.Net Programming Lab-3

Name : Ch Bala Gowtham

Regdno : 2000032067

Sec : 13

IN-LAB:

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

Low level Task:

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

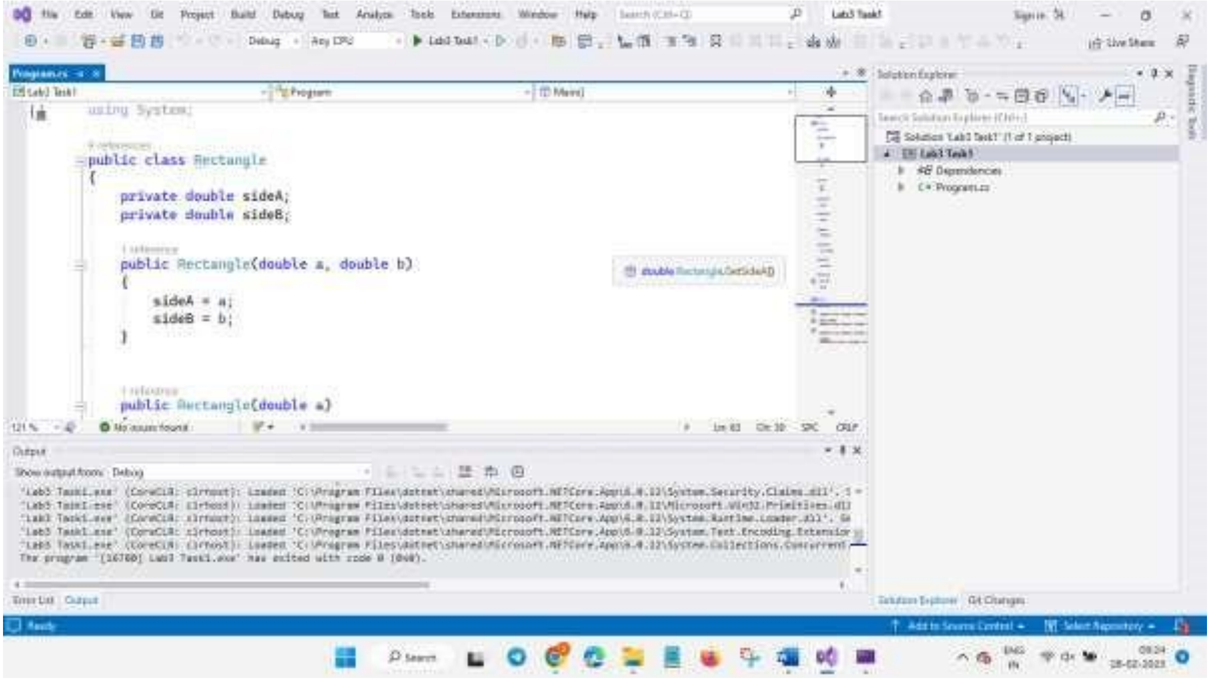
- 2 closed real fields **sideA** and **sideB** (sides A and B of the rectangle)
- Constructor with two real parameters **a** and **b** (parameters specify rectangle sides)
- Constructor with a real parameter **a** (parameter specify side A of a rectangle, side B is always equal to 5)
- Constructor without 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;

