App.java

```
package app;

/**
 *
 * @custom.application_name App
 * @custom.class_name App
 *
 * @custom.author Daniel C. Landon Jr.
 * @custom.instructor Dr. Bob Walsh
 * @custom.course CSCI 202 - Introduction to Software Systems
 * @custom.date_started 02.05.2020
 * @custom.date_due 02.06.2020
 *
 * @custom.class_notes Entry point/clinet for Matrix Class testing
 *
 * @custom.pre_condition none
 *
 * @custom.post_condition none
 *
 * @custom.javadoc_tags In order to use @custom.tag_name in javadocs you must include the folloinwing in th
e command line to generate the docs. This part must be after you have indicated what files to process;
 *
 *  ' -tag custom.tag_name:a:"tag_name" '
 *
 * The first part identifies the tag in the code, the second part in quotes indentifies what will be printed in the jav
adocs when they are generated. If you do not include this in the command to generate the docs you will get an e
rror/warning.
 *
 */

public class App {

    /**
     *
     * @custom.method_name main
     *
     * @custom.author Daniel C. Landon Jr.
     * @custom.date_started 02.05.2020
     *
     * @custom.method_notes entry point, no args
     *
     * @custom.pre_condition none
     *
     * @custom.post_condition PROGRAM TERMINATED. END OF LINE.
     *
     */
    public static void main(String[] args) throws Exception {
```

```
        // constants
        final int _ROWS = 2;
        final int _COLUMNS = 2;

        // create base matrix for processing
        Matrix _baseMatrix = new Matrix(_ROWS, _COLUMNS);
        System.out.println("\nBase Matrix Used For Processing In Application.\n" + _baseMatrix.toString());
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        // end base matrix creation

        // multiply matrix
        Matrix _multiplyMatrix = new Matrix(_ROWS, _COLUMNS);
        System.out.println("\nMatrix used to multiply against _baseMatrix.\n" + _multiplyMatrix.toString());
        System.out.println("\nResults of multiplying _multiplyMatrix against _baseMatrix.\n" + _baseMatrix.matrixMultiply(_multiplyMatrix));
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        // end multiply matrix

        // copy matrix
        Matrix _copyMatrix = new Matrix(_ROWS, _COLUMNS);
        System.out.println("\n_copyMatrix before copy operation: \n" + _copyMatrix.toString());
        _copyMatrix.matrixCopy(_baseMatrix);
        System.out.println("\n_copyMatrix after copy operation: \n" + _copyMatrix.toString());
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        // end copy matrix

        // equals matrix
        Matrix _equalsMatrix = new Matrix(_ROWS, _COLUMNS);
        System.out.println("\nMatrix used to compare against _baseMatrix:\n" + _equalsMatrix.toString());
        System.out.println("\nDoes _equalsMatrix equal _baseMatrix? " + _baseMatrix.matrixEquals(_equalsMatrix) + "\n");
        System.out.println("\nDoes _baseMatrix equal itself? " + _baseMatrix.matrixEquals(_baseMatrix) + "\n");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        // end equals matrix

        // scalar multiply
        int _scalarMultiplier = 2;
        System.out.println("\n_baseMatrix before scalar multiplication operation:\n" + _baseMatrix.toString());
        System.out.println("Multiply _baseMatrix by: " + _scalarMultiplier);
        _baseMatrix.matrixScalarMultiply(_scalarMultiplier);
        System.out.println("\n_baseMatrix after scalar multiplication operation: \n" + _baseMatrix.toString());
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        // end scalar multiply

        // matrix add
        Matrix _addMatrix = new Matrix(_ROWS, _COLUMNS);
        System.out.println("\n_baseMatrix before add operation:\n" + _baseMatrix.toString());
        System.out.println("\n_addMatrix before add operation:\n" + _addMatrix.toString());
        _baseMatrix.matrixAdd(_addMatrix);
```

```
        System.out.println("\n_baseMatrix after add operation:\n" + _baseMatrix.toString());
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        // end matrix add

    } // end main

}  // end app
```

