#### Обобщенные типы

class Bank<T>

{

    T[] clients;

    public Bank()

    {

    }

    public Bank(T[] \_clients)

    {

        this.clients = \_clients;

    }

}

Bank<int> bank = new Bank<int>(new int[] { 1, 2, 4, 5, 6 });

Bank<string> bank2 = new Bank<string>(new string[] {"13433", "342454", "21432"});

#### Делегаты

#### Именованные методы

class Program

{

delegate void Del(int x);

static void DoWork(int k) {

Console.WriteLine(k.ToString());

}

static void Main(string[] args)

{

Del d = DoWork;

d(1);

}

}

#### Анонимные методы

void StartThread()

{

System.Threading.Thread t1 = new System.Threading.Thread

(delegate()

{

System.Console.Write("Hello, ");

System.Console.WriteLine("World!");

});

t1.Start();

}

#### Применение методов экземпляра в качестве делегатов

using System;

delegate string StrMod(string str);

class StringOps {

public string ReplaceSpaces (string s) {

Console.WriteLine("Замена пробелов дефисами.");

return s.Replace(' ', '-');

}

public string RemoveSpaces(string s) {

string temp = "";

int i;

Console.WriteLine("Удаление пробелов.");

for(i=0; i < s.Length; i++)

if(s[i] != ' ') temp += s[i];

return temp;

}

public string Reverse(string s) {

string temp = "";

int i, j;

Console.WriteLine("Обращение строки.");

for(j=0, i=s.Length-1; i >= 0; i--, j++)

temp += s[i];

return temp;

}

}

class DelegateTest {

static void Main() {

StringOps so = new StringOps();

// объекта класса StringOps

// Инициализировать делегат.

StrMod strOp = so.ReplaceSpaces;

string str;

str = strOp("Это простой тест.");

Console.WriteLine("Результирующая строка: " + str);

Console.WriteLine();

strOp = so.RemoveSpaces;

str = strOp("Это простой тест.");

Console.WriteLine("Результирующая строка: " + str);

Console.WriteLine();

strOp = so.Reverse;

str = strOp("Это простой тест.");

Console.WriteLine("Результирующая строка: " + str);

}

}

#### Ковариантность и контравариантность

using System;

class X {

public int Val;

}

class Y : X { }

delegate X ChangeIt(Y obj);

class CoContraVariance {

static X IncrA(X obj) {

X temp = new X();

temp.Val = obj.Val + 1;

return temp;

}

static Y IncrB(Y obj) {

Y temp = new Y();

temp.Val = obj.Val + 1;

return temp;

}

static void Main() {

Y Yob = new Y();

// В данном случае параметром метода IncrA является объект класса X,

// а параметром делегата ChangeIt — объект класса Y. Но благодаря

// контравариантности следующая строка кода вполне допустима.

ChangeIt change = IncrA;

X Xob = change(Yob);

Console.WriteLine("Xob: " + Xob.Val);

// В этом случае возвращаемым типом метода IncrB служит объект класса Y,

// а возвращаемым типом делегата ChangeIt — объект класса X. Но благодаря

// ковариантности следующая строка кода оказывается вполне допустимой.

change = IncrB;

Yob = (Y) change (Yob);

Console.WriteLine("Yob: " + Yob.Val);

}

}

#### Возврат значения из анонимного метода

using System;

delegate int CountIt(int end);

class AnonMethDemo3 {

static void Main() {

int result;

CountIt count = delegate (int end) {

int sum = 0;

for(int i=0; i <= end; i++) {

Console.WriteLine (i);

sum += i;

}

return sum;

};

result = count(3);

Console.WriteLine("Сумма 3 равна " + result);

Console.WriteLine();

result = count (5);

Console.WriteLine("Сумма 5 равна " + result);

}

}

#### Одиночные лямбда-выражения

delegate int Incr(int v);

delegate bool IsEven(int v);

class SimpleLambdaDemo {

static void Main() {

Incr incr = count => count + 2;

