**Explanation of my solution**

**Content**

[**Basic Operations** 2](#_Toc105974238)

[1 FindMaxValue 2](#_Toc105974239)

[2 FindMinValue 3](#_Toc105974240)

[3 Swap 4](#_Toc105974241)

[4 ShifLeftByOne 5](#_Toc105974242)

[5 CreateRandomArray 6](#_Toc105974243)

[6 CreateRandomMatrix 7](#_Toc105974244)

[7 CoppyArray 8](#_Toc105974245)

[8 FindInSortedArray 9](#_Toc105974246)

[**Recursion** 10](#_Toc105974247)

[**Object Oriented Programming** 11](#_Toc105974248)

[1 Interfaces 11](#_Toc105974249)

[2 Task class 12](#_Toc105974250)

[3 TaskDriver class 13](#_Toc105974251)

## **Basic Operations**

For all the following methods, I’ve also created some UnitTests in Array.Tests.cs.

# 1 FindMaxValue

****Explanation:

This method takes 1 array of integers and 2 integer variables.

The goal of this method is to search the highest value in this array between 2 positions.

Position1 is the start position and position2 is the end position.

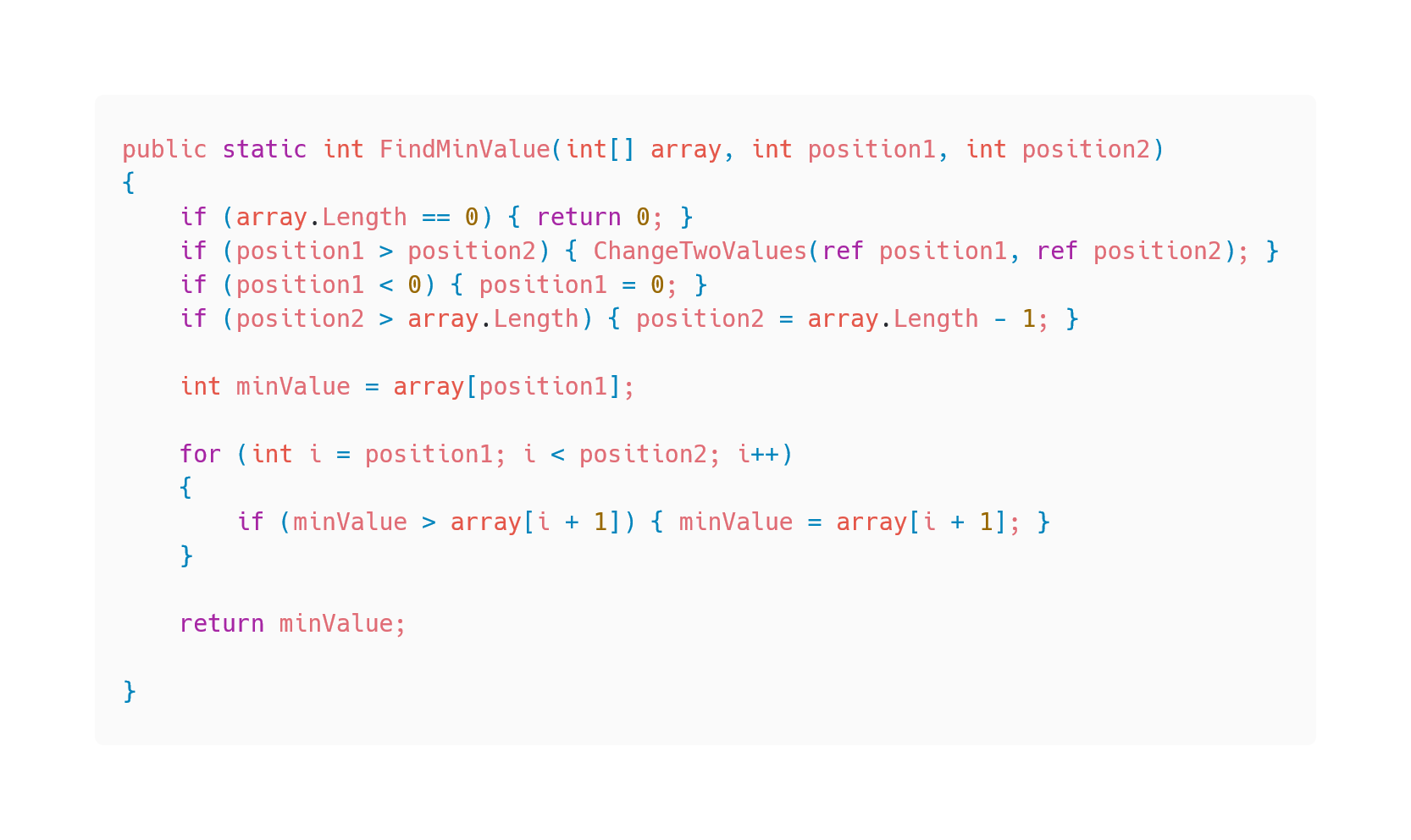
First, the method checks if the array is empty or not. If it’s empty it returns 0. After that it checks with another method changeTwoValues () if position1 is higher as position2, if a user enters the numbers that way I expect that he just messed up the order and it swaps the values.

