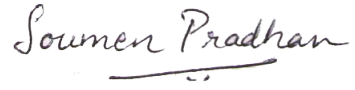

CS-212: Practical 2021

SOUMEN PRADHAN – 1912176

27 . 04 . 2021

1. Q-01: Matrix

```
1  import java.util.Random;
2
3  public class MatrixDemo {
4      public static void main(String[] args)
5      {
6          int max = 10;
7          int dim = 3;
8
9          Matrix a = new Matrix(dim, dim-1);
10         a.fillRand(max);
11         System.out.println("Mat a, " +
12                             a.getRows() + " x " + a.getCols());
13         a.print();
14
15         Matrix b = new Matrix(dim-1, dim);
16         b.fillRand(max);
17         System.out.println("Mat b, " +
18                             b.getRows() + " x " + b.getCols());
19         b.print();
20
21         Matrix r;
22
23         try {
24             r = a.addMats(b);
25             System.out.println("Result of Addition");
26             r.print();
27         } catch (Exception e) {
28             System.out.println();
29             e.printStackTrace();
30             System.out.println();
31             r = null;
32         }
33
34         try {
35             r = a.mulMats(b);
36             System.out.println("Result of Multiplication");
37             r.print();
38         } catch (Exception e) {
```

```
39              System.out.println();
40              e.printStackTrace();
41              System.out.println();
42              r = null;
43          }
44      }
45  }
46
47  class Matrix {
48      private int rows, cols;
49      private int[][] arr;
50
51      Matrix(int rows, int cols) {
52          this.rows = rows;
53          this.cols = cols;
54          this.arr = new int[rows][cols];
55      }
56
57      public int getRows() { return this.rows; }
58      public int getCols() { return this.cols; }
59      public int getElement(int i, int j) { return this.arr[i][j]; }
60
61      public boolean setElement(int i, int j, int data) {
62          if (i >= this.rows || j >= this.cols ||  i < 0 || j < 0)
63              return false;
64
65          this.arr[i][j] = data;
66          return true;
67      }
68
69      public void print() {
70          for (int i = 0; i < this.rows; i++) {
71              for (int j = 0; j < this.cols; j++) {
72                  System.out.print(this.arr[i][j] + " ");
73              }
74              System.out.println();
75          }
76          System.out.println();
77      }
78
79      public void fillRand(int max) {
80          Random rand = new Random();
81          for (int i = 0; i < this.rows; i++) {
82              for (int j = 0; j < this.cols; j++) {
83                  this.arr[i][j] = rand.nextInt(max);
84              }
85          }
```

```
86      }
87
88      public Matrix addMats(Matrix matB) {
89          if (this.rows != matB.rows || this.cols != matB.cols) {
90              System.out.println("Matrices cannot be added");
91              return null;
92          }
93
94          Matrix matR = new Matrix(this.rows, this.cols);
95          for (int i = 0; i < this.rows; i++)
96              for (int j = 0; j < this.cols; j++)
97                  matR.setElement(i, j,
98                      this.arr[i][j] + matB.getElement(i, j));
99
100         return matR;
101     }
102
103     public Matrix mulMats(Matrix matB) {
104         if (this.cols != matB.rows) {
105             System.out.println("Matrices cannot be multiplied");
106             return null;
107         }
108
109         Matrix matR = new Matrix(this.rows, matB.getCols());
110
111         for (int i = 0; i < this.rows; i++) {
112             for (int j = 0; j < matB.cols; j++) {
113                 int sum = 0;
114                 for (int k = 0; k < this.cols; k++) {
115                     sum += this.arr[i][k] * matB.getElement(k, j);
116                 }
117                 matR.setElement(i, j, sum);
118             }
119         }
120
121         return matR;
122     }
123 }
```