public class Rectangle
{     private double
sideA;
```

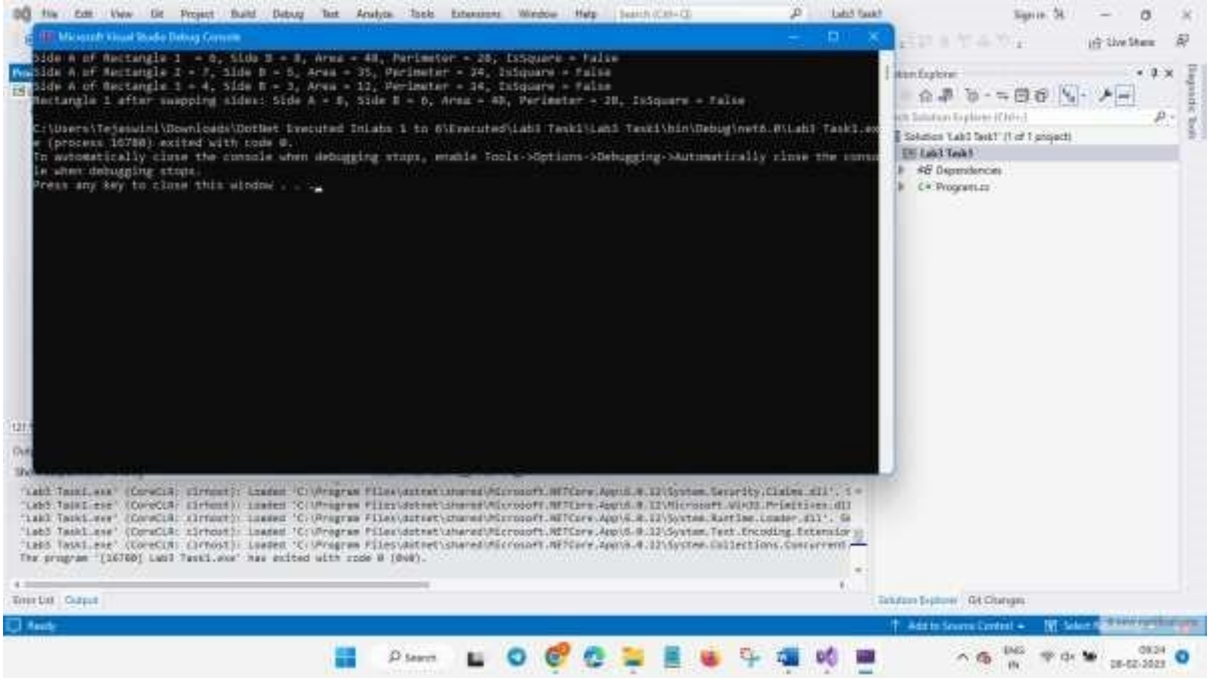
```
   private double sideB;

   public Rectangle(double a, double b)
   {         sideA
= a;         sideB
= b;
   }

   public Rectangle(double a)
   {         sideA
= a;         sideB
= 5;
   }


   public Rectangle()
   {         sideA
= 4;         sideB
= 3;
   }

   public double GetSideA()
   {         return
sideA;
   }

   public double GetSideB()
   {         return
sideB;
   }

   public double Area()
   {
      return sideA * sideB;
   }

   public double Perimeter()
   {
      return 2 * (sideA + sideB);
   }

   public bool IsSquare()
   {
      return sideA == sideB;
   }

   public void ReplaceSides()
```

```
    {
        double temp = sideA;
sideA = sideB;         sideB
= temp;
    }
}

public class Program
{
    public static void Main()
    {
        Rectangle rect1 = new Rectangle(6, 8);
        Console.WriteLine($"Side A of Rectangle 1  = {rect1.GetSideA()}, Side B =
{rect1.GetSideB()}, Area = {rect1.Area()}, Perimeter = {rect1.Perimeter()}, IsSquare =
{rect1.IsSquare()}");

        Rectangle rect2 = new Rectangle(7);
        Console.WriteLine($"Side A of Rectangle 2 = {rect2.GetSideA()}, Side B =
{rect2.GetSideB()}, Area = {rect2.Area()}, Perimeter = {rect2.Perimeter()}, IsSquare =
{rect2.IsSquare()}");

        Rectangle rect3 = new Rectangle();
        Console.WriteLine($"Side A of Rectangle 3 = {rect3.GetSideA()}, Side B =
{rect3.GetSideB()}, Area = {rect3.Area()}, Perimeter = {rect3.Perimeter()}, IsSquare =
{rect3.IsSquare()}");

        rect1.ReplaceSides();
        Console.WriteLine($"Rectangle 1 after swapping sides: Side A = {rect1.GetSideA()},
Side B = {rect1.GetSideB()}, Area = {rect1.Area()}, Perimeter = {rect1.Perimeter()},
IsSquare = {rect1.IsSquare()}");
    }
}
```



```
using System;

public class Rectangle
{
    private double sideA;
    private double sideB;

    public Rectangle(double a, double b)
    {
        sideA = a;
        sideB = b;
    }


    public Rectangle(double a)
```

```
'Lab3 Task1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Security.Claims.dll'. S
'Lab3 Task1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\Microsoft.Win32.Primitives.dll
'Lab3 Task1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Runtime.Loader.dll'. Sk
'Lab3 Task1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Text.Encoding.Extensior
'Lab3 Task1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Collections.Concurrent
The program '[16780] Lab3 Task1.exe' has exited with code 0 (0x0).
```



