```
/**********************************************************************
 *
 * File:        MagicSquare.java
 *
 * Author:      Dan Gerstl
 *
 * Date:        05/19/2018
 *
 * Purpose:     Project 02
 *
 * Description: CO to test if provided array holds a magic square
 *
 * Comment:     NA
 *
 **********************************************************************/

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

 public class MagicSquare
 {
     /*** Private Constructor ***/

     private MagicSquare()
     {

     }

     /*** Main Method ***/

     public static void checkIfMagicSquare(IntegerTextField[][] square,
                                           int squareSize)
                                           throws IntegerUserInputException,
                                           Exception
     {
         /*** Verify has no empty TextFields ***/

         verifyFullSquare(square, squareSize);

         /*** Verify all input is valid integer within range ***/

         verifyValidIntegers(square, squareSize);

         /*** Verify has no duplcate entries ***/

          verifyNoDuplicates(square, squareSize);

         /*** Verify is Magic Square ***/

          verifyIsMagicSquare(square, squareSize);
     }

     /*** Verification Methods ***/

     private static void verifyFullSquare(IntegerTextField[][] square,
                                          int squareSize) throws Exception
     {
         /*** Local Variables ***/

         boolean fullSquare = true;

         int xCoordinate = 0;
         int yCoordinate = 0;
```

```
        /*** Reset array color ***/

        setTextFieldArrayColor(square, Color.WHITE);

        /*** Trim array elements ***/

        for (int i = 0; i < squareSize; i++)
        {
            for (int j = 0; j < squareSize; j++)
            {
                square[i][j].setText(square[i][j].getText().trim());
            }
        }

        /*** Check for empty elements ***/

        for (int i = 0; i < squareSize; i++)
        {
            for (int j = 0; j < squareSize; j++)
            {
                if (square[i][j].getText().equals(""))
                {
                    /*** Get invalid element location ***/

                    if (fullSquare)
                    {
                        yCoordinate = i;
                        xCoordinate = j;
                    }

                    fullSquare = false;
                }
            }
        }

        /*** Correct coordinates for user orientation ***/

        xCoordinate = xCoordinate + 1;
        yCoordinate = yCoordinate + 1;

        if (!fullSquare)
        {
            /*** Change TextField color of invalid field ***/

            square[yCoordinate - 1][xCoordinate - 1].
            setBackground(Color.RED);

            /*** Throw exception for empty field ***/

            throw new Exception("Square is not complete. Empty field at " +
                                "row [" + yCoordinate + "], column [" +
                                xCoordinate + "].");
        }
    }

    private static void verifyValidIntegers(IntegerTextField[][] square,
                                            int squareSize) throws
                                            IntegerUserInputException
    {
        /*** Check to make sure all fields has valid integer ***/

        for (int i = 0; i < squareSize; i++)
        {
            for (int j = 0; j < squareSize; j++)
```

```
            {
                /*** Change color in case of error ***/

                square[i][j].setBackground(Color.RED);

                /*** Check if valid integer ***/

                square[i][j].getInteger();

                /*** Reset color if no error ***/

                square[i][j].setBackground(Color.WHITE);
            }
        }
    }

    private static void verifyNoDuplicates(IntegerTextField[][] square,
                                           int squareSize) throws Exception
    {
        /*** Local Variables ***/

        boolean noDuplicates = true;

        /*** Reset array color ***/

        setTextFieldArrayColor(square, Color.WHITE);

        /*** Loops for first element selection ***/

        for (int row1 = 0; row1 < squareSize; row1++)
        {
            for (int col1 = 0; col1 < squareSize; col1++)
            {
                /*** Loops for second element selection ***/

                for (int row2 = squareSize - 1; row2 >= row1; row2--)
                {
                    for (int col2 = squareSize - 1; col2 >= 0; col2--)
                    {
                        /*** Exclude same fields ***/

                        if (!(row1 == row2 && col1 == col2))
                        {
                            /*** Check if equal ***/

                            if (square[row1][col1].getInteger() ==
                                square[row2][col2].getInteger())
                            {
                                /*** Change color when duplicate ***/

                                square[row1][col1].setBackground(Color.RED);
                                square[row2][col2].setBackground(Color.RED);

                                noDuplicates = false;
                            }
                        }
                    }
                }
            }
        }

        if (!noDuplicates)
        {
            throw new Exception("Duplicate numbers entered!");
```

```
        }
    }

    private static void verifyIsMagicSquare(IntegerTextField[][] square,
                                            int squareSize) throws
