Лабораторная работа №3. "Принцип подстановки".

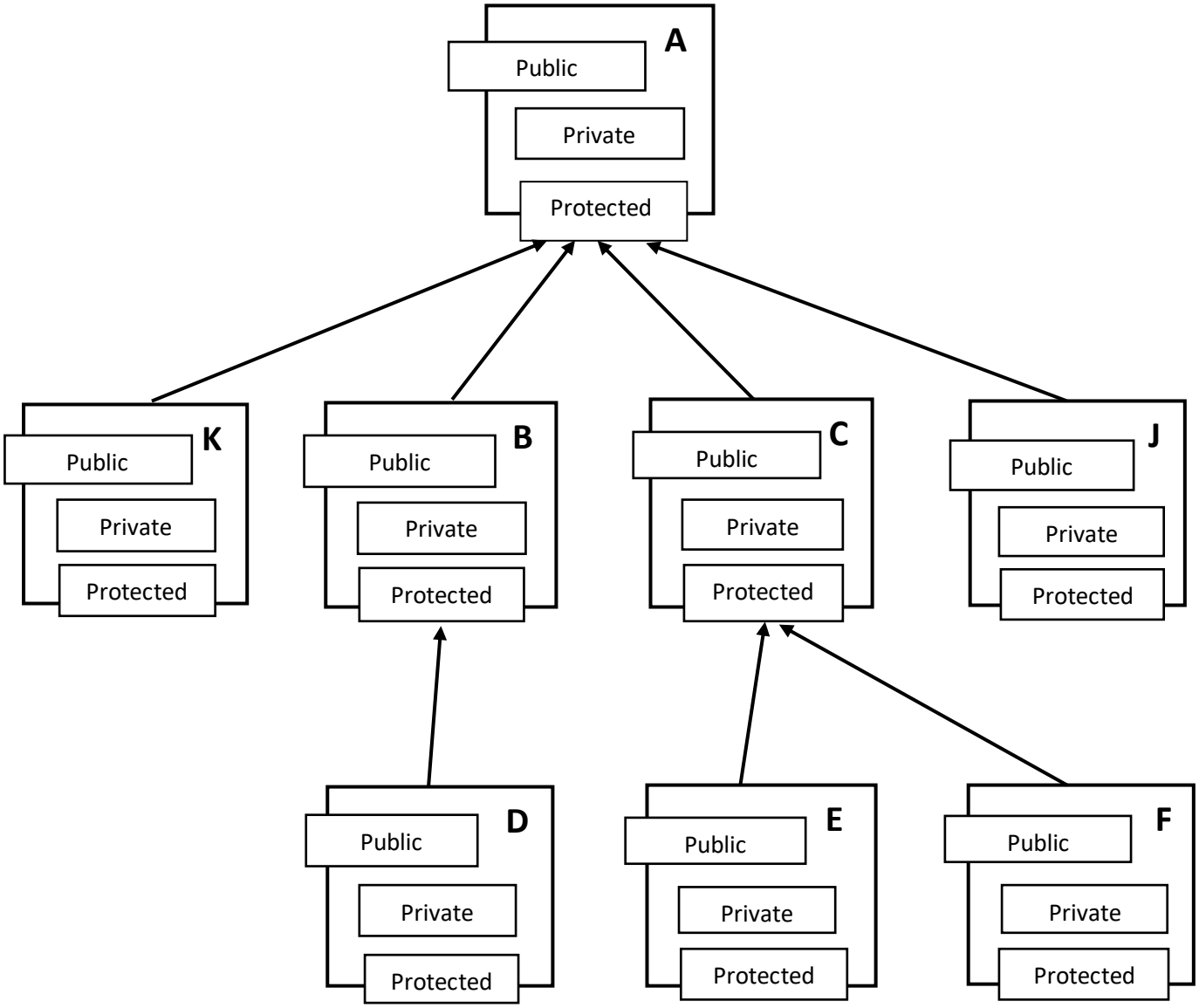
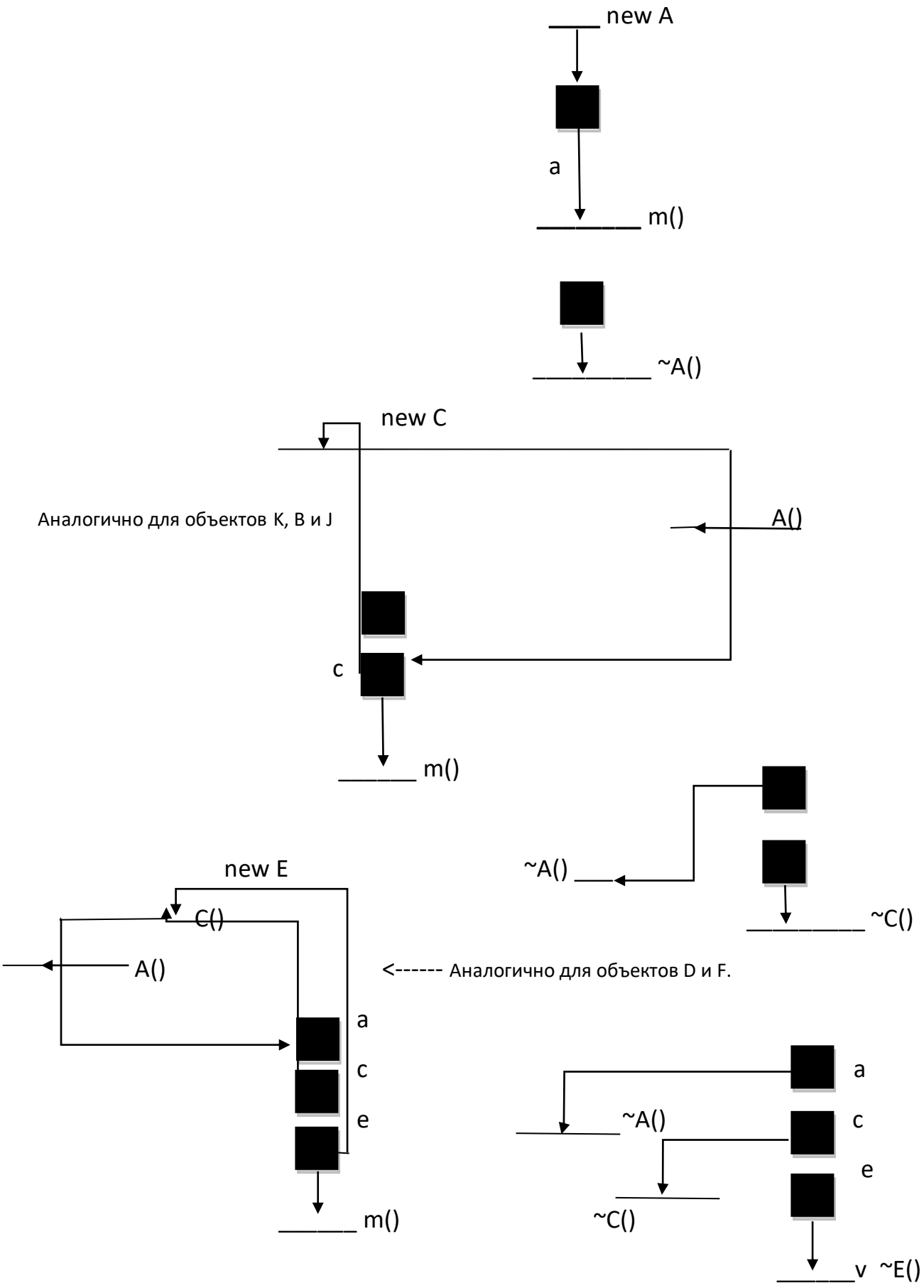


