Московский Авиационный Институт

(Национальный Исследовательский Университет)

**Факультет информационных технологий и прикладной математики**

Кафедра вычислительной математики и программирования

Лабораторная работа 2 по курсу ООП:

Основы программирования на языке C#

2.АГРЕГАЦИЯ ПО ЗНАЧЕНИЮ И ВЛОЖЕНИЯМ

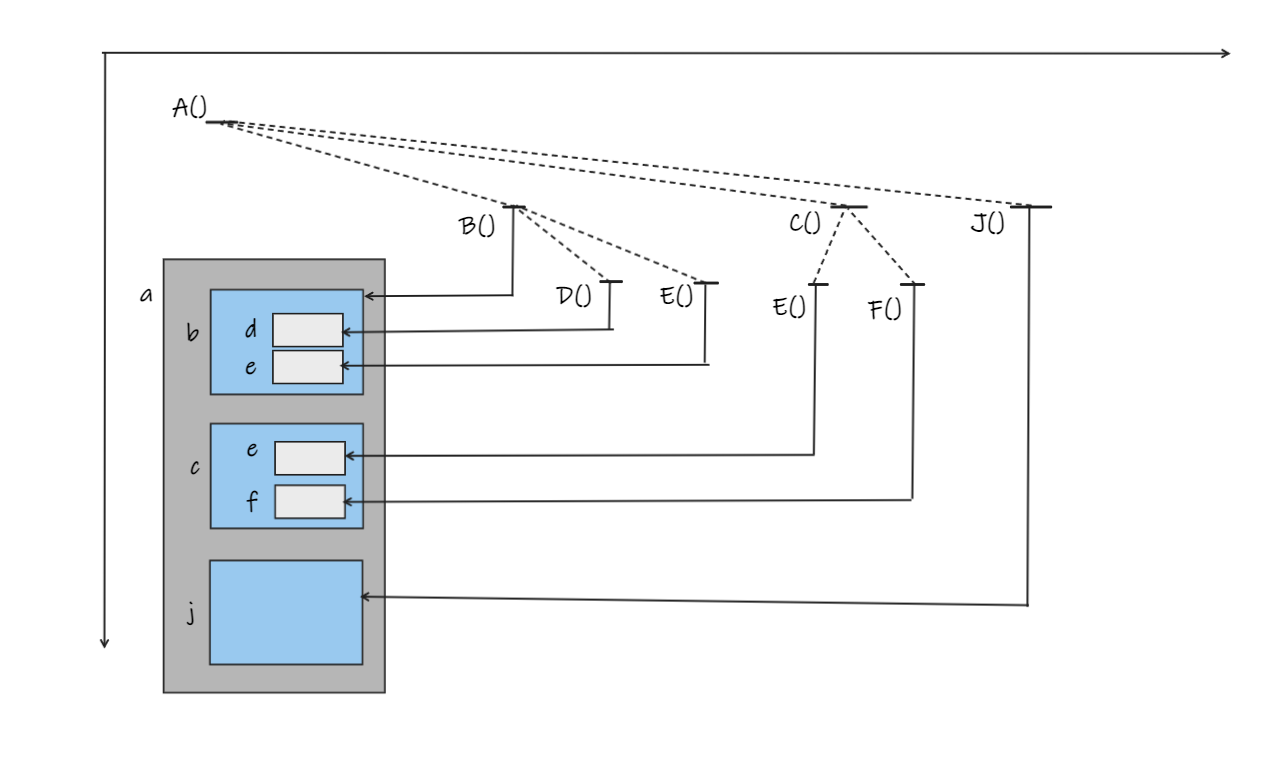
Работу выполнил:

