Тимофеева Наталья

М8О-202Б-19

ООП C#

Лабораторная работа № 2

**Описание работы**

Для категории агрегация по значению создается отдельный проект.

Для категории вложением создается отдельный проект.

1. Изучите категорию агрегация по значению. Представить граф древовидной структуры как иерархию “Целое-Часть”, используя категорию агрегация по значению реализуйте граф программно на С#, каждая вершина графа объект соответствующего класса. Пусть объект А – целое, а все остальные объекты его части в соответствии с иерархией графа, поставьте правильно стрелки на графе.

В главной программе должно быть реализовано движение по графу от целого к частям.

Обратите внимание, что объекты часть на С++, может быть реализован только с конструктором по умолчанию. Предложите вариант реализации.

2. Изучите категорию агрегация вложением. Представить граф древовидной структуры как иерархию “Целое-Часть”, используя категорию агрегация вложением реализуйте граф программно на С#, каждая вершина графа объект соответствующего класса. Пусть объект А – целое, а все остальные объекты его части в соответствии с иерархией графа, поставьте правильно стрелки на графе.

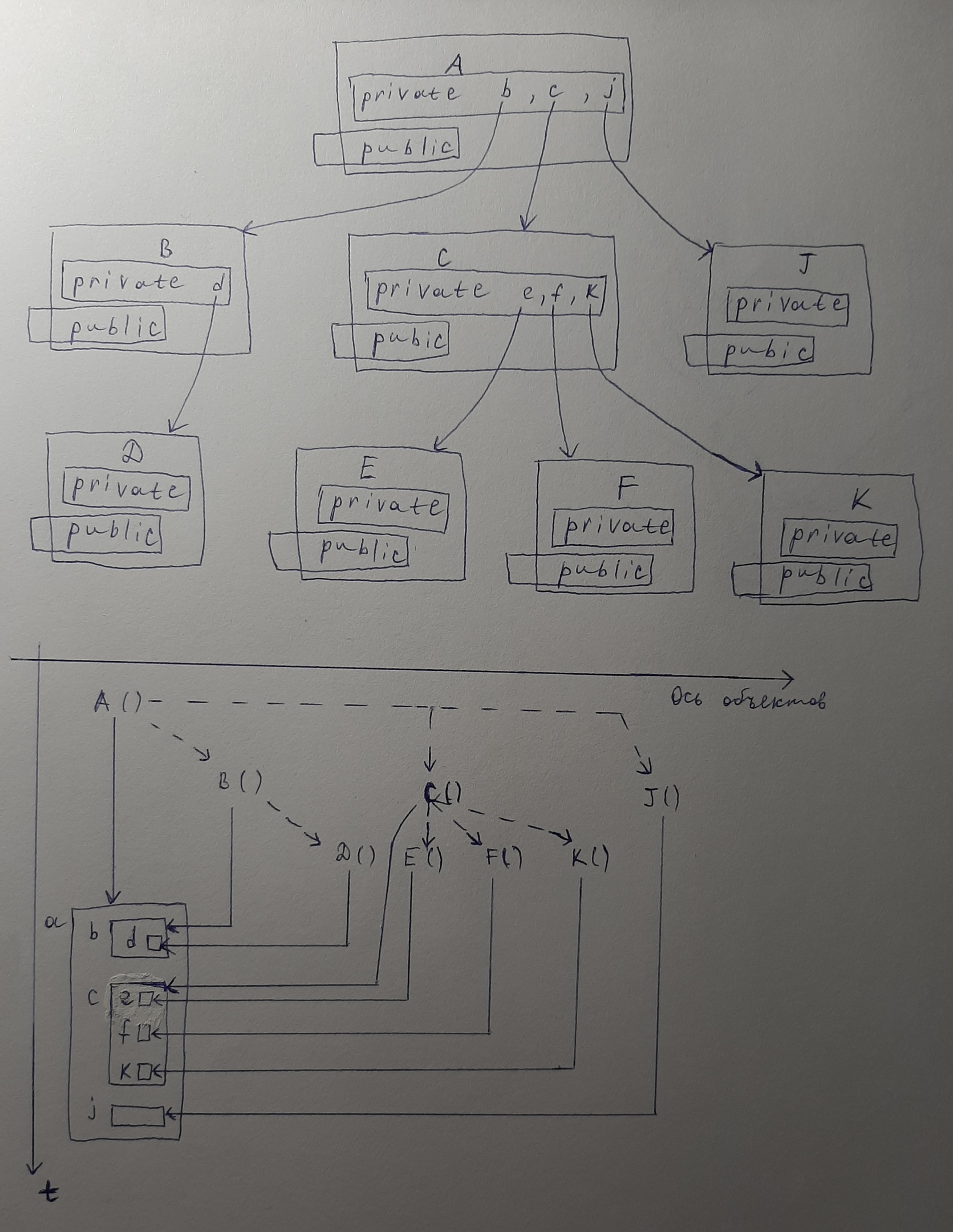
В главной программе должно быть реализовано движение по графу от целого к частям.