If a user enters numbers who are out of bounds of the array, like < 0, position1 will be 0 (start of the array) or > as the array length, position2 will have the value of the length of the array.

I added an integer variable maxValue, by default it has the value of the first element of the array. After that I wrote a for loop which is looping through the array and checks if maxValue is higher as the current element of the array, if it’s higher the loop goes to the next element, if maxValue is lower, then maxValue gets the value of the current element of the array.

After the for loop is finished, the method returns maxValue.

# 2 FindMinValue + ChangeTwoValues



Explanation:

For this method I used almost the same code as for the FindMaxValue method.

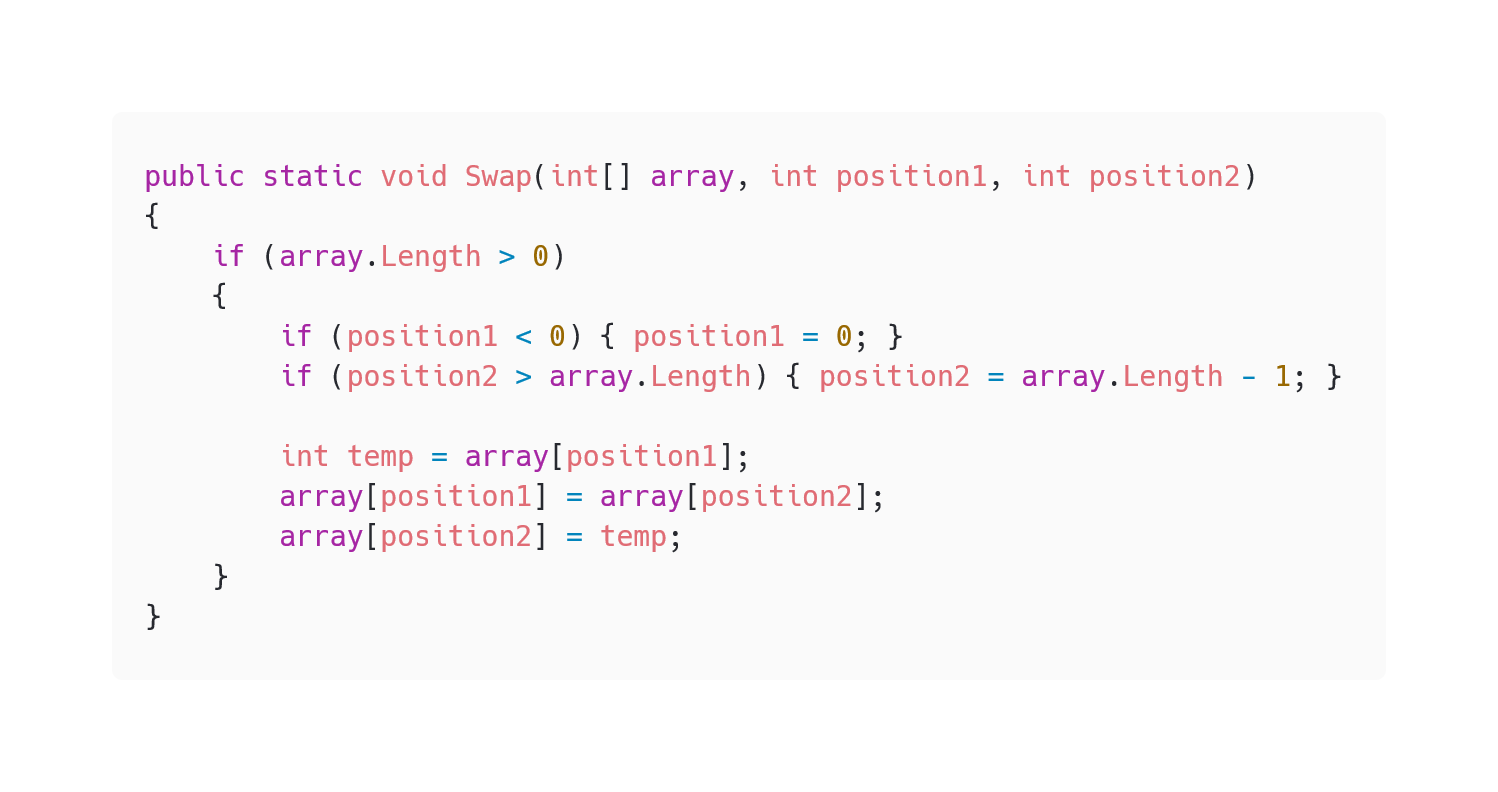
The only two things I changed are:

* The name of the return variable (maxValue 🡪 minValue)
* The operator in the if statement of the for loop (< 🡪 >)



Here we can see the other method which swaps the values of the variables.

# 3 Swap



Explanation:

This method takes 1 array of integers and 2 integer variables.

The goal of the method is to change the array values position1 and position2.

The 2 variables are positions of the array.

If position1 or position2 are out of bounds of the array, like < 0, position1 will be 0 (start of the array) or > as the array length, position2 will have the value of the length of the array.

The method switches the values of the positions with the help of the integer variable temp, which stores one value.

# 4 ShifLeftByOne



Explanation:

This method takes 1 array of integers and 2 integer variables.

The goal of the method is to shift all elements of the array between position1 and position2 to the left by one element.

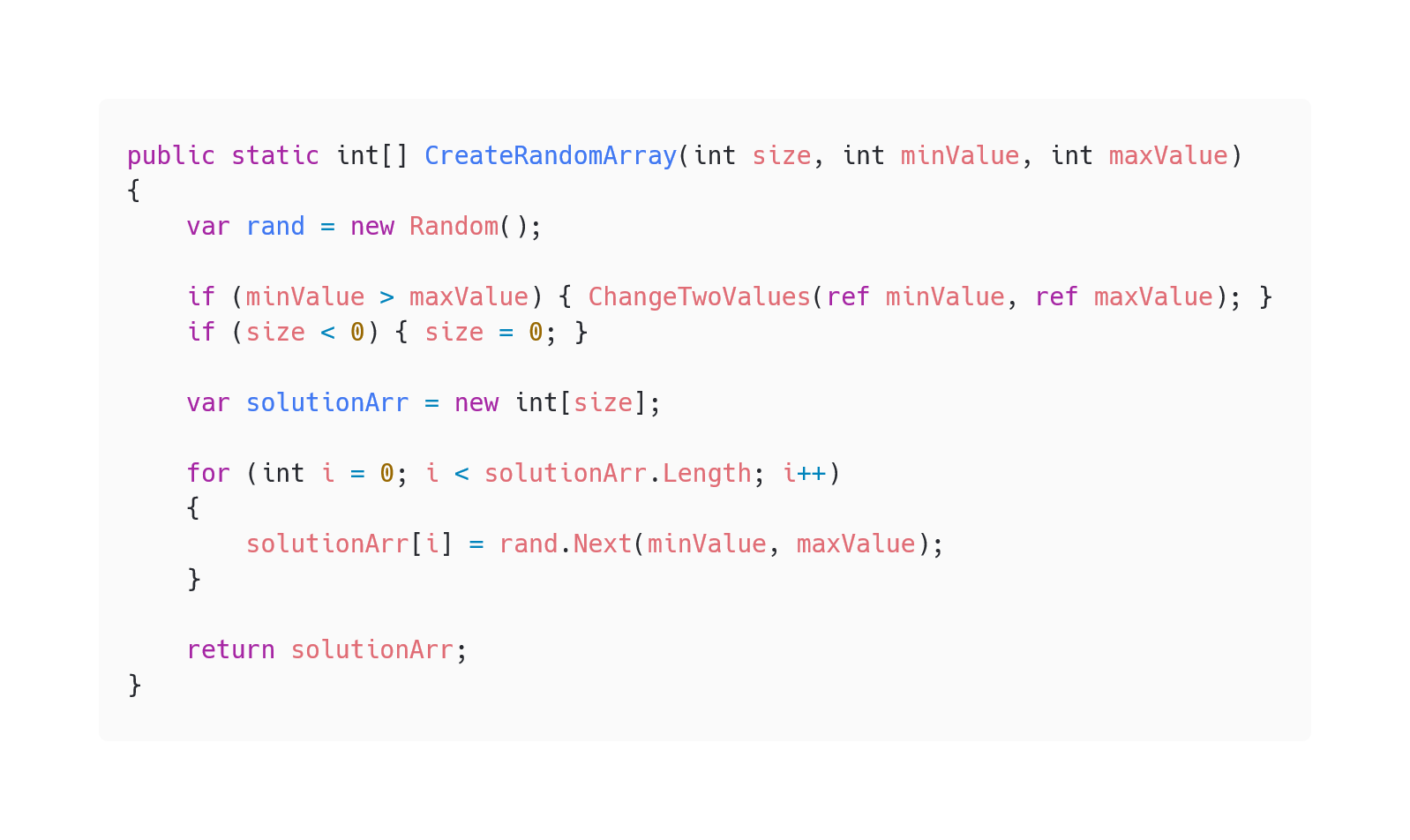
The 2 variables are positions of the array.

The element of position2 should never be changed or shifted. First the method checks with another method changeTwoValues () if position1 is higher as position2, if a user enters the numbers that way I expect that he just messed up the order and it swaps the values.

If a user enters numbers who are out of bounds of the array, like < 0, position1 will be 0 (start of the array) or > as the array length, position2 will have the value of the length of the array.

Then I added an if statement with a for loop which is looping through the elements of the array between position1 and position2. The current element gets the same value as the following one, then the loop goes on until he has reached the position2 element which shouldn’t be changed or shifted.

# 5 CreateRandomArray



Explanation:

This method takes 3 integer variables.

The goal of the method is to create a one-dimensional array with a given size, filled with random elements between the minValue and maxValue.

First, I made an object from the random class of C#, then it checks If a user enters numbers who are out of bounds of the array, like < 0, position1 will be 0 (start of the array) or > as the array length, position2 will have the value of the length of the array. If someone gives a value < 0 to size, then the size will always be 0.

After that I created an array solutionArr with the length of the size variable. A for loop fills the array by replacing the current element null with a random generated value between minValue and maxValue, then the loop goes on until solutionArr is filled up.

When the for loop is finished, the method returns the solutionArr.

# 6 CreateRandomMatrix



Explanation:

This method takes 4 integer variables.

