```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 = 0; i < arr.length; i++) {
            if (i != arr.length - 1) {
                int current = arr[i];
                int next = arr[i + 1];
                if (next < current) {
                    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[] combine = new int[(arr1.length + arr2.length)];
        for (int i = 0; i < combine.length; i++) {
            if (i < arr1.length) {
                combine[i] = arr1[i];
            } else {
                if (i - arr1.length < arr2.length) {
                    combine[i] = arr2[i - arr1.length];
                }
            }
        }
        return combine;
    }

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

```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[][] matrix = new int[m1.length][m1[0].length];
        for (int i = 0; i < m1.length; i++) {
            for (int j = 0; j < m1[i].length; j++) {
                if (j >= m2[i].length) {
                    return null;
                } else {
                    matrix[i][j] = m1[i][j] + m2[i][j];
                }
            }
        }
        return matrix;
    }

    /**
     * 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[][] matrix = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (j == i) {
                    matrix[i][j] = 1;
                } else {
                    matrix[i][j] = 0;
                }
            }
        }
        return matrix;
    }

    /**
     * Returns the matrix resulting from multiplying the two matrices, or null if
     * they have incompatible dimensions.
     */
    public static int[][] mult(int[][] m1, int[][] m2) {
        int[][] matrix = new int[m1.length][m2[0].length];
        if (m1[0].length != m2.length) {
            return null;
        }
        for (int i = 0; i < m1.length; i++) {
            for (int j = 0; j < m2[0].length; j++) {
                for (int k = 0; k < m1[0].length; k++) {
                    matrix[i][j] += m1[i][k] * m2[k][j];
                }
            }
        }
        return matrix;
    }

    /**
     * Returns a matrix which is the transpose of the given matrix.
     */
    public static int[][] transpose(int[][] m) {
        int[][] matrix = new int[m[0].length][m.length];
        for (int i = 0; i < m[0].length; i++) {
            for (int j = 0; j < m.length; j++) {
                matrix[i][j] = m[j][i];
            }
        }
        return matrix;
    }

    /**
     * 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[]) {
        // Creates two matrices, for testing
        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("A * B:");
        println(mult(a, b));
        System.out.println("I (a unit matrix of size 3):");
        println(unit(3));
        System.out.println("A * I: ");
        println(mult(a, unit(3)));
        int[][] c = { { 1, 2, 3 }, { 4, 5, 6 }, };
        System.out.println("Matrix C:");
        println(c);
        System.out.println("C, transposed:");
        println(transpose(c));
    }
}
```