
0 and 150.");
    }
}
```

**Status : Correct**

**Marks : 10/10**