Московский авиационный институт

(Национальный исследовательский университет)

**Факультет информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

**Лабораторные работы 1 — 9 по курсу ООП:  
основы программирования на С#**

0.Перегрузка

1.Агрегация по ссылке

2.Агрегация по значению и вложением

3.Принцип подстановки

4.Наследование: расширение, спецификация, специализация, конструирование и комбинирование

5.Наследование: комбинирование через общих предков

6.Ассоциация(один к одному, один ко многим)

7.Использование

8.Конкретизация

9. АНОНИМНЫЕ ФУНКЦИИ: delegate, ЛЯМБДА ВЫРАЖЕНИЕ. Event

Работу выполнил:

Руководитель: \_\_\_\_\_\_\_/Семенов А.С.

Дата: \_\_\_

**Лабораторная работа №1. Агрегация по ссылке.**



**A**

Public

Private

Public

**F**

Protected

Private

Public

**E**

Protected

Private

Public

**K**

Protected

Private

Public

**C**

Protected

Private

Public

**B**

Protected

Private

Private

Public

**J**

Protected

Protected

**Рис.1.** диаграмма **классов:** агрегация по ссылке.

**Текст программы.** Диаграмма **сборки объектов**

using System;

A

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks; A()

namespace lab1\_AgregationLink

{

B

C

J

class Program

{

class A B() C()

{

public A(J j,B b,C c)

{

F

E

K

this.j = j;

this.b = b;

this.c = c;

c.cq = 21;

}

public void mA()

{

Console.WriteLine("method of A");

}

public J jA

{

set { Console.WriteLine("set j"); j = value; }

get { Console.Write("get j ->"); return j; }

}

public B bA

{

set { Console.WriteLine("set b"); b = value; }

get { Console.Write("get b ->"); return b; }

}

public C cA

{

set { Console.WriteLine("set c"); c = value; }

get { Console.Write("get c ->"); return c; }

}

private B b = null;

private C c = null;

private J j = null;

}

class J

{

public J() { }

~J() { }

public void mJ()

{

Console.WriteLine("Method of J");

}

}

class B

{

public B(K k)

{

this.k = k;

}

public void mB()

{

Console.WriteLine("method of B");

}

public K kA

{

set { Console.WriteLine("set k"); k = value; }

get { Console.Write("get k ->"); return k; }

}

private K k = null;

}

class C

{

public C(E e, F f)

{

this.e = e;

this.f = f;

}

public void mC()

{

Console.WriteLine("method of C");

}

public E eA

{

set { Console.WriteLine("set e"); e = value; }

get { Console.Write("get e ->"); return e; }

}

public F fA

{

set { Console.WriteLine("set f"); f = value; }

get { Console.Write("get f ->"); return f; }

}

public int cq { get; set; }

private E e = null;

private F f = null;

}

class K

{

public K() { }

~K() { }

public void mK()

{

Console.WriteLine("Method of K");

}

}

class E

{

public E() { }

~E() { }

public void mE()

{

Console.WriteLine("Method of E");

}

}

class F

{

public F() { }

~F() { }

public void mF()

{

Console.WriteLine("Method of F");

}

}

static void Main(string[] args)

{

E e = new E();

F f = new F();

J j = new J();

K k = new K();

B b = new B(k);

C c = new C(e, f);

A a = new A(j, b, c);

Console.WriteLine("печать атрибута доступа:");

Console.WriteLine(c.cq);

Console.WriteLine("передача по ссылке:");

a.mA();

a.bA.mB();

a.cA.mC();

a.jA.mJ();

a.bA.kA.mK();

a.cA.eA.mE();

a.cA.fA.mF();

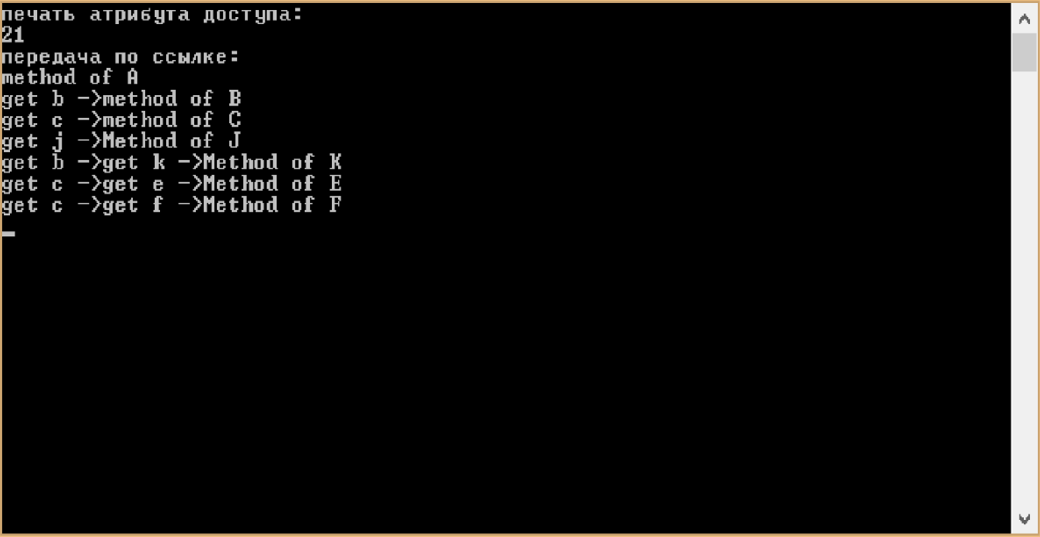
Console.ReadKey();

}

}

}

**Вывод программы:**

****

**Вывод:** *Агрегация по ссылке – когда один объект имеет ссылку на другой объект*.Объекты всех классов существуют независимо друг от друга. Связывание объектов происходит с помощью конструктора. Например, *b, c, j, —* параметры для конструктора класса A; *k* — для конструктора класса B; *e, f —* для конструктора класса C. Объекты могут быть уничтожены по отдельности. Это нарушит целостность структуры. Если удалить объект *a,* объекты *b, c, j* и т.д. будут продолжать существовать и дальше. Агрегация по ссылке очень выгодна, так как она расходует очень мало памяти.