Matrix.java

```
package app;

import java.util.Arrays;

/**
 *
 * @custom.class_name Matrix
 *
 * @custom.author Daniel C. Landon Jr.
 * @custom.instructor Dr. Bob Walsh
 * @custom.course CSCI 202 - Introduction to Software Systems
 * @custom.date_started 02.05.2020
 * @custom.date_due 02.06.2020
 *
 * @custom.class_notes This class generates a matrix using a constructor with
 *                     argumens. It has various methods that can be called to
 *                     manipulate the matrix
 *
 * @custom.pre_condition none
 *
 * @custom.post_condition none
 *
 * @custom.javadoc_tags In order to use @custom.tag_name in javadocs you must
 *                      include the folloinwing in the command line to generate
 *                      the docs. This part must be after you have indicated
 *                      what files to process;
 *
 *                      ' -tag custom.tag_name:a:"tag_name" '
 *
 *                      The first part identifies the tag in the code, the
 *                      second part in quotes indentifies what will be printed
 *                      in the javadocs when they are generated. If you do not
 *                      include this in the command to generate the docs you
 *                      will get an error/warning.
 *
 */

public class Matrix {

    // class variables
    private int _row = 0; // rows
    private int _col = 0; // columns
    private int[][] _data; // Keanu

    /**
     *
     * @custom.method_name Matrix constructor
```

```
     *
     * @custom.author Daniel C. Landon Jr.
     * @custom.date_started 02.05.2020
     *
     * @custom.method_notes Constructor that creates the initial matrix
     *
     * @custom.pre_condition create instance of object and supply starting values to create a matrix.
     *
     * @custom.post_condition matrix is created
     *
     * @param _row the number of rows to create
     * @param _col the number of columns to create
     */
    public Matrix(int _row, int _col) {
        // set internal variables
        this._row = _row;
        this._col = _col;

        //initialize the matrix
        this._data = new int[this._row][this._col]; // keanu lives

        // loop _loopRow
        for(int _loopRow = 0; _loopRow < this._row; _loopRow++){

            // loop _loopCol
            for(int _loopCol = 0; _loopCol < this._col; _loopCol++){

                // populate the matrix
                this._data[_loopRow][_loopCol] = (int)(Math.random() * 4 + 1);

            } // end for _loopCol

        } // end for _loopRow

    } // end Matrix constructor

    /**
     *
     * @custom.method_name toString
     *
     * @custom.author Daniel C. Landon Jr.
     * @custom.date_started 02.05.2020
     *
     * @custom.method_notes Returns a string that contains the matrice.
     *
     * @custom.pre_condition matrix must be created
     *
     * @custom.post_condition string containing the matrix is returned
     *
     * @return the matrix in string format
```

```
     */
    public String toString(){

        // varaibles
        String _ans = "";

        // loop row
        for(int _loopRow = 0; _loopRow < this._row; _loopRow++){

            // loop col
            for(int _loopCol = 0; _loopCol < this._col; _loopCol++){

                // create return for matrix
                _ans+= _data[_loopRow][_loopCol] + "\t";

            } // end for _loopCol

            // create final output
            _ans+= "\n";

        } // end for _loopRow

        return _ans;

    } // end toString

    /**
     *
     * @custom.method_name matrixAdd
     *
     * @custom.author Daniel C. Landon Jr.
     * @custom.date_started 02.05.2020
     *
     * @custom.method_notes Takes two matricies and adds them together
     *
     * @custom.pre_condition two matrix must be supplied
     *
     * @custom.post_condition return a matrix containing the results of adding two supplied matrix together.
     *
     * @param _matrix matrix to add to _data matrix
     * @return matrix with added values otherwise echos back supplied matrix from args
     */
    public Matrix matrixAdd(Matrix _matrix) {

        // loop the row
        for(int _rowLoop = 0; _rowLoop < this._row; _rowLoop++) {

            // loop column
            for(int _colLoop = 0; _colLoop < this._col; _colLoop++) {
```

```
            // do the math
            this._data[_rowLoop][_colLoop] = this._data[_rowLoop][_colLoop] + _matrix._data[_rowLoop][_col
Loop];

        } // end _colLoop

    } // end _rowLoop

    // if we get here there is a problem so echo _matrixArgs
    return _matrix;
  } // end matrixAdd

  /**
   *
   * @custom.method_name matrixScalarMultiply
   *
   * @custom.author Daniel C. Landon Jr.
   * @custom.date_started 02.05.2020
   *
   * @custom.method_notes multiples this._data by supplied number