```
Side A of Rectangle 1  = 6, Side B = 8, Area = 48, Perimeter = 28, IsSquare = False
Side A of Rectangle 2 = 7, Side B = 5, Area = 35, Perimeter = 24, IsSquare = False
Side A of Rectangle 3 = 4, Side B = 3, Area = 12, Perimeter = 14, IsSquare = False
Rectangle 1 after swapping sides: Side A = 8, Side B = 6, Area = 48, Perimeter = 28, IsSquare = False

C:\Users\Tejaswini\Downloads\DotNet Executed Inlabs 1 to 6\Executed\Lab3 Task1\Lab3 Task1\bin\Debug\net6.0\Lab3 Task1.ex
e (process 16780) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Advanced level Task:

TASK 2: Develop class **ArrayRectangles**, in which declare:

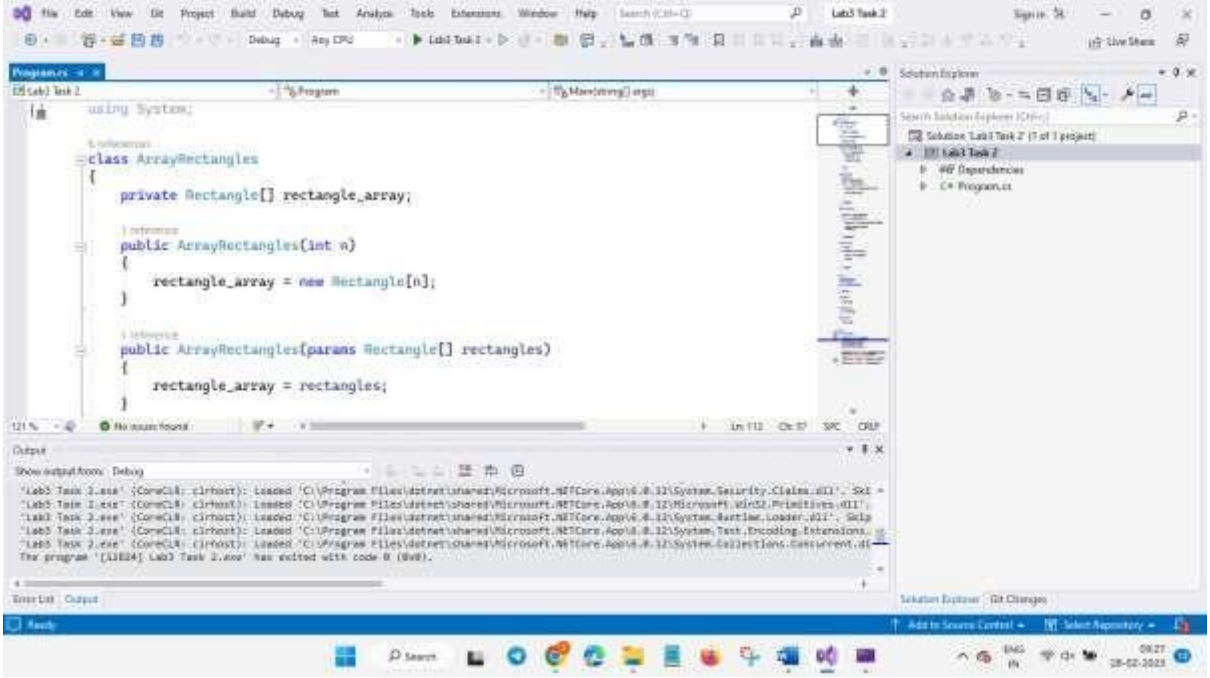
- Private field **rectangle_array** - array of rectangles
- Constructor creating an empty array of rectangles with length n
- Constructor that receives an arbitrary amount of objects of type **Rectangle** or an array of objects of type **Rectangle**.

