CMR
INSTITUTE OF
TECHNOLOGY

USN

Internal Assessment Test 3 – June 2022

Sub:	**Programming Using C#**							**Sub Code:**	**20MCA42**
Date:	**18/06/2022**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem:**	**IV**	**Branch:**	**MCA**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
	PART I		CO	RBT
1	What is meant by the .Net delegate type? Explain the concept with example code. **OR**	[5] [5]	CO2	L2
2	What are delegates? Explain with code example, the concept of multicasting with delegates.	[4] [6]	CO2	L3
3	**PART II** Describe how to implement the multiple events. Give example **OR**	[10]	CO2	L3
4	How to detach the event from the event source? Give an example	[10]	CO2	L2

5	**PART III** Explain exception handling with a sample program to handle multiple exceptions. **OR**	[10]	CO2	L2
6	Explain .NET Exception handling with valid example code	[10]	CO2	L2
7	**PART IV** Explain the architecture of ADO.NET with a neat diagram **OR**	[10]	CO2	L1
8	Explain WPF.What are the advantages of WPF?	[10]	CO2	L2
9	**PARTV** With the help of code explain : a) Listbox control b) Checked Listbox control **OR**	[5] [5]	CO2	L3
10	Explain Keyboard and Mouse event handling.	[5] [5]	CO2	L3

CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | |

Internal Assessment Test 3 – June 2022

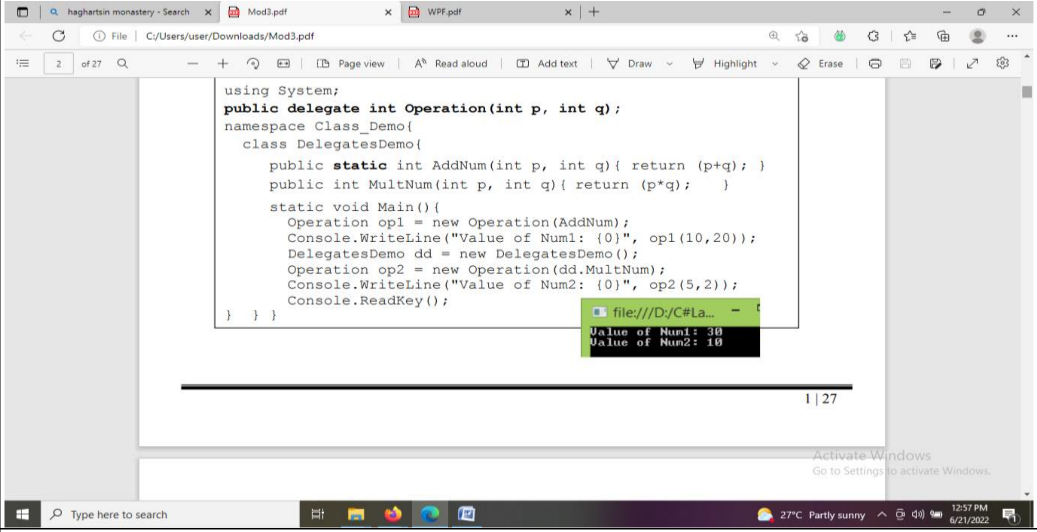
Sub:	**Programming Using C#**							**Sub Code:**	**20MC**
Date:	**18/06/2022**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem:**	**IV**	**Branch:**	**MC**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	R

1 What is meant by the .Net delegate type? Explain the concept with example code. [5] [5] CO2

Delegates: A delegate is special type of object that contains the details of a method rather than data. OR A delegate is a class type object, which is used to invoke a method that has been encapsulated into it at the All delegates are implicitly derived from the System.Delegate class.• Delegates are especially used for implementing events and the call-back methods. • A delegate is a reference type variable that holds the references to a method of any class. •time of its creation. Uses: Suppose you need to create a program that requires data, such as student information, to display it on a website. This data can be retrieved by calling a method without having to know to compiler time which method is to be invoked. In this case, you need to create an object of delegate and encapsulate a reference to that method inside the delegate object