                                            IntegerUserInputException,
                                            Exception
    {
        /*** Local Variables ***/

        boolean validRows        = false;
        boolean validColumns     = false;
        boolean validDiagonals   = false;
        boolean isMagicSquare    = false;

        int sum = 0;

        /*** Calculate sum for square size ***/

        sum = (squareSize * ((squareSize * squareSize) + 1)) / 2;

        /*** Check horizontal numbers ***/

        validRows = validateRows(square, squareSize, sum);

        /*** Check vertical numbers ***/

        validColumns = validateColumns(square, squareSize, sum);

        /*** Check diagonal numbers ***/

        validDiagonals = validateDiagonals(square, squareSize, sum);

        /*** Check if is Magic Square ***/

        if (validRows && validColumns && validDiagonals)
        {
            isMagicSquare = true;
        }

        /*** Provide feedback on if is magic square ***/

        if (isMagicSquare)
        {
            throw new Exception("Square is a Magic Square!");
        }

        else if (!isMagicSquare)
        {
            throw new Exception("Square is not a Magic Square!");
        }
    }

    /*** Validation Helper Methods ***/

    private static boolean validateRows(IntegerTextField[][] array,
                                        int squareSize, int squareSum)
                                        throws IntegerUserInputException
    {
        /*** Local Variables ***/

        boolean allRowsValid = true;

        int rowSum = 0;
```

```
        /*** Loop through array to get row sum ***/

        for (int i = 0; i < squareSize; i++)
        {
            for (int j = 0; j < squareSize; j++)
            {
                /*** Change color in case exception is thrown ***/

                array[i][j].setBackground(Color.RED);

                /*** Calculate sum ***/

                rowSum = rowSum + array[i][j].getInteger();

                /*** Revert color if exception isnt thrown ***/

                array[i][j].setBackground(Color.WHITE);
            }

            /*** Check if row sum equals desired square sum ***/

            if (rowSum != squareSum)
            {
                allRowsValid = false;
            }

            /*** Reset sum ***/

            rowSum = 0;
        }

        return allRowsValid;
    }

    private static boolean validateColumns(IntegerTextField[][] array,
                                           int squareSize, int squareSum)
                                           throws IntegerUserInputException
    {
        /*** Local Variables ***/

        boolean allColumnsValid = true;

        int columnSum = 0;

        /*** Loop through array to get column sum ***/

        for (int i = 0; i < squareSize; i++)
        {
            for (int j = 0; j < squareSize; j++)
            {
                /*** Change color in case exception is thrown ***/

                array[j][i].setBackground(Color.RED);

                /*** Calculate sum ***/

                columnSum = columnSum + array[j][i].getInteger();

                /*** Revert color if exception isnt thrown ***/

                array[j][i].setBackground(Color.WHITE);
            }
```

```
            /*** Check if column sum equals desired square sum ***/

            if (columnSum != squareSum)
            {
                allColumnsValid = false;
            }

            /*** Reset sum ***/

            columnSum = 0;
        }

        return allColumnsValid;
    }

    private static boolean validateDiagonals(IntegerTextField[][] array,
                                             int squareSize, int squareSum)
                                             throws IntegerUserInputException
    {
        /*** Local Variables ***/

        boolean allDiagonalsValid = true;

        int diagonalSum = 0;

        /*** Check top left to bottom right diagonal sum ***/

        for (int i = 0; i < squareSize; i++)
        {
            /*** Change color in case exception is thrown ***/

            array[i][i].setBackground(Color.RED);

            /*** Calculate sum ***/

            diagonalSum = diagonalSum + array[i][i].getInteger();

            /*** Revert color if exception isnt thrown ***/

            array[i][i].setBackground(Color.WHITE);
        }

        /*** Check if diagonal sum equals desired square sum ***/

        if (diagonalSum != squareSum)
        {
            allDiagonalsValid = false;
        }

        /*** Reset sum ***/

        diagonalSum = 0;

        /*** Check bottom left to top right diagonal sum ***/

        for (int i = squareSize - 1; i >= 0; i--)
        {
            /*** Change color in case exception is thrown ***/

            array[i][i].setBackground(Color.RED);

            /*** Calculate sum ***/

            diagonalSum = diagonalSum + array[i][i].getInteger();
```

```
            /*** Revert color if exception isnt thrown ***/

            array[i][i].setBackground(Color.WHITE);
        }

        /*** Check if diagonal sum equals desired square sum ***/

        if (diagonalSum != squareSum)
        {
            allDiagonalsValid = false;
        }

        return allDiagonalsValid;
    }

    /*** Helper Methods ***/

    public static void setTextFieldArrayColor(IntegerTextField[][] array,
                                              Color color)
    {
        for (int i = 0; i < array.length; i++)
        {
            for (int j = 0; j < array.length; j++)
            {
                array[i][j].setBackground(color);
            }
        }
    }
}
```