LAB ASSIGNMENT-3

NAME- Nandini Kumari

ROLL NO – 2401201085

COURSE- BCA(AI&DS)

Input-

```java
import java.util.*;


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


class Loader implements Runnable {
    @Override
    public void run() {
        try {
            System.out.print(s:"Loading");
            for (int i = 0; i < 5; i++) {
                Thread.sleep(millis:300);
                System.out.print(s:".");
            }
            System.out.println();
        } catch (InterruptedException e) {
            System.out.println(x:"Loading interrupted!");
        }
    }
}


```

```java
interface RecordActions {
    void addStudent(Student s);
    Student getStudent(Integer roll) throws StudentNotFoundException;
}


class Student {
    private Integer roll;
    private String name;
    private String email;
    private String course;
    private Double marks;

    public Student(Integer roll, String name, String email, String course, Double marks) {
        this.roll = roll;
        this.name = name;
        this.email = email;
        this.course = course;
        this.marks = marks;
    }


    public Integer getRoll() {
        return roll;
    }

    public Double calculateGrade() {
        if (marks >= 90) return 4.0;
```

```java
        else if (marks >= 80) return 3.5;
        else if (marks >= 70) return 3.0;
        else if (marks >= 60) return 2.5;
        else return 2.0;
    }

    public String getLetterGrade() {
        if (marks >= 90) return "A";
        else if (marks >= 80) return "B+";
        else if (marks >= 70) return "B";
        else if (marks >= 60) return "C";
        else return "D";
    }

    public void display() {
        System.out.println("\nRoll No: " + roll);
        System.out.println("Name: " + name);
        System.out.println("Email: " + email);
        System.out.println("Course: " + course);
        System.out.println("Marks: " + marks);
        System.out.println("Grade: " + getLetterGrade());
    }
}


class StudentManager implements RecordActions {
```

```
83      private Map<Integer, Student> records = new HashMap<>();
84
85      @Override
86      public void addStudent(Student s) {
87          records.put(s.getRoll(), s);
88      }
89
90      @Override
91      public Student getStudent(Integer roll) throws StudentNotFoundException {
92          if (!records.containsKey(roll)) {
93              throw new StudentNotFoundException("Student with roll " + roll + " not found!");
94          }
95          return records.get(roll);
96      }
97  }
98
99
100 public class Main {
        Run | Debug
101     public static void main(String[] args) {
102         Scanner sc = new Scanner(System.in);
103         StudentManager manager = new StudentManager();
104
105         try {
106
107             System.out.print(s:"Enter Roll No (Integer): ");
108             Integer roll = Integer.valueOf(sc.nextInt());
```

```
109             sc.nextLine();
110
111             System.out.print(s:"Enter Name: ");
112             String name = sc.nextLine();
113             if (name.trim().isEmpty())
114                 throw new Exception(message:"Name cannot be empty!");
115
116             System.out.print(s:"Enter Email: ");
117             String email = sc.nextLine();
118             if (email.trim().isEmpty())
119                 throw new Exception(message:"Email cannot be empty!");
120
121             System.out.print(s:"Enter Course: ");
122             String course = sc.nextLine();
123             if (course.trim().isEmpty())
124                 throw new Exception(message:"Course cannot be empty!");
125
126             System.out.print(s:"Enter Marks: ");
127             Double marks = Double.valueOf(sc.nextDouble());
128             if (marks < 0 || marks > 100)
129                 throw new Exception(message:"Marks must be between 0 and 100!");
130
131
132             Thread t = new Thread(new Loader());
133             t.start();
134             t.join();
```

```
135
136
137             Student s = new Student(roll, name, email, course, marks);
138             manager.addStudent(s);
139
140             
141             Student stored = manager.getStudent(roll);
142             stored.display();
143
144         } catch (StudentNotFoundException e) {
145             System.out.println("Error: " + e.getMessage());
146         } catch (InputMismatchException e) {
147             System.out.println(x:"Error: Invalid input type!");
148         } catch (Exception e) {
149             System.out.println("Validation Error: " + e.getMessage());
150         } finally {
151             sc.close();
152             System.out.println(x:"\nProgram Finished.");
153         }
154     }
155 }
156
```

OUTPUT-

```
Enter Roll No (Integer): 85
Enter Name: Nandini Kumari
Enter Email: Nandiniii428@gmail.com
Enter Course: BCA
Enter Marks: 62
Loading.....

Roll No: 85
Name: Nandini Kumari
Email: Nandiniii428@gmail.com
Course: BCA
```

```
Name: Nandini Kumari
Email: Nandiniii428@gmail.com
Course: BCA
Marks: 62.0
Grade: C

Program Finished.
```