Ex. No. 04	OPERATOR OVERLOADING
28.08.2023	

Aim

To develop C# console application using operator overloading concept.

Description

Operator Overloading:

- Gives the ability to use the same operator to do various operations, provides additional capabilities to C# operators when they are applied to user defined class.
- It is achieved by defining like a function
- Syntax:
  <access_specifier> static <return_type> operator <operator>(<parameters>){…}
- It should be static and it does not reference to a class, All the basic operations can be overloaded through it.

Source Code

1.

```
using System;

namespace E4{

    internal class Matrix{

        public int[ , ] mat=new int[2,2];

        public Matrix(int a, int b, int c, int d) {

            this.mat[0, 0] = a;

            this.mat[0, 1] = b;

            this.mat[1, 0] = c;
            this.mat[1, 1] = d;
        }
        public void Matrix_display(){
            Console.WriteLine("Displaying Matrix");
            Console.WriteLine(this.mat[0, 0] + " "+this.mat[0, 1]);
            Console.WriteLine(this.mat[1, 0] + " " + this.mat[1, 1]);
        }
        public static Matrix operator +(Matrix A, Matrix B){
            Matrix C=new Matrix(A.mat[0, 0] + B.mat[0,0], A.mat[0, 1] + B.mat[0, 1], A.mat[1, 
0] + B.mat[1, 0], A.mat[1,1] + B.mat[1,1]);
            return C;
        }
        public static Matrix operator *(Matrix A, Matrix B){
            Matrix C = new Matrix(A.mat[0, 0] * B.mat[0, 0] + A.mat[0, 1] * B.mat[1,0], A.mat[0, 
0] * B.mat[0, 1] + A.mat[0, 1] * B.mat[1,1], A.mat[1, 0] * B.mat[0, 0] + A.mat[1, 1] * 
B.mat[1, 0], A.mat[1, 0] * B.mat[0, 1] + A.mat[1, 1] * B.mat[1,1]);
            return C;
        }
static void Main(string[] args){
            int n1, n2, n3, n4;
            Console.WriteLine("Enter Matrix M1 Elements: ");
            Console.Write("[0,0] = ");
            n1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("[0,1] = ");
            n2 = Convert.ToInt32(Console.ReadLine());
```

```
Console.Write("[1,0] = ");
n3 = Convert.ToInt32(Console.ReadLine());
Console.Write("[1,1] = ");
n4 = Convert.ToInt32(Console.ReadLine());
Matrix M1 = new Matrix(n1, n2, n3, n4);
int o1, o2, o3, o4;
Console.WriteLine("\nEnter Matrix M2 Elements: ");
Console.Write("[0,0] = ");
o1 = Convert.ToInt32(Console.ReadLine());
Console.Write("[0,1] = ");
o2 = Convert.ToInt32(Console.ReadLine());
Console.Write("[1,0] = ");
o3 = Convert.ToInt32(Console.ReadLine());
Console.Write("[1,1] = ");
o4 = Convert.ToInt32(Console.ReadLine());
Matrix M2 = new Matrix(o1,o2,o3,o4);
Console.Write("M1 ");
M1.Matrix_display();
Console.Write("M2 ");
M2.Matrix_display();
Matrix M3 = M1 + M2;
Console.Write("M3 Matrix(M1+M2) ");
M3.Matrix_display();
Matrix M4 = M1 * M2;
```

```
            Console.Write("M4 Matrix(M1*2) ");
            M4.Matrix_display();
            Console.ReadKey();
        }
    }
}
```

2.