Result

Figure 1.1: *No Exceptions*

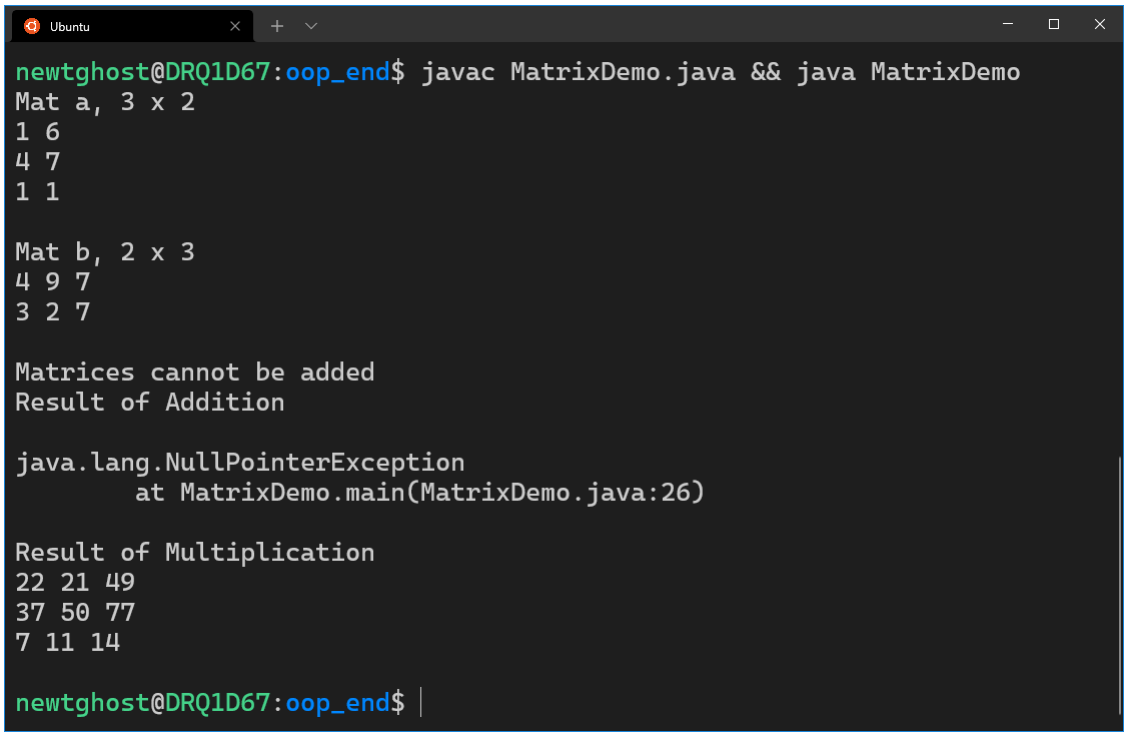

Figure 1.2: *Exception in Addition*

2. Q-02: Shape

```
1  public class ShapeDemo {
2      public static void main(String[] args)
3      {
4          shape s;
5
6          s = new circle(5);
7          s.display();
8          System.out.println("Area = " + s.area() + "\n");
9
10         s = new rectangle(5, 10);
11         s.display();
12         System.out.println("Area = " + s.area() + "\n");
13
14         s = new trapezoid(5, 10, 2);
15         s.display();
16         System.out.println("Area = " + s.area() + "\n");
17     }
18 }
19
20 abstract class shape {
21     protected double length, breadth;
22
23     shape(double l, double b) {
24         this.length = l;
25         this.breadth = b;
26     }
27
28     public abstract double area();
29     public abstract void display();
30 }
31
32 class circle extends shape {
33     circle(double r) {
34         super(r, r);
35     }
36
37     public double area() {
38         return Math.PI * Math.pow(length, 2);
```

```
39      }
40
41      public void display() {
42          System.out.println("Circle of radius: " + length);
43      }
44  }
45
46  class rectangle extends shape {
47      rectangle(double l, double b) {
48          super(l, b);
49      }
50
51      public double area() {
52          return length * breadth;
53      }
54
55      public void display() {
56          System.out.println("Rectangle of dim: " +
57                             length + ", " + breadth);
58      }
59  }
60
61  class trapezoid extends shape {
62      protected
63      double height;
64
65      trapezoid(double l, double b, double h) {
66          super(l, b);
67          this.height = h;
68      }
69
70      public double area() {
71          return (length + breadth) * height / 2;
72      }
73
74      public void display() {
75          System.out.println("Trapezoid of sides: " +
76                             length + ", " + breadth +
77                             " and height: " + height);
78      }
79  }
```

Result

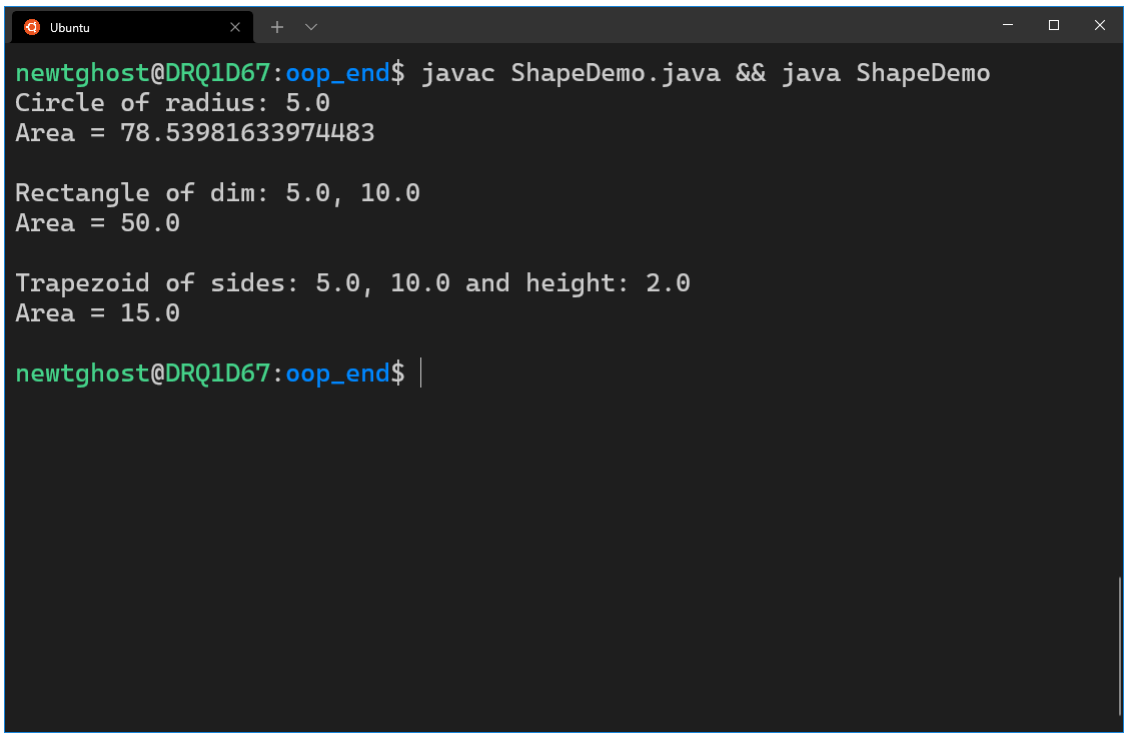

Figure 2.1: *Shapes area and display*