

**Лабораторная работа №2. "Агрегация по значению и вложением.
Агрегация по значению".**

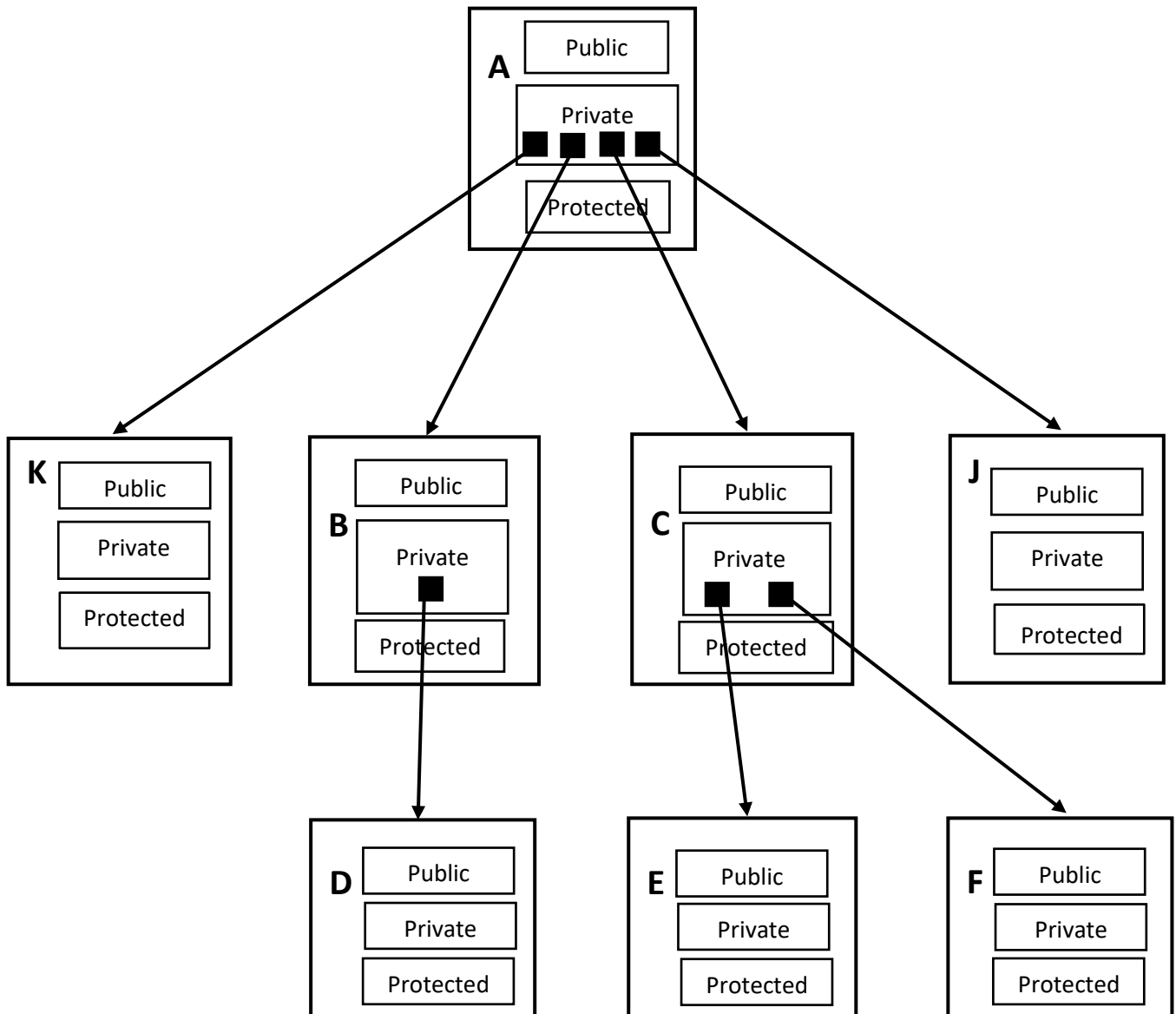
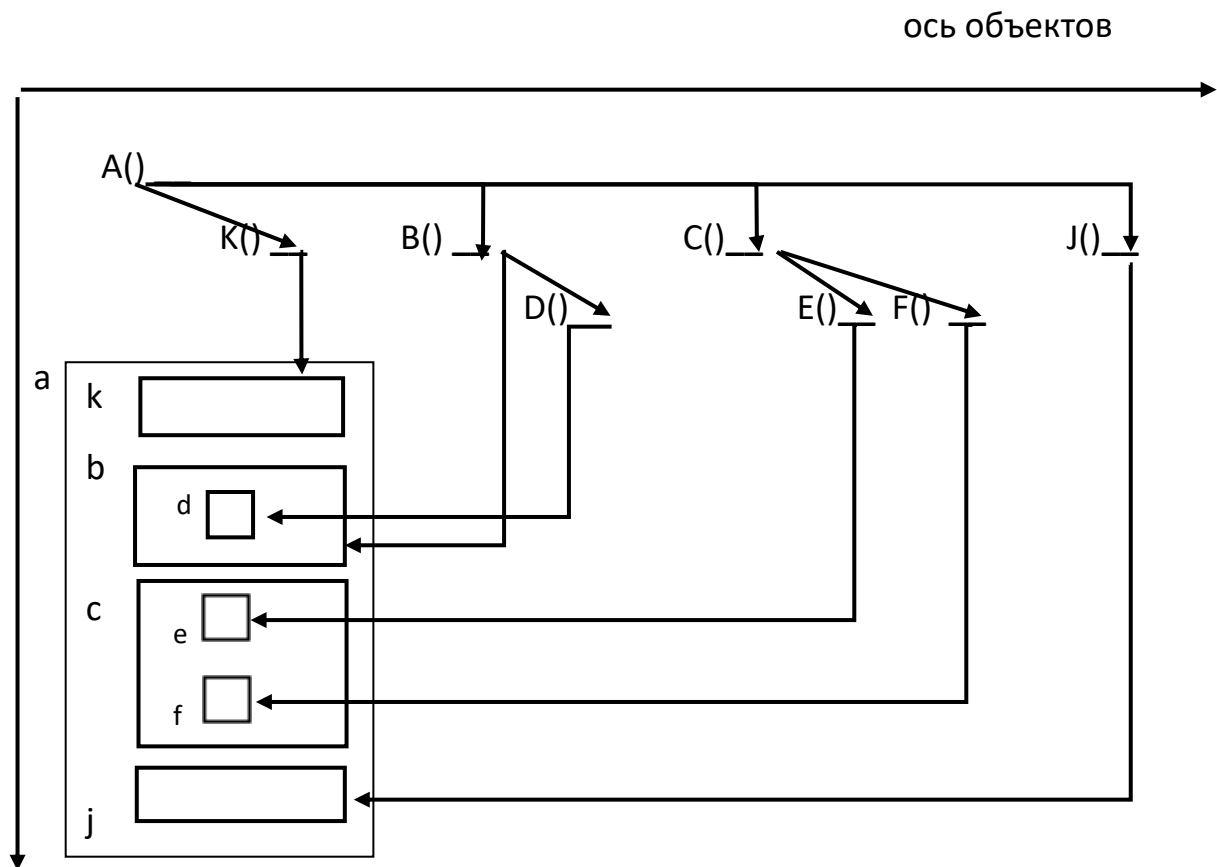


