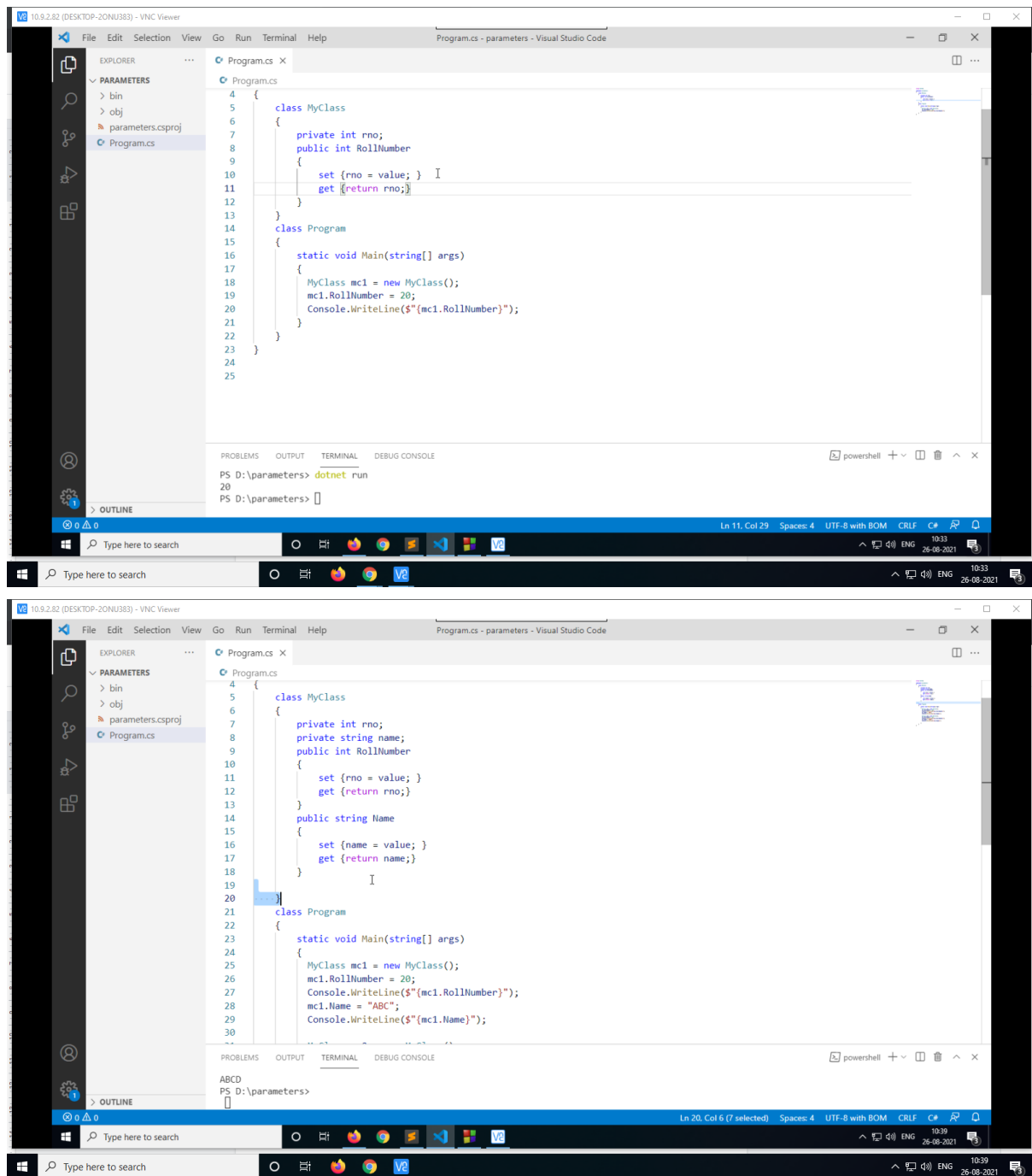
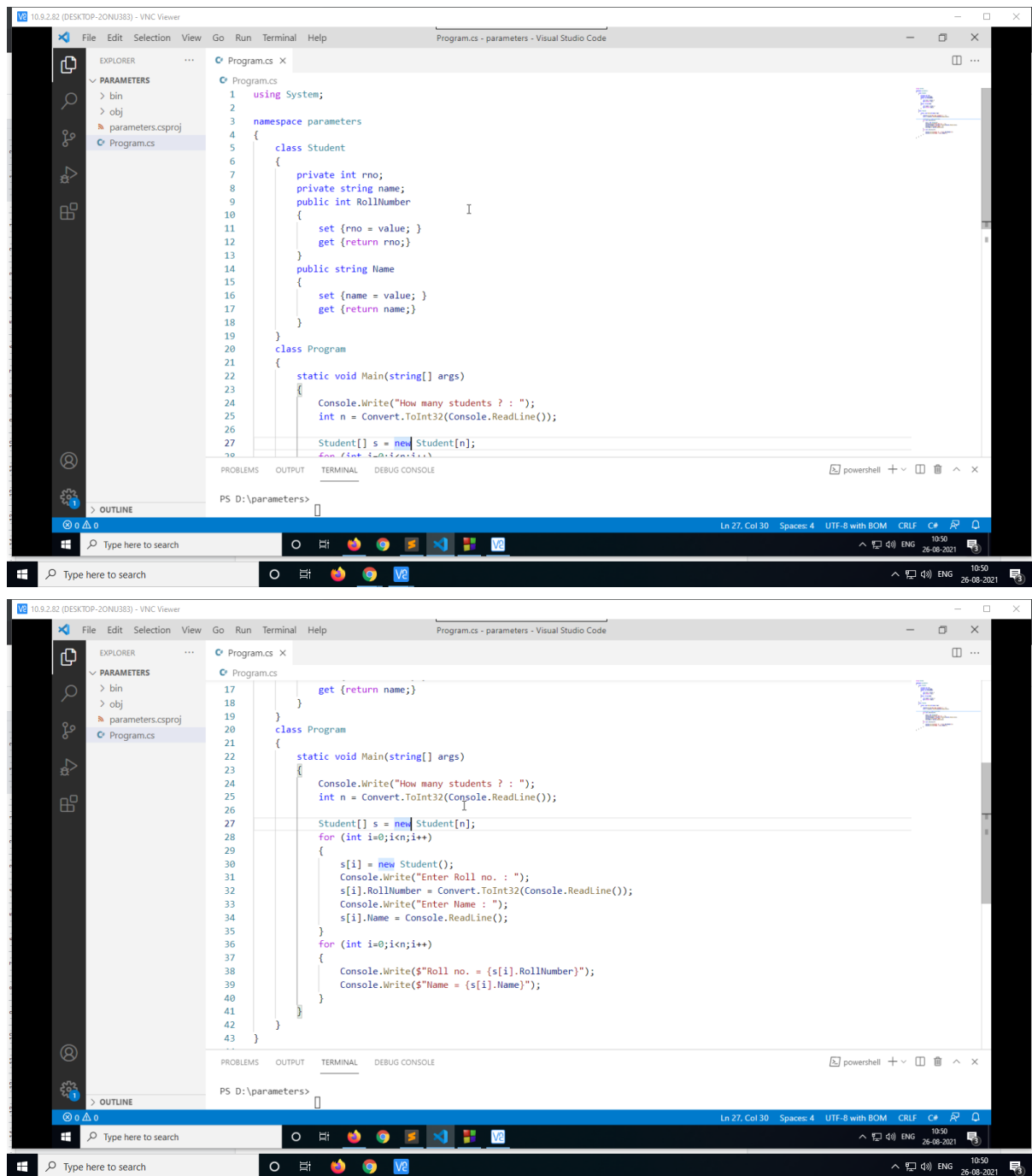


