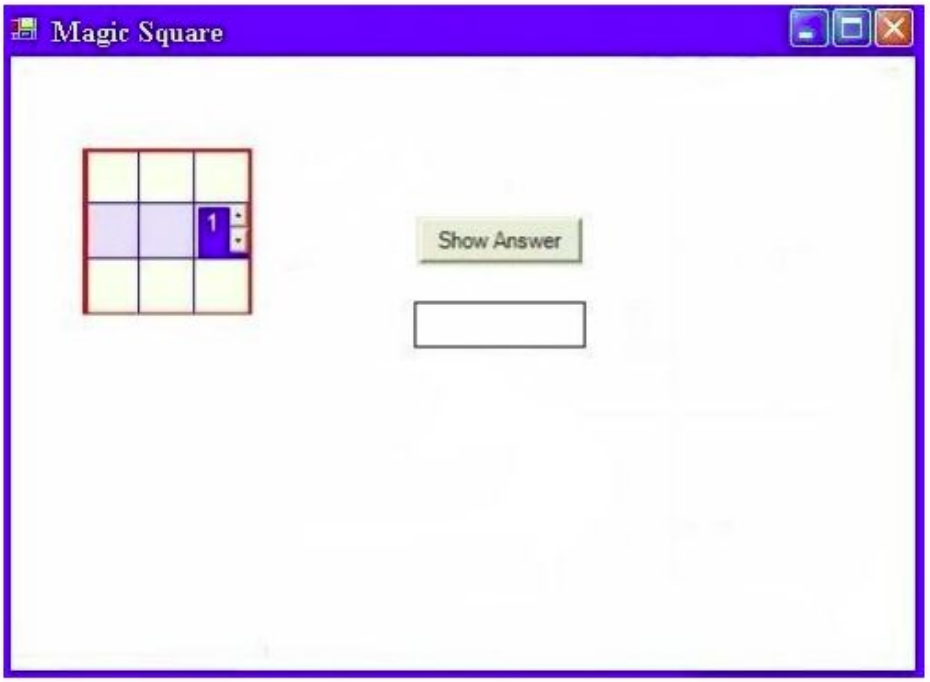
ST Assignment 3

SAHIL BONDRE: U18CO021

Write a java program to enter an integer number ‘n’. Create a grid of size ‘n*n’. Enter the value of each cell using a dropdown list. Check whether the created grid is a magic square or not. Display the result into the given text box by clicking on "Show Answer" button.

Magic Square:

A square matrix is said to be a Magic Square, if the sum of each row, each column and each diagonal is same.



