Московский Авиационный Институт

(Национальный Исследовательский Университет)

**Факультет информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

Лабораторная работа 1 по курсу ООП:

Основы программирования на языке C#

1.АГРЕГАЦИЯ ПО ССЫЛКЕ

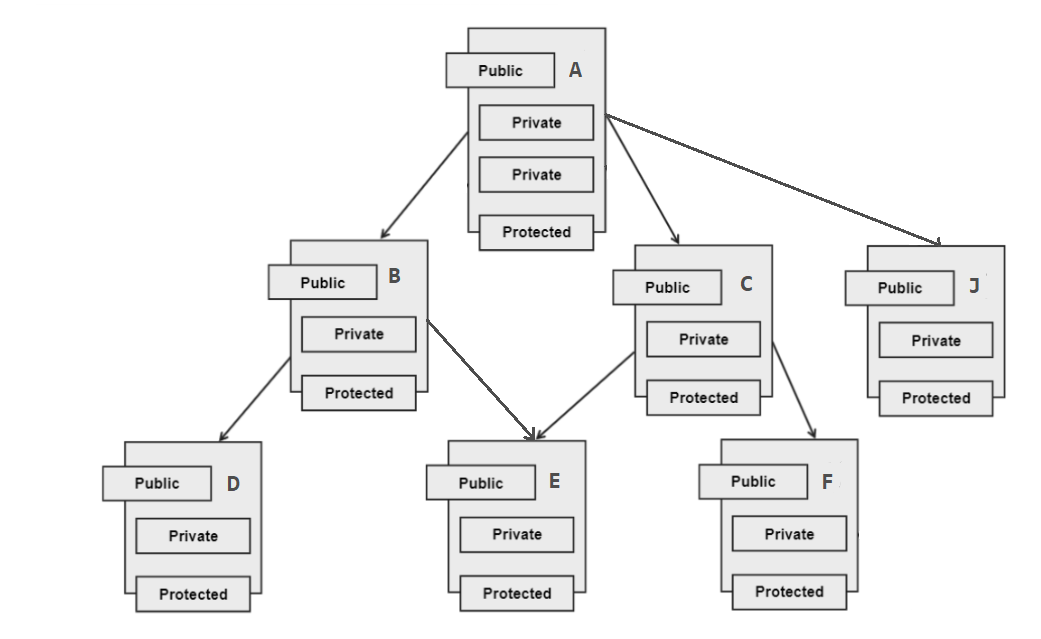
Работу выполнил:

М8О-205Б-21 Жилов А.А. \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

Руководитель: \_\_\_\_\_\_\_\_\_\_\_/Кузнецова С.В.

Дата: \_\_\_Октября 2022

**Агрегация по ссылке**



**Текст программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_1

{

class A

{

private B b = null;

private C c = null;

private J j = null;

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

{

this.b = b;

this.c = c;

this.j = j;

}

public void mA()

{

Console.WriteLine("Metod of A");

}

public B bA

{

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

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

}

public C cA

{

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

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

}

public J jA

{

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

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

}

}

class B

{

private D d = null;

private E e = null;

public B(D d, E e)

{

this.d = d;

this.e = e;

}

public void mB()

{

Console.WriteLine("Metod of B");

}

public D dB

{

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

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

}

public E eB

{

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

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

}

}

class C

{

private F f = null;

private E e = null;

public C(F f, E e)

{

this.f = f;

this.e = e;

this.c\_value = 0;

}

public void mC()

{

Console.WriteLine("Metod of C");

}

public F fC

{

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

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

}

public E eC

{

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

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

}

public int c\_value { set; get; }

}

class J

{

public J() { }

public void mJ()

{

Console.WriteLine(" method of J");

}

}

class D

{

public D() { }

public void mD()

{

Console.WriteLine(" method of D");

}

}

class E

{

private D d = null;

public E(D d)

{

this.d = d;

}

public void mE()

{

Console.WriteLine(" Method of E");

}

public D dE

{

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

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

}

}

class F

{

public F() { }

public void mF()

{

Console.WriteLine(" method of F");

}

}

internal class Program

{

static void Main(string[] args)

{

J j = new J();

F f = new F();

D d = new D();

E e = new E(d);

B b = new B(d, e);

C c = new C(f, e);

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

Console.WriteLine($"C value is {a.cA.c\_value}");

a.cA.c\_value = 1;

Console.WriteLine($"C vaue is {a.cA.c\_value}");

Console.WriteLine();

a.mA();

a.bA.mB();

a.cA.mC();

a.jA.mJ();

a.bA.dB.mD();

a.bA.eB.mE();

a.cA.eC.mE();

a.cA.fC.mF();

Console.ReadKey();

}

}

}

**Результат работы  
**

**Пример использования**

Данный метод можно использовать для описания структуры производства каких-либо устройств. Например, главный класс пусть будет станок. Сначала мы проинициализируем все его компоненты, затем привязываем их к более крупным и так пока станок не соберется. Причем эти компоненты мы сможем использовать и для других видов станков.

**Вывод**

Объекты классов A, B, C, D, E, F, J, K существуют независимо друг от друга. При этом, например связывание объекта а с объектами b, c, j происходит с помощью конструктора; b, c, j – параметры для конструктора A. Аналогичным образом происходит связывание b c d, e, f и с c e, f. Но при этом объекты могут быть уничтожены по отдельности, что нарушает целостность структуры.