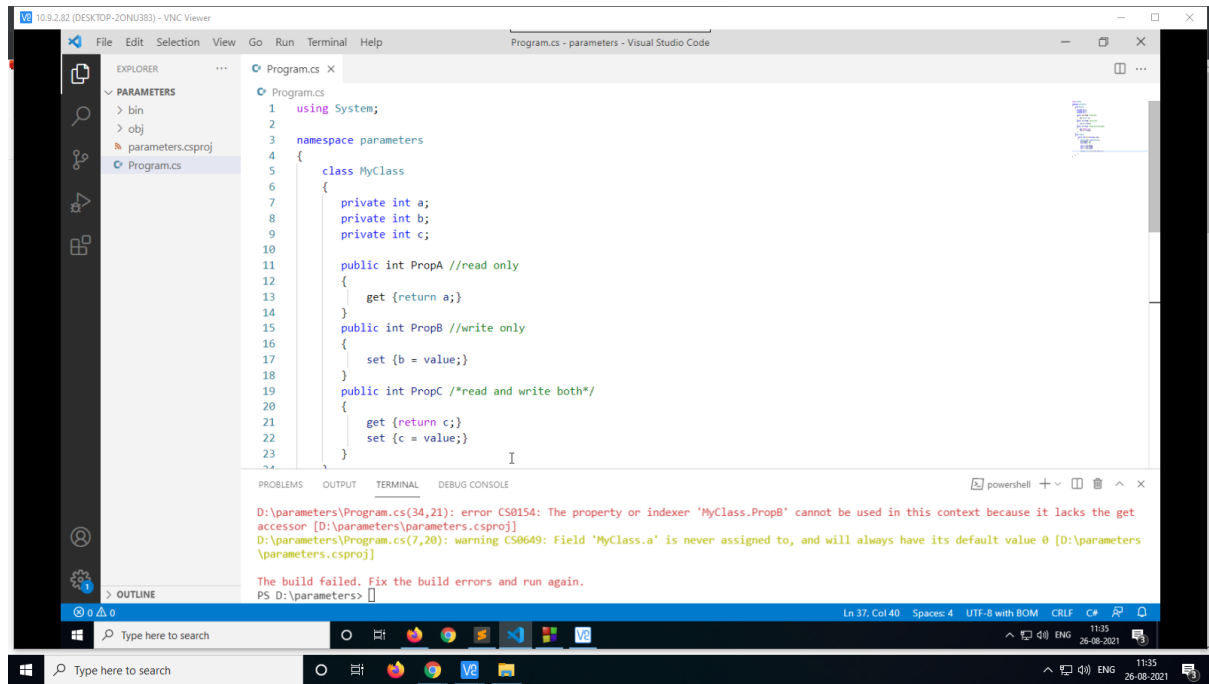
## Properties