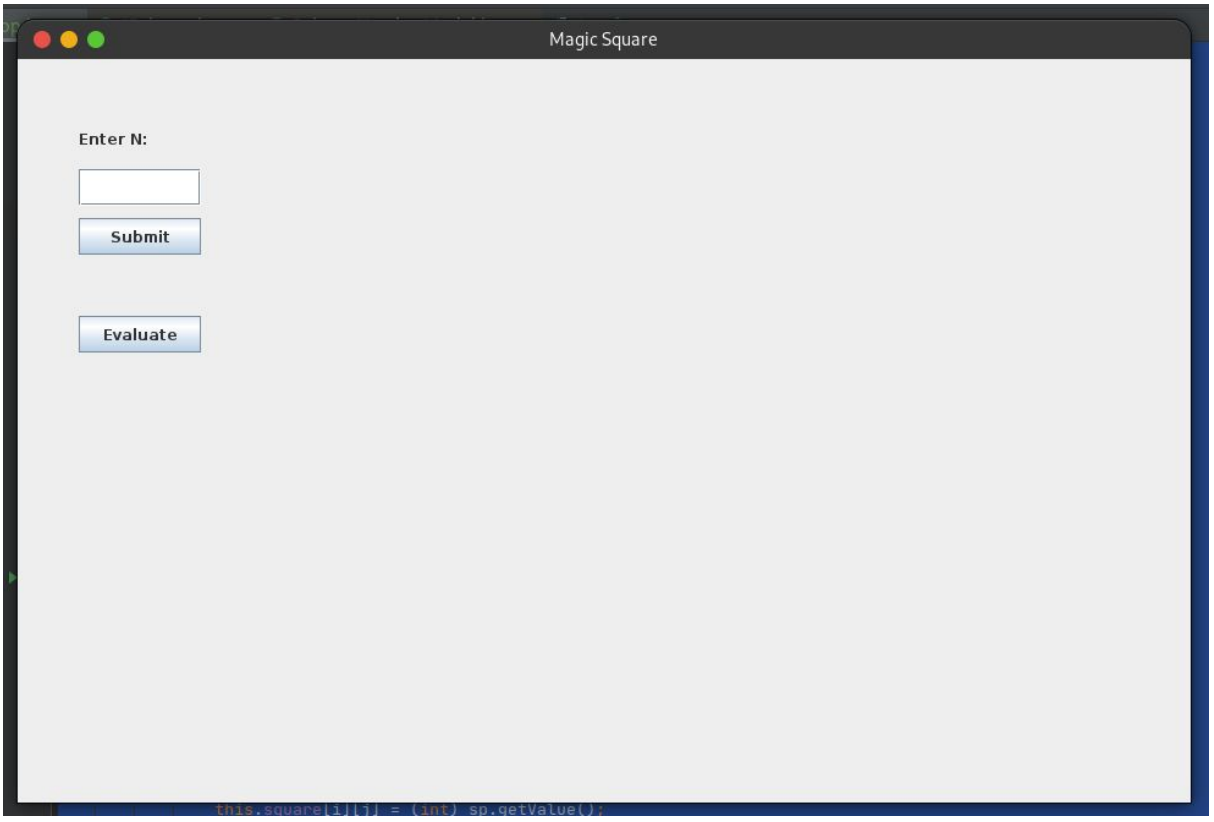

```
package lab_3;

import javax.swing.*;
import java.awt.*;

public class App {
    static int n = 0;
    static int x = 50;
    static int y = 50;
    static int dy = 40;
    static int w = 100;
    static int h = 30;
    JComponent[][] matrix = new JComponent[0][0];
```

```
int[][] square = new int[0][0];
JFrame f = new JFrame("Magic Square");
JPanel panel = new JPanel();
JDialog d = new JDialog(f, "Message", true);
JLabel dMessage = new JLabel("Warning: N must be an integer.");

public App() {
    var l = new JLabel("Enter N: ");
    l.setBounds(x, y, w, h);
    f.add(l);
    y += dy;

    var tf1 = new JTextField();
    tf1.setBounds(x, y, w, h);
    f.add(tf1);
    y += dy;


    d.setLayout(new FlowLayout());
    d.add(dMessage);
    var b2 = new JButton("OK");
    b2.addActionListener(e -> d.setVisible(false));
    d.add(b2);
    d.setSize(250, 100);


    panel.setBounds(300, 50, 500, 500);

    var b1 = new JButton("Submit");
    b1.setBounds(x, y, w, h);
    b1.addActionListener(e -> {
        try {
            n = Integer.parseInt(tf1.getText());
            buildGrid();

        } catch (NumberFormatException err) {
            d.setVisible(true);
        }
    });
    f.add(b1);
    y += dy;
    y += dy;

    var b3 = new JButton("Evaluate");
    b3.setBounds(x, y, w, h);
    b3.addActionListener(e -> fillSquare());
```

