Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Практична робота №1

за курсом: «Робота з даними на платформи .Nеt»

за темою: «Створення делегатів і подій у власних класах»

|  |  |
| --- | --- |
|  | Виконав:  студент гр. ПЗПІ-16-3  Сачек О.А. |
|  | Перевірила:  ст.викл. каф. ПІ  Широкопетлєва М.С. |

Харків 2019 р.

Практична робота №1. Створення делегатів і подій у власних класах

Мета роботи: Ознайомитися з поняттями делегатів, набути практичні навички створення та роботи з делегатами

Постановка задачі:

У власній збірці створити делегат. Продемонструвати коваріатівность та / або контрваріатівность посилальних типів при прив'язці методу до делегату. Продемонструвати використання делегатів для виклику статичних і екземплярні методів. Розібрати класи з використанням утиліти ILDasm.exe. Продемонструвати тотожність виклику делегатів з використанням методу Invoke() та без нього.

Хід роботи.

Було створено додаток та продемостровано роботу с коваріантними на контрваріантними делегатами. Код програми:

using System;

namespace PZ1

{

class Program

{

delegate T CovariantDelegate<out T>();

delegate void ContravariantDelegate<in T>(T value);

static void Main(string[] args)

{

var del1 = new CovariantDelegate<MyParentType>(MyStaticMethodWithReturnParentType);

var del2 = new CovariantDelegate<MyParentType>(MyStaticMethodWithReturnChildType);

// Error

//var del3 = new CovariantDelegate<MyChildType>(MyStaticMethodWithReturnParentType);

var del4 = new CovariantDelegate<MyChildType>(MyStaticMethodWithReturnChildType);

var del5 = new ContravariantDelegate<MyParentType>(MyStaticMethodWithInParentType);

// Error

//var del6 = new ContravariantDelegate<MyParentType>(MyStaticMethodWithInChildType);

var del7 = new ContravariantDelegate<MyChildType>(MyStaticMethodWithInParentType);

var del8 = new ContravariantDelegate<MyChildType>(MyStaticMethodWithInChildType);

del1();

del2.Invoke();

del4();

del5(new MyParentType());

del7(new MyChildType());

del8(new MyChildType());

var myType = new MyType();

var del9= new CovariantDelegate<MyParentType>(myType.MyMethodWithReturnParentType);

var del10 = new CovariantDelegate<MyParentType>(myType.MyMethodWithReturnChildType);

del9();

del10();

Console.ReadLine();

}

static MyParentType MyStaticMethodWithReturnParentType()

{

Console.WriteLine("MyStaticMethodWithReturnParentType called.");

return new MyParentType();

}

static MyChildType MyStaticMethodWithReturnChildType()

{

Console.WriteLine("MyStaticMethodWithReturnChildType called.");

return new MyChildType();

}

static void MyStaticMethodWithInParentType(MyParentType parentType)

{

Console.WriteLine("MyStaticMethodWithInParentType called.");

}

static void MyStaticMethodWithInChildType(MyChildType childType)

{

Console.WriteLine("MyStaticMethodWithInChildType called.");

}

static MyChildType MyStaticMethodWithChildType(MyChildType childType)

{

Console.WriteLine($"MyStaticMethodWithChildType called with {childType.GetType().Name} argument type.");

return childType;

}

}

public class MyParentType { }

public class MyChildType : MyParentType { }

public class MyType

{

public MyParentType MyMethodWithReturnParentType()

{

Console.WriteLine("MyMethodWithReturnParentType called.");

return new MyParentType();

}

public MyChildType MyMethodWithReturnChildType()

{

Console.WriteLine("MyMethodWithReturnChildType called.");

return new MyChildType();

}

}

}

Висновки: на практичній роботі було розроблено додаток для демонстрації роботи с коваріантними та контрваріантними делегатами, роботу з статичними методами та методами класів, використано утілиту ILDasm.exe для перегляду IL коду, згенерованого для додатку.