Рис.2. Диаграмма классов: агрегация по значению.



Текст программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LabochkaTwo
{
    class Program
    {
        class A
        {
            public A() { c.cq = 55; }
            ~A() { }
            public void mA() { Console.WriteLine("method of A"); }

            public K kA { get { Console.Write("get k -> "); return k; } }
            public B bA { get { Console.Write("get b -> "); return b; } }
            public C cA { get { Console.Write("get c -> "); return c; } }
            public J jA { get { Console.Write("get j -> "); return j; } }
        }
    }
}
```

```

        private K k = new K();
        private B b = new B();
        private C c = new C();
        private J j = new J();
    }
    class B
    {
        public B() { }
        ~B() { }
        public void mB() { Console.WriteLine("method of B "); }
        public D dA { get { Console.WriteLine("get d -> "); return d; } }

        private D d = new D();
    }
    class C
    {
        public C() { this.cq = 11; }
        ~C() { }

        public void mC() { Console.WriteLine("method of C "); }

        public E eA { get { Console.WriteLine("get e -> "); return e; } }
        public F fA { get { Console.WriteLine("get f -> "); return f; } }

        private E e = new E();
        private F f = new F();

        public int cq { get; set; }
    }
    class D
    {
        public D() { }
        ~D() { }
        public void mD() { Console.WriteLine("method of D "); }
    }
    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 "); }
    }
    class J
    {
        public J() { }
        ~J() { }
        public void mJ() { Console.WriteLine("method of J "); }
    }
    class K
    {
        public K() { }
        ~K() { }
        public void mK() { Console.WriteLine("method of K "); }
    }

```