Обратите внимание на применение статической функции, для чего она может быть использована.

3. Сравните полученные иерархические объекты, также используйте первую лабораторную работу. В каком случае объект целое можно создать сразу? В каком случае объект целое обладает гибкостью? Какие существуют разновидности категории агрегации, в чем их достоинство и недостатки?

**Агрегация по значению**

**Граф**



**Программа 1 (пункт 1)**

using System;

namespace Lab2\_1

{

class J {

public J() { }

public void print() {

Console.WriteLine("method of J");

}

}

class D {

public D() { }

public void print() {

Console.WriteLine("method of D");

}

}

class E {

public E() { }

public void print() {

Console.WriteLine("method of E");

}

}

class F {

public F() { }

public void print() {

Console.WriteLine("method of F");

}

}

class K {

public K() { }

public void print() {

Console.WriteLine("method of K");

}

}

class B {

D objD = null;

public D d {

get {

Console.Write("-> d ");

return objD;

}

}

public B() {

objD = new D();

}

public void print() {

Console.WriteLine("method of B");

}

}

class C {

E objE = null;

F objF = null;

K objK = null;

public E e {

get {

Console.Write("-> e ");

return objE;

}

}

public F f {

get {

Console.Write("-> f ");

return objF;

}

}

public K k {

get {

Console.Write("-> k ");

return objK;

}

}

public C() {

objE = new E();

objF = new F();

objK = new K();

}

public void print() {

Console.WriteLine("method of C");

}

}

class A {

B objB = null;

C objC = null;

J objJ = null;

public B b {

get {

Console.Write("a -> b ");

return objB;

}

}

public C c {

get {

Console.Write("a -> c ");

return objC;

}

}

public J j {

get {

Console.Write("a -> j ");

return objJ;

}

}

public A() {

objB = new B();

objC = new C();

objJ = new J();

}

public void print() {

Console.WriteLine("method of A");

}

}

class Program {

static void Main()

{

A objA = new A();

K objK = new K();

F objF = new F();

E objE = new E();

D objD = new D();

J objJ = new J();

C objC = new C();

B objB = new B();

objA.print();

objB.print();

objA.b.print();

objD.print();

objA.b.d.print();

objC.print();

objA.c.print();

objE.print();

objA.c.e.print();

objF.print();

objA.c.f.print();

objK.print();

objA.c.k.print();

objJ.print();

objA.j.print();

