**2100030320**

**Bhavana**

# EPAM – LAB-3

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

What's New? Program.cs*

lab3_task1-rectangle Rectangle

```
// See https://aka.ms/new-console-template for more information
using System;
9 references
class Rectangle
{
    private double sideA;
    private double sideB;
    1 reference
    public Rectangle(double a, double b){
        sideA = a;
        sideB = b; }
    1 reference
    public Rectangle(double a){
        sideA = a;
        sideB = 5; }
    1 reference
    public Rectangle() {
        sideA = 4;
        sideB = 3;    }
    0 references
    public double GetSideA() {
        return sideA;    }
    0 references
    public double GetSideB(){
        return sideB;}
    4 references
    public double Area(){
        return sideA * sideB; }
    4 references
    public double Perimeter(){
        return 2 * (sideA + sideB);}
    4 references
    public bool IsSquare(){
        return sideA == sideB; }
```

100 % No issues found

```
lab3_task1-rectangle        Rectangle        Rectangle()
        1 reference
        public void ReplaceSides() {
            double temp = sideA;
            sideA = sideB;
            sideB = temp; }
    }
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            Rectangle rectangle1 = new Rectangle(5, 8);
            Console.WriteLine("Rectangle 1 - Area: " + rectangle1.Area());
            Console.WriteLine("Rectangle 1 - Permeter: " + rectangle1.Perimeter());
            Console.WriteLine("Rectangle 1 - issquare " + rectangle1.IsSquare());

           Rectangle rectangle2 = new Rectangle(4);
            Console.WriteLine("Rectangle 2 - Area: " + rectangle2.Area());
            Console.WriteLine("Rectangle 2 - Permeter: " + rectangle2.Perimeter());
            Console.WriteLine("Rectangle 2 - issquare " + rectangle2.IsSquare());

            Rectangle rectangle3 = new Rectangle();
            Console.WriteLine("Rectangle 3 - Area: " + rectangle3.Area());
            Console.WriteLine("Rectangle 3 - Permeter: " + rectangle3.Perimeter());
            Console.WriteLine("Rectangle 3 - issquare " + rectangle3.IsSquare());

            rectangle3.ReplaceSides();
            Console.WriteLine("After swapping sides - Rectangle 3 - Area: " + rectangle3.Area() + ", Perimeter: " + rectangle3.Perimeter(
        }
    }
```

# OUTPUT:

```
Microsoft Visual Studio Debu

Rectangle 1 - Area: 40
Rectangle 1 - Permeter: 26
Rectangle 1 - issquare False
Rectangle 2 - Area: 20
Rectangle 2 - Permeter: 18
Rectangle 2 - issquare False
Rectangle 3 - Area: 12
Rectangle 3 - Permeter: 14
Rectangle 3 - issquare False
After swapping sides - Rectangle 3 - Area: 12, Perimeter: 14, Is Square? False

C:\Users\Bhavana\Documents\EPAM\lab3_task1-rectangle\bin\Debug\net8.0\lab3_task1-rectangle.exe (process
th code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatical
le when debugging stops.
Press any key to close this window . . .
```

**TASK 2:** 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

LAB3-TASK-2 | Rectangle

```
// See https://aka.ms/new-console-template for more information
using System;
11 references
public class Rectangle
{
    4 references
    public double Width { get; }
    4 references
    public double Height { get; }
    3 references
    public Rectangle(double width, double height){
        Width = width;
        Height = height;
    }
    1 reference
    public double CalculateArea() {
        return Width * Height; }
    1 reference
    public double CalculatePerimeter(){
        return 2 * (Width + Height);}
    1 reference
    public bool IsSquare(){
        return Width == Height;}
}
4 references
public class ArrayRectangles
{
    private Rectangle[] rectangleArray;
    1 reference
    public ArrayRectangles(int n){
        rectangleArray = new Rectangle[n]; }
    0 references
    public ArrayRectangles(params Rectangle[] rectangles){
        rectangleArray = rectangles;    }
```

100 % | No issues found

Output

LAB3-TASK-2 | Rectangle | Width

```
    3 references
    public bool AddRectangle(Rectangle rectangle)  {
        for (int i = 0; i < rectangleArray.Length; i++)       {
            if (rectangleArray[i] == null)          {
                rectangleArray[i] = rectangle;
                return true;          }   }
        return false;
    }
    1 reference
    public int NumberMaxArea()    {
        if (rectangleArray.Length == 0)
            throw new InvalidOperationException("Array is empty.");
        double maxArea = double.MinValue;
        int maxIndex = 0;
        for (int i = 0; i < rectangleArray.Length; i++)       {
            if (rectangleArray[i] != null)          {
                double area = rectangleArray[i].CalculateArea();
                if (area > maxArea)               {
                    maxArea = area;
                    maxIndex = i;               }           }       }
        return maxIndex;
    }
    1 reference
    public int NumberMinPerimeter()  {
        if (rectangleArray.Length == 0)
            throw new InvalidOperationException("Array is empty.");
        double minPerimeter = double.MaxValue;
        int minIndex = 0;
        for (int i = 0; i < rectangleArray.Length; i++)      {
            if (rectangleArray[i] != null)          {
                double perimeter = rectangleArray[i].CalculatePerimeter();
                if (perimeter < minPerimeter)               {
                    minPerimeter = perimeter;
```

```
                minIndex = i;                }            }        }
            return minIndex;
        }
        1 reference
        public int NumberSquare()  {
            int squareCount = 0;
            foreach (var rectangle in rectangleArray)      {
                if (rectangle != null && rectangle.IsSquare())           {
                    squareCount++;            }        }
            return squareCount;
        }
    }
    0 references
class Program{
    0 references
    static void Main()  {
        ArrayRectangles arrayRectangles = new ArrayRectangles(5);
        Rectangle rectangle1 = new Rectangle(3, 4);
        Rectangle rectangle2 = new Rectangle(2, 2);
        Rectangle rectangle3 = new Rectangle(5, 5);
        arrayRectangles.AddRectangle(rectangle1);
        arrayRectangles.AddRectangle(rectangle2);
        arrayRectangles.AddRectangle(rectangle3);
        int maxAreaIndex = arrayRectangles.NumberMaxArea();
        int minPerimeterIndex = arrayRectangles.NumberMinPerimeter();
        int squareCount = arrayRectangles.NumberSquare();
        Console.WriteLine($"Rectangle with max area is at index: {maxAreaIndex}");
        Console.WriteLine($"Rectangle with min perimeter is at index: {minPerimeterIndex}");
        Console.WriteLine($"Number of squares in the array: {squareCount}");
    }
}
```

# OUTPUT:

```
Microsoft Visual Studio Debu

Rectangle with max area is at index: 2
Rectangle with min perimeter is at index: 1
Number of squares in the array: 2

C:\Users\Bhavana\Documents\EPAM\LAB3-TASK-2\bin\Debug\net8.0\LAB3-TASK-2.exe (p
To automatically close the console when debugging stops, enable Tools->Options-
le when debugging stops.
Press any key to close this window . . .
```