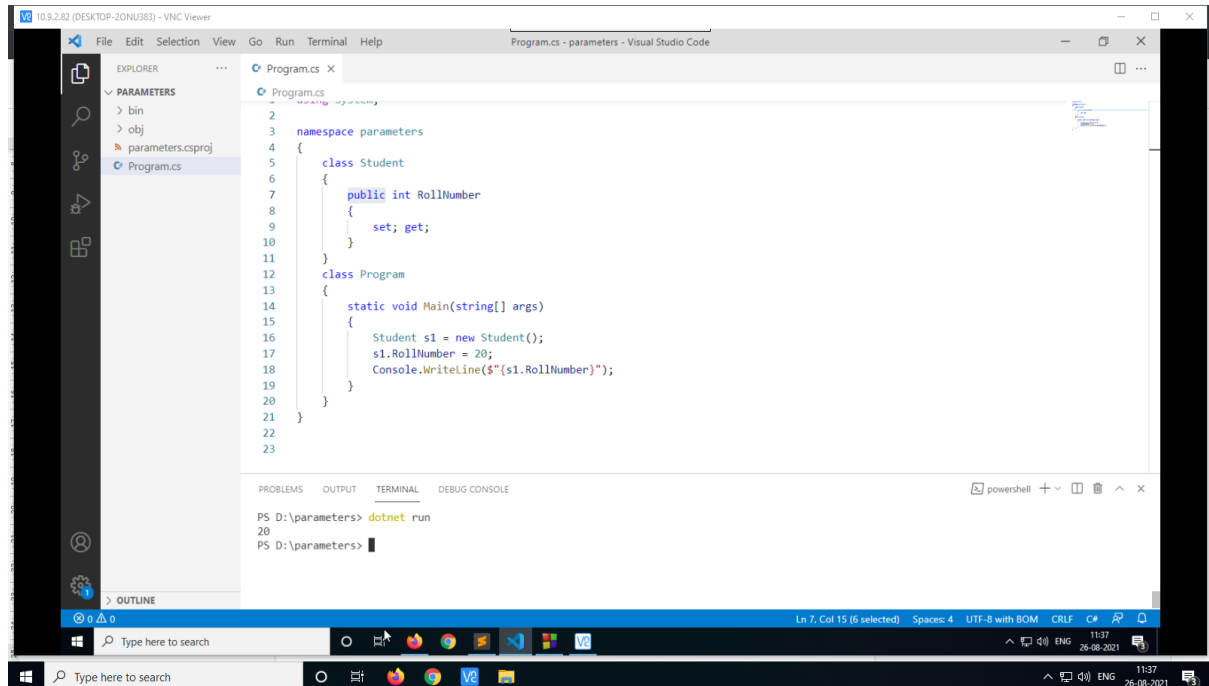
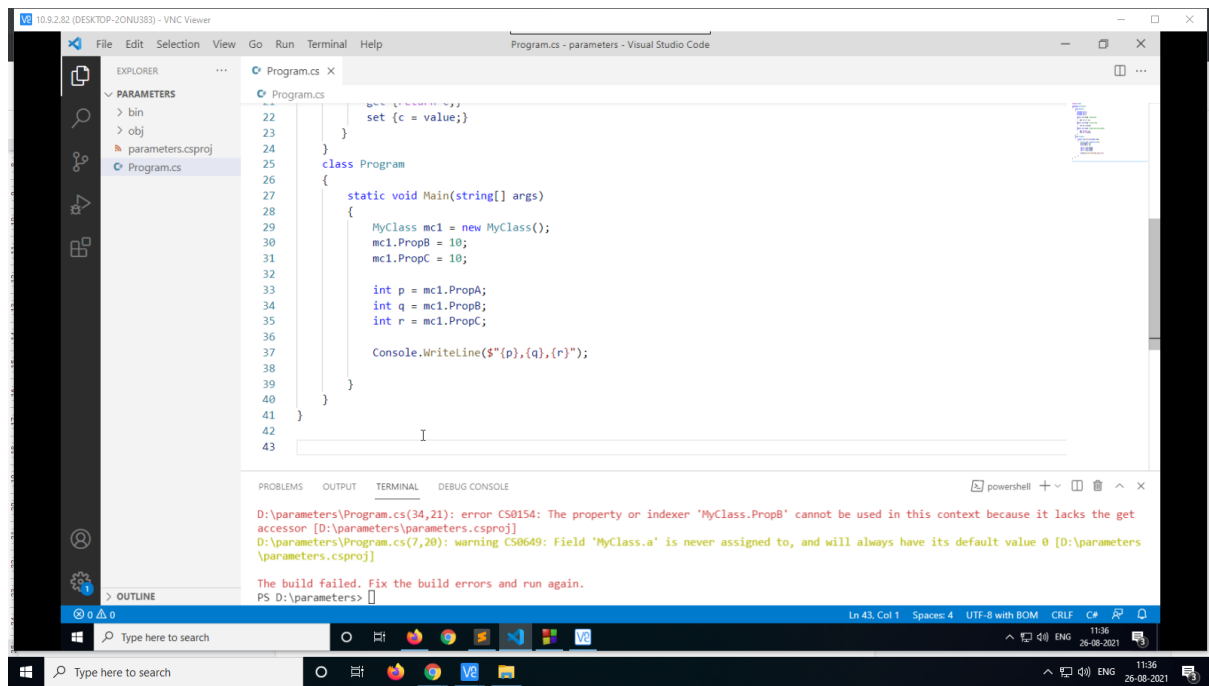
## Properties