}

}

}

**Вывод с консоли**

method of A

method of B

a -> b method of B

method of D

a -> b -> d method of D

method of C

a -> c method of C

method of E

a -> c -> e method of E

method of F

a -> c -> f method of F

method of K

a -> c -> k method of K

method of J

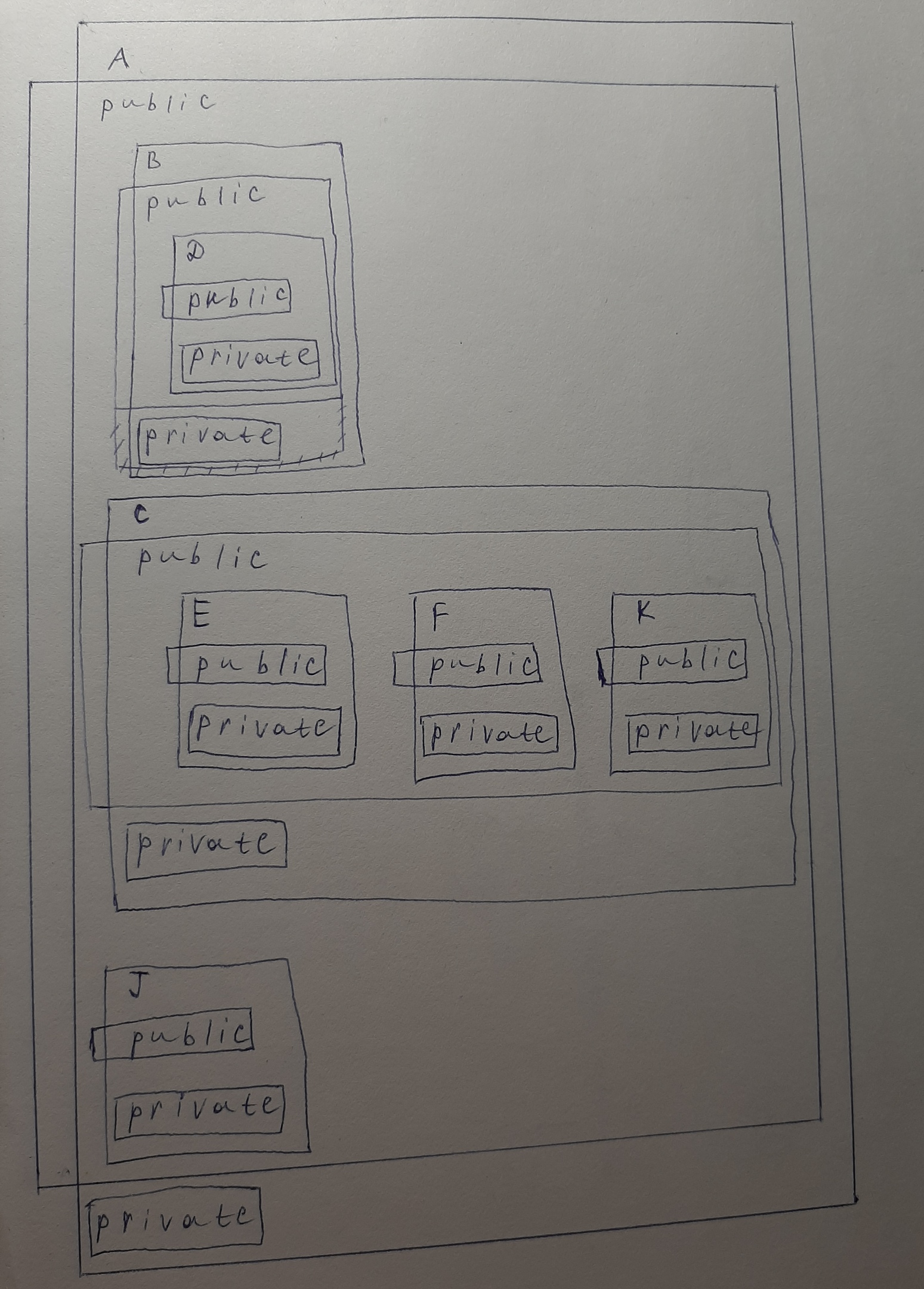
a -> j method of J

**Вывод**

При агрегации по значению уничтожение объекта класса отдельно от объекта, частью которого он является, невозможно. Его уничтожение происходит при уничтожение самого верхнего в иерархии объекта.

**Агрегация вложением**

**Граф**



**Программа 2 (пункт 2)**

using System;

namespace Lab2\_2

{

class A {

public class B {

public class D {

public D() { }

public void print() {

Console.WriteLine("method of D");

}

}

D objD = null;

public D d {

get {

Console.Write("-> d ");

return objD;

}

}

public B() {

objD = new D();

}

public void print() {

Console.WriteLine("method of B");

}

}

public class C {

public class E {

public E() { }

public void print() {

Console.WriteLine("method of E");

}

}

public class F {

public F() { }

public void print() {

Console.WriteLine("method of F");

}

}

public class K {

public K() { }

public void print() {

Console.WriteLine("method of K");

}

}

E objE = null;

F objF = null;

K objK = null;

public E e {

get {

Console.Write("-> e ");

return objE;

}

}

public F f {

get {

Console.Write("-> f ");

return objF;

}

}

public K k {

get {

Console.Write("-> k ");

return objK;

}

}

public C() {

objE = new E();

objF = new F();

objK = new K();

}

public void print() {

Console.WriteLine("method of C");

}

}

public class J {

public J() { }

public void print() {

Console.WriteLine("method of J");

}

}

B objB = null;

C objC = null;

J objJ = null;

public B b {

get {

Console.Write("a -> b ");

return objB;

}

}

public C c {

get {

Console.Write("a -> c ");

return objC;

}

}

public J j {

get {

Console.Write("a -> j ");

return objJ;

}

}

public A() {

objB = new B();

objC = new C();

objJ = new J();

}

public void print() {

Console.WriteLine("method of A");

}

}

class Program

{

static void Main()

{

A objA = new A();

objA.print();

objA.b.print();

objA.b.d.print();

objA.c.print();

objA.c.e.print();

objA.c.f.print();

objA.c.k.print();

objA.j.print();

}

}

}

**Вывод с консоли**

method of A

a -> b method of B

a -> b -> d method of D

a -> c method of C

a -> c -> e method of E

a -> c -> f method of F

a -> c -> k method of K

a -> j method of J

**Вывод**

При агрегации вложением, классы объявляются внутри классов, стоящих выше их по иерархии. Уничтожение объекта класса, находящегося внутри другого невозможно без удаления внешнего класса. Уничтожается сразу всё дерево