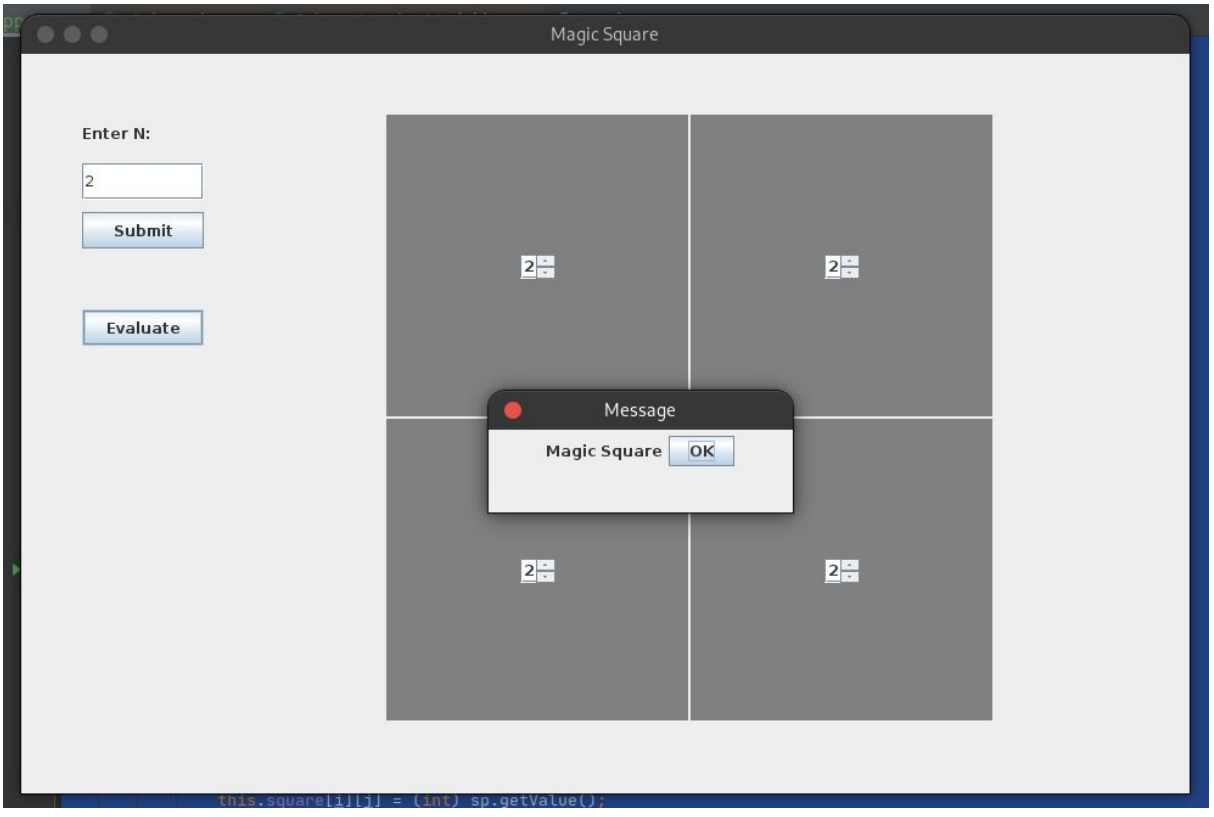
```
        f.add(b3);
        y += dy;

        f.setSize(960, 640);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        var app = new App();
    }

    public void fillSquare() {
        this.square = new int[n][n];
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                JSpinner sp = (JSpinner)
matrix[i][j].getComponents()[0];
                this.square[i][j] = (int) sp.getValue();
            }
        }

        if (isMagicSquare(this.square)) {
            dMessage.setText("Magic Square");
        } else {
            dMessage.setText("Not Magic Square");
        }

        d.setVisible(true);
    }

    boolean isMagicSquare(int[][] mat) {

        // calculate the sum of
        // the prime diagonal
        int N = mat.length;
        int sum = 0, sum2 = 0;
        for (int i = 0; i < N; i++)
            sum = sum + mat[i][i];

        // the secondary diagonal
        for (int i = 0; i < N; i++)
            sum2 = sum2 + mat[i][N - 1 - i];

        if (sum != sum2)
```