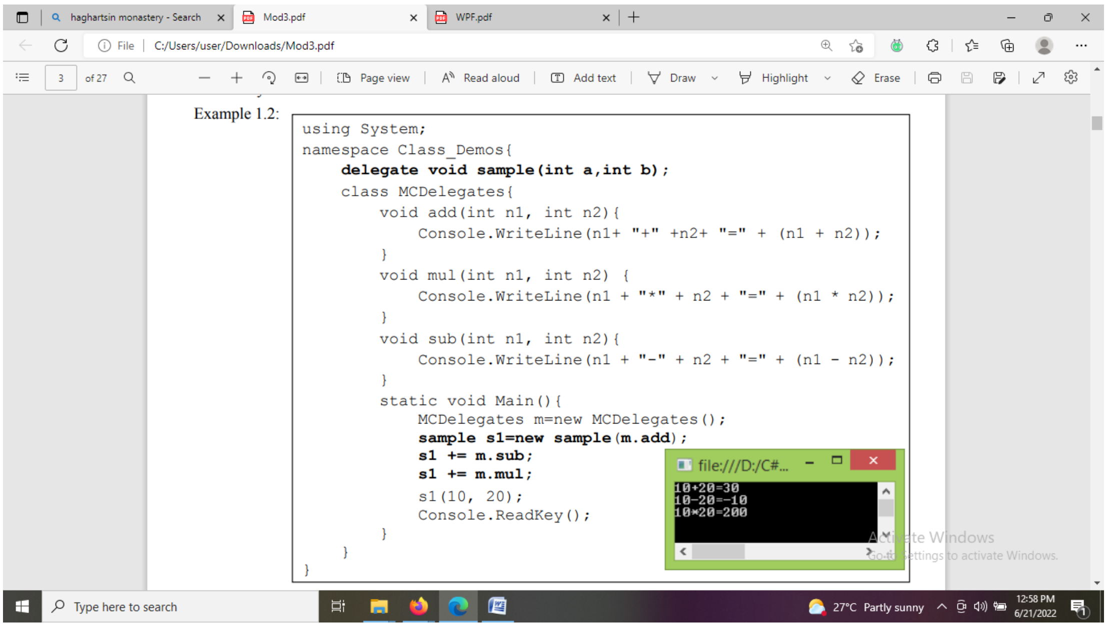


```
using System;
public delegate int Operation(int p, int q);
namespace Class_Demo{
  class DelegatesDemo{
     public static int AddNum(int p, int q){ return (p+q); }
     public int MultNum(int p, int q){ return (p*q);    }
     static void Main(){
       Operation op1 = new Operation(AddNum);
       Console.WriteLine("Value of Num1: {0}", op1(10,20));
       DelegatesDemo dd = new DelegatesDemo();
       Operation op2 = new Operation(dd.MultNum);
       Console.WriteLine("Value of Num2: {0}", op2(5,2));
       Console.ReadKey();
}   } }
```

```
Value of Num1: 30
Value of Num2: 10
```

2 What are delegates? Explain with code example, the concept of multicasting with delegates. [4] [6] CO2

Delegates: A delegate is special type of object that contains the details of a method rather than data. OR A delegate is a class type object, which is used to invoke a method that has been encapsulated into it at the All delegates are implicitly derived from the System.Delegate class.• Delegates are especially used for implementing events and the call-back methods. • A delegate is a reference type variable that holds the references to a method of any class. •time of its creation. Uses: Suppose you need to create a program that requires data, such as student information, to display it on a website. This data can be retrieved by calling a method without having to know to compiler time which method is to be invoked. In this case, you need to create an object of delegate and encapsulate a reference to that method inside the delegate object



Example 1.2:

```
using System;
namespace Class_Demos{
    delegate void sample(int a,int b);
    class MCDelegates{
        void add(int n1, int n2){
            Console.WriteLine(n1+ "+" +n2+ "=" + (n1 + n2));
        }
        void mul(int n1, int n2) {
            Console.WriteLine(n1 + "*" + n2 + "=" + (n1 * n2));
        }
        void sub(int n1, int n2){
            Console.WriteLine(n1 + "-" + n2 + "=" + (n1 - n2));
        }
        static void Main(){
            MCDelegates m=new MCDelegates();
            sample s1=new sample(m.add);
            s1 += m.sub;
            s1 += m.mul;
            s1(10, 20);
            Console.ReadKey();
        }
    }
}
```

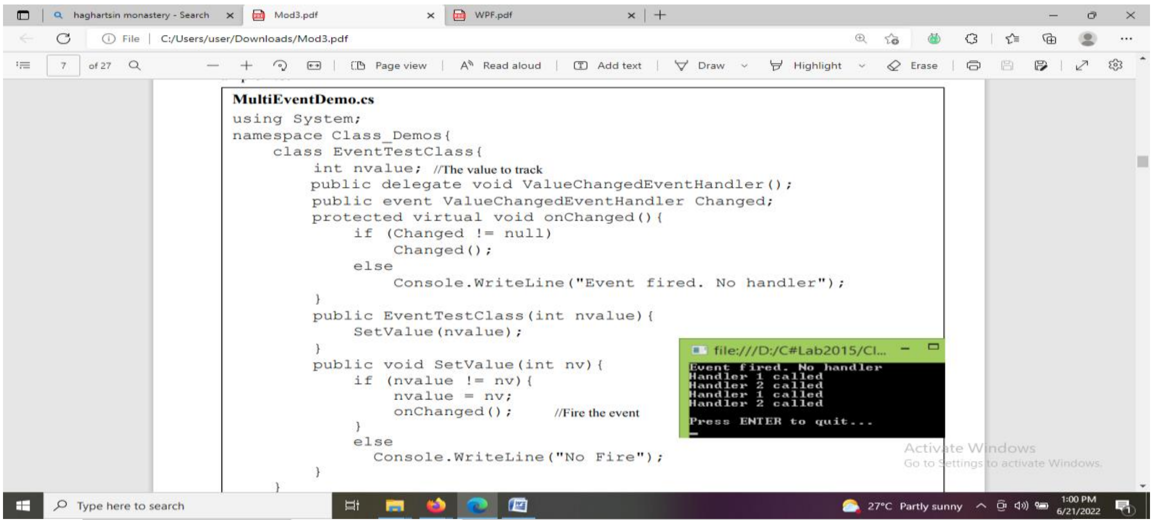
```
10+20=30
10-20=-10
10*20=200
```

PART II

[10]