# Properties

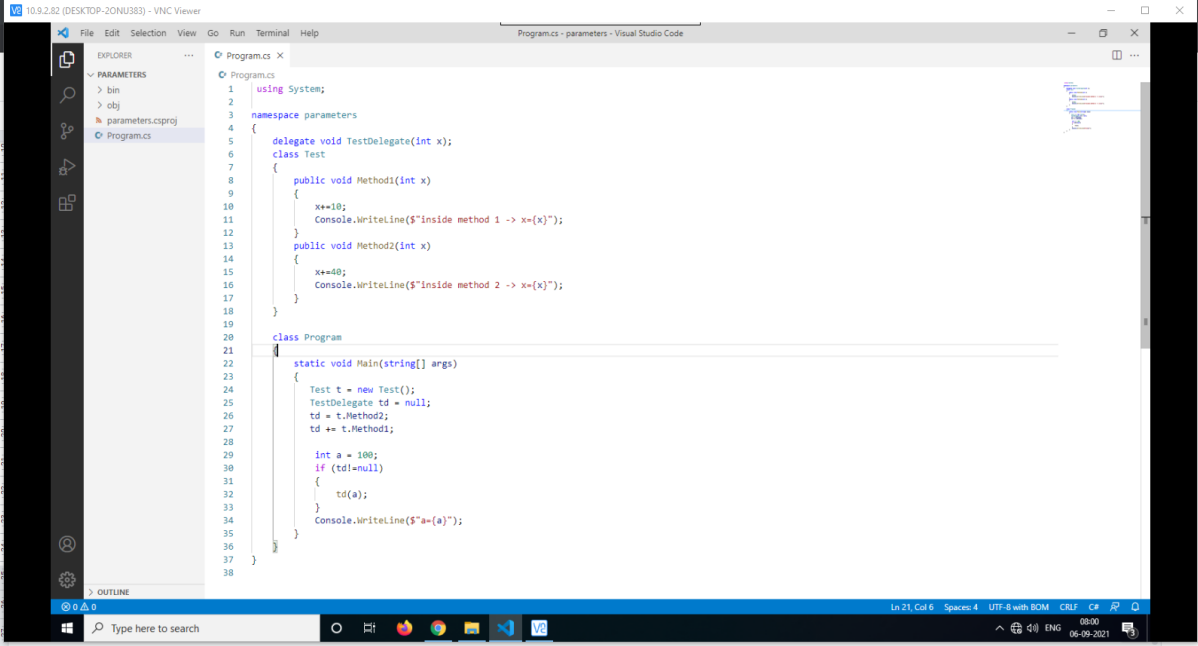


## Properties



## Delegate

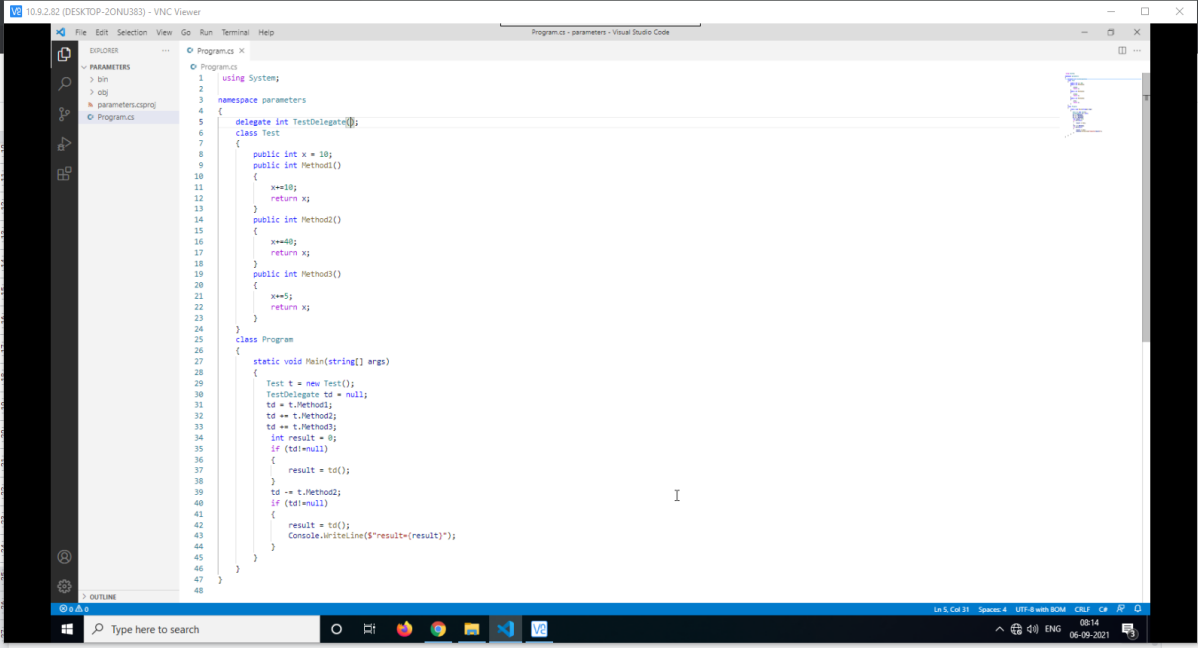
# Properties



This screenshot shows a Visual Studio Code editor window titled "Programs - parameters - Visual Studio Code". The Explorer sidebar on the left shows a project named "parameters" with files "bin", "obj", "parameters.csproj", and "Program.cs". The main editor displays the code in "Program.cs".

```
1 using System;
2
3 namespace parameters
4 {
5     delegate void TestDelegate(int x);
6     class Test
7     {
8         public void Method1(int x)
9         {
10             x+=10;
11             Console.WriteLine($"inside method 1 -> x={x}");
12         }
13         public void Method2(int x)
14         {
15             x+=40;
16             Console.WriteLine($"inside method 2 -> x={x}");
17         }
18     }
19
20     class Program
21     {
22         static void Main(string[] args)
23         {
24             Test t = new Test();
25             TestDelegate td = null;
26             td = t.Method2;
27             td += t.Method1;
28
29             int a = 100;
30             if (td!=null)
31             {
32                 td(a);
33             }
34             Console.WriteLine($"a={a}");
35         }
36     }
37 }
38
```

The status bar at the bottom indicates "Ln 21, Col 6", "Spaces: 4", "UTF-8 with BOM", "C# 7.3", and the date "08:00 06-09-2021".