   *
   * @custom.pre_condition this._data must exist
   *
   * @custom.post_condition new matrix containg the multipled values
   *
   * @param _numberToMultiply number to multiply against this._data
   */
  public void matrixScalarMultiply(int _numberToMultiply) {

    for(int _rowLoop = 0; _rowLoop < this._row; _rowLoop++){

        for(int _colLoop = 0; _colLoop < this._col; _colLoop++) {

            // multiply contents of matrix element by multiplier
            this._data[_rowLoop][_colLoop] = this._data[_rowLoop][_colLoop] * _numberToMultiply;

        } // end _columnLoop

    } // end _rowLoop

  } // end matrixScalarMultiply

  /**
   *
   * @custom.method_name matrixEquals
   *
   * @custom.author Daniel C. Landon Jr.
   * @custom.date_started 02.05.2020
   *
   * @custom.method_notes takes a supplied matrix and compares it to this._data
```

```
     *
     * @custom.pre_condition this._data must exist
     *
     * @custom.post_condition successfull comparison
     *
     * @param _matrix matrix to compare to this._data
     * @return true if the match false otherwise
     */
    public boolean matrixEquals(Matrix _matrix) {

        boolean _testCondition =  Arrays.equals(this._data, _matrix._data) ?  true : false;

        return _testCondition; // we get here there was a problem

    } // end matrixEquals

    /**
     *
     * @custom.method_name matrixCopy
     *
     * @custom.author Daniel C. Landon Jr.
     * @custom.date_started 02.05.2020
     *
     * @custom.method_notes none
     *
     * @custom.pre_condition matrix must be supplied to copy to additionally the default data matrix must also e
xist
     *
     * @custom.post_condition return a copy of existing matrix
     *
     * @param _matrix the matrix that we will copy too
     */
    public void matrixCopy(Matrix _matrix) {

        // this works, i can conceptualize it but I cannot explain it.
        // i need to work on this one
        this._data = Arrays.stream(_matrix._data)
            .map((int[] row) -> row.clone())
            .toArray((int length) -> new int[length][]);

    } // end matrixCopy

    /**
     *
     * @custom.method_name matrixMultiply
     *
     * @custom.author Daniel C. Landon Jr.
     * @custom.date_started 02.05.2020
     *
     * @custom.method_notes takes two matrices and multiplies them together
```

```
    *
    * @custom.pre_condition two matrix must be supplied
    *
    * @custom.post_condition return a matrix containing the results of multiplying two supplied matrix together
.
    *
    * @param _m2 matrix to multiply against this._data
    * @return matrix containing multipled matrix, if problem will echo back supplied matrix from args
    */
   public Matrix matrixMultiply(Matrix _m2) {

      try {

         // check to see if this._col = _m2._row
         if(this._col != _m2._row){

            System.out.println("Matrix Size Incorrect!");

            throw new RuntimeException();

         } // end if

         // create new matrix to contain new values
         Matrix _m3 = new Matrix(this._row, _m2._col);

         // loop row
         for(int _loopRow = 0; _loopRow < this._data.length; _loopRow++){

            // loop col
            for(int _loopCol = 0; _loopCol < this._data.length; _loopCol++) {

               // new matrix contains daat so set to zero
               _m3._data[_loopRow][_loopCol] = 0;

               // loop k
               for(int _x = 0; _x < _m2._data.length; _x++) {

                  // do the math and update matrix
                  _m3._data[_loopRow][_loopCol] += this._data[_loopRow][_x] * _m2._data[_x][_loopCol];

               } // end for _x

            } // end for _loopCol

         } // end for _loopRow

         return _m3;

      } // end try
```

```
        catch (Exception e) {
            System.out.println("WOOT BAM! ... Sumo Ninja Strikes");
        } // end catch

        // if we get here the arrays were not multiplied so echo arg array
        return _m2;

    } // end mulltiplyMatrix

} // end Matrix
```

CONSOLE OUTPUT

Base Matrix Used For Processing In Application.
4 3
3 1

~~~~~~

Matrix used to multiply against _baseMatrix.
4 4
3 4

Results of multiplying _multiplyMatrix against _baseMatrix.
25 28
15 16

~~~~~~

_copyMatrix before copy operation:
4 1
3 2

_copyMatrix after copy operation:
4 3
3 1

~~~~~~

Matrix used to compare against _baseMatrix:
1 4
4 3

Does _equalsMatrix equal _baseMatrix? false

Does _baseMatrix equal itself? true

~~~~~~

_baseMatrix before scalar multiplication operation:
4 3
3 1

Multiply _baseMatrix by: 2

_baseMatrix after scalar multiplication operation:

8 6
6 2

~~~~~~

_baseMatrix before add operation:
8 6
6 2

_addMatrix before add operation:
4 1
1 3

_baseMatrix after add operation:
12 7
7 5

~~~~~~