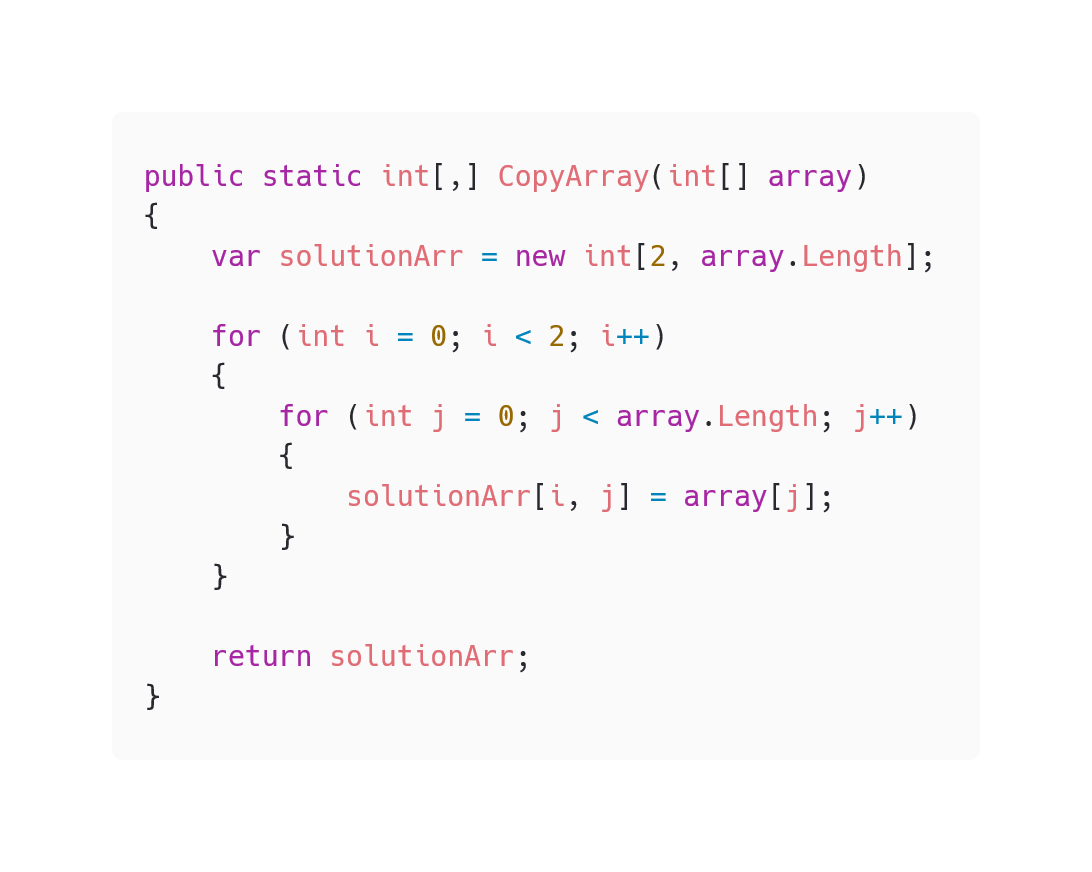
The goal of the method is to create a two-dimensional array with given rows and columns values, filled with random elements between the minValue and maxValue.

First, I made an object from the random class of C#, then it checks If a user enters numbers who are out of bounds of the array, like < 0, position1 will be 0 (start of the array) or > as the array length, position2 will have the value of the length of the array. If someone gives a value < 0 to size, then the size will always be 0.

After that I created an array solutionArr with the rows and columns values. A double for loop fills the array by replacing the current element null with a random generated value between minValue and maxValue, then the loop goes on until solutionArr is filled up.

When the double for loop is finished, the method returns the solutionArr.

# 7 CoppyArray



Explanation:

This method takes an array of integers.

The goal of the method is to copy array in a two-dimensional array.

I created an array solutionArr with the rows value of 2 and columns value of the length of array.

A double for loop fills solutionArr by replacing all null elements in the first row with all element of array, then the loop goes to the second row and does the same again.

