

.NET PROGRAMMING LAB 8

Name: CH BALA GOWTHAM

Id: 2000032067

Section: S-13

Task1: Task Delegates and events refers to task Collections

To add the following new functionalities to the project created in task Collections:

Include two events in the CustomArray type:

- The OnChangeElement event occurs when the indexer changes the element value(if the old and new element values match, the event is not raised)
- The OnChangeEqualElement event occurs if a value equal to the index of the changed element is written to the element(if the old and new values of the element match, the event is not raise)

Use the ArrayHandler delegate to create the event

The event handler takes two parameters: an object sender – a reference to the CustomArray instance that generated the event, and an event argument ArrayEventArgs<T> e. In the event argument, write in the id field the index by which the user changes the element of the CustomArray array, in the value field - the new value of the element by the Id index, in the message field – an arbitrary string message.

Code:

```
using System;
```

```
namespace CustomArrayProject
```

```
{
```

```
    public class ArrayEventArgs<T> : EventArgs
```

```
    {
```

```
        public int Id { get; set; }
```

```
public T Value { get; set; }  
public string Message { get; set; }  
}
```

```
public delegate void ArrayHandler<T>(object sender, EventArgs<T> e);
```

```
public class CustomArray<T>  
{  
    private T[] array;  
  
    public event ArrayHandler<T> OnChangeElement;  
    public event ArrayHandler<T> OnChangeEqualElement;
```

```
    public CustomArray(int size)  
    {  
        array = new T[size];  
    }
```

```
    public T this[int index]  
    {  
        get { return array[index]; }  
        set  
        {  
            T oldValue = array[index];  
            array[index] = value;
```

```

        if (!object.Equals(oldValue, value))
        {
            OnChangeElement?.Invoke(this, new ArrayEventArgs<T> { Id =
index, Value = value, Message = "Element value changed" });
        }
        else if (object.Equals(value, index))
        {
            OnChangeEqualElement?.Invoke(this, new ArrayEventArgs<T> { Id
= index, Value = value, Message = "Element value equal to index changed" });
        }
    }
}
}

```

class Program

```

{
    static void Main(string[] args)
    {
        CustomArray<int> myArray = new CustomArray<int>(5);

        myArray.OnChangeElement += MyArray_OnChangeElement;
        myArray.OnChangeEqualElement += MyArray_OnChangeEqualElement;

        myArray[0] = 1;
        myArray[1] = 2;
    }
}

```

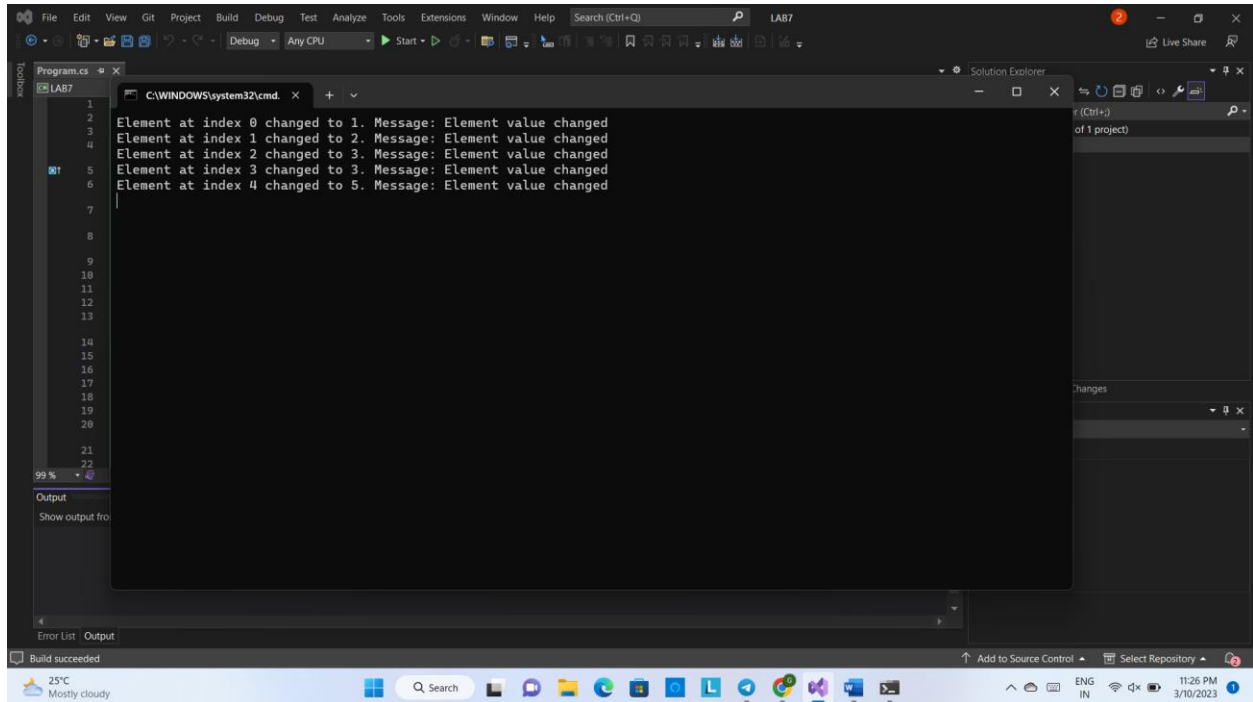
```
myArray[2] = 3;
myArray[3] = 3; // this will raise the OnChangeEqualElement event
myArray[4] = 5;
```

```
    Console.ReadKey();
}
```

```
private static void MyArray_OnChangeElement(object sender,
ArrayEventArgs<int> e)
{
    Console.WriteLine($"Element at index {e.Id} changed to {e.Value}.
Message: {e.Message}");
}
```

```
private static void MyArray_OnChangeEqualElement(object sender,
ArrayEventArgs<int> e)
{
    Console.WriteLine($"Element at index {e.Id} equal to {e.Value} changed.
Message: {e.Message}");
}
}
```

Output:



The screenshot shows the Visual Studio Code interface with a C# program named 'LAB7' open in the editor. The program is a console application that iterates through an array of 5 elements, changing their values and printing a message for each change. The output window shows the following messages:

```
Element at index 0 changed to 1. Message: Element value changed
Element at index 1 changed to 2. Message: Element value changed
Element at index 2 changed to 3. Message: Element value changed
Element at index 3 changed to 3. Message: Element value changed
Element at index 4 changed to 5. Message: Element value changed
```

The status bar at the bottom indicates 'Build succeeded'.