Date / /
Page No.

Name = Komal
Roll NO = 240142004 7
Program = BTech CSE (DS)
Semester = III

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

public class StudentResultManagementSystem {

    private static final int MAX_STUDENTS = 100;
    private Student[] students = new Students[MAX_STUDENTS];
    private int count = 0;
    private final Scanner input = new Scanner(System.in);
    public static final int PASS_MARK = 33;

public void AddStudent() throws InvalidMarksException {
      try {

          Sout("Enter Roll Number:");
          int roll = ~~in~~ ~~next~~ input.nextInt();
          input.nextLine();

      if (FindIndexByRoll(roll) != -1) {
        Sout("roll");
        return;
      }

      Sout(" Enter Student Name:");
      String name = input.nextLine();
```

Date / /
Page No.

```
int[] marks = new int[3];
for(int i=0; i<3; i++) {

    Sout (" Enter marks for subject " +(i+1) +":");
    marks[i] = input.nextInt();

    if (marks[i]<0 || marks[i]>100) {
        input.nextLine();
        throw new InvalidMarksException (" Invalid
            marks" + (i+1)+":"+ marks[i])
    }
}

input.nextLine();

Student s= new Student (roll, name, marks);
if (count < MAX_STUDENTS) {
    students[count++]= s;
    Sout (" Student added successfully ");
}
else {
    Sout (" Storage full");
} }

} catch (InputMismatchException ime) {

    Sout(" Input error : Enter numeric values only");
}

}
```

Date / /
Page No.

```
public void ShowStudentDetails() {
    try {
        Sout("Enter Roll Number to search:");
        int roll = input.nextInt();
        input.nextIntline();

        int idx = FinalIndexByRoll(roll);
        if (idx==-1) {

            Sout("Student with roll" +roll);
        } else {

            Students[idx].DisplayResult();
            Sout("Search Completed");

        }
    } catch (InputMismatchException ime) {

        Sout("Input error: roll number must be numeric");
        input.nextLine();

    }
}

private int FinalIndexByRoll(int roll) {
    for (int i=0; i<count; i++) {
        if (Student[i]!=null && student[i].GetRoll()==
                                                roll) {

            return i;

        }

    }

    return -1;

}
```

```
public void Menu() {

    boolean running = true;
    try {

    while (running) {

        Sout ();
        Sout (" Student Result Management System");
        Sout ("1. Add Student");
        Sout ("2. Show Student Details");
        Sout ("3. Exit");
        Sout ("Enter your choice");

    int choice;
    try {

        choice= input. nextInt();
        input.nextLine();

    }
    catch (InputMismatch Exception IME){

        Sout (" Invalid input. Please enter a no.
        input. nextLine();
        continue;
    }

    Switch (choice) {

        case 1:
            try {
                Add Student();
            }
```

```
Catch( Invalid_Marks_Exception IME ) {
        Sout ( IME.getMessage() };
}
 break;

Case 2:
   ~~reg~~ Show Student Details();
      break;
Case 3:
      Sout(" Exiting program. Thank you");
      running = False;
      break;
default:
      Sout (" Invalid choice");
   }
}

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

public static void main (String [] args) {

   Student Result Management System system = new Student Result
                                             Management System();

   system.menu();
}
```

```
Static class Student {

    private int Roll;
    private String Name;
    private int[] Marks;

    public Student(int Roll, String Name, int [] Marks){

        this.Roll = Roll;
        this.Name = Name;
        this.Marks = Marks;
    }

    public int GetRoll(){
        return roll;
    }

public double CalculateAverage(){

    double sum=0;
    for(int m: Marks) sum += m;
    return sum / Marks.length;

}

public void DisplayResult(){

    Sout("Roll Number" + Roll);
    Sout("Student Name" + &Name);
    Sout("Marks");
    for(int m: Marks){
        Sout(m + " ");
    }
```

```
Sout ();

double avg = CalculateAverage();
Sout ("Average" + avg);

boolean pass = true;
for (int m: Marks) {

    if ( m < StudentResultManagementSystem.PassMARK) {
        pass = false;
        break;
    }
}

Sout (" Result" + (pass? "pass": "Fail"));
}
}

Static class InvalidMarksException extends Exception {

    public InvalidMarksException (String Message) {

        super (message);
    }
}
}
```