برمجة متقدمة ADVANCE PROGRAMMING IN C#

المحاضرة الثانية

علوم حاسوب وتقنية المعلومات - مستوى ثالث – ترم ثاني

أ/ نجم الدين الدغار najmuddin.developer@gmail.com

المفوضات DELEGATES

*** تعريف المفوض هو نوع في لغة السي شارب يستخدم للإشارة الى دوا ل او مجموعة من الدوال -يمكن اعتباره ك إشارة لدالة- مما يسمح لك بتمرير الدوال كبارامترات الى دوال أخرى او تخزينها في متغيرات او استدعاءها لاحقا .**

* (إمكانياته) (Delegates)فوائد استخدام المفوضات

- يمكنك التاشير على داله .
- يمكنك تغيير الدالة التي يتم استدعاؤها في وقت التشغيل .
- يمكنك تمرير الدوال كمعاملات إلى دوال أخرى .
- تسمح لك بالاشتراك في أكثر من دالة باستخدام العامل =+ وإلغاء الاشتراك من دالة باستخدام العامل =- وعندما تستدعي المفوض، سيتم تنفيذ جميع الدوال المسجلة فيه.
- القدرة على بناء دوال سطريه inline code block من نوع

Delegate & Anonymous function & lambda expression

- القدر على تفسير تعابير ال

lambda expression

- يُسهّل كتابة أكواد أكثر تنظيمًا وقابلية للصيانة .

*اغلب الظروف التي تجعلني أستخدم المفوض

1-عند استخدام نمط تصميم الأحداث :

- إذا كنت ترغب في تنفيذ أحداث، مثل الضغط على زر، يمكنك استخدام المفوض لربط الأحداث بالأساليب المناسبة.

2- عند الرغبة في تغليف دالة ثابتة :

- إذا كان لديك دالة ثابتة وترغب في تمريرها كمفوض، يمكنك فعل ذلك بسهولة .

3-عندما لا يحتاج المتصل إلى الوصول إلى خصائص أو طرق أو واجهات أخرى على الكائن الذي ينفذ الدالة.

4- عندما ترغب في تكوين سهل .

- يسمح لك المفوض بإنشاء تركيبات سهلة للدوال، مما يسهل التعامل مع وظيفة متعددة.

5-عندما قد تحتاج الفئة إلى أكثر من تنفيذ واحد للدالة

- إذا كانت لديك فئة تتطلب استخدام أكثر من دالة لتنفيذ نفس الوظيفة، يمكنك استخدام المفوض لتحقيق ذلك.

ملاحظة/ هناك مثال لكل ضرف من هذه الظروف نهاية الملف.

* الشروط المطلوبة لتعريف المفوض

عند تعريف المفوض، يجب مراعاة الشروط التالية:

- يجب تحديد نوع القيمة التي ستعيدها الدالة المرتبطة بالمفوض مثل (`int`، `void`) إلخ .

- يجب تحديد المعلمات التي ستقبلها الدالة، إذا كانت هناك أي معلمات.

- لتعريف المفوض يجب استخدام كلمة.

DELEGATE

* الصيغة العامة لتعرف المفوض

[access_modifier] delegate [return_type] [delegate_name]([parameter_type1] [parameter_name1], [parameter_type2] [parameter_name2], ...);

Public – praivet – protected … <= access_modifier

مثال على تعريف المفوض:

```
public delegate int MyDelegate(int x, int y);
```

يؤشر على دالة تقبل 2 بارامترات من نوع صحيح وترجع قيمة من نوع صحيح

التطبيق العملي

1. البداية مع المفوضات من خلال بناء مفوضات بسيطة خاصة لكل دالة ومن ثم التدرج لمفاهيم المفوضات خطوة بخطوة.

```
using System;
namespace delegates {

    public class DelegateOne
    {
        // الدالة الأولى: طباعة نص ثابت
        public static void PrintMessage()
        {
            Console.WriteLine("Delegate in C#");
        }
        // الدالة الثانية: طباعة نص يتم استقباله
        public static void PrintText(string text)
        {
            Console.WriteLine(text);
        }
        // الدالة الثالثة: حساب مضروب العدد
        public static int Factorial(int number)
        {
            int result = 1;
            for (int i = 1; i <= number; i++)
            {
                result *= i;
            }
            return result;
        }
        // تعريف المفوضات
        public delegate void PrintMessageDelegate();
        public delegate void PrintTextDelegate(string text);
        public delegate int FactorialDelegate(int number);

//Delegate دالة لاستقبال بارامتر من نوع مفوض
     static   void resevDeleget(PrintTextDelegate del)
        {
            del("delegate from function");// اطلاق / تنفيذ المفوض
        }
        public static void Main()
        {
            // استخدام المفوضات
            PrintMessageDelegate messageDelegate = new
PrintMessageDelegate(PrintMessage);
            PrintTextDelegate textDelegate = PrintText;
            FactorialDelegate factorialDelegate =(Factorial);

            // استدعاء الدوال عبر المفوضات
            Console.WriteLine("Result for function");
            PrintMessage();
            Console.WriteLine("Result for delegate");
            messageDelegate();//messageDelegate.Invoke();// اطلاق / تنفيذ المفوض
            textDelegate("Hello, Delegates!"); // اطلاق / تنفيذ المفوض
            Console.WriteLine("Factorial of 5: " + factorialDelegate(5));
            //ارسال المفوض الى دالة
            resevDeleget(PrintText);
            resevDeleget(textDelegate);
            Console.ReadKey();
        }
    }}
```

2. استخدام المفوض مع الدوال السطرية (lambda و delegate) .

```
using System;
namespace delegates
{
    public class DelegateTwo
    {
        // تعريف المفوض
        public delegate void CustomDelegate(string message);
        public static void Main()
        {
            // استخدام delegate
            CustomDelegate delegateMethod = delegate (string message)
            {
                Console.WriteLine("Anonymous method executed: " + message);
            };
            delegateMethod("Hello from delegate!"); // اطلاق / تنفيذ المفوض

            //----------------------------------------

            // استخدام lambda مختصرة
            delegateMethod =  (message)=>
            {// اذا في اكثر من سطر{}
                Console.WriteLine("Anonymous method executed: " + message);
            };
            delegateMethod("Hello from delegate!"); // اطلاق / تنفيذ المفوض

            //----------------------------------------

            // اذا كان سطر برمجي واحد بدون{}
            CustomDelegate lambdaMethod = message => Console.WriteLine("Lambda
executed: " + message);
            lambdaMethod("Hello from lambda!"); // اطلاق / تنفيذ المفوض
        }
    }

}
```

3. بناء مفوض خاص يتم تمريره لدالة أخرى ومن ثم استخدامة.

```
using System;
namespace delegates
{
public class DelegateThree
    {
        // تعريف المفوض
        public delegate bool NumberConditionDelegate(int number);
        // دالة لحساب عدد الأرقام بناءً على شرط معين
        public static int CountNumbers(int[] numbers, NumberConditionDelegate
condition)
        {
            int count = 0;
            foreach (var number in numbers)
            {
                if (condition(number))
                {
                    count++;
                }
            }
            return count;
        }
        public static void Main()
        {
            // مصفوفة لاختبار الأرقام
            int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
            // استخدام المفوض لحساب الأرقام الزوجية
            NumberConditionDelegate isEven = (int n) => { return n % 2 == 0;
};//n => n % 2 == 0;
            Console.WriteLine("Even numbers: " + CountNumbers(numbers,
isEven));

            // استخدام المفوض لحساب الأرقام الفردية
           // NumberConditionDelegate isOdd = n => n % 2 != 0;
            Console.WriteLine("Odd numbers: " + CountNumbers(numbers, n => n %
2 != 0));

            // استخدام المفوض لحساب الأرقام الأولية
            Console.WriteLine("Prime numbers: " + CountNumbers(numbers, n =>
            {
                if (n <= 1) return false;
                for (int i = 2; i < n; i++)
                {
                    if (n % i == 0) return false;
                }
                return true;
            }));
            Console.ReadKey();
        }
    }

}
```

4- كيفية استخدام المفوض مع عدة دوال باستخدام =+ و=-

```
using System;
namespace delegates
{
public class DelegateFure
    {
        // الدوال المختلفة
        public static void FirstMethod() { Console.WriteLine("First Method
Executed."); }
        public static void SecondMethod() => Console.WriteLine("Second Method
Executed.");
        public static void ThirdMethod() => Console.WriteLine("Third Method
Executed.");
        // تعريف المفوض
        public delegate void MultiMethodDelegate();

        public static void Main()
        {
            // تعريف مفوض يشير إلى عدة دوال
            MultiMethodDelegate multiDelegate = FirstMethod;
            multiDelegate += SecondMethod;
            multiDelegate += ThirdMethod;

            // تنفيذ المفوض
            Console.WriteLine("Executing all methods:");
            multiDelegate();

            // إزالة الإشارة إلى إحدى الدوال
            multiDelegate -= SecondMethod;

            // تنفيذ المفوض بعد التعديل
            Console.WriteLine("Executing after removing SecondMethod:");
            multiDelegate();
            Console.WriteLine("------------multi delegate wthe return type
function--------------");
// عندما يتم تنفيذ مفوض لعدة دوال ترجع قيمة فانه يتم نفيذ جميع الدوال ولكن فقط يتم الاحتفاض باخر قيمة لاخر دالة
 MultiMethodDelegatereturn multiDelegatereturn = FourdMethod;//()=>10;
            multiDelegatereturn += FiveddMethod;//() => 20;
            Console.WriteLine("multiDelegatereturn ="+ multiDelegatereturn());

            Console.ReadKey();
        }
        public static int FourdMethod() => 10;//   public static int
FourdMethod(){return 10;}
        public static int FiveddMethod() => 20;//   public static int
FiveddMethod(){return 20;}
        public delegate int MultiMethodDelegatereturn();
    }

}
```

5. استخدام مثالًا بسيطًا على template generic لبناء دوال وكلاس.

```
using System;
namespace delegates
{// تعريف كلاس Generic
    public class ClassUsesGeneric<T1, T2>
    {
        private T1 firstValue;
        private T2 secondValue;
      public ClassUsesGeneric(T1 first,T2 second) {
            firstValue = first;
            secondValue = second;
      }
        public void SetFirst(T1 value)
        {
            firstValue = value;
        }
        public T1 GetFirst()
        {
            return firstValue;
        }
        public void SetSecond(T2 value)
        {
            secondValue = value;
        }
        public T2 GetSecond()
        {
            return secondValue;
        }
        // دالة لطباعة القيم
        public void Display()
        {
            Console.WriteLine($"First Value: {firstValue}, Second Value:
{secondValue}");
        }}
    public class DelegateFive
    {
      static  T Genericfunction<T>(T value)
        {
            return value;
        }
        public static void Main()
        {
            // إنشاء كائن من الكلاس مع النصوص
            ClassUsesGeneric<string, string> textPair = new
ClassUsesGeneric<string, string>("Hello", "Delegate");
            textPair.Display();
            Console.WriteLine("First Value: " + textPair.GetFirst());
            Console.WriteLine("Second Value: " + textPair.GetSecond());
            //-------------------------------------------------------------
            // إنشاء كائن من الكلاس مع الأرقام
            ClassUsesGeneric<int, int> numberPair = new ClassUsesGeneric<int,
int>(10,20);
            numberPair.Display();
            Console.WriteLine("First Value: " + numberPair.GetFirst());
            Console.WriteLine("Second Value: " + numberPair.GetSecond());
            //-------------------------------------------------------------
            Console.WriteLine($"use Generic function (int)
{Genericfunction<int>(100)}");
            Console.WriteLine($"use Generic function (string)
{Genericfunction<string>("Advance Programming")}");
            Console.ReadKey();}}}
```

6. إعادة بناء مثال الكلاس الأول باستخدام مفوضات Generic من بنائنا، ثم مقارنة ذلك مع Func, Action, وPredicate المبنية داخل اللغة.

```
using System;
namespace delegates
{
    public class DelegateSix
    {
        // تعريف مفوض Generic
        public delegate void myAction<T>(T param);
        public delegate TResult myFunc<T, TResult>(T param);
        public delegate bool mypredicat<T>(T param);

        // دالة للطباعة
        public static void Print(string message) => Console.WriteLine(message);

        // دالة لحساب مضروب العدد
        public static int Factorial(int number)
        {
            int result = 1;
            for (int i = 1; i <= number; i++)
            {
                result *= i;
            }
            return result;
        }
        public static void Main()
        {
            // استخدام المفوضات التي قمنا ببنائها
            myAction<string> myPrint = Print;
            myPrint("Using my action delegate");

            myFunc<int, int> myFactorial = Factorial;
            Console.WriteLine("Factorial of 5 using my func: " +
myFactorial(5));

            mypredicat<string> mylength = s => s.Length > 10;
            Console.WriteLine("Using my Predicate delegate:  " +
mylength("Advance Programming"));
            // مقارنة مع المفوضات الجاهزة
            Action<string> actionDelegate = Print;
            actionDelegate("Using Action delegate");

            Func<int, int> funcDelegate = Factorial;
            Console.WriteLine("Factorial of 5 using Func: " + funcDelegate(5));

             Predicate<string> predDelegate= s => s.Length > 10;
             Console.WriteLine("Using  Predicate delegate:  " +
predDelegate("Advance Programming"));
             Console.ReadKey();

            /*
             //ضروري على الاقل الارجاع _يدعم من 0 الى 16 مدخل
               //func مع مدخلات
            Func<int, int, int> add = (a, b) => a + b;
              Console.WriteLine("Sum: " + add(5, 10));
            //func   بدون مدخلات
             Func<int> getNumber = () => 42;
              Console.WriteLine(getNumber());

             */
```

```
            //----------------------------------
            /*
             //Action يدعم من 0 الى 16 مدخل_ لا يرجع قيمة
             //Action مع مدخلات
                Action<string> printMessage = message =>
Console.WriteLine(message);
                printMessage("Hello, Action!");
            //   //Action   بدون مدخلات
                Action printHello = () => Console.WriteLine("Hello, World!");

             */
            //----------------------------------
            /*
            //Predicate يدعم مدخل واحد فقط يرجع قيمة منطقية ترو او فولس
            Predicate<int> isEven = num => num % 2 == 0;
                    Console.WriteLine(isEven(4));
             */}}}
```

7. جمع أكثر من مفهم من المفاهيم السابقة

```
using System;
namespace delegates
{
 public class Operation<T>
    {

        public T PerformOperation(T a, T b, Func<T, T, T> operation)
        {
            return operation(a, b);
        }
    }

    public class DelegateTen
    {
        public static void Main()
        {
            // استخدام الكلاس مع نوع int
            var intOperation = new Operation<int>();
            Console.WriteLine("Addition: " + intOperation.PerformOperation(5,
10, (x, y) => x + y));

            // استخدام الكلاس مع نوع string
            var stringOperation = new Operation<string>();
            Console.WriteLine("Concatenation: " +
stringOperation.PerformOperation("Hello", " World", (x, y) => x + y));
        }
    }

}
```

8. توضيح كيفية استخدام المفوضات كخاصية (Delegate as a Property) وتطبيقها مع الأحداث(Events) .

```
using System;
using System.Windows.Forms;
namespace delegates
{
    public delegate void clickdelegate(object sender, EventArgs e);
   public class ClickButtonDelegate
    {
        public clickdelegate Click;
        void OnClick()
        {

           // Click?(null, null);error
            if (Click != null)//اذا لم يكون فاضي نفذ الحدث
            {
                Click?.Invoke(null, null);
              //  Click(null, null);
            }
            //يمنك الاستغناء عن السابق بالسطر التالي
            // Click?.Invoke(null, null);//?اذا لم يكون فاضي نفذ الحدث
        }
        public static void Main()
        {
            ClickButtonDelegate button = new ClickButtonDelegate() ;
            button.Click += (s, e) =>
            {
                MessageBox.Show("button clicked");
            };

            button.OnClick();
            //----------------
            TextBox t = new TextBox();
         //   button.Click -=
           button.Click += t.textchange;
            button.OnClick();
            //----------------------------
            Write cout = Console.WriteLine;
            cout("simple ");
            Read cin = Console.ReadLine;
            cout(cin());
            Console.ReadKey();
        }

    }


    class TextBox
    {
        public void textchange(object sender, EventArgs e)
        {
            MessageBox.Show("inserting... in TextBox ");
        }
    }
    delegate void Write(string s);//
    delegate string Read();
}
```

- **نمذج بسيط لتوظيف بعض من المفاهيم السابقة للمفوض**

الكود

```
using System;
using System.Windows.Forms;
using System.Drawing;
namespace Najmuddin_lct2
{
    public delegate long factdel(int x);
    delegate void del2(string s);
    delegate bool check(int x);
    public class Program : Form
    {
        TextBox t1;
        Button[] arrb;
        del2 msg = (s) => MessageBox.Show(s.ToString());
        Program()
        {//form
           this. Text = "DELEGATE";
            this.Size = new Size(450, 250);this.ResumeLayout(false);
            //text
            t1 = new TextBox();
            t1.Bounds =new Rectangle(100,50,100,40); Controls.Add(t1);
            t1.KeyPress += (s, e) => { if
((e.KeyChar<48||e.KeyChar>57)&&(e.KeyChar!=8)&&(e.KeyChar!='-')){ e.Handled =
true; }};
            string[] str = { "فردي" ,"زوجي" ,"تربيع" ,"القيمة المطلقة" ,"المضروب" };
            //button
            arrb = new Button[5];
            for (int i=0; i < arrb.Length; i++)
            {
                arrb[i] = new Button();
                arrb[i].Bounds = new Rectangle((80 * i) + 10, 100, 70, 40);
                arrb[i].Text = str[i];arrb[i].Click += call;
                Controls.Add(arrb[i]);
            }
        }
        public long fact(int n) { return (n > 1) ? n * fact(n - 1) : 1; }
        public long abs(int n) { return (n >=0) ? n : n*-1; }
        //button click
        public void call(object s, EventArgs e)
```

```
        {
            Button b = (Button)s;
            check checkv;
            factdel fd;
            if (t1.Text.Trim() != "")
            {
                if (b.Text == "المضروب")
                {
                    fd = new factdel(fact);
                    msg(fd(int.Parse(t1.Text)).ToString());
                }
                else if (b.Text == "القيمة المطلقة")
                {
                    fd = new factdel(abs);
                    msg(fd(int.Parse(t1.Text)).ToString());
                }
                else if (b.Text== "تربيع")
                {
                    Func<int, int> f = ((int a) => a * a);
                     msg(f(int.Parse(t1.Text)).ToString());
                }
                else if ((b.Text == "زوجي"))
                {
                     checkv= (x) => x % 2 ==0;
                     msg(checkv(Convert.ToInt32(t1.Text)).ToString());
              //      msg(count(new int[] { 1, 2, 3, 4, 5, 6 }, (x) => x % 2 ==
              //0).ToString());
                }
                else if ((b.Text == "فردي"))
                {
                    checkv = (x) => x % 2 != 0;
                    msg(checkv(Convert.ToInt32(t1.Text)).ToString());
                   // msg(count(new int[] { 1, 2, 3, 4, 5, 6 }, (x) => x % 2 !=
                   //0).ToString());
                }
            }
        }
        static int count(int[] l, check ch)
        {
            int c = 0;
            foreach (int x in l)
                if (ch(x))
                    c++;
            return c;
        }
        static void Main()
        {
            Application.Run(new Program());
        }
    }
}
```

- **الأمثلة البرمجية التي توضح استخدام المفوضات في ظروف مختلفة**

1. استخدام نمط تصميم الأحداث(Eventing Design Pattern)

```
using System;

public delegate void Notify(); // تعريف المفوض

public class Publisher
{
    public event Notify OnNotify; // تعريف الحدث باستخدام المفوض

    public void TriggerEvent()
    {
        Console.WriteLine("Triggering event...");
        OnNotify?.Invoke(); // استدعاء الحدث
    }
}

public class Subscriber
{
    public void Respond()
    {
        Console.WriteLine("Event received!");
    }
}

// الاستخدام
var publisher = new Publisher();
var subscriber = new Subscriber();
publisher.OnNotify += subscriber.Respond; // الاشتراك في الحدث
                                                    publisher.TriggerEvent();
```

2. تغليف الدوال الثابتة encapsulate a static method

```
using System;

public delegate void StaticDelegate(); // تعريف المفوض

public class MyClass
{
    public static void StaticMethod()
    {
        Console.WriteLine("Static method called!");
    }
}

// الاستخدام
StaticDelegate del = new StaticDelegate(MyClass.StaticMethod);
                                                استدعاء الدالة الثابتةdel.Invoke(); //
```

3. عدم الحاجة للوصول إلى خصائص أو دوال أخرى

```
using System;

public delegate void SimpleDelegate(); // تعريف المفوض

public class MyClass
{
    public void SimpleMethod()
    {
        Console.WriteLine("Simple method called!");
    }
}

// الاستخدام
public class Program
{
    public static void Execute(SimpleDelegate del)
    {
        del.Invoke(); // استدعاء المفوض
    }

    public static void Main()
    {
        MyClass myClass = new MyClass();
        Execute(myClass.SimpleMethod); // تمرير الدالة
    }
}
```

4. سهولة التركيب(Easy Composition)

```
using System;

public delegate int MathOperation(int a, int b); // تعريف المفوض

public class Math
{
    public int Add(int a, int b) => a + b;
    public int Multiply(int a, int b) => a * b;
}

// الاستخدام
public class Program
{
    public static void ExecuteOperation(MathOperation operation, int x, int y)
    {
        Console.WriteLine($"Result: {operation(x, y)}");
    }

    public static void Main()
    {
        Math math = new Math();
        ExecuteOperation(math.Add, 5, 3); // استخدام عملية الجمع
        ExecuteOperation(math.Multiply, 5, 3); // استخدام عملية الضرب
    }
}
```

5. الحاجة إلى أكثر من تنفيذ لنفس الدالة

```
using System;

public delegate void MessageDelegate(string message); // تعريف المفوض

public class Messenger
{
    public void SendEmail(string message)
    {
        Console.WriteLine($"Email sent: {message}");
    }

    public void SendSMS(string message)
    {
        Console.WriteLine($"SMS sent: {message}");
    }
}

// الاستخدام
public class Program
{
    public static void Notify(MessageDelegate messageDelegate, string message)
    {
        messageDelegate(message); // استدعاء المفوض
    }

    public static void Main()
    {
        Messenger messenger = new Messenger();
        Notify(messenger.SendEmail, "Hello via Email!"); //  إرسال رسالة عبر البريد
الإلكتروني
        Notify(messenger.SendSMS, "Hello via SMS!"); // إرسال رسالة عبر الرسائل القصيرة
    }
                                                                                    }
```