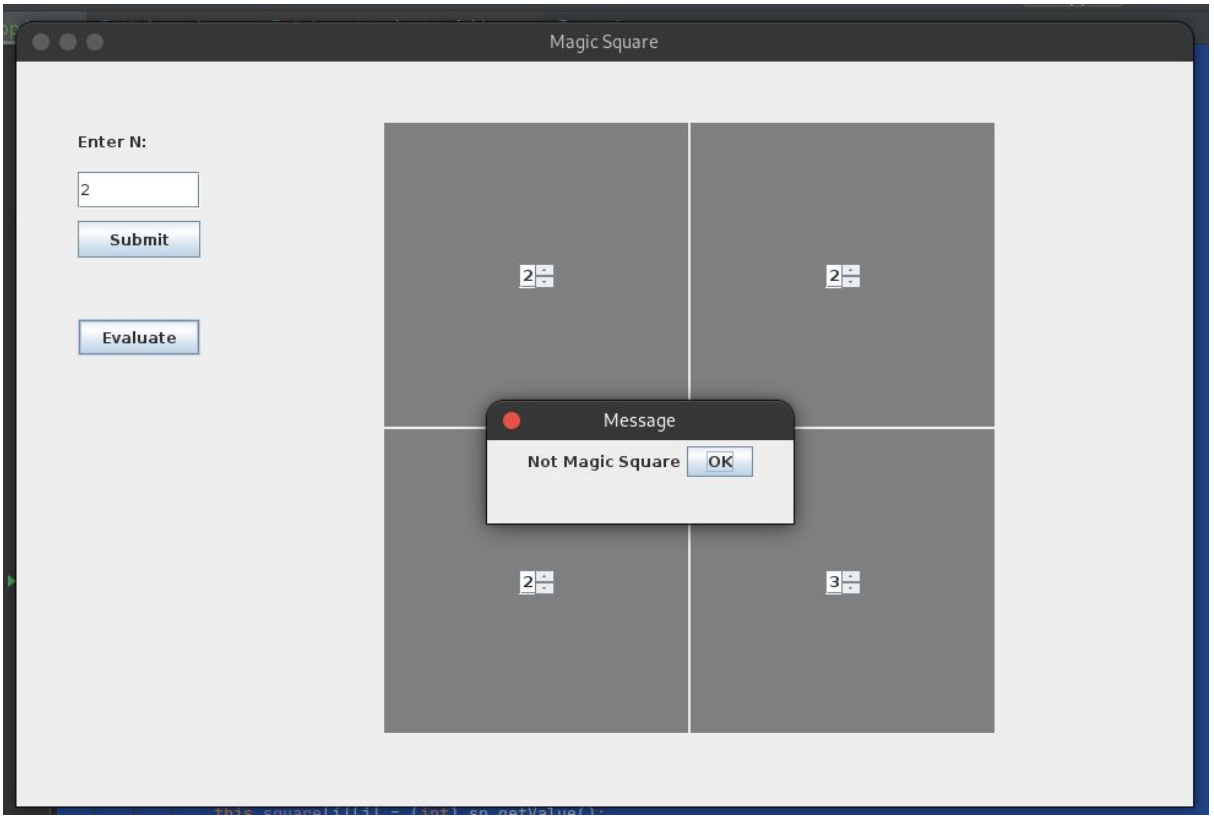
```
            return false;

        // For sums of Rows
        for (int[] ints : mat) {

            int rowSum = 0;
            for (int j = 0; j < N; j++)
                rowSum += ints[j];

            // check if every row sum is
            // equal to prime diagonal sum
            if (rowSum != sum)
                return false;
        }

        // For sums of Columns
        for (int i = 0; i < N; i++) {

            int colSum = 0;
            for (int[] ints : mat) colSum += ints[i];

            // check if every column sum is
            // equal to prime diagonal sum
            if (sum != colSum)
                return false;
        }

        return true;
    }

    public void buildGrid() {
        // clear matrix
        int curr_n = this.matrix.length;

        for (JComponent[] jComponents : this.matrix) {
            for (int j = 0; j < curr_n; ++j) {
                this.f.remove(jComponents[j]);
            }
        }

        this.f.revalidate();


        int dx = 500 / n;
        int dy = 500 / n;
        this.matrix = new JComponent[n][n];
```

```
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                var temp = new JPanel();
                temp.setLayout(new GridBagLayout());
                var gbc = new GridBagConstraints();
                temp.setBounds(300 + i * dx, 50 + j * dy, dx, dy);
                temp.setBackground(Color.GRAY);
                var sp = new JSpinner();
                sp.setSize(dx - 2, dy - 2);
                temp.add(sp, gbc);
                temp.validate();
                matrix[i][j] = temp;
                this.f.add(temp);
                this.f.validate();
            }
        }
    }
}
```




Magic Square
Enter N:
2
Submit
Evaluate
0
0
0
0
this.square[i][j] = (int) sp.getValue();


Magic Square
Enter N:
2
Submit
Evaluate
2
2
Message
Magic Square
OK
2
2
this.square[i][j] = (int) sp.getValue();


Magic Square
Enter N:
2
Submit
Evaluate
2
2
Message
Not Magic Square
OK
2
3