3 Describe how to implement the multiple events. Give example

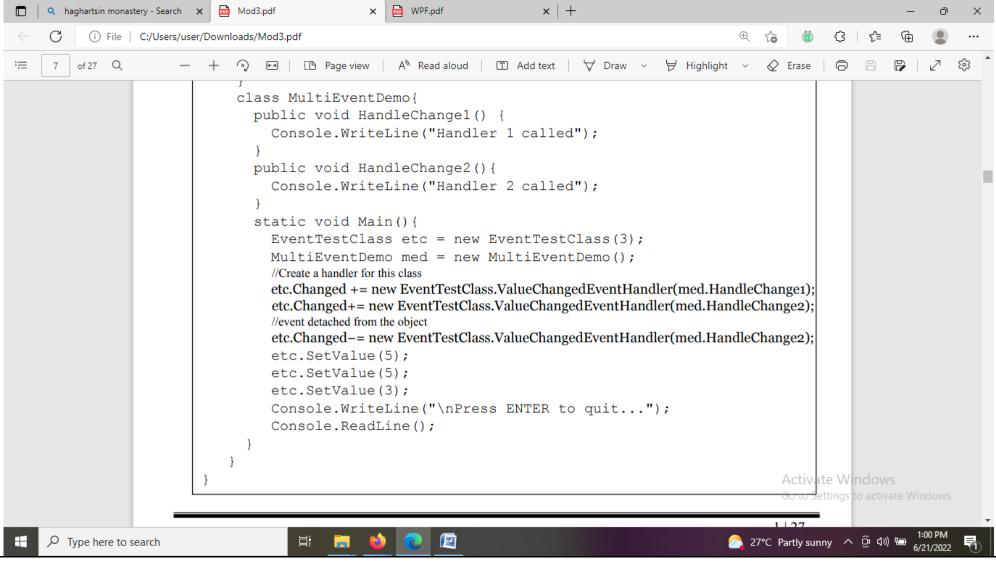
Like delegates, events can be multicast. This enables multiple objects to respond to an event notification, you used the += operator to add the event handler



MultiEventDemo.cs

```
using System;
namespace Class_Demos{
    class EventTestClass{
        int nvalue;  //The value to track
        public delegate void ValueChangedEventHandler();
        public event ValueChangedEventHandler Changed;
        protected virtual void onChanged(){
            if (Changed != null)
                Changed();
            else
                Console.WriteLine("Event fired. No handler");
        }
        public EventTestClass(int nvalue){
            SetValue(nvalue);
        }
        public void SetValue(int nv){
            if (nvalue != nv){
                nvalue = nv;
                onChanged();       //Fire the event
            }
            else
              Console.WriteLine("No Fire");
        }
    }
```

```
Event fired. No handler
Handler 1 called
Handler 2 called
Handler 1 called
Handler 2 called

Press ENTER to quit...
_
```



```
    class MultiEventDemo{
      public void HandleChange1() {
        Console.WriteLine("Handler 1 called");
      }
      public void HandleChange2(){
        Console.WriteLine("Handler 2 called");
      }
      static void Main(){
        EventTestClass etc = new EventTestClass(3);
        MultiEventDemo med = new MultiEventDemo();
        //Create a handler for this class
        etc.Changed += new EventTestClass.ValueChangedEventHandler(med.HandleChange1);
        etc.Changed+= new EventTestClass.ValueChangedEventHandler(med.HandleChange2);
        //event detached from the object
        etc.Changed-= new EventTestClass.ValueChangedEventHandler(med.HandleChange2);
        etc.SetValue(5);
        etc.SetValue(5);
        etc.SetValue(3);
        Console.WriteLine("\nPress ENTER to quit...");
        Console.ReadLine();
      }
    }
}
```

