Московский авиационный институт

(национальный исследовательский университет)

**Факультет информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

**Лабораторная работа 5 по курсу ОOП:**

**основы программирования на языке С#**

КОМБИНИРОВАНИЕ ЧЕРЕЗ ОБЩИХ ПРЕДКОВ

Работу выполнил:

Березнев Никита Вадимович

Группа: М8О-203Б-21

Подпись: \_\_\_\_\_\_\_\_

Дата: 29 октября 2022 г.

**5. Наследование. Комбинирование через общих предков**

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CSharpLab5

{

class Program

{

public interface A

{

void Va();

int Fa();

}

public interface F : A

{

int Ff();

void Vf();

}

public class C : A

{

public C() { Console.WriteLine("Constructor C"); this.c = 1; }

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

public virtual int Fa() { return 0; }

public virtual void Va() { Console.WriteLine("method A in class C"); }

protected int Fc(int c) { return this.c; }

protected int c { get; set; }

}

public class D : C

{

public D() { Console.WriteLine("Constructor D"); this.d = 2; this.c = 2; }

public D(int d) { Console.WriteLine("Another construct of D"); this.d = d; }

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

public override int Fa() { return 100; }

public int Fd(int n) { return this.d \* this.c + n; }

protected int d { get; set; }

}

public class E : D, F

{

public E() { Console.WriteLine("Constructor E"); this.e = 1; this.d = 3; }

// значение x передается в конструктор E. Значения параметра конструктора производного класса, передаётся в конструктор базового класса.

public E(int x, int y) : base(x) { Console.WriteLine("Another construct of E"); this.e = x; this.d = y; }

~E() { Console.WriteLine("Destructor D"); }

public override int Fa() { return this.e + this.d; }

public int Ff() { return this.e + this.d + 100; }

public void Vf() { Console.WriteLine("Method B in class E"); }

public int e { get; set; }

}

static void Main(string[] args)

{

A a = null;

a = new C();

Console.WriteLine("C: a.Fa() = {0}", a.Fa());

a = new D(); //в ин-с A передаём указатель на объект класса D

Console.WriteLine("D: a.Fa() = {0}", a.Fa());//должно вывести 100

Console.WriteLine();

a = new E(); //в ин-с A передаём указатель на объект класса E

Console.WriteLine("E: a.Fa() = {0}", a.Fa()); //должно вывести 4

Console.WriteLine();

a = new E(12, 15);

Console.WriteLine("a.Fa() = {0}", a.Fa());//должно вывести 27

Console.ReadKey();

//комбинирование

Console.WriteLine();

F f = null;

f = new E();

Console.WriteLine("f.Ff() = {0}", f.Ff());//должно вывести 100

f.Vf();

Console.WriteLine("f.Ff() + f.Fa() = {0}", f.Ff() + f.Fa());//должно вывести 104 + 4

Console.WriteLine();

Console.ReadKey();

//преобразую объект e в F

E e = new E();

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

Console.WriteLine("f = {0}", ((F)e).Fa());

Console.WriteLine("f = {0}", ((A)((F)e)).Fa());

Console.WriteLine();

Console.ReadKey();

//созданные объекты с помощью конструктора E разные, потому что по разному проинициализированы

F f1 = new E();

A a1 = new E();

Console.WriteLine("f = {0}", f1.Fa());

Console.WriteLine("f = {0}", a1.Fa());

Console.WriteLine("f = {0}", f1.Ff());

Console.WriteLine("f = {0}", ((F)a1).Ff());

//

Console.WriteLine(((D)f1).Fd(1));

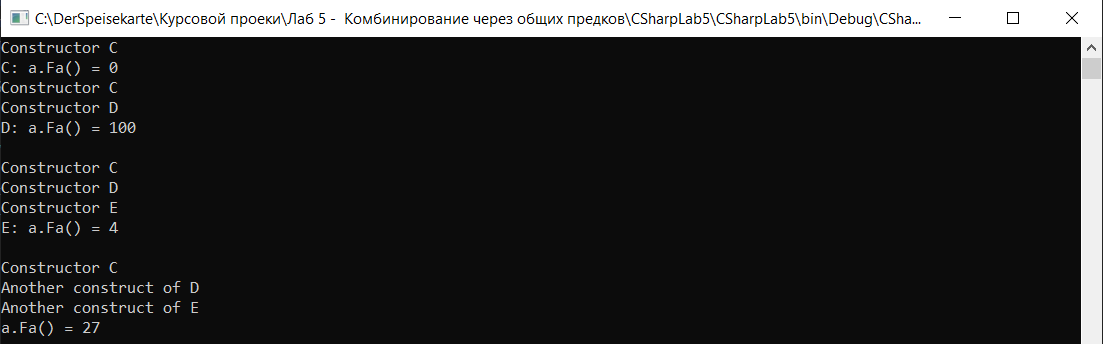
Console.ReadKey();

}

}

}

**Результат:**



**Вывод:**

Комбинирование через общих предков позволяет реализовать множественное наследование через общий суперкласс; комбинирует в себе два класса с общим суперклассом, который, в свою очередь, сам присутствует только в единственном экземпляре благодаря virtual наследованию, позволяющему избежать дублирование этого суперкласса.