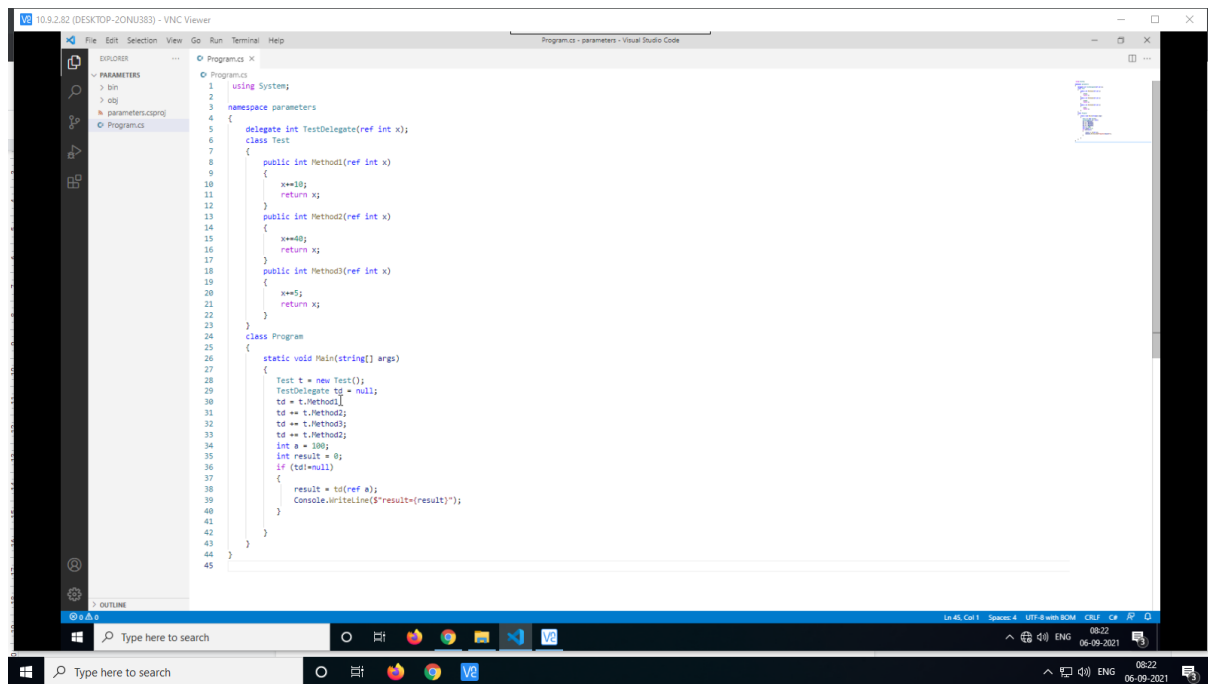


This screenshot shows the same Visual Studio Code editor window, but with a cursor positioned at line 41. The code in "Program.cs" is as follows:

```
1 using System;
2
3 namespace parameters
4 {
5     delegate int TestDelegate();
6     class Test
7     {
8         public int x = 10;
9         public int Method1()
10         {
11             x+=10;
12             return x;
13         }
14         public int Method2()
15         {
16             x+=40;
17             return x;
18         }
19         public int Method3()
20         {
21             x+=5;
22             return x;
23         }
24     }
25
26     class Program
27     {
28         static void Main(string[] args)
29         {
30             Test t = new Test();
31             TestDelegate td = null;
32             td = t.Method1;
33             td += t.Method2;
34             int result = 0;
35             if (td!=null)
36             {
37                 result = td();
38             }
39             td += t.Method3;
40             if (td!=null)
41             {
42                 result = td();
43                 Console.WriteLine($"result={result}");
44             }
45         }
46     }
47 }
48
```

The status bar at the bottom indicates "Ln 5, Col 31", "Spaces: 4", "UTF-8 with BOM", "C# 7.3", and the date "08:14 06-09-2021".

## Properties



write a c# program to create a class Test which has two methods Addition and Subtraction that takes three parameteres (int) but does not return any value

create a delegate that matches the signature of above two methods

create a another class to check Test class ,create a invocation list and excute the daleget

```
using System;

namespace LabProgram
{
    public delegate void dg_cal(int a,int b,int c);
    class Test
    {
        public void Addition(int a,int b,int c)
        {
            int add = a+b+c;
            Console.WriteLine($"Addition : {add}");
        }

        public void Subtraction(int a,int b,int c)
        {
            int sub = a-b;
            Console.WriteLine($"Subtraction : {sub}");
        }
    }
}
```

## Properties

```
        {  
            int add = a-b-c;  
            Console.WriteLine($"Subtraction : {add}");  
        }  
    }  
  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Test t = new Test();  
            dg_cal dc;  
  
            dc = t.Addition;  
            dc +=t.Subtraction;  
  
            if(dc!=null){  
                dc(2,2,2);  
            }  
        }  
    }  
}
```

## Event

10.9.2.82 (DESKTOP-2ONU383) - VNC Viewer

Dr. hetal thakar - atmija university, rajkot

### EVNETS IN C#.NET

#### WHAT IS AN EVENT?

- An object to indicate that particular action is going to happen.
- Object that raises an event is called "event sender".

