My string

```java
public class MyString {
    public static void main(String []args) {
        // Calls parseInt, and adds 1 to the returned value,
        // to verify that the returned value is indeed the correct int.
        System.out.println(parseInt("5613") + 1);
        System.out.println(parseInt("9a7"));
    }

    /**
     * Returns the integer value of the given string of digit characters,
     * or -1 if the string contains one or more non-digit characters.
     */
    public static int parseInt(String str)
    {
        int nnum=0;
        for( int i=0; i<str.length(); i++)
        {
            if((int)str.charAt(i)<=47||(int)str.charAt(i)>=58)
                return -1;
        }

        for( int j=0; j<str.length(); j++)
        {
            int realN= ((int)str.charAt(j)-48);
            nnum= nnum + (realN*(int)(Math.pow(10,str.length()-(j+1))));
        }
        return nnum;
    }
}
```

My Arrays

MatrixOps.java

```java
public class MyArrays {

    // Two arrays, for testing purposes. Used by the testing methods in this class.
    private static final int[] a = { 2, 4, 2, 5};
    private static final int[] b = { 3, 6, 9};

    /**
     * If every element in the array is greater than or equal to the previous element, returns true.
     * Otherwise, returns false.
     */
    public static boolean isInIncreasingOrder(int[] arr)
    {
        for(int i=1; i<arr.length; i++)
        {
        if(arr[i]<arr[i-1])
            return false;
        }
        return true;
    }

    /**
     * Returns an array whose elements consist of all the elements of arr1,
     * followed by all the elements of arr2.
     */
    public static int[] concat(int[] arr1, int[] arr2)
    {
        int[] bigarr = new int [arr1.length+arr2.length];

        for (int k=0; k<arr1.length; k++)
            bigarr[k]=arr1[k];
```

INS UTF-8 Windows (CR LF) Ln : 2 Col : 1 Pos : 26 length : 2,637 lines : 90

MatrixOps.j

```java
     */
    public static int[] concat(int[] arr1, int[] arr2)
    {
        int[] bigarr = new int [arr1.length+arr2.length];

        for (int k=0; k<arr1.length; k++)
            bigarr[k]=arr1[k];

        for (int j=0; j<arr2.length; j++)
            bigarr[arr1.length+j]=arr2[j];

        return bigarr;
    }

    /** If the given array contains an element that appears more than once, returns true.
     *  Otherwise, returns false. */
    public static boolean hasDuplicates(int[] arr)
    {
        for( int i=0; i<arr.length; i++)
        {
            for ( int j=i+1; j<arr.length; j++)
            {
                if(arr[i]==arr[j])
                    return true;
            }
        }
        return false;
    }

    // Prints the given int array, and then prints an empty line.
    public static void println(int[] arr) {
```

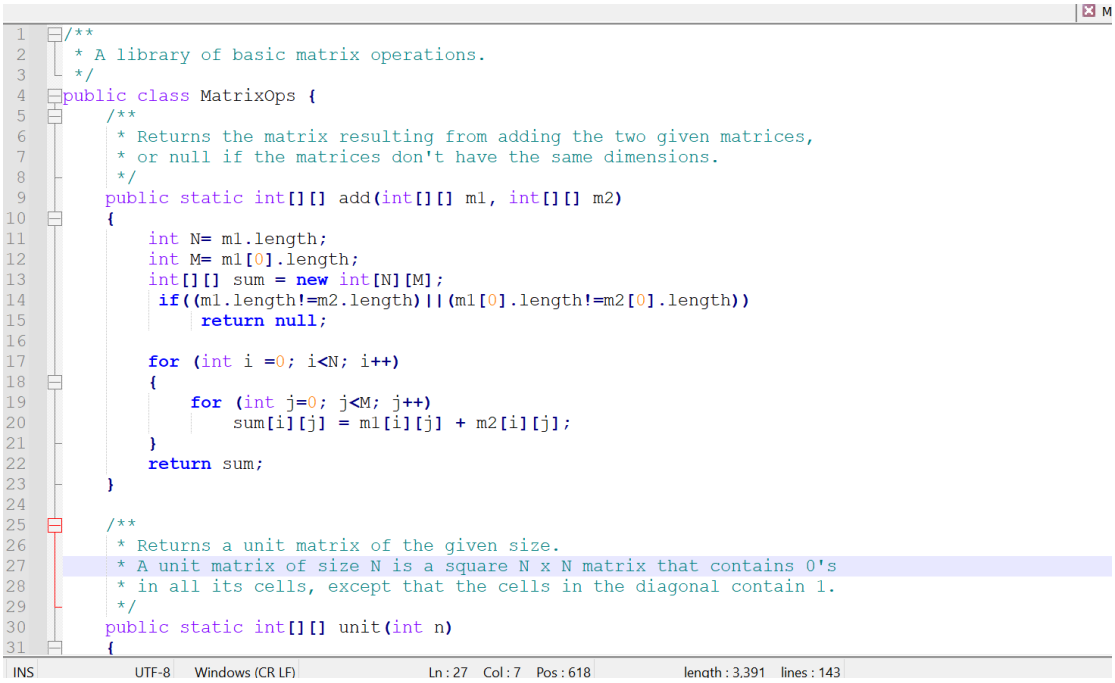
INS UTF-8 Windows (CR LF) Ln : 2 Col : 1 Pos : 26 length : 2,637 lines : 90

```
51      }
52
53      // Prints the given int array, and then prints an empty line.
54      public static void println(int[] arr) {
55          for (int i = 0; i < arr.length; i++) {
56              System.out.print(arr[i] + " ");
57          }
58          System.out.println();
59      }
60
61      public static void main(String[] args)
62      {
63          System.out.print("Array a: "); println(a);
64          System.out.print("Array b: "); println(b);
65          //// Uncomment the test that you wish to execute
66          testIsInIncreasingOrder();
67          testConcat();
68          testHasDuplicates();
69          ////testIsInIncreasingOrder();
70          ////testConcat();
71          ////testHasDuplicates();
72      }
73
74      private static void testIsInIncreasingOrder() {
75          System.out.println();
76          System.out.println("Array a is " + ((isInIncreasingOrder(a)) ? "" : "not ") + "in order");
77          System.out.println("Array b is " + ((isInIncreasingOrder(b)) ? "" : "not ") + "in order");
78      }
79
80      private static void testConcat() {
81          System.out.println();
```

