```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 m1Rows = m1.length;
        int m1Colums = m1[0].length;
        int m2Rows = m2.length;
        int m2Colums = m2[0].length;
        String message = "" + m1Colums + " " + m2Rows;
        System.out.println(message);
        if (m1Colums != m2Colums || m2Rows != m1Rows) {
            return null;
        }

        int[][] newMatrix = new int[m1Rows][m1Colums];
        for (int i = 0; i < m1Rows; i++) {
            for (int j = 0; j < m1Colums; j++) {
                newMatrix[i][j] = m1[i][j] + m2[i][j];
            }
        }
        return newMatrix;
    }

    /**
     * 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) {
        int[][] imatrix;
        imatrix = new int[n][n];
        for (int row = 0; row < n; row++) {
            for (int col = 0; col < n; col++) {
                if (row == col) {
                    imatrix[row][col] = 1;
                } else {
                    imatrix[row][col] = 0;
                }
            }
        }
        return imatrix;
    }

    /**
     * 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 m1Rows = m1.length;
        int m1Colums = m1[0].length;
        int m2Rows = m2.length;
        int m2Colums = m2[0].length;
        int[][] product = new int[m1Rows][m2Colums];

        if (m1Colums != m2Rows) {
            return null;
        }

        for(int i = 0; i < m1Rows; i++) {
            for (int j = 0; j < m2Colums; j++) {
                for (int k = 0; k < m1Colums; k++) {
                    product[i][j] += m1[i][k] * m2[k][j];
                }
            }
        }

        return product;
    }

    /**
     * Returns a matrix which is the transpose of the given matrix.
     */
    public static int[][] transpose(int[][] m) {
        int mRows = m.length;
        int mColums = m[0].length;
        int transpose[][] = new int[mColums][mRows];

        for (int i = 0; i < mColums; i++){
            for (int j = 0; j < mRows; j++){
                transpose[i][j] = m[j][i];
            }
        }

        return transpose;
    }

    /**
     * Prints the given matrix, and then prints an empty line.
     */
    public static void println(int[][] m) {
        for (int row = 0; row < m.length; row++) {
            for (int col = 0; col < m[1].length; col++) {
                System.out.print(m[row][col] + "  ");
            }
            System.out.println();
        }
        System.out.println();
    }

    /**
     * Tests all the matrix operations featured by this class.
     */
    public static void main(String args[]) {
        int[][] a = { { 1, 2, 1 },
                      { 0, 1, 1 },
                      { 2, 0, 1 } };

        int[][] b = { { 1, 0, 2 },
                      { 1, 2, 0 },
                      { 2, 0, 1 } };

        System.out.println("Matrix A:");  println(a);
        System.out.println("Matrix B:");  println(b);

        System.out.println("A + B:");  println(add(a, b));
        System.out.println("B + A:");  println(add(b, a));
        System.out.println("I (a unit matrix of size 3):"); println(unit(3));

        int[][] c = { { 1, 2, 3 },
                      { 4, 5, 6 }, };
        System.out.println("A * B:");  println(mult(a,b));
        System.out.println("A * I: "); println(mult(a,unit(3)));

        System.out.println("Matrix C:"); println(c);
        System.out.println("C, transposed:"); println(transpose(c));
    }
}
```

```java

import java.util.Arrays;
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) {
        boolean isInIncreasingOrder = true;
        for (int i = 0; i < arr.length; i++) {
            if (i > 0 && arr[i] < arr[i - 1]) {
                isInIncreasingOrder = false;
            }
        }
        return isInIncreasingOrder;
    }

    /**
     * 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[] both = Arrays.copyOf(arr1, arr1.length + arr2.length);
        System.arraycopy(arr2, 0, both, arr1.length, arr2.length);
        return both;
    }

    /** If the given array contains an element that appears more than once, returns true.
     *  Otherwise, returns false. */
    public static boolean hasDuplicates(int[] arr) {
        //// Replace the following statement with your code
        boolean duplicates = false;
        for (int firstCounter = 0; firstCounter < arr.length; firstCounter++) {
            for (int secondCounter = firstCounter + 1; secondCounter < arr.length; secondCounter++) {
                if (secondCounter != firstCounter && arr[secondCounter] == arr[firstCounter]) {
                    duplicates = true;
                }
            }
        }

        return duplicates;
    }

    // Prints the given int array, and then prints an empty line.
    public static void println(int[] arr) {
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }

    public static void main(String[] args) {
        System.out.print("Array a: "); println(a);
        System.out.print("Array b: "); println(b);
        //// Uncomment the test that you wish to execute
        testIsInIncreasingOrder();
        testConcat();
        testHasDuplicates();
    }

    private static void testIsInIncreasingOrder() {
        System.out.println();
        System.out.println("Array a is " + ((isInIncreasingOrder(a)) ? "" : "not ") + "in order");
        System.out.println("Array b is " + ((isInIncreasingOrder(b)) ? "" : "not ") + "in order");
    }

    private static void testConcat() {
        System.out.println();
        System.out.print("Concatenantion of a and b: "); println(concat(a, b));
    }

    private static void testHasDuplicates() {
        System.out.println();
        System.out.println("Array a has " + ((hasDuplicates(a)) ? "" : "no ") + "duplicates");
        System.out.println("Array b has " + ((hasDuplicates(b)) ? "" : "no ") + "duplicates");
    }
}
```

```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 num = 0;
        for(int i = 0; i < str.length(); i++) {
            if(str.charAt(i) >= 48 && str.charAt(i) <= 57) {
                num = num * 10 + str.charAt(i) - 48;
            } else {
                return -1;
            }
        }

        return num;
    }
}
```