#### PUBLISHER-SUBSCRIBER MODEL

- There are two parts in any event handling program.
- One part is Publisher that contains definition of events and delegates.
- And another part is Subscriber that accepts the event and provides an event handler.
- Publisher publishes / notifies about an event.
- All subscriber will get notification about event.
- Event will be notified to only those who have subscribed.
- E.g. If Google does webinar on particular date, then updates about this event will be notified via email only if you have subscribed to this particular publisher.
- Here in above example Google is a Publisher, webinar is an event and as you have subscribed to receive email from google you are subscriber.
- Publisher will determine when to raise event.
- Subscriber can determine what actions to take as a response to that event.
- In C# events follows publisher-subscriber model.

These terminologies can be defined as below:

- All subscriber will get notification about event.
- Event will be notified to only those who have subscribed.
- E.g. If Google does webinar on particular date, then updates about this event will be notified via email only if you have subscribed to this particular publisher.
- Here in above example Google is a Publisher, webinar is an event and as you have subscribed to receive email from google you are subscriber.
- Publisher will determine when to raise event.
- Subscriber can determine what actions to take as a response to that event.
- In C# events follows publisher-subscriber model.

These terminologies can be defined as below:

- **Publisher:** A class or struct that publishes an event so that other classes can be notified when the event occurs.
- **Subscriber:** A class or struct that registers to be notified when the event occurs
- **Event handler:** A method that is registered with the publisher, by the subscriber, and is executed when the publisher raises the event. The event handler method can be declared in the same class or struct as the event or in a different class or struct.
- **Raising an event:** The term for invoking or firing an event. When an event is raised, all the methods registered with that event are invoked.

#### EVENTS WITH DELEGATES

- Delegates are used to reference a method.



# Properties

The screenshot shows a PDF document titled "EVENTS" with the following content and annotations:

- registration** (annotated with a red bracket)
- Code that raises the event** (annotated with a red bracket)
- notified when it has been raised.** (annotated with a green box)
- This is the code that connects the event handlers to the event.** (annotated with a green box)
- This is the code in the publisher that "fires" the event, causing it to invoke all the event handlers registered with it.** (annotated with a green box)

**KEY POINTS ABOUT EVENTS**

- An Event is created using event keyword.
- An Event has no return type and it is always void.
- All events are based on delegates.
- All the published events must have a listening object.
- All Events should be defined starting with "On" keyword.

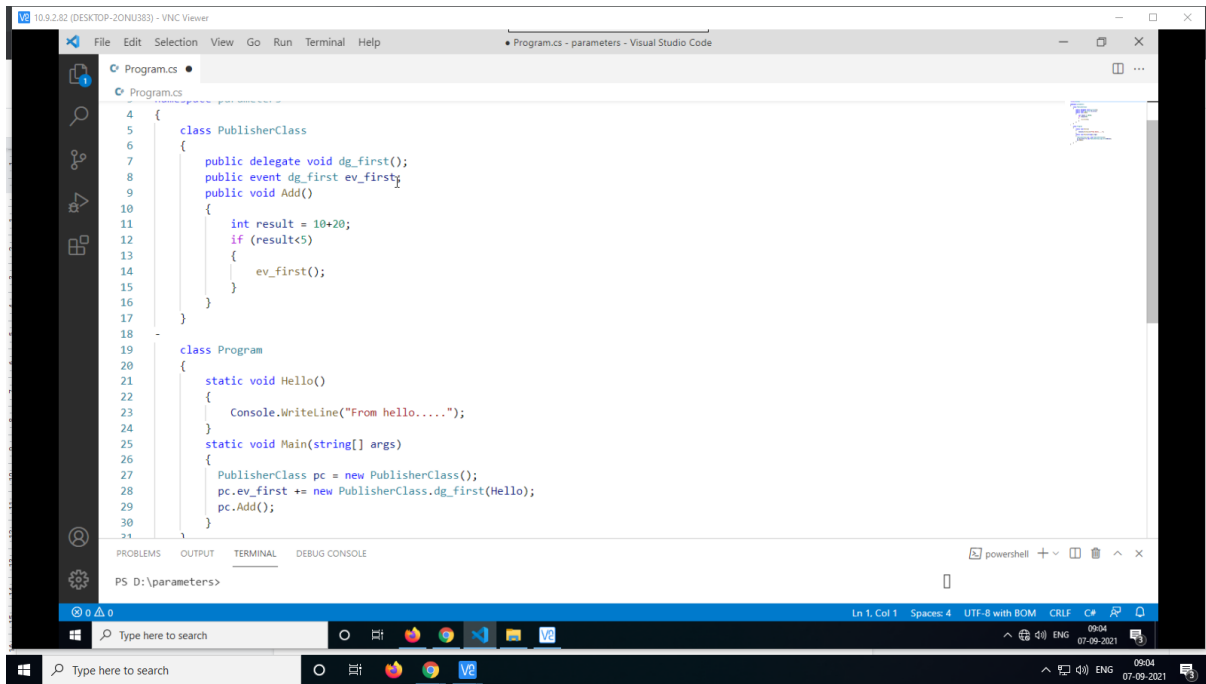
**STEPS TO WRITE EVENT RELATED PROGRAM IN C#**

- Step 1: Define a Delegate
- Step 2: Define an Event with same name of Delegates.
- Step 3: Define an Event Handler that respond when event raised.
- Step 4: You must have method ready for delegates.

