Московский авиационный институт

(национальный исследовательский университет)

**Факультет информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

**Лабораторная работа 6 по курсу ОOП:**

**основы программирования на языке С#**

АССОЦИАЦИЯ

Работу выполнил:

Березнев Никита Вадимович

Группа: М8О-203Б-21

Подпись: \_\_\_\_\_\_\_\_

Дата: 29 октября 2022 г.

**6.1 Ассоциация 1:1**

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharpLab6\_1

{

class Program

{

//1 : 1

//ссылка используется

//в F на K. и наоборот

class F

{

public F() { }

~F() { }

public K k { set; get; }

public int J() { return 0; }

}

class K

{

public K()

{

//this.i = 1;

}

~K() { }

public F f { set; get; }

public int i { set; get; }

public int J() { return 1; }

}

static void Main(string[] args)

{

//создаются объекты

F f = new F();

K k = new K();

//пока что эти объекты классов не связаны. их надо связать с помощью ассоциации

//ассоциация

//объект k обращается к атрибуту доступа класса F( и присваивает ему f

k.f = f;

f.k = k;

Console.WriteLine("f.J() = {0}", f.J());

Console.WriteLine("k.J() = {0}", k.J());

//обращение к функциям.нельзя так делать, пока не проведена ассоциация, иначе бо-бо:)

Console.WriteLine("k.f.J = {0}", k.f.J());

Console.WriteLine("f.k.J = {0}", f.k.J());

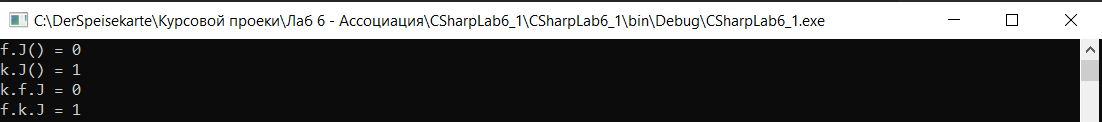
Console.ReadKey();

}

}

}

**Результат:**



**Вывод:**

Ассоциация - двусторонняя зависимость. При ассоциации взаимодействующие объекты между собой условно находятся на одном уровне. Ни один из объектов не имеет превосходства над другим, и из любого можно получить другой связанный с ним объект.

**6.2 Ассоциация 1:N**

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharpLab6\_2

{

class Program

{

class F

{

//конструктор номер 1

public F()

{

this.k = new K[N];

Console.WriteLine("Constructor1 F");

}

//конструктор номер 2, если N вводится пользователем, например

public F(int N)

{

this.N = N;

this.k = new K[N];//массив скрытых ссылок на объекты класса K

Console.WriteLine("Constructor2 F");

}

~F() { Console.WriteLine("Destructor F"); }

public void setK(K k)

{

//ассоциация

if (size < N)

{

Console.WriteLine("Proverka");

this.k[size] = k;

size++;

}

}

//метод, позволяющий просматривать последовательно объекты класса K, связанные с объектом класса F

public K getNext(int index)

{

if (index < size)

{

return k[index];//возвращаем ссылку

}

return null;

}

private K[] k = null;

private int size = 0;

//количество объектов класса K

private int N = 7;

}

class K

{

//конструктор 1

public K() { Console.WriteLine("Constructor1 K"); }

//конструктор 2

public K(F f)

{

f.setK(this);

Console.WriteLine("Constructor2 K");

}

~K() { Console.WriteLine("Destructor K"); }

public int j() { return v; }

private int v = 11;

//атрибут доступа

public F f { set; get; }

}

static void Main(string[] args)

{

F f1 = new F(); //создание объекта класса D с помощью конструктора номер 1

F f2 = new F(8);//создание объекта класса D с помощью конструктора номер 2

K k1 = new K();

f1.setK(k1);

K k3 = new K();

f1.setK(k3);

k1.f = f1;

Console.WriteLine("k1.f = {0}", k1.f);

Console.WriteLine("k1.f.getNext(0) = {0}", k1.f.getNext(0));

Console.WriteLine(" f1.getNext().j() = {0}", f1.getNext(0).j());

Console.WriteLine(" f1.getNext().j() = {0}", f1.getNext(1).j() + 1);

K k2 = new K(f1);

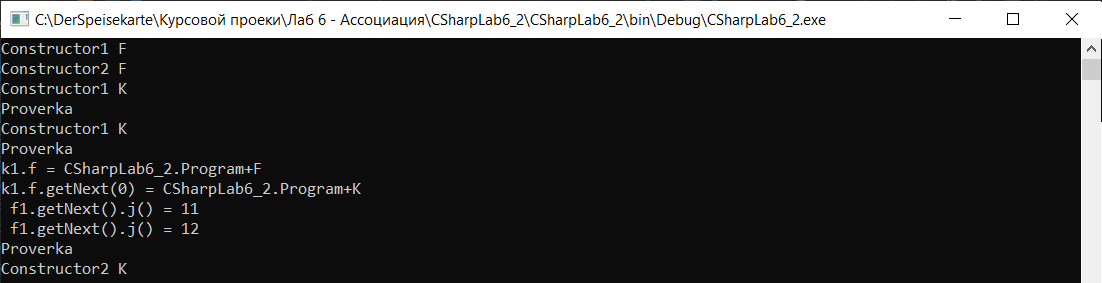
Console.ReadKey();

}

}

}

**Результат:**



**Вывод:**

В ассоциации 1 : N  каждый из N объектов указывает на 1 общий для всех объект, который в свою очередь имеет N указателей на другие объекты. При этом из одного объекта группы с помощью общего можно перейти к другим объектам этой группы