- 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

Solution:

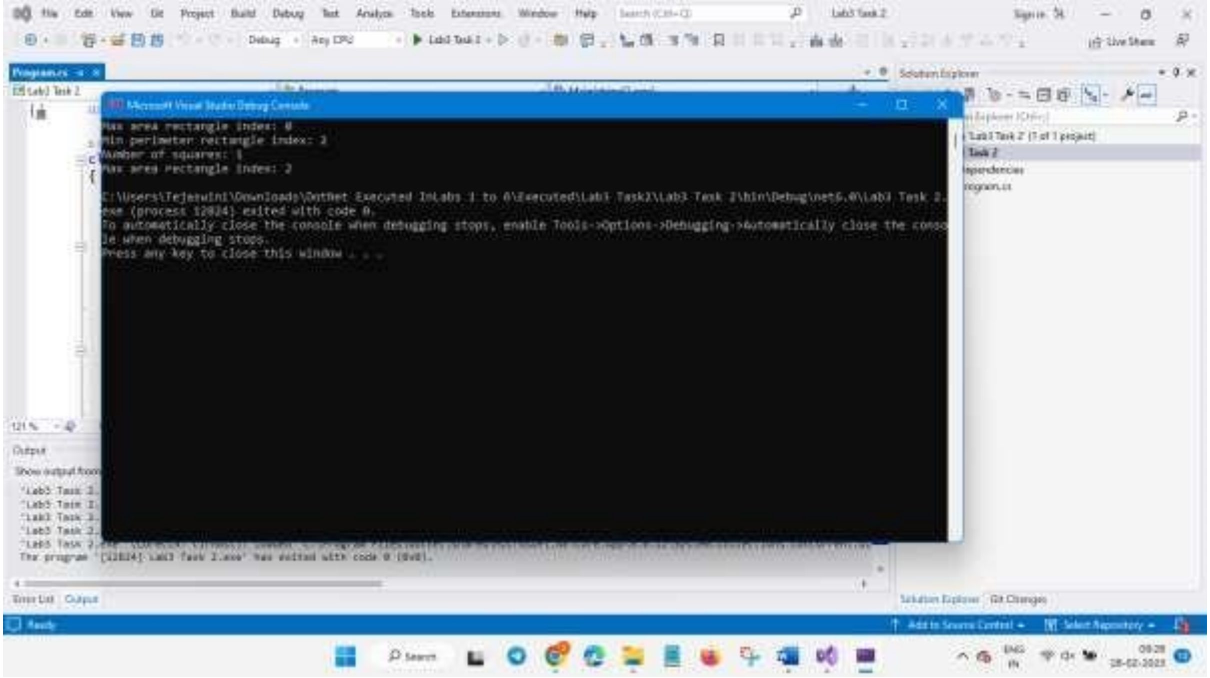
```
using System;

class ArrayRectangles
{
    private Rectangle[] rectangle_array;

    public ArrayRectangles(int n)
    {
        rectangle_array = new Rectangle[n];
    }

    public ArrayRectangles(params Rectangle[] rectangles)
    {
        rectangle_array = rectangles;
    }

    public bool AddRectangle(Rectangle rectangle)
    {
        for (int i = 0; i < rectangle_array.Length; i++)
        {
            if (rectangle_array[i] == null)
            {
                rectangle_array[i] = rectangle;
                return true;
            }
}
        return false;
    }

    public int NumberMaxArea()
    {
        int maxAreaIndex = 0;
        double maxArea = rectangle_array[0].Area();

        for (int i = 1; i < rectangle_array.Length; i++)
        {
            if (rectangle_array[i] != null && rectangle_array[i].Area() > maxArea)
            {
                maxArea = rectangle_array[i].Area();
maxAreaIndex = i;
            }
        }

        return maxAreaIndex;
    }

    public int NumberMinPerimeter()
    {
        int minPerimeterIndex = 0;
        double minPerimeter = rectangle_array[0].Perimeter();

        for (int i = 1; i < rectangle_array.Length; i++)
        {
            if (rectangle_array[i] != null && rectangle_array[i].Perimeter() < minPerimeter)
            {
                minPerimeter = rectangle_array[i].Perimeter();
minPerimeterIndex = i;
            }
        }

        return minPerimeterIndex;
    }

    public int NumberSquare()
    {
        int squareCount = 0;

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

        return squareCount;
    }
```

```
}

class Rectangle
{
    public double Width { get; set; }
public double Height { get; set; }

    public Rectangle(double width, double height)
    {
        Width = width;
        Height = height;
    }

    public double Area()
    {
        return Width * Height;
    }

    public double Perimeter()
    {
        return 2 * (Width + Height);
    }

    public bool IsSquare()
    {
        return Width == Height;
    }
}

class Program
{
    static void Main(string[] args)
    {
        ArrayRectangles arrRectangles = new ArrayRectangles(3);
arrRectangles.AddRectangle(new Rectangle(2, 3));         arrRectangles.AddRectangle(new
Rectangle(4, 1));         arrRectangles.AddRectangle(new Rectangle(1, 1));
arrRectangles.AddRectangle(new Rectangle(5, 5));

        Console.WriteLine("Max area rectangle index: " + arrRectangles.NumberMaxArea());
        Console.WriteLine("Min perimeter rectangle index: " +
arrRectangles.NumberMinPerimeter());
        Console.WriteLine("Number of squares: " + arrRectangles.NumberSquare());

        Rectangle[] rectangles = { new Rectangle(3, 3), new Rectangle(4, 4), new Rectangle(5,
5) };
        ArrayRectangles arrRectangles2 = new ArrayRectangles(rectangles);
        Console.WriteLine("Max area rectangle index: " + arrRectangles2.NumberMaxArea());
} }
```



```
using System;

class ArrayRectangles
{
    private Rectangle[] rectangle_array;

    public ArrayRectangles(int n)
    {
        rectangle_array = new Rectangle[n];
    }

    public ArrayRectangles(params Rectangle[] rectangles)
    {
        rectangle_array = rectangles;
    }
```

```
'Lab3 Task 2.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Security.Claims.dll'. Skl
'Lab3 Task 2.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\Microsoft.Win32.Primitives.dll'.
'Lab3 Task 2.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Runtime.Loader.dll'. Skip
'Lab3 Task 2.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Text.Encoding.Extensions.