**PROGRAM 1 : DEMO OF CREATING AND RAISING EVENT WITH DELEGATES**

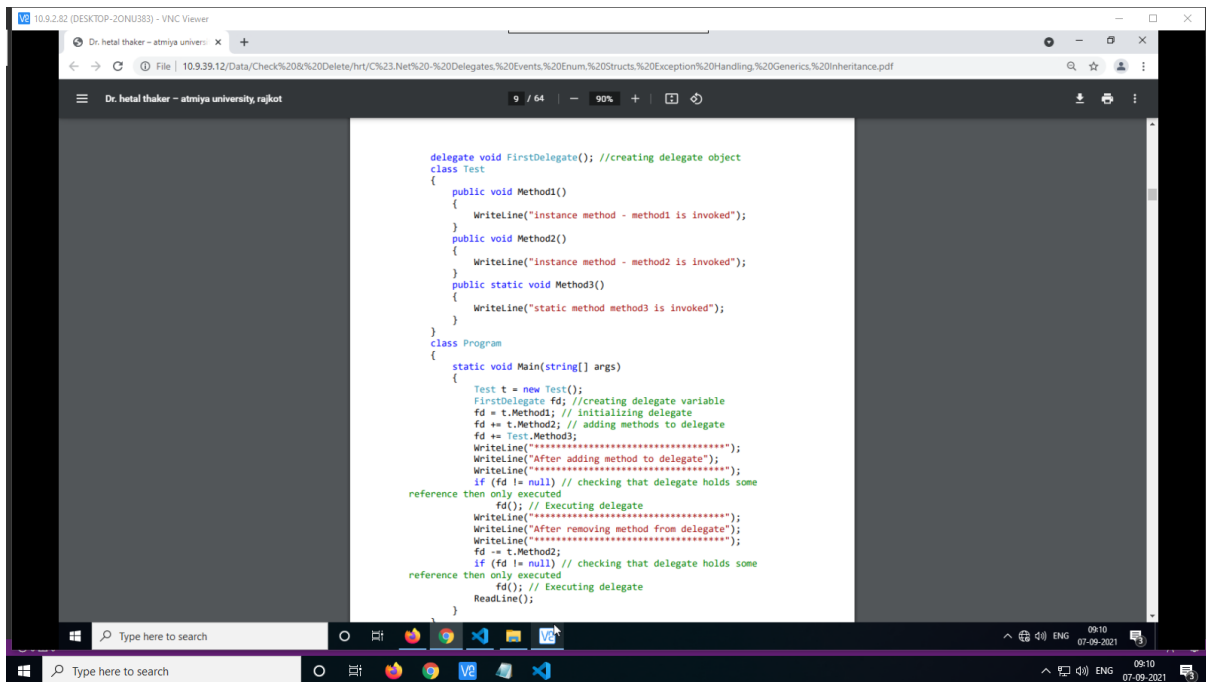
This program adds two numbers. Only condition is if the sum of number is even it fires an

# Properties



The screenshot shows the Visual Studio Code editor with a C# file named Program.cs. The code defines a PublisherClass with a delegate dg\_first, an event dg\_first ev\_first, and an Add() method. It also defines a Program class with a static Hello() method and a Main() method. The Main() method creates a PublisherClass object, adds the Hello() method to the dg\_first event, and calls Add().

```
4 {
5     class PublisherClass
6     {
7         public delegate void dg_first();
8         public event dg_first ev_first;
9         public void Add()
10        {
11            int result = 10+20;
12            if (result<5)
13            {
14                ev_first();
15            }
16        }
17    }
18
19    class Program
20    {
21        static void Hello()
22        {
23            Console.WriteLine("From hello.....");
24        }
25        static void Main(string[] args)
26        {
27            PublisherClass pc = new PublisherClass();
28            pc.ev_first += new PublisherClass.dg_first(Hello);
29            pc.Add();
30        }
31    }
```



The screenshot shows a web browser displaying a PDF document titled "Dr. hetal thaker - atmija university, rajkot". The document contains C# code examples for delegates. The code defines a FirstDelegate delegate, a Test class with three methods (Method1, Method2, Method3), and a Program class with a Main method. The Main method demonstrates how to create a delegate object, add methods to it, and execute it.

```
delegate void FirstDelegate(); //creating delegate object
class Test
{
    public void Method1()
    {
        WriteLine("Instance method - method1 is invoked");
    }
    public void Method2()
    {
        WriteLine("Instance method - method2 is invoked");
    }
    public static void Method3()
    {
        WriteLine("static method method3 is invoked");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Test t = new Test();
        FirstDelegate fd; //creating delegate variable
        fd = t.Method1; //initializing delegate
        fd += t.Method2; //adding methods to delegate
        fd += t.Method3;
        WriteLine("*****");
        WriteLine("After adding method to delegate");
        WriteLine("*****");
        if (fd != null) //checking that delegate holds some
            reference then only executed
            fd(); //Executing delegate
        WriteLine("*****");
        WriteLine("After removing method from delegate");
        WriteLine("*****");
        fd -= t.Method2;
        if (fd != null) //checking that delegate holds some
            reference then only executed
            fd(); //Executing delegate
        ReadLine();
    }
}
```

