```
public class ArrayOps {
    public static void main(String[] args) {

        int[] testCase1 = {2,8,3,7,8};

        System.out.println(secondMaxValue(testCase1));
    }

    public static int findMissingInt (int[] array) {
        // Write your code here:
        int n = array.length;

            // for loops have a condition / as long as the condition is true - the for loop
runs
        for (int i = 0; i <= n; i++) {
            boolean number = false;
            for (int index = 0; index < array.length; index++) {
                if (array[index] == i) {
                    number = true;
                }
            }
            if (number == false) {
               return i;
            }
        }
        return 1;
    }

    public static int secondMaxValue(int [] array) {
        int l = array.length;
        int large = 0;
        int secondLarger = 0;

        for (int i = 0; i < l; ++i) {
            if (array[i] > secondLarger) {
                secondLarger = large;
                large = array[i];
            }
        }
        return secondLarger;
    }


    public static boolean containsTheSameElements(int [] array1,int [] array2) {
        int n = array1.length;
        int m = array2.length;

        for (int i = 1; i < n; i++ ) { // first array check
           for (int j = 0; j < m; j++) { // second array check
            if (array1[i] == array2[j]) {
                break;
            }
            else if (j == (m - 1)) {
                return false;
            }
           }
        }

        for (int j = 1; j < m; j++ ) { // first array check
            for (int i = 0; i < n; i++) { // second array check
             if (array2[j] == array1[i]) {
                 break;
             }
             else if (i == (n - 1)) {
                 return false;
             }
            }
         }
        return true;
    }



    public static boolean isSorted(int [] array) {
        // Write your code here:
        if (array.length <= 1) {
            return true;
        }

        boolean minOrMax = array[0] < array[1];

        for (int i = 1; i < array.length; i++) {
            if (minOrMax) {
                if (array[i - 1] > array[i]) {
                    return false;
                }
            } else if (array[i - 1] < array[i]) {
                    return false;
                }
            }
            return true;

    }
}

public class StringOps {
public static void main(String[] args) {

    }
```

```
private static char lowerCaseXter(char c) {
    if (c >= 'A' && c <= 'Z') {
        return (char)(c + 32);
    }
    return c;
}

private static char upperCaseXter(char c) {
    if (c >= 'a' && c <= 'z') {
        return (char)(c - 32);
    }
    return c;
}

private static boolean characterIsVowel(char c) {
    switch (c) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U':
            return true;
        default:
            return false;
    }
}

public static String capVowelsLowRest (String string) {
    String result = "";

    for (int i = 0; i < string.length(); ++i) {
        if (characterIsVowel(string.charAt(i))) {
            result += upperCaseXter(string.charAt(i));
        } else {
            result += lowerCaseXter(string.charAt(i));
        }
    }
    return result;
}

public static String camelCase (String string) {
    String result = "";

    for (int i = 0; i < string.length(); ++i) {
        // trim spaces
        while (string.charAt(i) == ' ') {
            ++i;
        }

        // we found first character already, make first character
        // of new word uppercase
        if (result != "" && string.charAt(i - 1) == ' ') {
            result += upperCaseXter(string.charAt(i));
        } else {
            result += lowerCaseXter(string.charAt(i));
        }
    }
    return result;
}

public static int[] allIndexOf (String string, char chr) {
    int indexOfCharacter = -1;
    int indexesFound = 0;

    // count number of characters that we have in the string
    do {
        indexOfCharacter = string.indexOf(chr, indexOfCharacter + 1);
        if (indexOfCharacter != -1) {
            indexesFound++;
        }
    } while (indexOfCharacter != -1); // didn't find character
    // result is amount of indexes found :shrug:
    int[] result = new int[indexesFound];
    int indexIndex = 0;

    // set the indices in the result array
    do {
        indexOfCharacter = string.indexOf(chr, indexOfCharacter + 1);
        if (indexOfCharacter != -1) {
            result[indexIndex++] = indexOfCharacter;
        }
    } while (indexOfCharacter != -1); // didn't find character

    return result;
    }
}
```