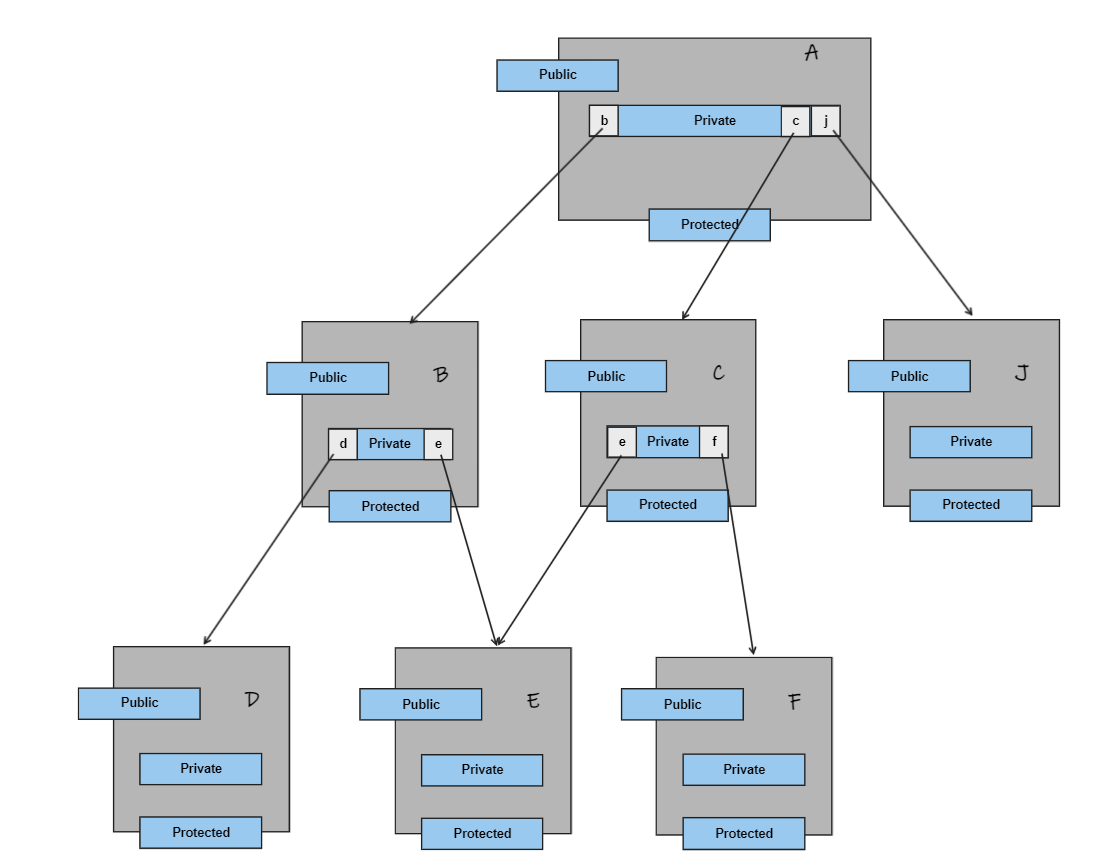
М8О-205Б-21 Жилов А.А. \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

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

Дата: \_\_\_Октября 2022

**Агрегация по значению**

****



**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 = new B();

private C c = new C();

private J j = new J();

public void mA()

{

Console.WriteLine("Metod of A");

}

public B bA

{

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

}

public C cA

{

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

}

public J jA

{

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

}

}

class B

{

private D d = new D();

private E e = new E();

public void mB()

{

Console.WriteLine("Metod of B");

}

public D dB

{

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

}

public E eB

{

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

}

}

class C

{

private F f = new F();

private E e = new E();

public C()

{

this.c\_value = 0;

}

public void mC()

{

Console.WriteLine("Metod of C");

}

public F fC

{

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

}

public E eC

{

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 = new 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)

{

A a = new A();

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

a.cA.c\_value = 1;

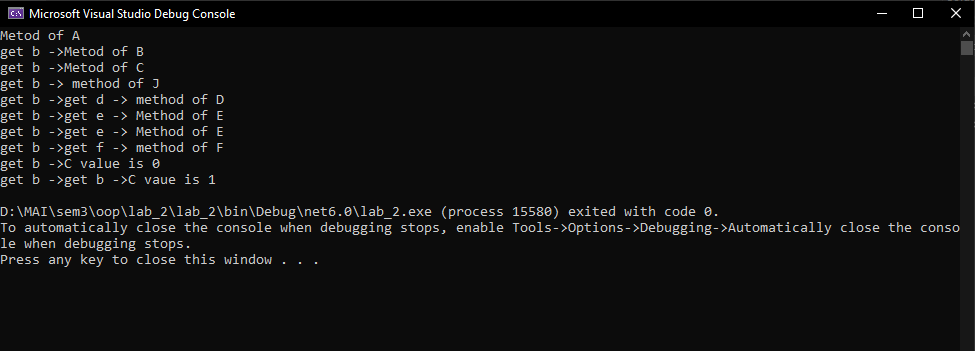
Console.WriteLine($"C vaue is {a.cA.c\_value}");

Console.ReadKey();