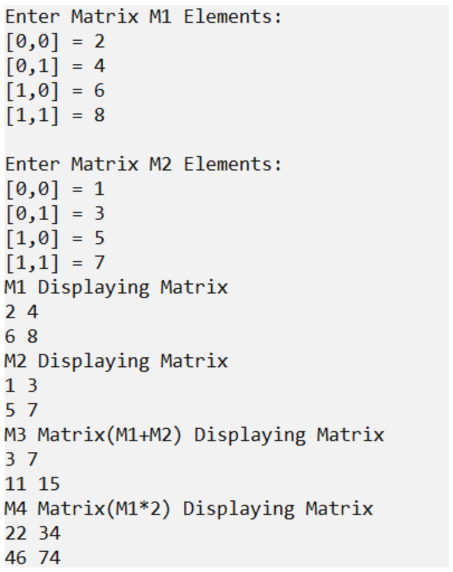
```
using System;
namespace E4{
    internal class Rectangle{
        int l, b;
        public Rectangle(int length, int breadth){
            this.l = length;
            this.b = breadth;
        }
        public void display(){Console.WriteLine("Length= " + this.l + "  Breadth= " + this.b);}
        public static Rectangle operator +(Rectangle first, Rectangle second){
            return new Rectangle(first.l + second.l,second.b+ second.b);
        }
        public static Boolean operator ==(Rectangle first, Rectangle second){
            if (first.l == second.l && first.b == second.b) return true;
            else return false;
        }
        public static Boolean operator !=(Rectangle first, Rectangle second){
            if (first.l != second.l && first.b != second.b) return true;
            else return false;
        }
        public static Boolean operator >(Rectangle first, Rectangle second){
            if (first.l > second.l && first.b > second.b) return true;
            else return false;
        }
        public static Boolean operator <(Rectangle first, Rectangle second){
            if (first.l < second.l && first.b < second.b) return true;
            else return false;
        }
        public static Boolean operator >=(Rectangle first, Rectangle second){
            if (first.l >= second.l && first.b >= second.b) return true;
            else return false;
        }
        public static Boolean operator <=(Rectangle first, Rectangle second){
            if (first.l <= second.l && first.b <= second.b) return true;
            else return false;
        }
public static void Main(string[] args){
            Console.WriteLine("Enter the Dimensions for R1");
            Console.Write("Length: ");
            int r1_l=Convert.ToInt32(Console.ReadLine());
            Console.Write("Breadth: ");
```

}

```
            int r1_b = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            Rectangle R1 = new Rectangle(r1_l, r1_b);
            Console.WriteLine("Enter the Dimensions for R2");
            Console.Write("Length: ");
            int r2_l = Convert.ToInt32(Console.ReadLine());
            Console.Write("Breadth: ");
            int r2_b = Convert.ToInt32(Console.ReadLine());
            Rectangle R2 = new Rectangle(r2_l, r2_b);
            Console.WriteLine();
            Console.WriteLine("R1 Dimensions");
            R1.display();
            Console.WriteLine("\nR2 Dimensions");
            R2.display();
            Rectangle R3 = R1 + R2;
            Console.WriteLine("\nR3 (R1+R2) Dimensions");
            R3.display();
            Console.WriteLine("\nDisplaying Results");
            Console.Write("R1==R2: ");Console.WriteLine(R1==R2);
            Console.Write("R1<R2: "); Console.WriteLine(R1 < R2);
            Console.Write("R1>=R2: "); Console.WriteLine(R1 >= R2);
        }
    }
}
```

Output

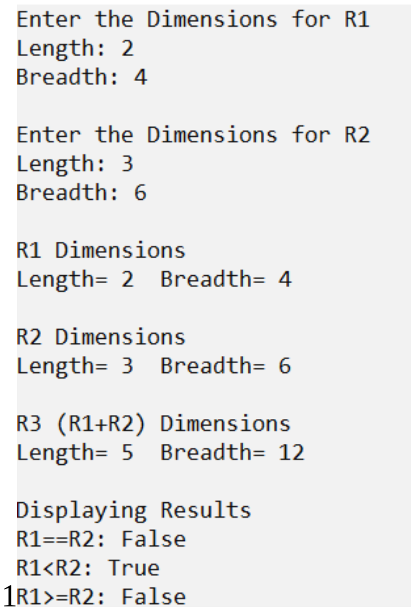
1.



```
Enter Matrix M1 Elements:
[0,0] = 2
[0,1] = 4
[1,0] = 6
[1,1] = 8

Enter Matrix M2 Elements:
[0,0] = 1
[0,1] = 3
[1,0] = 5
[1,1] = 7
M1 Displaying Matrix
2 4
6 8
M2 Displaying Matrix
1 3
5 7
M3 Matrix(M1+M2) Displaying Matrix
3 7
11 15
M4 Matrix(M1*2) Displaying Matrix
22 34
46 74
```

2.



```
Enter the Dimensions for R1
Length: 2
Breadth: 4

Enter the Dimensions for R2
Length: 3
Breadth: 6

R1 Dimensions
Length= 2  Breadth= 4

R2 Dimensions
Length= 3  Breadth= 6

R3 (R1+R2) Dimensions
Length= 5  Breadth= 12

Displaying Results
R1==R2: False
R1<R2: True
R1>=R2: False
```

1

Result

The C# console application using operator overloading has been executed successfully and the desired output is displayed on the screen.