When the double for loop is finished, the method returns the solutionArr.

# 8 FindInSortedArray



Explanation:

This method takes one array and one integer variable.

The goal of this method is, to search the value of number in array and return the position of the element with the same value.

If no element in array has the same value as number, the method should return -1.

I implemented here 2 ways, a linear search solution and a binary search solution.

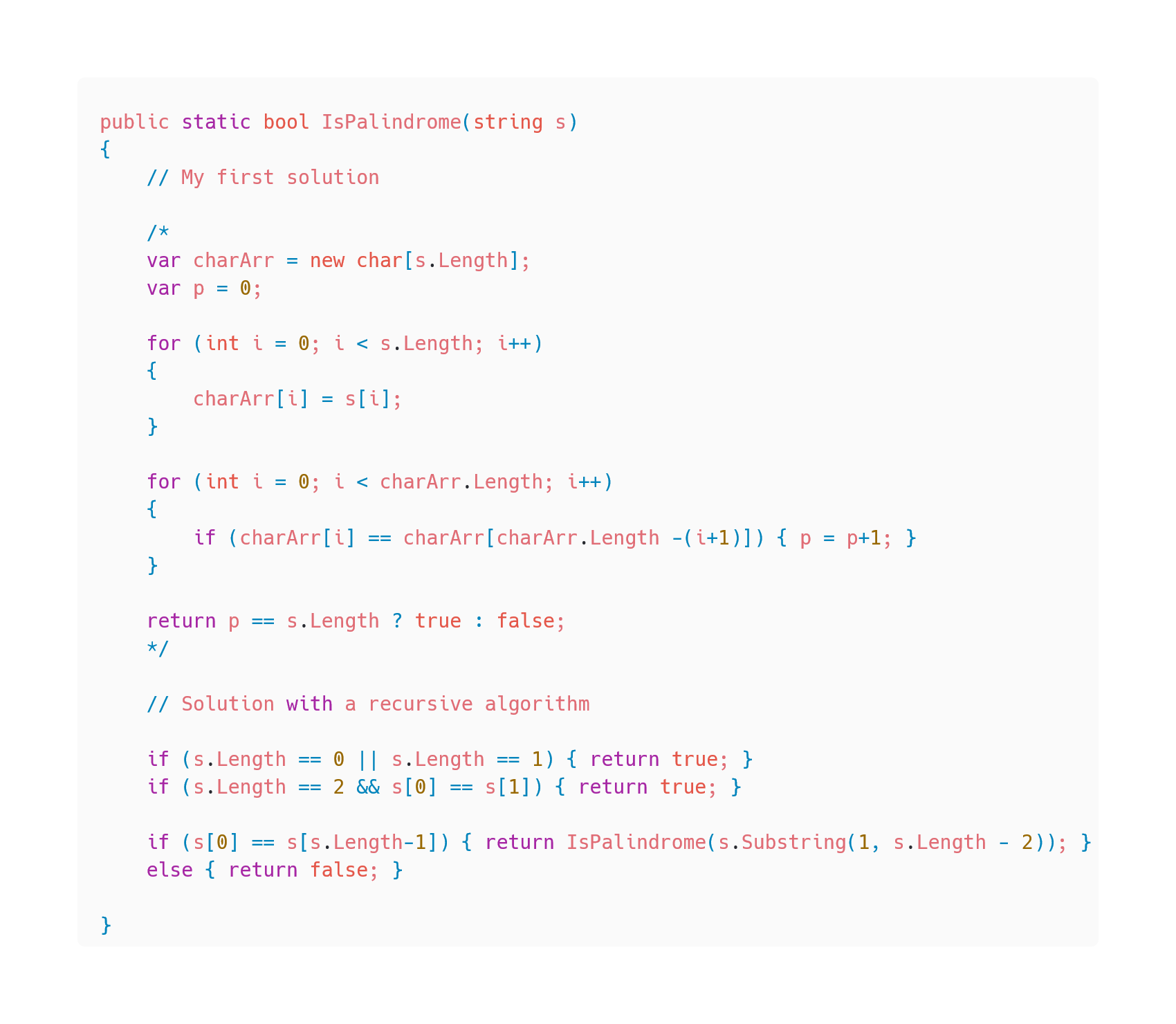
For the linear search solution, I wrote a simple for loop, looping through array until it finds the same value as number. If the loop finds the same value as number, it returns the position of the element, if not it returns -1.

