**Лабораторная работа №5. Комбинирование через общих предков.**

**Диаграмма классов: Диаграмма объектов:**

Public

Private

Protected

F

E

Int Fe(); int Ff();

f

e

Public

D void mf();

Private

Protected

B

Protected

Private

Public

C

Int Fe(); int Fb();

b

Int Fc(); void mb();

c

Protected

Private

Public

Int Ff();

Void mf();

Protected

Private

Public

A

Int Fe();

Int Fc();

Int Ff();

c

Int Fb();

Void mf();

Void mb();

**Текст программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Form\_1

{

class Program

{

public interface F

{

void mf();

int Ff();

}

public interface B : F// при наследовании интерфейса, методы определенные в них, могут быть определены или нет, уже по усмотрению, но они существуют в унаследованном интерфейсе

{

void mb();

int Fb();

}

class E

{

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

~E() { }

public virtual int Fe(int e) { return this.e = e + 1; }

protected int e { get; set; }

}

class C : E, F

{

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

~C() { }

public virtual void mf() { Console.WriteLine("method F of class C"); }

public virtual int Ff() { return this.c = this.c + 1; }

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

public virtual void Fc() { Console.WriteLine("Fucntion of class C"); }

protected int c { get; set; }

}

class A : C, B

{

public A(int n) : base(n) { this.a = n; Console.WriteLine("Constructor A"); }

~A() { }

public void mb() { Console.WriteLine("Method B of class A"); }

public int Fb() { return this.a = this.a + 2; }

public override int Fe(int e) { return base.Fe(e); }

public override void Fc() { Console.WriteLine("Fucntion of class C in class A"); }

public override void mf() { Console.WriteLine("method F of class A"); }

public override int Ff() { return this.a + 5; }

protected int a { get; set; }

}

static void Main(string[] args)

{

F f = null;

f = new C();

Console.WriteLine("f.Ff={0}", f.Ff()); // выведет 3

f.mf();

Console.WriteLine();

f = new A(4);

Console.WriteLine("f.Ff={0}", f.Ff());// выведет 9

f.mf();

Console.WriteLine();

A a = new A(3);

a.mf();

a.Fc();

Console.WriteLine("a.Fe()={0}", a.Fe(10));// выведет 11

Console.WriteLine();

B b = null;

b = new A(5);

b.mb();

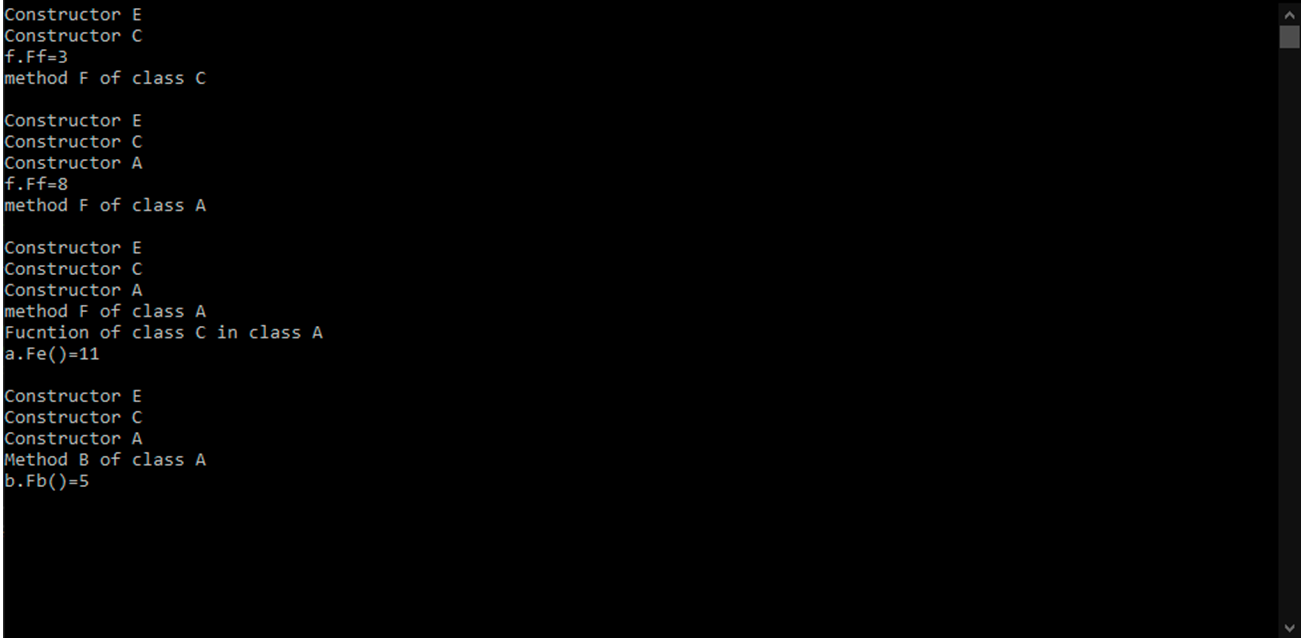
Console.WriteLine("b.Fb()={0}", b.Fb());// выведет 7

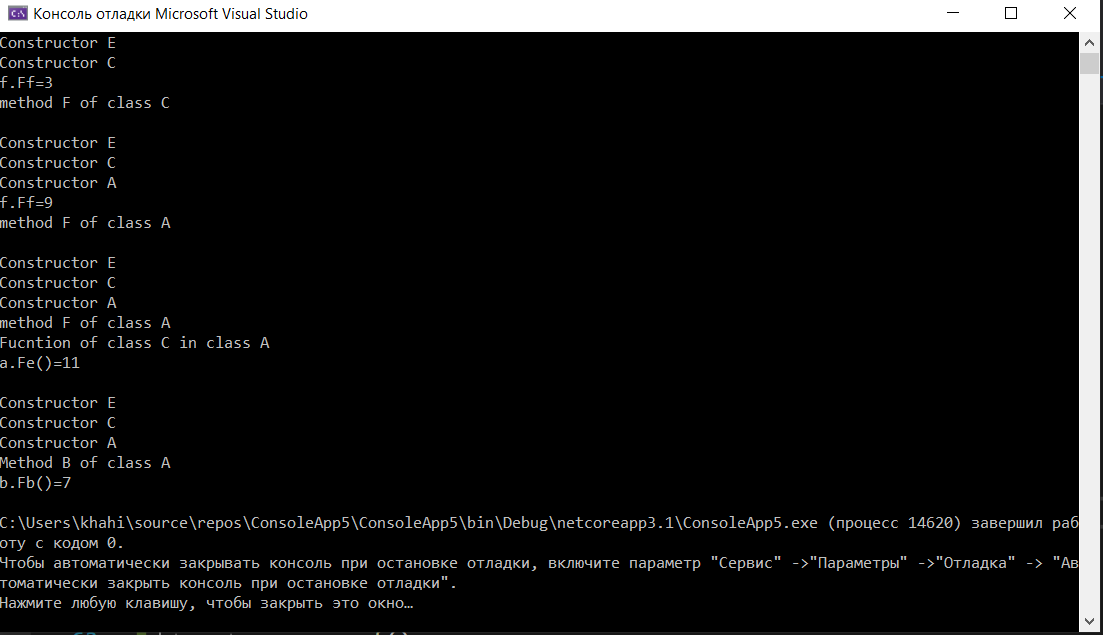
Console.ReadKey();

}

}

}





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, позволяющее избежать дублирование этого суперкласса.