CO2

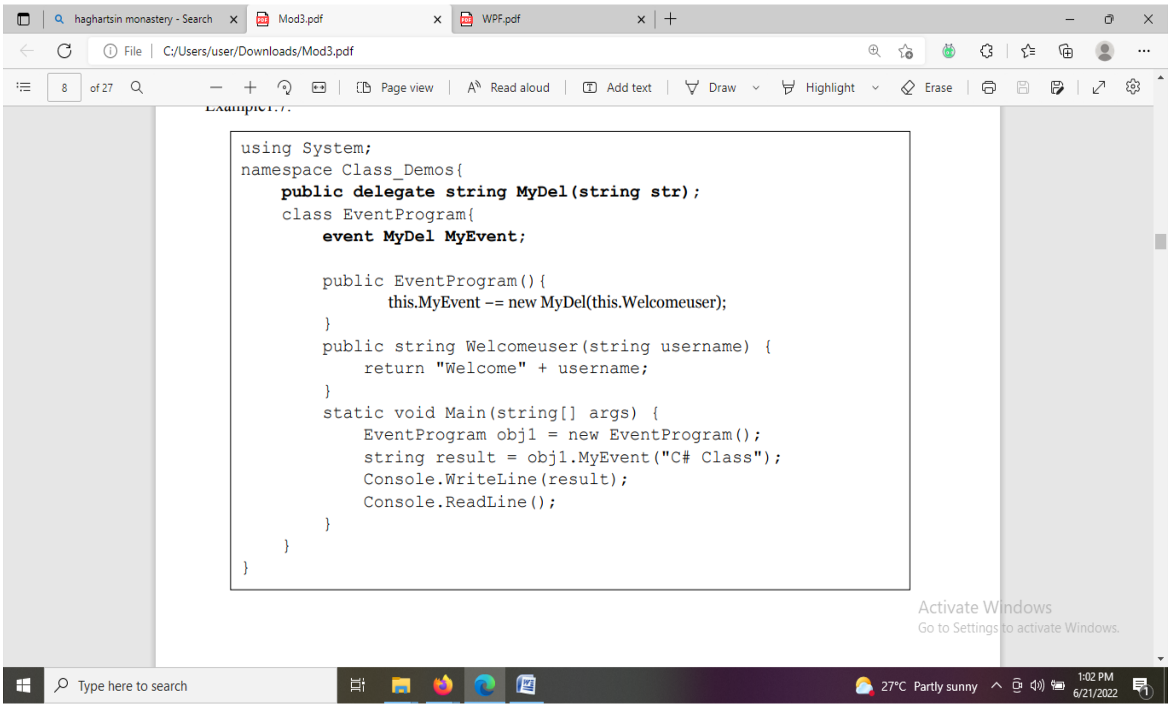
4

How to detach the event from the event source? Give an example

[10]

When you talk about events, the equation always has two tasks that as follows below. i. An event source pays attention to firing events and then detects the timing when an event should be fired. ii. The event handler deals with receiving the information that an event has fired and then verifies the information that the event is present. Event Source: Event source is an object that inform other objects or tasks that something has changed. The event source sends out a general message when it wants to notify other objects about a change, and•Notification in C# takes the form of callbacks. The event handling is in the form of publish–and–subscribe, which requires a generalized broadcast•then any object interested in the message can read and interpret it. The occurrence of a callback depends upon the event-firing object rather than its caller.• Events are considered to be asynchronous from the perspective of the caller. • Each event source maintains all the events published by it. •mechanism and that all objects receive all messages. Event handler: receive event notifications. When an event source notifies that an event is called, all the event handlers that have registered for the notification of that event start executing. An event handler may be either a static or a non-static method of a class. In case of a static method, event handlers simply respond to a given event source and take a course of action based on the information passed to the handler

CO2



```
using System;
namespace Class_Demos{
    public delegate string MyDel(string str);
    class EventProgram{
        event MyDel MyEvent;

        public EventProgram(){
                this.MyEvent -= new MyDel(this.Welcomeuser);
        }
        public string Welcomeuser(string username) {
            return "Welcome" + username;
        }
        static void Main(string[] args) {
            EventProgram obj1 = new EventProgram();
            string result = obj1.MyEvent("C# Class");
            Console.WriteLine(result);
            Console.ReadLine();
        }
    }
}
```

5

PART III

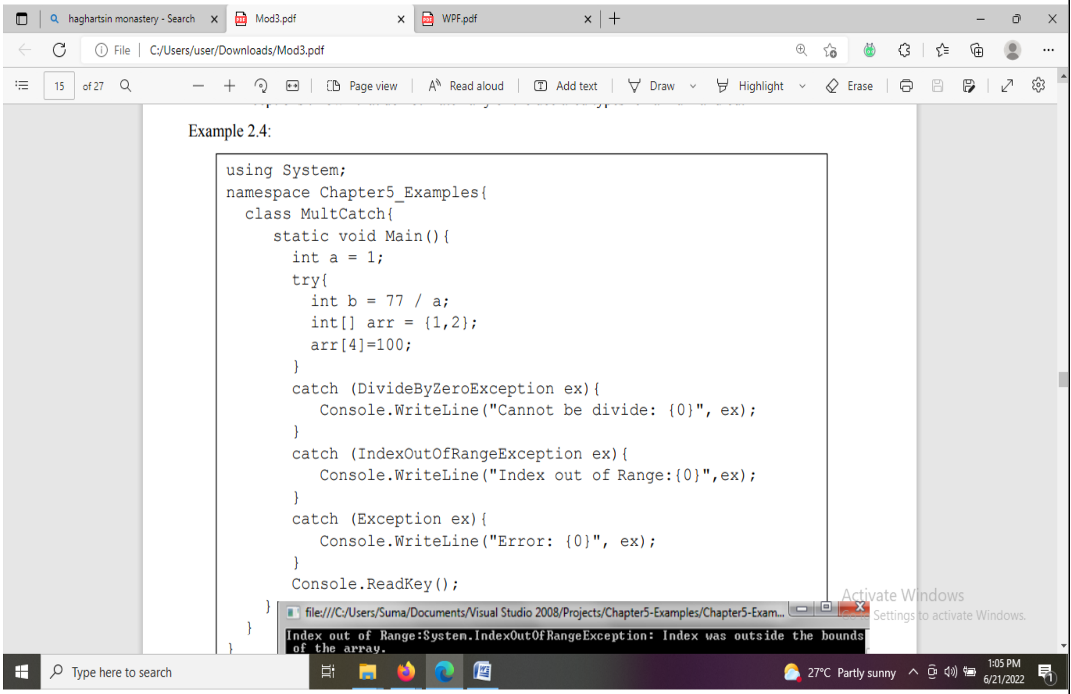
Explain exception handling with a sample program to handle multiple exceptions.