Рис. 1: диаграмма классов

Рис.2: диаграмма объектов



Текст программы:

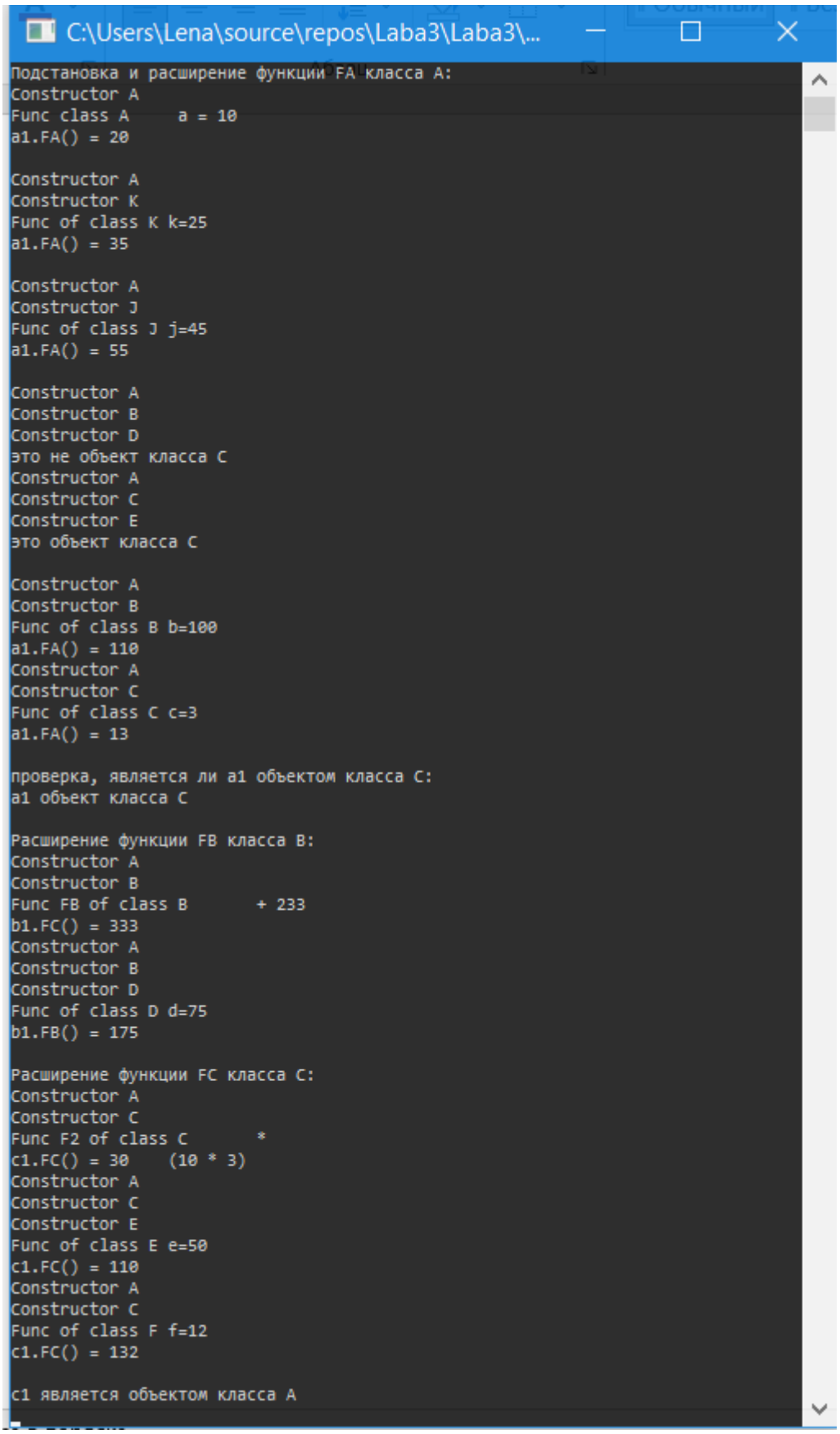
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3
{
    public class A
    {
        public A()
        {
            Console.WriteLine("Constructor A");
            this.varA = 10;
        }
        ~A() { }

        public virtual int FA()
        {
            Console.WriteLine("Func class A\t a = 10");
            return this.varA + this.varA;
        }
         protected int varA { get; set; }
    }

    public class K : A
    {
       public K()
       {
           Console.WriteLine("Constructor K");
           this.varK = 25;
       }
       ~K() { }

       public override int FA()
       {
           Console.WriteLine("Func of class K\tk=25");
           return this.varA + this.varK;
       }
       protected int varK { get; set; }
    }
    public class B : A
    {
       public B()
       {
           Console.WriteLine("Constructor B");
           this.varB = 100;
       }
       ~B() { }

       //расширение функции FA, наследуемую классом B из класса A

        public override int FA()
        {
           Console.WriteLine("Func of class B\tb=100");
           return this.varB + this.varA;
        }

        public virtual int FB()
        {
```

```
        Console.WriteLine("Func FB of class B\t + 233");
        return varB + 233;
    }
    protected int varB { get; set; }
}

public class C : A
{
    public C()
    {
        Console.WriteLine("Constructor C");
        this.varC = 3;
    }
    ~C() { }