```

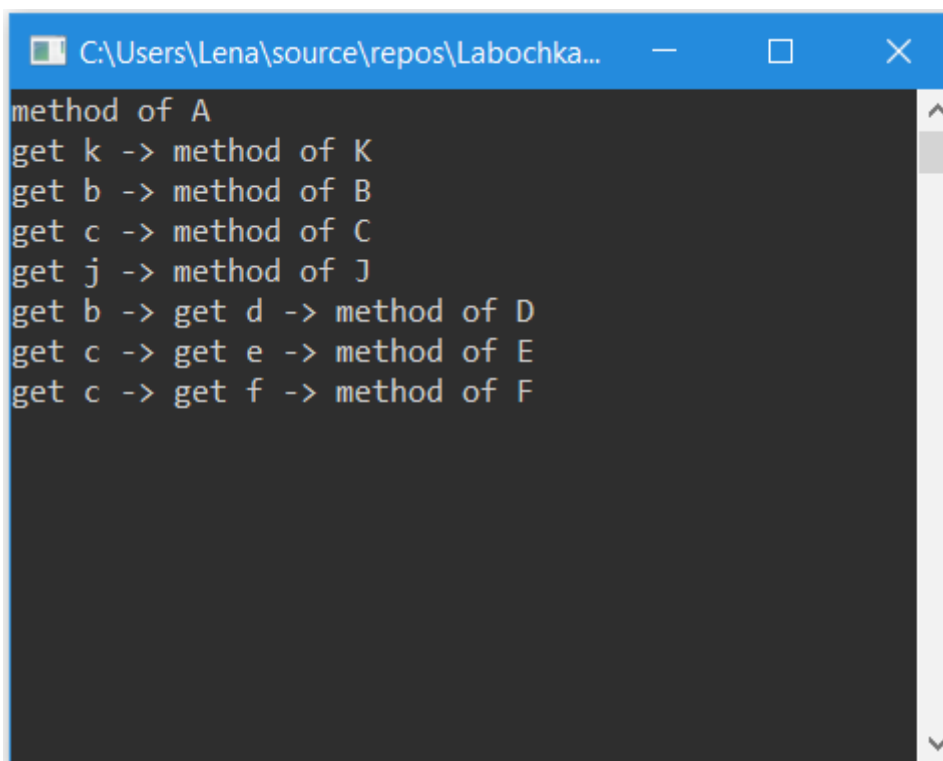
    }

    static void Main(string[] args)
    {
        A a = new A();
        a.mA();
        a.kA.mK();
        a.bA.mB();
        a.cA.mC();
        a.jA.mJ();
        a.bA.dA.mD();
        a.cA.eA.mE();
        a.cA.fA.mF();

        Console.ReadKey();
    }
}

```

Результат работы программы:



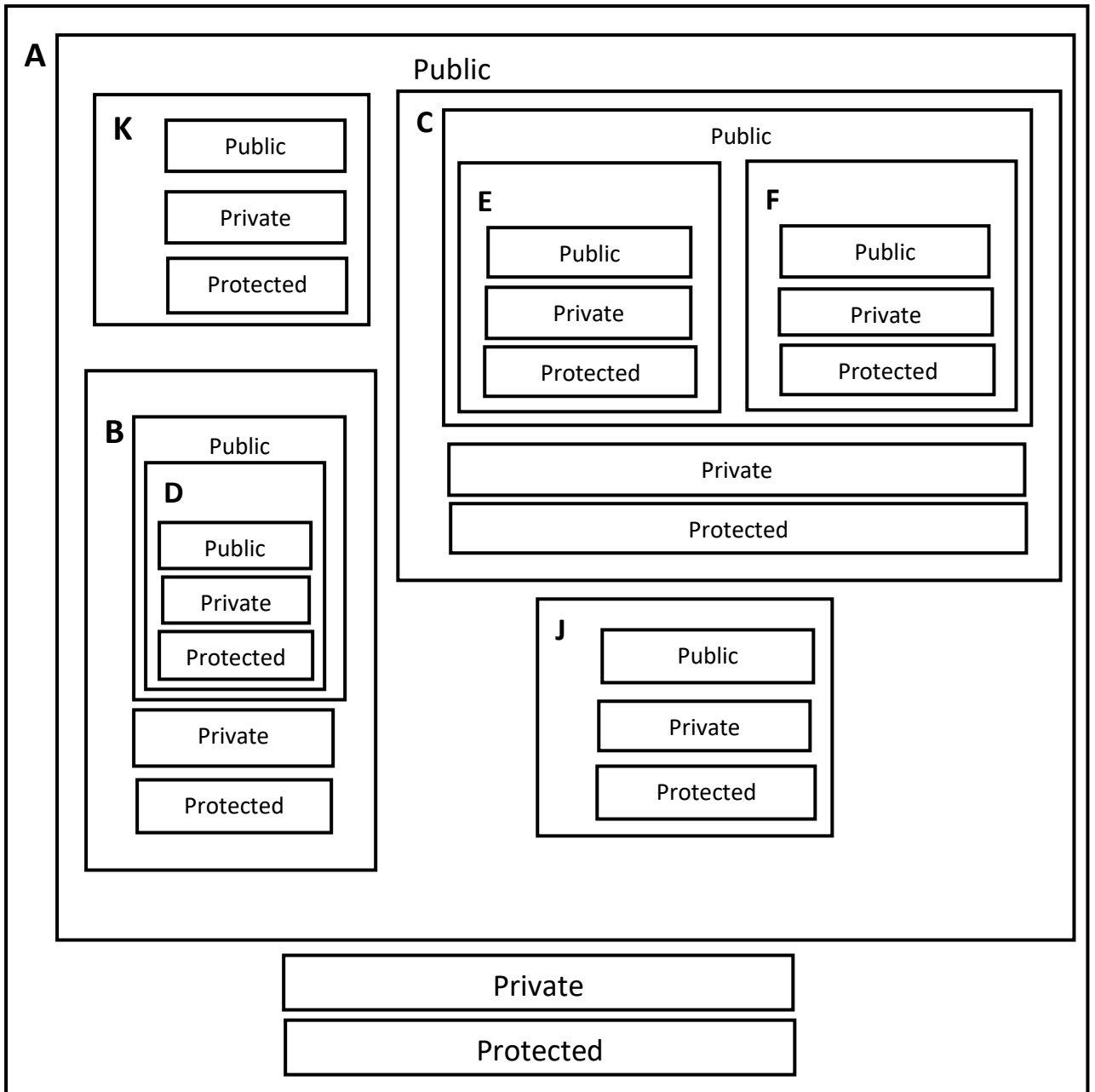
```