For the binary search solution, I implemented 3 variables, h which is the highest point, l which is the lowest point and m which is the middle of h and l. After that we find a while loop, the loop does not end until the loop finds an element with the same value as number or l or h equals m.

The loop checks if the current m is higher or lower as number, if it’s higher, l will be the current m and m will be determined. If it’s lower, h will be the current m and m will be determined.

When the while loop is finished, the method returns m (element position of array) or -1.

## **Recursion**



Explanation:

This method takes one string variable.

The goal of this method is, to check if s is a palindrome.

I implemented here 2 ways, my first solution what came in my mind and then a solution with a recursive algorithm like asked in the Task.

In my solution I created a char array and filled it with the letters of s. After that I looped through the array and checked if the first and last letter is equal, if it’s equal check the second and the second last letter and so on. Every time the loop finds equal letter it counts up the integer variable p, if p has the same value and s has letters, s is a palindrome.

For the recursive algorithm the method checks first s has zero, one or two letters. It returns always true except the 2 letters are not equal. If s has more as 2 letters, the third if statement will check the first and last letter if they’re equal, if they are the method repeats itself with a new string. The new string is the same as the original one but without the first and last letter. If the original string was “radar” the new method just checks “ada”. This method is repeating itself until the word has zero to two letters or finds an unequal combination. If something is unequal the method returns false, if not, the method returns true which means the string is a palindrome.

## **Object Oriented Programming**

# 1 Interfaces







Explanation:

For this task, I implemented 3 interfaces IPriority, IComplexity and IComparable. All 3 interfaces have abstract methods and IPriority has 3 constant variables (MIN\_PRIORITY, MED\_PRIORITY & MAX\_PRIORITY). For the Priority system, in my solution 1 is the lowest priority (MIN\_PRIORITY), 10 is the highest priority (MAX\_PRIORITY) and 5 is medium priority (MED\_PRIORITY). Then we have normal getter and setter methods for complexity and priority, getter methods (GetPriority & GetComplexity) return the objects given value and setter method (SetPriority & SetComplexity) are setting a value to the object. Last but not least we have the CompareTo method in IComparable. This Method takes and returns a Task object, more explanation in the Task class.

# 2 Task class



Explanation:

The Task class has implemented all 3 interfaces with the abstract methods and variables. Every Task has three variables: name, priority and complexity. I made an empty constructor and one where a name has to be passed to the task. Complexity will be added through the SetComplexity method afterwards and every tasks priority has MED\_PRIORITY as default priority level. Priority and complexity can always be changed or requested with the getter and setter methods. Then we can see the CompareTo method which takes and returns a Task object. This method compares a Task with another one. The comparison is based on the tasks’ priority. Basically, it’s just an if statement which compares both integers of priority, the higher one will be returned.

# 3 TaskDriver class



Explanation:

Here we can see the TaskDriver class with the main method. First the method creates three instances of Task. Due to the fact that the constructor only takes a name, I had to set the priority and complexity in the following 2 lines of code. Once all 3 instances of Task are made, they get all added up to an array of Task. Next, I defined a Task variable called highestTask which has a default value as the first element of the Task array. After that we find a for loop, which is looping through the array and checks if highestTask is higher as the current array Task. If the current array Task is higher, that one is the new highestTask, otherwise the loop goes on.

Then there is just the Output as shown in the exercise.

In TaskTests.cs I made some tests of the 3 created instances if the CompareTo method is working as intended.