[10]

CO2

L2

More than one exception could be raise by a single piece of code. To handle this type of situation, When an exception is thrown, each catch block is checked in turn to see if the exception thrown is the•you can specify two or more catch block, each having a different type to handle various exceptions. When a match is found, the code within the catch block is executed. Only one catch block's code is ever•same type as, or derives from, that declared in the catch statement. Exceptions thrown that do not match any of the declared types remain unhandled.



Example 2.4:

```
using System;
namespace Chapter5_Examples{
  class MultCatch{
     static void Main(){
       int a = 1;
       try{
         int b = 77 / a;
         int[] arr = {1,2};
         arr[4]=100;
       }
       catch (DivideByZeroException ex){
          Console.WriteLine("Cannot be divide: {0}", ex);
       }
       catch (IndexOutOfRangeException ex){
          Console.WriteLine("Index out of Range:{0}",ex);
       }
       catch (Exception ex){
          Console.WriteLine("Error: {0}", ex);
       }
       Console.ReadKey();
     }
   }
 }
```

file:///C:/Users/Suma/Documents/Visual Studio 2008/Projects/Chapter5-Examples/Chapter5-Exam...

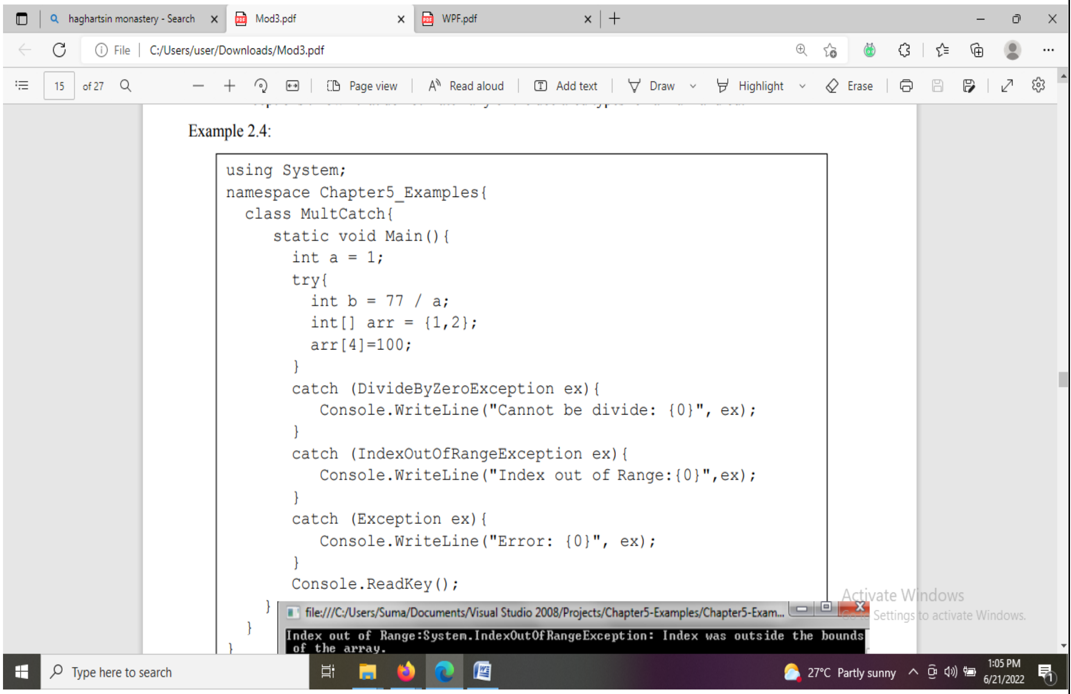
```
Index out of Range:System.IndexOutOfRangeException: Index was outside the bounds
 of the array.
```

6

Explain .NET Exception handling with valid example code

[10]

CO2 L2



Example 2.4:

```
using System;
namespace Chapter5_Examples{
  class MultCatch{
     static void Main(){
       int a = 1;
       try{
         int b = 77 / a;
         int[] arr = {1,2};
         arr[4]=100;
       }
       catch (DivideByZeroException ex){
          Console.WriteLine("Cannot be divide: {0}", ex);
       }
       catch (IndexOutOfRangeException ex){
          Console.WriteLine("Index out of Range:{0}",ex);
       }
       catch (Exception ex){
          Console.WriteLine("Error: {0}", ex);
       }
       Console.ReadKey();
     }
   }
 }
```

file:///C:/Users/Suma/Documents/Visual Studio 2008/Projects/Chapter5-Examples/Chapter5-Exam...