INS UTF-8 Windows (CR LF) Ln : 2 Col : 1 Pos : 26 length : 2,637 lines : 90

```
64          System.out.print("Array b: "); println(b);
65          //// Uncomment the test that you wish to execute
66          testIsInIncreasingOrder();
67          testConcat();
68          testHasDuplicates();
69          ////testIsInIncreasingOrder();
70          ////testConcat();
71          ////testHasDuplicates();
72      }
73
74      private static void testIsInIncreasingOrder() {
75          System.out.println();
76          System.out.println("Array a is " + ((isInIncreasingOrder(a)) ? "" : "not ") + "in order");
77          System.out.println("Array b is " + ((isInIncreasingOrder(b)) ? "" : "not ") + "in order");
78      }
79
80      private static void testConcat() {
81          System.out.println();
82          System.out.print("Concatenantion of a and b: "); println(concat(a, b));
83      }
84
85      private static void testHasDuplicates() {
86          System.out.println();
87          System.out.println("Array a has " + ((hasDuplicates(a)) ? "" : "no ") + "duplicates");
88          System.out.println("Array b has " + ((hasDuplicates(b)) ? "" : "no ") + "duplicates");
89      }
90  }
```

MatrixOps

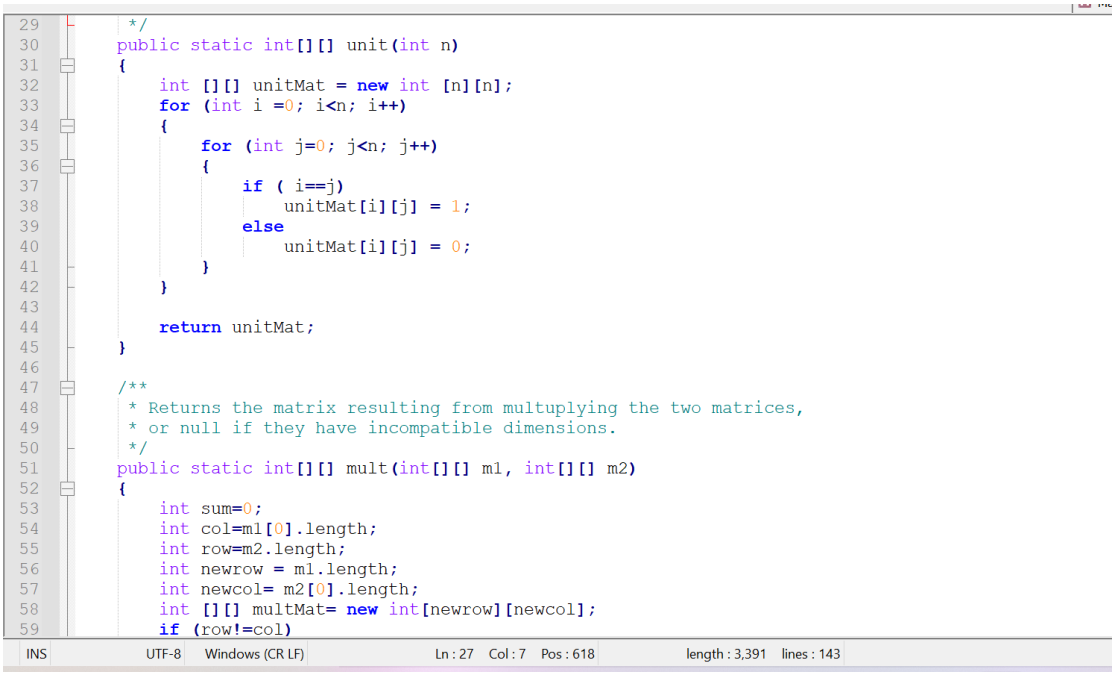


```java
/**
 * A library of basic matrix operations.
 */
public class MatrixOps {
    /**
     * Returns the matrix resulting from adding the two given matrices,
     * or null if the matrices don't have the same dimensions.
     */
    public static int[][] add(int[][] m1, int[][] m2)
    {
        int N= m1.length;
        int M= m1[0].length;
        int[][] sum = new int[N][M];
         if((m1.length!=m2.length)||(m1[0].length!=m2[0].length))
             return null;

        for (int i =0; i<N; i++)
        {
            for (int j=0; j<M; j++)
                sum[i][j] = m1[i][j] + m2[i][j];
        }
        return sum;
    }

    /**
     * Returns a unit matrix of the given size.
     * A unit matrix of size N is a square N x N matrix that contains 0's
     * in all its cells, except that the cells in the diagonal contain 1.
     */
    public static int[][] unit(int n)
    {
```