Console.WriteLine("Использование лямбда-выражения incr: ");

int x = -10;

while(x <= 0) {

Console.Write(x + " ");

x = incr(x);

}

Console.WriteLine ("\n");

IsEven isEven = n => n % 2 == 0;

Console.WriteLine("Использование лямбда-выражения isEven: ");

for(int i=1; i <= 10; i++)

if(isEven(i)) Console.WriteLine(i + " четное.");

}

}

#### Блочные лямбда-выражения

using System;

delegate int IntOp(int end);

class StatementLambdaDemo {

static void Main() {

IntOp fact = n => {

int r = 1;

for(int i=1; i <= n; i++)

r = i \* r;

return r;

};

Console.WriteLine("Факториал 3 равен " + fact(3));

Console.WriteLine("Факториал 5 равен " + fact(5));

}

}

#### События

delegate void MyEventHandler();

class MyEventClass

{

public event MyEventHandler SomeEvent;

public void OnSomeEvent()

{

if (SomeEvent != null)

SomeEvent();

}

}

class EventDemo

{

static void Handler()

{

Console.WriteLine("Произошло событие");

}

static void Main()

{

MyEventClass evt = new MyEventClass();

evt.SomeEvent += Handler;

evt.OnSomeEvent();

}

}

#### Пример групповой адресации события

delegate void MyEventHandler();

class MyEventClass {

public event MyEventHandler SomeEvent;

public void OnSomeEvent() {

if(SomeEvent != null)

SomeEvent();

}

}

class X {

public void Xhandler() {

Console.WriteLine("Событие получено объектом класса X");

}

}

class Y {

public void Yhandler() {

Console.WriteLine("Событие получено объектом класса Y");

}

}

class EventDemo2 {

static void Handler() {

Console.WriteLine("Событие получено объектом класса EventDemo");

}

static void Main() {

MyEventClass evt = new MyEventClass();

X xOb = new X();

Y yOb = new Y();

evt.SomeEvent += Handler;

evt.SomeEvent += xOb.Xhandler;

evt.SomeEvent += yOb.Yhandler;

evt.OnSomeEvent();

Console.WriteLine();

evt.SomeEvent -= xOb.Xhandler;

evt.OnSomeEvent();

}

}

#### Применение анонимных методов и лямбда-выражений вместе с событиями

delegate void MyEventHandler (int n);

class MyEvent {

public event MyEventHandler SomeEvent;

public void OnSomeEvent(int n) {

if (SomeEvent != null)

SomeEvent(n);

}

}

class LambdaEventDemo {

static void Main() {

MyEvent evt = new MyEvent();

evt.SomeEvent += (n) =>

Console.WriteLine("Событие получено. Значение равно " + n);

// Запустить событие.

evt.OnSomeEvent(1);

evt.OnSomeEvent(2);

}

}

#### Рекомендации по обработке событий в среде .NET Framework

class MyEventArgs : EventArgs {

public int EventNum;

}

delegate void MyEventHandler(object source, MyEventArgs arg);

class MyEvent {

static int count = 0;

public event MyEventHandler SomeEvent;

public void OnSomeEvent() {

MyEventArgs arg = new MyEventArgs();

if(SomeEvent != null) {

arg.EventNum = count++;

SomeEvent(this, arg);

}

}

}

class X {

public void Handler(object source, MyEventArgs arg) {

Console.WriteLine("Событие " + arg.EventNum +

" получено объектом класса X.");

Console.WriteLine("Источник: " + source);

Console.WriteLine();

}

}

class Y {

public void Handler(object source, MyEventArgs arg) {

Console.WriteLine("Событие " + arg.EventNum +

" получено объектом класса Y.");

Console.WriteLine("Источник: " + source);

Console.WriteLine();

}

}

class EventDemo6 {

static void Main() {

X ob1 = new X();

Y ob2 = new Y();

MyEvent evt = new MyEvent();

evt.SomeEvent += ob1.Handler;

evt.SomeEvent += ob2.Handler;

evt.OnSomeEvent();

evt.OnSomeEvent();

}

}