Name: Gondrala Mani Sai
Id num: 2100031545

Lab3: OOP Concepts Classes(Low and Advanced)

1. Develop **Rectangle** and **ArrayRectangles** with a predefined functionality.

Low level Task:

TASK 1: To develop **Rectangle** class with following content:

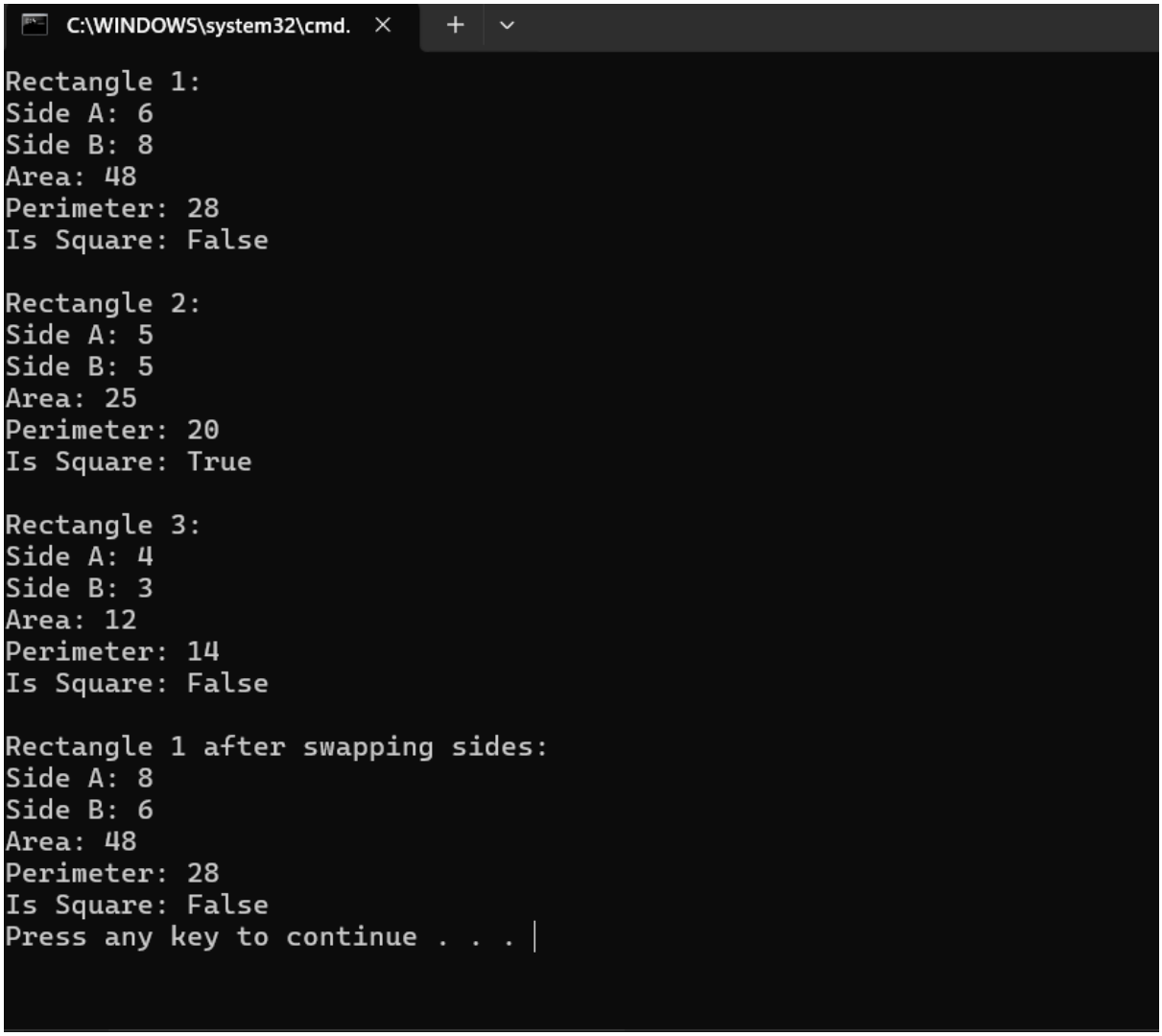
- 2 closed real fields **sideA** and **sideB** (sides A and B of the rectangle)
- Constructor with two real parameters **a** and **b** (parameters specify rectangle sides)
- Constructor with a real parameter **a** (parameter specify side A of a rectangle, side B is always equal to 5)
- Constructor without parameters (side A of a rectangle equals to 4, side B - 3)
- Method **GetSideA**, returning value of the side A
- Method **GetSideB**, returning value of the side B
- Method **Area**, calculating and returning the area value
- Method **Perimeter**, calculating and returning the perimeter value
- Method **IsSquare**, checking whether current rectangle is shape square or not. Returns true if the shape is square and false in another case.
- Method **ReplaceSides**, swapping rectangle sides