INS UTF-8 Windows (CR LF) Ln : 27 Col : 7 Pos : 618 length : 3,391 lines : 143



```java
     */
    public static int[][] unit(int n)
    {
        int [][] unitMat = new int [n][n];
        for (int i =0; i<n; i++)
        {
            for (int j=0; j<n; j++)
            {
                if ( i==j)
                    unitMat[i][j] = 1;
                else
                    unitMat[i][j] = 0;
            }
        }

        return unitMat;
    }

    /**
     * Returns the matrix resulting from multuplying the two matrices,
     * or null if they have incompatible dimensions.
     */
    public static int[][] mult(int[][] m1, int[][] m2)
    {
        int sum=0;
        int col=m1[0].length;
        int row=m2.length;
        int newrow = m1.length;
        int newcol= m2[0].length;
        int [][] multMat= new int[newrow][newcol];
        if (row!=col)
```

INS UTF-8 Windows (CR LF) Ln : 27 Col : 7 Pos : 618 length : 3,391 lines : 143

```
57          int newcol= m2[0].length;
58          int [][] multMat= new int[newrow][newcol];
59          if (row!=col)
60              return null;
61          else
62          {
63              for(int i=0; i<newrow; i++)
64              {
65                  for (int j=0; j<newcol; j++)
66                  {
67                      for (int k=0; k<col; k++)
68                      {
69                          sum=sum+(m1[i][k]*m2[k][j]);
70                      }
71                      multMat[i][j]=sum;
72                      sum=0;
73                  }
74              }
75
76          }
77
78          return multMat;
79      }
80
81      /**
82       * Returns a matrix which is the transpose of the given matrix.
83       */
84      public static int[][] transpose(int[][] m)
85      {
86          int rows = m.length;
87          int columns = m[0].length;
```

```
83       */
84      public static int[][] transpose(int[][] m)
85      {
86          int rows = m.length;
87          int columns = m[0].length;
88          int [][] transMat = new int [columns][rows];
89
90          for(int i=0; i<rows; i++)
91          {
92              for (int j=0; j<columns; j++)
93              {
94                  transMat[j][i]=m[i][j];
95              }
96          }
97
98          return transMat;
99      }
100
101     /**
102      * Prints the given matrix, and then prints an empty line.
103     l */
104     public static void println(int[][] m) {
105         for (int row = 0; row < m.length; row++) {
106             for (int col = 0; col < m[1].length; col++) {
107                 System.out.print(m[row][col] + "  ");
108             }
109             System.out.println();
110         }
111         System.out.println();
112     }
113
```

```
113
114     /**
115      * Tests all the matrix operations featured by this class.
116      */
117     public static void main(String args[]) {
118         // Creates two matrices, for testing
119         int[][] a = { { 1, 2, 1 },
120                       { 0, 1, 1 },
121                       { 2, 0, 1 } };
122
123         int[][] b = { { 1, 0, 2 },
124                       { 1, 2, 0 },
125                       { 2, 0, 1 } };
126         System.out.println("Matrix A:");  println(a);
127         System.out.println("Matrix B:");  println(b);
128
129         //// Uncomment the statements that you wish to execute.
130         System.out.println("A + B:");  println(add(a,b));
131         System.out.println("A * B:");  println(mult(a,b));
132         System.out.println("I (a unit matrix of size 3):"); println(unit(3));
133         System.out.println("A * I: "); println(mult(a,unit(3)));
134
135         int[][] c = { { 1, 2, 3 },
136                       { 4, 5, 6 }, };
137         System.out.println("Matrix C:"); println(c);
138         System.out.println("C, transposed:"); println(transpose(c));
139
140         ////System.out.println("Random sum test, using matrix C:");
141         ////sumRandom(c);
142     }
143 }
```

INS UTF-8 Windows (CR LF) Ln : 27 Col : 7 Pos : 618 length : 3,391 lines : 143