}

}

}

**Результат работы  
**

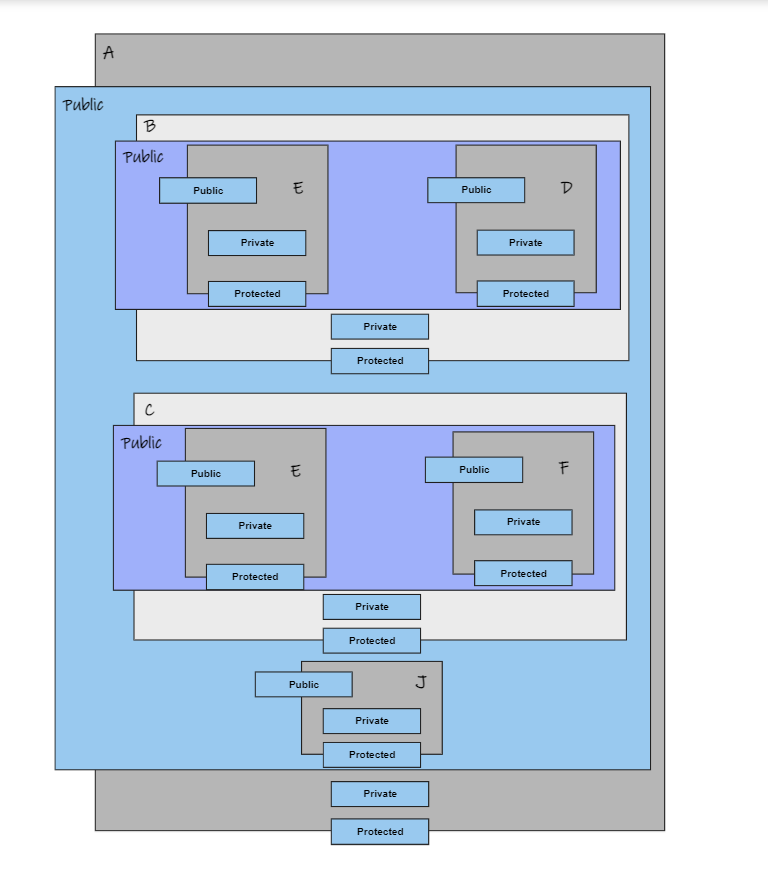
**Пример использования**

Используя данный метод можно описать устройство станка. Например, при инициализации станка создаются все необходимые компоненты. Но мы можем отдельно создать этот компонент, не создавая станок.

**Вывод**

При агрегации по значению, все объекты создаваемого класса существуют внутри него самого, уничтожение их отдельно от объекта, частью которого они являются невозможно. Их уничтожение происходит при уничтожение самого верхнего в иерархии объекта. Например: b, c, j – части a, они создаются в конструкторе a, а их уничтожение происходит при вызове деструктора a.

**Агрегация вложением**

****

**Текст программы**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static lab\_2\_2.A;

using static lab\_2\_2.A.B;

namespace lab\_2\_2

{

class A

{

public A() { }

public class B

{

public B() { }

public class D

{

public D() { }

public void mD()

{

Console.WriteLine("metod of D");

}

}

public class E

{

public E() { }

public void mE()

{

Console.WriteLine("metod of E");

}

}

public void mB()

{

Console.WriteLine("metod of B");

}

public D dB

{

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

}

public E eB

{

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

}

D d =new D();

E e =new E();

}

public class C

{

public C()

{

this.c\_value = 0;

}

public class F

{

public F() { }

public void mF()

{

Console.WriteLine("metod of F");

}

}

public class E

{

public E() { }

public void mE()

{

Console.WriteLine("metod of E");

}

}

public F fC

{

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

}

public E eC

{

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

}

public void mC()

{

Console.WriteLine("Metod of C");

}

F f = new F();

E e = new E();

public int c\_value{set; get;}

}

public class J

{

public J() { }

public void mJ()

{

Console.WriteLine("metod of J");

}

}

public void mA()

{

Console.WriteLine("Metod of A");

}

public B bA

{

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

}

public C cA

{

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

}

public J jA

{

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

}

private B b = new B();

private C c = new C();

private J j = new J();

}

internal class Program

{

static void Main(string[] args)

{

A a = new A();

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

a.cA.c\_value = 1;

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

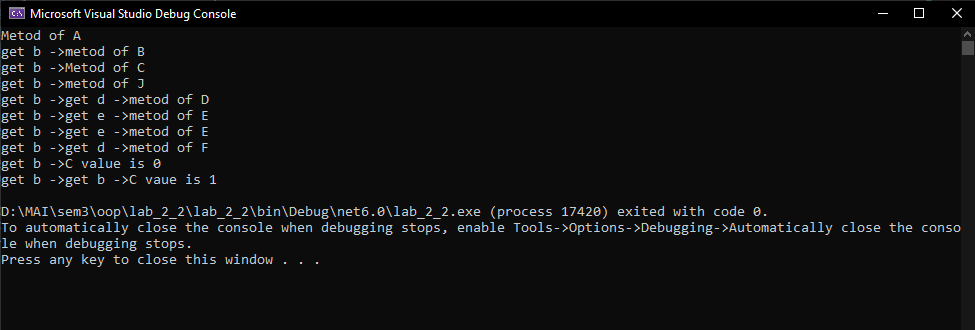
Console.ReadKey();

}

}

}

**Результат работы**

****

**Пример использования**

Данный метод можно использовать для описания пациента. В этой иерархии главным будет пациент, а его болезни будут создаваться внутри этого класса, при этом мы не сможем создать болезни отдельно от пациента.

**Вывод**

При агрегации вложением, объявление классов происходит внутри классов, стоящих выше их по иерархии. Все объекты создаваемого класса существуют внутри него самого, уничтожение их отдельно от объекта, частью которого они являются невозможно. Их уничтожение происходит при уничтожение самого верхнего в иерархии объекта. Например: b, c, j – части a, они создаются в конструкторе a, а их уничтожение происходит при вызове деструктора a.