method of A
get k -> method of K
get b -> method of B
get c -> method of C
get j -> method of J
get b -> get d -> method of D
get c -> get e -> method of E
get c -> get f -> method of F

```

Вывод: при агрегации по значению все объекты класса существуют непосредственно внутри объявленного класса. При таком виде агрегации невозможно удалить объекты, которые являются частью объекта первого по иерархии класса. *Например*, частями объекта `a` класса `A` (первый класс по иерархии) являются `k`, `b`, `c`, `j`; они создаются только при вызове конструктора класса `A`, а уничтожаются — при вызове деструктора `A`.

Агрегация вложением.



Текст программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laborat2
{
    class Program
    {
        class A
        {
            public A() { c.c1 = 5; }
            ~A() { }

            public class K
            {
                public K() { }
                ~K() { }
                public void mK() { Console.WriteLine("method of K "); }
            }

            public class B
            {
                public B() { }
                ~B() { }
                public class D
                {
                    public D() { }
                    ~D() { }
                    public void mD() { Console.WriteLine("method of D "); }
                }
                public void mB() { Console.WriteLine("method of B "); }
                public D dA { get { Console.Write("get d -> "); return d; } }
                private D d = new D();
            }
            public class C
            {
                public C() { this.c1 = 10; }
                ~C() { }
                public class E
                {
                    public E() { }
                    ~E() { }
                    public void mE() { Console.WriteLine("method of E "); }
                }
                public class F
                {
                    public F() { }
                    ~F() { }
                    public void mF() { Console.WriteLine("method of F "); }
                }

                public void mC() { Console.WriteLine("method of C "); }
                public E eA { get { Console.Write("get e -> "); return e; } }
                public F fA { get { Console.Write("get f -> "); return f; } }

                private E e = new E();
            }
        }
    }
}
```

```

        private F f = new F();

        public int c1 { set; get; }
    }
    public class J
    {
        public J() { }
        ~J() { }
        public void mJ() { Console.WriteLine("method of J "); }
    }

    public void mA() { Console.WriteLine("method of A "); }

    public K kA { get { Console.Write("get k -> "); return k; } }
    public B bA { get { Console.Write("get b -> "); return b; } }
    public C cA { get { Console.Write("get c -> "); return c; } }
    public J jA { get { Console.Write("get j -> "); return j; } }

    private K k = new K();
    private B b = new B();
    private C c = new C();
    private J j = new J();
}
static void Main(string[] args)
{
    A a = new A();
    a.mA();
    a.kA.mK();
    a.bA.mB();
    a.cA.mC();
    a.jA.mJ();
    a.bA.dA.mD();
    a.cA.eA.mE();
    a.cA.fA.mF();

    Console.ReadKey();
}
}
}

```

Результат работы программы:

```

C:\Users\Lena\source\repos\Laborat2\...
method of A
get k -> method of K
get b -> method of B
get c -> method of C
get j -> method of J
get b -> get d -> method of D
get c -> get e -> method of E
get c -> get f -> method of F
_

```

Вывод: при агрегации вложением определение классов происходит внутри классов, стоящих выше по иерархии. Все объекты создаваемого класса существуют непосредственно внутри него самого. Как и в случае агрегации по значению, уничтожение объектов невозможно без уничтожения класса, который стоит выше по иерархии. Агрегация по ссылке расходует меньше памяти. Именно поэтому агрегация по ссылке предпочтительнее агрегации по значению или вложению.