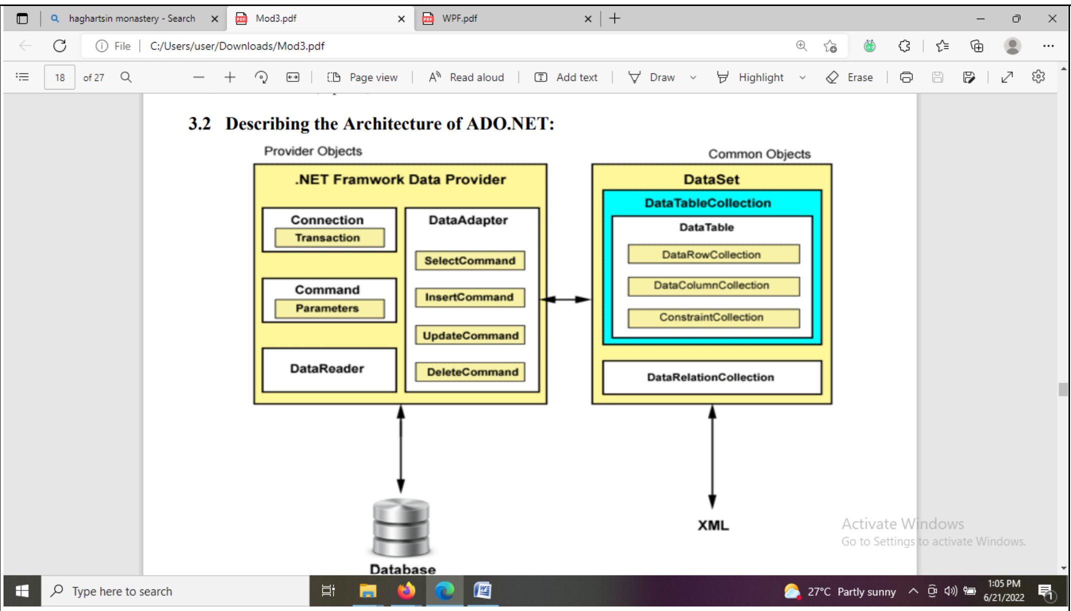
```
Index out of Range:System.IndexOutOfRangeException: Index was outside the bounds
 of the array.
```

PART IV

7 Explain the architecture of ADO.NET with a neat diagram

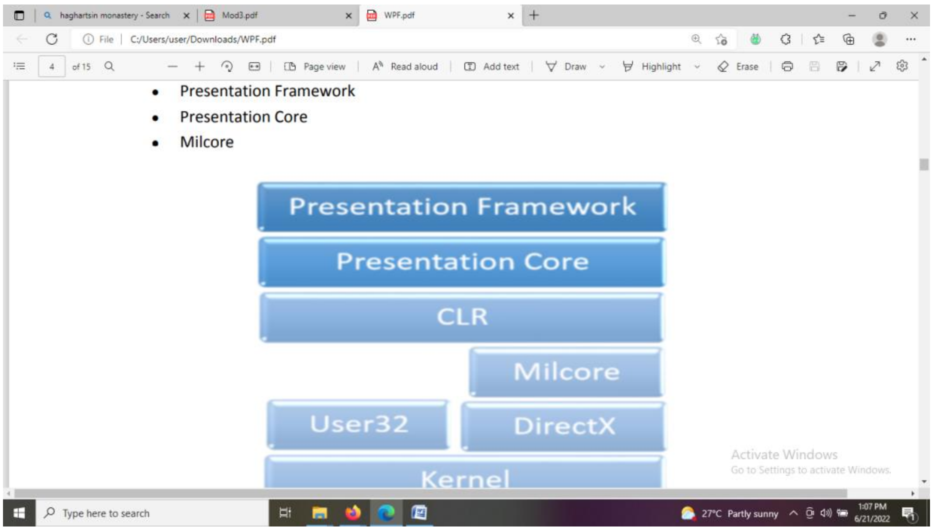
[10]

CO2 L1

<table>
<tr>
<td></td>
<td>


3.2 Describing the Architecture of ADO.NET:

</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>
Explain WPF.What are the advantages of WPF?