## Delegate Demo

```
using System;

namespace Event
{
    public delegate void dg_test(int a,int b);
    class Program
    {
        static void Addition(int a,int b)
        {
            Console.WriteLine($"{a} + {b} : {a+b}");
        }

        static void Multiplication(int a,int b)
        {
            Console.WriteLine($"{a} * {b} : {a*b}");
        }

        static void Main(string[] args)
        {
            dg_test dt = Addition;
            dt += Multiplication;

            if(dt!=null){
                dt(20,5);
            }
        }
    }
}
```

## Event Demo

```
using System;

namespace Event
{
    public delegate void dg_test();

    class Program
    {
        public static event dg_test ev_test;

        static void india() {
            Console.WriteLine("I'm From India");
        }

        static void canada() {
            Console.WriteLine("Hello Canada");
        }

        static void usa() {
            Console.WriteLine("Hi USA");
        }

        static void Main(string[] args)
        {
            ev_test += new dg_test(india);
            ev_test += new dg_test(canada);
            ev_test += new dg_test(usa);

            ev_test();
        }
    }
}
```

## Event Demo 2

```
using System;

namespace Event
{
    public delegate void dg_test(string s);
    //Publiser Class
    public class Operation
    {
        public event dg_test ev_test;
        public void Action(string s)
        {
            if(ev_test != null)
            {
                ev_test(s);
                Console.WriteLine($"{s}");
            }
            else{
                Console.WriteLine("Event Not Register");
            }
        }
    }

    class Program
    {
        public static void CatchEvent(string s)
        {
            Console.WriteLine("Catch Event Method Calling");
        }
        static void Main(string[] args)
        {
            Operation o = new Operation();
            o.ev_test += new dg_test(CatchEvent); //Binding Event
            o.Action("Event Calling");
        }
    }
}
```

## Structure

### Struct demo-1

```
using System;

namespace Structure
{
    struct Student{
        public int rno;
        public string name;
        public string email_id;
    }

    class Program
    {
        static void Main(string[] args)
        {
            Student s1 = new Student();
            s1.rno = 1;
            s1.name = "Ankit";
            s1.email_id = "ankitrudani@gmail.com";
            Console.WriteLine($"Roll No : {s1.rno}\nName : {s1.name}\nEmail Id : {s1.email_id}");
        }
    }
}
```

```
using System;

namespace Structure
{
    struct Employee{
        public int emp_id;
        public string name;
    }
}
```

## Properties

```
class Program
{
    static void Main(string[] args)
    {
        Employee e1,e2;
        e1.emp_id = 1;
        e1.name = "Ankit";

        e2.emp_id = 2;
        e2.name = "Raj";
        Console.WriteLine($"Employee
:\n{e1.emp_id}\t{e1.name}\nEmployee : \n{e2.emp_id}\t{e2.name}");
    }
}
```

```
using System;

namespace Structure
{
    v
}
```

```
using System;

namespace Structure
{
    struct Employee{
        private int emp_id;
        private string name;

        public Employee(int emp_id,string name){
            this.emp_id = emp_id;
            this.name = name;
        }

        public void PrintData(){
```

## Properties

```
        Console.WriteLine($"Id : {emp_id} , Name : {name}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        int n;
        int emp_id;
        string name;

        Console.Write("Enter How Many Employees : ");
        n = Convert.ToInt32(Console.ReadLine());

        Employee [] e = new Employee [n];

        for(int i=0;i<n;i++){
            Console.Write("Enter Employee Id : ");
            emp_id = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Employee Name : ");
            name = Console.ReadLine();
            e[i] = new Employee(emp_id,name);
        }

        for(int i=0;i<n;i++){
            e[i].PrintData();
        }
    }
}
```

```
using System;

namespace Structure
{
    class ClassDemo
    {
        public int a;
        public int b;
    }
}
```



## Properties

```
struct StuctDemo
{
    public int x;
    public int y;
}

class Program
{
    static void Main(string[] args)
    {
        // ClassDemo c1 = new ClassDemo();
        // ClassDemo c2 = null;
        // c1.a = 10;
        // c1.b = 20;
        // c2 = c1;
        // Console.WriteLine("Class Output : ");
        // Console.WriteLine($"{c1.a},{c1.b}");
        // Console.WriteLine($"{c2.a},{c2.b}");

        // c1.a = c1.a+10;
        // c1.b = c1.b+20;

        // Console.WriteLine("Class Output : ");
        // Console.WriteLine($"{c1.a},{c1.b}");
        // Console.WriteLine($"{c2.a},{c2.b}");

        StuctDemo s1 = new StuctDemo();
        StuctDemo s2;

        s1.x = 10;
        s1.y = 20;

        s2 = s1;

        Console.WriteLine("Structure Output : ");
        Console.WriteLine($"{s1.x},{s1.y}");
        Console.WriteLine($"{s2.x},{s2.y}");

        s1.x = s1.x+10;
        s1.y = s1.y+20;

        Console.WriteLine("Structure Output : ");
```

## Properties

```
        Console.WriteLine($"{s1.x},{s1.y}");  
        Console.WriteLine($"{s2.x},{s2.y}");  
    }  
}  
}
```