Solution:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab_3
{
    internal class Rectangle
    {
        private double sideA;
        private double sideB;

        // Constructor with two real parameters a and b
        public Rectangle(double a, double b)
        {
            sideA = a;
            sideB = b;
        }

        // Constructor with a real parameter a (parameter specify side A of a
rectangle, side B is always equal to 5)
        public Rectangle(double a)
```

```
        {
            sideA = a;
            sideB = 5;
        }

        // Constructor without parameters (side A of a rectangle equals to 4,
side B - 3)
        public Rectangle()
        {
            sideA = 4;
            sideB = 3;
        }

        // Method GetSideA, returning value of the side A
        public double GetSideA()
        {
            return sideA;
        }

        // Method GetSideB, returning value of the side B
        public double GetSideB()
        {
            return sideB;
        }

        // Method Area, calculating and returning the area value
        public double Area()
        {
            return sideA * sideB;
        }

        // Method Perimeter, calculating and returning the perimeter value
        public double Perimeter()
        {
            return 2 * (sideA + sideB);
        }

        // Method IsSquare, checking whether current rectangle is shape square or
not
        // Returns true if the shape is square and false in another case
        public bool IsSquare()
        {
            return sideA == sideB;
        }

        // Method ReplaceSides, swapping rectangle sides
        public void ReplaceSides()
        {
            double temp = sideA;
            sideA = sideB;
            sideB = temp;
        }
    }

}
```

Output:



Develop class **ArrayRectangles**, in which declare:

- Private field **rectangle_array -** array of rectangles
- Constructor creating an empty array of rectangles with length n
- Constructor that receives an arbitrary amount of objects of type **Rectangle** or an array of objects of type **Rectangle**.
- Method **AddRectangle** that adds a rectangle of type Rectangle to the array on the nearest free place and returning true, or returning false, if there is no free space in the array
- Method **NumberMaxArea**, that returns order number (index) of the rectangle with the maximum area value (numeration starts from zero)
- Method **NumberMinPerimeter**, that returns order number(index) of the rectangle with the minimum area value (numeration starts from zero)
- Method **NumberSquare**, that returns the number of squares in the array of rectangles

Solution:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab_3
{
    internal class ArrayRectangles
    {
        private Rectangle[] rectangleArray;

        // Constructor creating an empty array of rectangles with length n
        public ArrayRectangles(int n)
        {
            rectangleArray = new Rectangle[n];
        }

        // Constructor that receives an arbitrary amount of objects of type
Rectangle or an array of objects of type Rectangle
        public ArrayRectangles(params Rectangle[] rectangles)
        {
            rectangleArray = rectangles;
        }

        // Method to add a rectangle to the array on the nearest free place
        // Returns true if added successfully, false if there is no free space in
the array
        public bool AddRectangle(Rectangle rectangle)
        {
            for (int i = 0; i < rectangleArray.Length; i++)
            {
                if (rectangleArray[i] == null)
                {
                    rectangleArray[i] = rectangle;
                    return true;
                }
            }
            return false;
        }

        // Method to find the order number (index) of the rectangle with the
maximum area value
        public int NumberMaxArea()
        {
            double maxArea = double.MinValue;
            int index = -1;

            for (int i = 0; i < rectangleArray.Length; i++)
            {
                if (rectangleArray[i] != null && rectangleArray[i].Area() >
maxArea)
                {
                    maxArea = rectangleArray[i].Area();
                    index = i;
                }
            }

            return index;
```

```
        }

        // Method to find the order number (index) of the rectangle with the
minimum perimeter value
        public int NumberMinPerimeter()
        {
            double minPerimeter = double.MaxValue;
            int index = -1;

            for (int i = 0; i < rectangleArray.Length; i++)
            {
                if (rectangleArray[i] != null && rectangleArray[i].Perimeter() <
minPerimeter)
                {
                    minPerimeter = rectangleArray[i].Perimeter();
                    index = i;
                }
            }

            return index;
        }

        // Method to count the number of squares in the array of rectangles
        public int NumberSquare()
        {
            int count = 0;

            foreach (Rectangle rectangle in rectangleArray)
            {
                if (rectangle != null && rectangle.IsSquare())
                {
                    count++;
                }
            }

            return count;
        }
    }

}
```

Output: