



# LẬP TRÌNH C# 1

## BÀI 7: DELEGATE, EXCEPTIONS

- ⦿ Delegate và event
- ⦿ Xử lý ngoại lệ



## Phần I: Delegate và event

 Delegate Types

 Single Cast Delegates

 Multicast Delegates

 Event

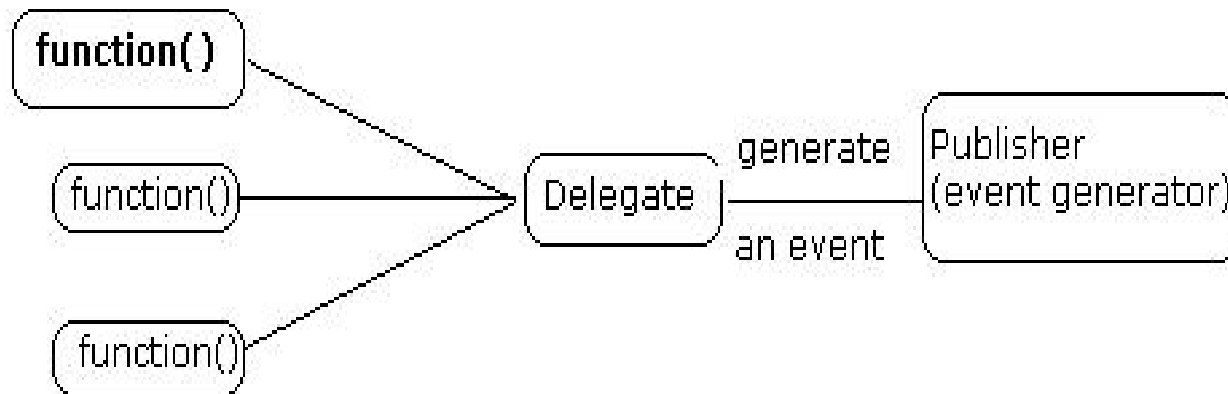
## Phần II: Exception

 Try – catch - finally

 Exception User defined



- ❑ Delegate là một đối tượng chứa tham chiếu đến phương thức cần thực thi.
- ❑ Một delegate có thể trỏ tới một hoặc nhiều phương thức
- ❑ Delegate có thể gọi bất kỳ phương thức nào nó trỏ tới tại thời điểm thực thi.



❑ Để liên kết một delegate với một phương thức cụ thể thì phương thức và delegate phải giống nhau ở kiểu trả về và kiểu tham số

❑ Cú pháp:

```
<access_modifier> delegate <return_type> <delegate_name>(<parameters>)
```

❑ Ví dụ:

```
public class BasicMaths
{
    public static double Add(double value1, double value2)
    {
        return value1 + value2;
    }

    public static double Sustract(double value1, double value2)
    {
        return value1 - value2;
    }
}

public delegate double MathDelegate(double value1, double value2);
```

Same return type

Same method signature  
i.e. Same no of  
parameters and thier  
types

## □ Sử dụng delegate

- ❖ Khai báo delegate
- ❖ Thực hiện delegate tham chiếu tới phương thức
- ❖ Tạo thể hiện của delegate
- ❖ Gọi phương thức thông qua thể hiện delegate

```
// Declare Delegate
public delegate void SampleDelegate(int a, int b);
class MathOperations
{
    public void Add(int a, int b)
    {
        Console.WriteLine("Add Result: {0}", a + b);
    }
    public void Subtract(int x, int y)
    {
        Console.WriteLine("Subtract Result: {0}", x - y);
    }
}

static void Main(string[] args)
{
    Console.WriteLine("****Delegate Example****");
    MathOperations m = new MathOperations();
    // Instantiate delegate with add method
    SampleDelegate dlgt = m.Add;
    dlgt(10, 90);
    // Instantiate delegate with subtract method
    dlgt = m.Subtract;
    dlgt(10, 90);
    Console.ReadLine();
}
```

```
****Delegate Example****
Add Result: 100
Subtract Result: -80
```



# DEMO

Hiện thực hóa ví dụ của slide trước





- ❑ Single Cast Delegates: Một delegate chỉ tham chiếu đến một phương thức tại một thời điểm

```
SampleDelegate dlgt = m.Add; // thời điểm t1  
dlgt(10, 90);  
  
// Instantiate delegate with subtract method  
  
dlgt = m.Subtract;  
  
dlgt(10, 90); // thời điểm t2
```

- ❑ C# Multicast Delegates:

- ❖ Có thể tham chiếu đến nhiều phương thức tại cùng một thời điểm
- ❖ Kiểu trả về của multicast delegate phải là kiểu void
- ❖ Dùng toán tử "+" để thêm phương thức vào delegate

## ❑ C# Multicast Delegates:

```
// Declare Delegate
public delegate void SampleDelegate(int a, int b);
class MathOperations
{
    public void Add(int a, int b)
    {
        Console.WriteLine("Add Result: {0}", a + b);
    }
    public void Subtract(int x, int y)
    {
        Console.WriteLine("Subtract Result: {0}", x - y);
    }
    public void Multiply(int x, int y)
    {
        Console.WriteLine("Multiply Result: {0}", x * y);
    }
}
```

```
static void Main(string[] args)
{
    Console.WriteLine("****Delegate Example****");
    MathOperations m = new MathOperations();
    // Instantiate delegate with add method
    SampleDelegate dlgt = m.Add;
    dlgt += m.Subtract;
    dlgt += m.Multiply;
    dlgt(10, 90);
    Console.ReadLine();
}
```

```
****Delegate Example****
Add Result: 100
Subtract Result: -80
Multiply Result: 900
```

- ❑ Truyền delegate (tham chiếu đến phương thức) vào phương thức, delegate đóng vai trò tham số.

```
class MathOperations
{
    public void Add(int a, int b)
    {
        Console.WriteLine("Add Result: {0}", a + b);
    }
    public void Subtract(int x, int y)
    {
        Console.WriteLine("Subtract Result: {0}", x - y);
    }
    public void Multiply(int x, int y)
    {
        Console.WriteLine("Multiply Result: {0}", x * y);
    }
}

static void Main(string[] args)
{
    Console.WriteLine("****Delegate Example****");
    MathOperations m = new MathOperations();
    SampleMethod(m.Add, 10, 90);
    SampleMethod(m.Subtract, 10, 90);
    SampleMethod(m.Multiply, 10, 90);
    Console.ReadLine();
}

static void SampleMethod(SampleDelegate dlgt, int a, int b)
{
    dlgt(a, b);
}
```

```
****Delegate Example****
Add Result: 100
Subtract Result: -80
Multiply Result: 900
```



# DEMO

Hiện thực hóa ví dụ của slide trước



- ❑ Sự kiện (Event) là các hành động, ví dụ như nhấn phím, click, di chuyển chuột...
- ❑ Trong C#, Event là một đối tượng đặc biệt của Delegate, nó là nơi chứa các phương thức, các phương thức này sẽ được thực thi khi sự kiện xảy ra
- ❑ Đặc điểm của event:
  - ❖ Được khai báo trong các lớp hoặc interface
  - ❖ Được khai báo là abstract hoặc sealed, virtual
  - ❖ Được thực thi thông qua delegate

## □ Tạo và sử dụng event

- ❖ Định nghĩa delegate cho event
- ❖ Tạo event thông qua delegate
- ❖ Đăng ký để lắng nghe và xử lý event
- ❖ Kích hoạt event



# Events

## □ Định nghĩa delegate và event

Cú pháp:

```
Khai báo delegate:  
Bổ_từ_truy_cập delegate Kiểu_trả_về Tên_delegate (Danh_sách_tham_số);  
Khai báo sự kiện:  
Bổ_từ_truy_cập event Tên_delegate Tên_sự_kiện;
```

Ví dụ:

```
using System;  
  
public delegate void PrintDetails(); //khai báo delegate trước  
  
class TestEvent  
{  
    event PrintDetails Print; //rồi khai báo event  
}
```

## □ Định nghĩa delegate và event

Cú pháp:

```
Khai báo delegate:  
Bổ_từ_truy_cập delegate Kiểu_trả_về Tên_delegate (Danh_sách_tham_số);  
Khai báo sự kiện:  
Bổ_từ_truy_cập event Tên_delegate Tên_sự_kiện;
```

Ví dụ:

```
using System;  
  
public delegate void PrintDetails(); //khai báo delegate trước  
  
class TestEvent  
{  
    event PrintDetails Print; //rồi khai báo event  
}
```



## Kích hoạt event

Cú pháp:

```
Tên_đối_tượng.Tên_sự_kiện(Danh_sách_tham_số);
```

Ví dụ:

```
using System;
public delegate void PrintDetails();
class TestEvent
{
    event PrintDetails Print;
    void Show()
    {
        Console.WriteLine("Hay hien thi toi ra man hinh.");
    }
    static void Main(string[] args)
    {
        TestEvent objTestEvent = new TestEvent();
        objTestEvent.Print += new PrintDetails(objTestEvent.Show);
        objTestEvent.Print(); //câu lệnh kích hoạt sự kiện
    }
}
```



# DEMO

Hiện thực hóa ví dụ của các slide trước





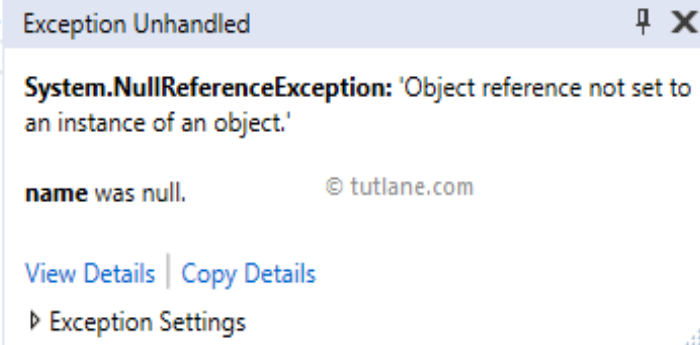
# LẬP TRÌNH C# 1

## BÀI 7: DELEGATE VÀ EXCEPTION (P2)

- ❑ Exception là các vấn đề phát sinh trong quá trình thực hiện chương trình như: không đọc được tập tin, kiểu dữ liệu sai...
- ❑ Các exception được sinh ra bởi .NET framework CLR hoặc lập trình viên

```
static void Main(string[] args)
{
    string name = null;
    // Null Reference Exception Error
    if (name.Length > 0)
    {
        Console.WriteLine("Name: "+name);
    }
}
```

```
static void Main(string[] args)
{
    string name = null;
    if (name.Length > 0)
    {
        Console.WriteLine
    }
}
```



- ❑ Xử lý ngoại lệ trong C# được xây dựng chủ yếu trên bốn từ khoá try, catch, finally và throw.

```
try
{
    // code that may cause exception
}
catch (Exception ex)
{
    // exception handling
}
finally
{
    // cleanup resources
}
```

## ❑ Một số Exception class thường gặp

Exception class	Mô tả
IOException	Xử lý các lỗi nhập xuất file
IndexOutOfRangeException	Xử lý các lỗi phát sinh khi truy cập vượt chỉ số mảng
NullReferenceException	Xử lý các lỗi phát sinh khi sử dụng một object chưa được tạo
InvalidCastException	Xử lý lỗi phát sinh khi chuyển đổi sai kiểu dữ liệu
Exception	Ngoại lệ tổng quát

□ Ví dụ:

```
static void Main(string[] args)
{
    string fpath = @"D:\Test.txt";
    StreamReader sr = new StreamReader(fpath);
    try
    {
        string txt;
        while ((txt = sr.ReadLine()) != null)
        {
            Console.WriteLine(txt);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Exception: {0}", ex.Message);
    }
    finally
    {
        if (sr != null)
        {
            sr.Close();
        }
    }
    Console.ReadLine();
}
```

- ❑ Lập trình viên có thể tự định nghĩa một ngoại lệ cho chính mình thay vì sử dụng những ngoại lệ có sẵn bằng cách kế thừa lớp Exception

```
-----  
class MyException : Exception  
{  
    0 references  
    public MyException(string msg) : base(msg)  
    {  
    }  
}
```



## ❑ Sử dụng exception người dùng định nghĩa

```
class Scores
{
    2 references
    public static void showScore(float avg)
    {
        if (avg < 0)
        {
            throw (new MyException("Average score must be greater than 0"));
        }
        else
        {
            Console.WriteLine("Average score: {0}", avg);
        }
    }
    0 references
    static void Main(string[] args)
    {
        try
        {
            // Ok
            showScore(7.8F);
            // Exception
            showScore(-10.0F);
        }
        catch (MyException me)
        {
            Console.WriteLine("{0}", me.Message);
        }
        Console.ReadLine();
    }
}
```



# DEMO

Hiện thực hóa các ví dụ trong slide



# Tổng kết bài học

## Phần I: Delegate và event

 Delegate Types


 Single Cast Delegates

 Multicast Delegates

 Event

## Phần II: Exception

 Try – catch - finally

 Exception User defined





**KẾT THÚC**