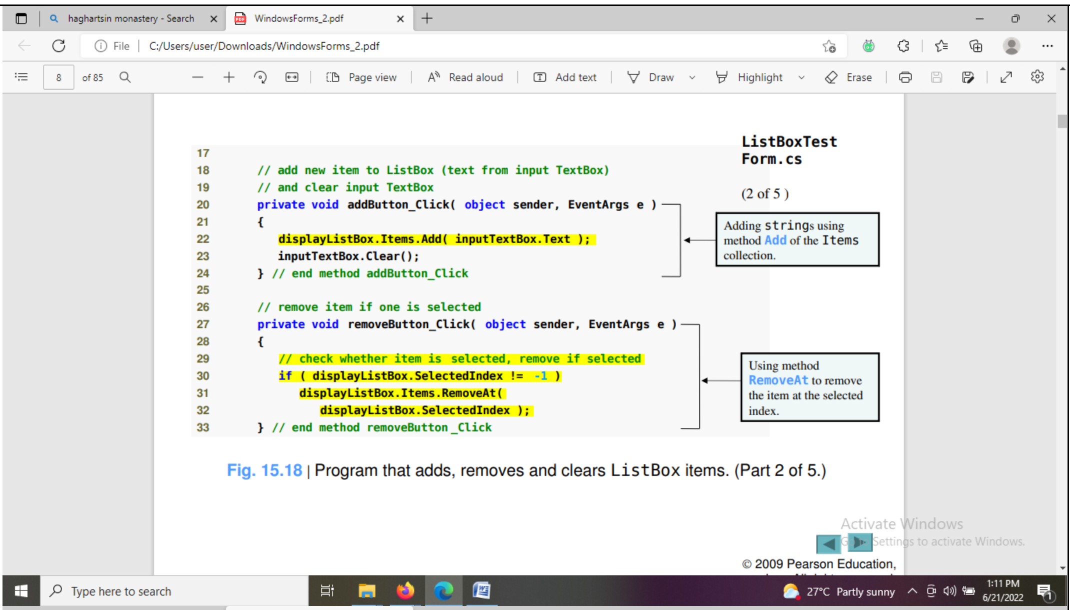


- Presentation Framework
- Presentation Core
- Milcore

In the earlier GUI frameworks, there was no real separation between how an application looks like and how it behaved. Both GUI and behavior was created in the same language, e.g. C# or VB.Net which would require more effort from the developer to implement both UI and behavior associated with it. In WPF, UI elements are designed in XAML while behaviors can be implemented in procedural languages such C# and VB.Net. So it very easy to separate behavior from the designer code. With XAML, the programmers can work in parallel with the designers. The separation between a GUI and its behavior can allow us to easily change the look of a control by using styles and templates
</td>
<td>[10]</td>
<td>CO2</td>
<td>L2</td>
</tr>
<tr>
<td>9</td>
<td>
PARTV

With the help of code explain :

a) Listbox control
</td>
<td>[5]
[5]</td>
<td>CO2</td>
<td>L3</td>
</tr>
</table>



ListBoxTest Form.cs

(2 of 5)

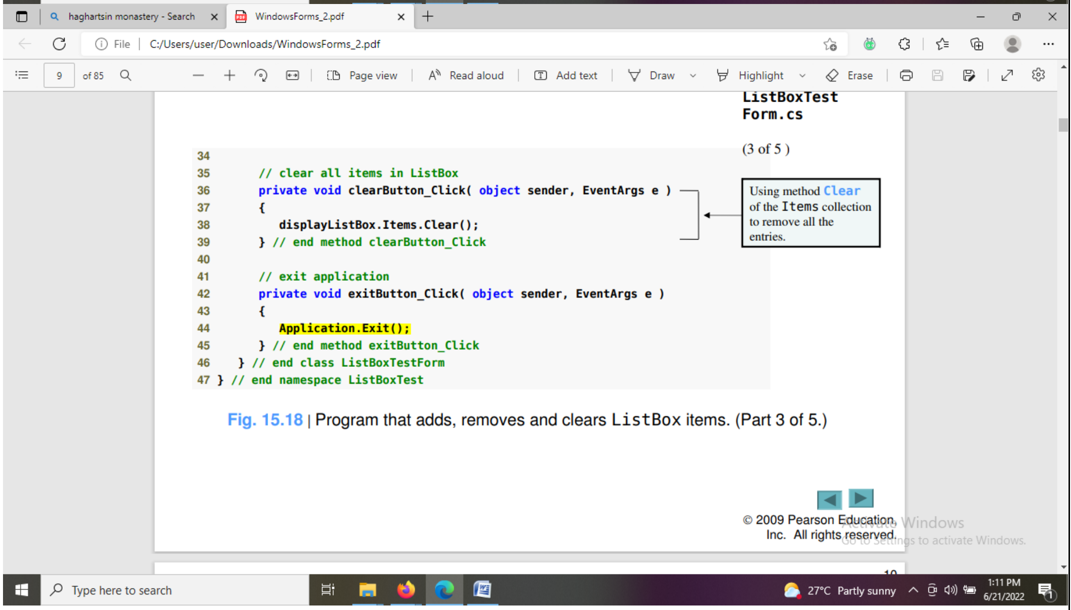
```
       // add new item to ListBox (text from input TextBox)
       // and clear input TextBox
       private void addButton_Click( object sender, EventArgs e )
       {
          displayListBox.Items.Add( inputTextBox.Text );
          inputTextBox.Clear();
       } // end method addButton_Click

       // remove item if one is selected
       private void removeButton_Click( object sender, EventArgs e )
       {
          // check whether item is selected, remove if selected
          if ( displayListBox.SelectedIndex != -1 )
             displayListBox.Items.RemoveAt(
                displayListBox.SelectedIndex );
       } // end method removeButton_Click
```

Adding strings using method Add of the Items collection.

Using method RemoveAt to remove the item at the selected index.

Fig. 15.18 | Program that adds, removes and clears ListBox items. (Part 2 of 5.)