'Lab3 Task 2.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.12\System.Collections.Concurrent.dl
The program '[12824] Lab3 Task 2.exe' has exited with code 0 (0x0).
```



```
Max area rectangle index: 0
Min perimeter rectangle index: 2
Number of squares: 1
Max area rectangle index: 2

C:\Users\Tejeswini\Downloads\DotNet Executed InLabs 1 to 6\Executed\Lab3 Task2\Lab3 Task 2\bin\Debug\net6.0\Lab3 Task 2.
exe (process 12824) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

POST-LAB

1. What are the building blocks of an OOP Application ,Design an Application and find the low-level and Advanced classes in that Application along with specifications?

   Note: here you can take any real time user defined class and supporting methods to implement low level and advanced level classes.

Solution:

The building blocks of an OOP application are classes, objects, inheritance, encapsulation, and polymorphism.

To design an OOP application, we will consider a simple example of a library management system. The application will have the following classes:

Book Class: This class will contain the book's properties such as title, author, ISBN, publication date, and number of copies.

Low-level methods:
get_title(): returns the book's title.

get_author(): returns the book's author. get_isbn():

returns the book's ISBN. get_pub_date(): returns

the book's publication date.

get_num_copies(): returns the number of copies of the book. set_title(title):

sets the book's title to the given value.

set_author(author): sets the book's author to the given value. set_isbn(isbn): sets the

book's ISBN to the given value. set_pub_date(pub_date): sets the book's publication

date to the given value. set_num_copies(num_copies): sets the number of copies of the

book to the given value.

Advanced-level methods:

increase_num_copies(): increases the number of copies of the book by 1.

decrease_num_copies(): decreases the number of copies of the book by 1.

Library Class: This class will contain the library's properties such as name, address, and a list of books.

Low-level methods:

get_name(): returns the library's name. get_address():

returns the library's address.

get_books(): returns the list of books in the library.

set_name(name): sets the library's name to the given value.

set_address(address): sets the library's address to the given value.

set_books(books): sets the list of books in the library to the given value.

Advanced-level methods:

add_book(book): adds a book to the library's list of books.

remove_book(book): removes a book from the library's list of books.

get_available_books(): returns a list of books that are currently available in the library.

Member Class: This class will contain the member's properties such as name, address, and a list of books borrowed by the member.

Low-level methods:
get_name(): returns the member's name. get_address(): returns the

member's address. get_borrowed_books(): returns the list of books

borrowed by the member. set_name(name): sets the member's name to the

given value. set_address(address): sets the member's address to the given

value.

set_borrowed_books(borrowed_books): sets the list of books borrowed by the member to

the given value. Advanced-level methods:

borrow_book(book): borrows a book from the library and adds it to the member's list of borrowed books.

return_book(book): returns a borrowed book to the library and removes it from the member's list of borrowed books.

get_overdue_books(): returns a list of books that are overdue and need to be returned.

By using the above classes, we can design a library management system that allows members to borrow and return books from the library. The low-level methods provide basic