© 2009 Pearson Education,



ListBoxTest Form.cs

(3 of 5)

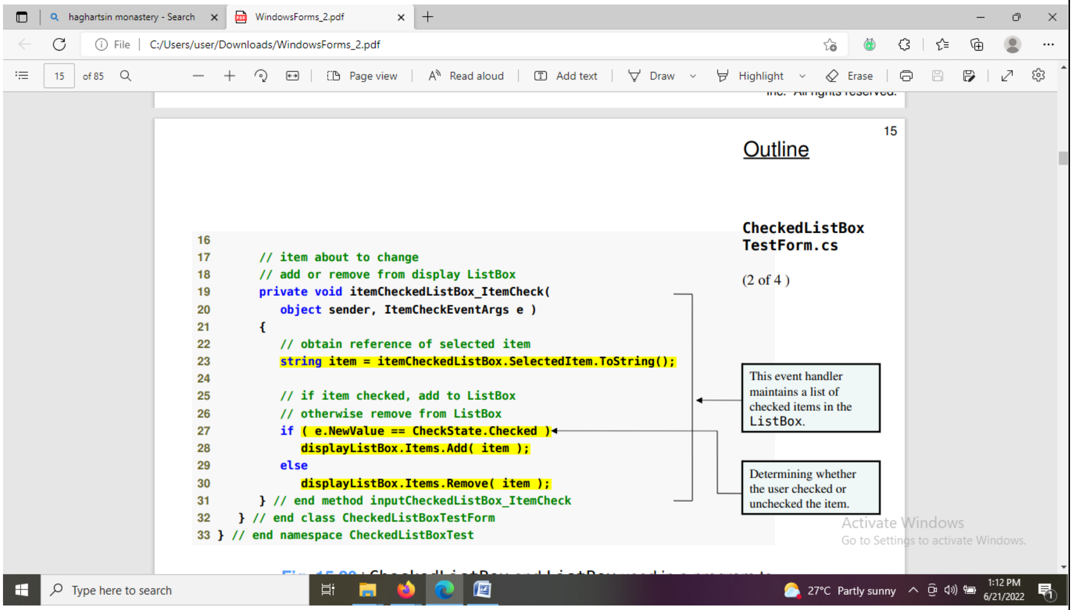
```
       // clear all items in ListBox
       private void clearButton_Click( object sender, EventArgs e )
       {
          displayListBox.Items.Clear();
       } // end method clearButton_Click

       // exit application
       private void exitButton_Click( object sender, EventArgs e )
       {
          Application.Exit();
       } // end method exitButton_Click
    } // end class ListBoxTestForm
 } // end namespace ListBoxTest
```

Using method Clear of the Items collection to remove all the entries.

Fig. 15.18 | Program that adds, removes and clears ListBox items. (Part 3 of 5.)

© 2009 Pearson Education, Inc. All rights reserved.

a) Checked Listbox control



15

Outline

CheckedListBox TestForm.cs

(2 of 4)

```
       // item about to change
       // add or remove from display ListBox
       private void itemCheckedListBox_ItemCheck(
          object sender, ItemCheckEventArgs e )
       {
          // obtain reference of selected item
          string item = itemCheckedListBox.SelectedItem.ToString();

          // if item checked, add to ListBox
          // otherwise remove from ListBox
          if ( e.NewValue == CheckState.Checked )
             displayListBox.Items.Add( item );
          else
             displayListBox.Items.Remove( item );
       } // end method inputCheckedListBox_ItemCheck
    } // end class CheckedListBoxTestForm
 } // end namespace CheckedListBoxTest
```

This event handler maintains a list of checked items in the ListBox.

Determining whether the user checked or unchecked the item.

10	Explain Keyboard and Mouse event handling.	[5] [5]	CO2	L3

Using KeyBoard Events

Every modern programming language contains all necessary functions for handling KeyBoard related events. C# also provides us with three events Keypress, KeyUp, and KeyDown, which you can use to handle Keyboard events. Listing – 3, below, shows the usage of the KeyUp Event. Copy the code given below, compile, and observe the output.

Listing – 3

```
using System;
using System.Windows.forms;
using System.Drawing;

public class Keydemo:form {

    public Keydemo() {

        this.KeyUp += new KeyEventHandler(OnKeypress);
    }

    public void OnKeypress(object sender, KeyEventArgs e)     {

        MessageBox.Show(e.KeyCode.ToString(), "Your input");

    }

    public static void Main()     {

        Application.Run(new Keydemo());
    }
}
```

Handling MouseEvents

You can handle various Mouse actions by using the events specified in the Control class. The following listing shows how to handle a simple MouseUp Event:

Listing – 2

```
using System;
using System.Windows.forms;
using System.Drawing;

public class Mousedemo:form  {

    public Mousedemo()  {

        this.MouseUp += new MouseEventHandler(OnMouseup);

    }

    public void OnMouseup(object sender,MouseEventArgs e)    {

        this.Text = "Current Position (" +e.X + " , " + e.Y +")";

    }
    public static void Main()  {

         Application.Run(new Mousedemo());

    }
}
```