    //расширение функции FA, наследуемую классом C из класса A

    public override int FA()
    {
        Console.WriteLine("Func of class C\tc=3");
        return this.varC + this.varA;
    }

    public virtual int FC()
    {
        Console.WriteLine("Func F2 of class C\t * ");
        return (this.varC * this.varA);
    }
    protected int varC { get; set; }
}

public class J : A
{
    public J()
    {
        Console.WriteLine("Constructor J");
        this.varJ = 45;
    }
    ~J() { }

    public override int FA()
    {
        Console.WriteLine("Func of class J\tj=45");
        return this.varA + this.varJ;
    }
    protected int varJ { get; set; }
}

public class D : B
{
    public D()
    {
        Console.WriteLine("Constructor D");
        this.varD = 75;
    }

    ~D() { }

    public override int FA()
    {
        return this.varA + this.varD;
    }
    public override int FB()
    {
        Console.WriteLine("Func of class D\td=75 ");
```

```
            return this.varD + this.varB;
        }

        protected int varD { get; set; }
    }

    class E : C
    {
       public E()
       {
           Console.WriteLine("Constructor E");
           this.varE = 50;
       }
       ~E() { }

       public override int FC()
       {
           Console.WriteLine("Func of class E\te=50");
           return (this.varC* this.varA* 2 + this.varE);
       }
        protected int varE { get; set; }
    }

    class F : C
    {
        public F()
        {
            Console.WriteLine("Constructor F");
            this.varF = 12;
        }
        ~F() { }

        public override int FC()
        {
            Console.WriteLine("Func of class F\tf=12");
            return (this.varC* this.varA* 4 + this.varF);
        }
        protected int varF { get; set; }
    }


class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Подстановка и расширение функции FA класса A: ");

        A a1 = new A();
        Console.WriteLine("a1.FA() = {0} ", a1.FA());
        Console.WriteLine();

        a1 = new K();
        Console.WriteLine("a1.FA() = {0}", a1.FA());
        Console.WriteLine();

        a1 = new J();
        Console.WriteLine("a1.FA() = {0}", a1.FA());
        Console.WriteLine();

        {
            A a = new D();

            if (a is C)
                Console.WriteLine("это объект класса C");  //определение типа,
который в объект подставлен
```

```
            else Console.WriteLine("это не объект класса C ");

            a = new E();

            if (a is C)
                Console.WriteLine("это объект класса C "); //определение типа,
который в объект подставлен
            else Console.WriteLine("это не объект класса C ");
        }

        Console.WriteLine();
        a1 = new B();
        Console.WriteLine("a1.FA() = {0}", a1.FA());
        //подстановка.
        a1 = new C();
        Console.WriteLine("a1.FA() = {0}", a1.FA());
        //подстановка.
        Console.WriteLine();
        Console.WriteLine("проверка, является ли a1 объектом класса C: ");
        //проверка, является ли a1 объектом класса C
        //какой объект в этой ссылке

        if (a1 is C)
            Console.WriteLine("a1 объект класса C ");
        else Console.WriteLine("a1 не является объектом класса C ");

        Console.WriteLine();

        Console.WriteLine("Расширение функции FB класса B: ");
        B b1 = new B();
        Console.WriteLine("b1.FC() = {0}", b1.FB());

        //подстановка объекта b1 в класс D
        b1 = new D();
        Console.WriteLine("b1.FB() = {0}", b1.FB());

        Console.WriteLine();

        Console.WriteLine("Расширение функции FC класса C: ");
        C c1 = new C();
        Console.WriteLine("c1.FC() = {0}\t(10 * 3) ", c1.FC());

        //подстановка объекта c1 в класс E
        c1 = new E();
        Console.WriteLine("c1.FC() = {0}", c1.FC());

        //подстановка объекта c1 в класс F
        c1 = new F();
        Console.WriteLine("c1.FC() = {0}", c1.FC());
        Console.WriteLine();

        if (c1 is A)
            Console.WriteLine("c1 является объектом класса A ");
        else Console.WriteLine("c1 не является объектом класса A ");


        Console.ReadKey();
    }
}

}
```

Результат работы программы:



Вывод: в этой программе используется метод подстановки и метод замещения. *Принцип подстановки:* вместо объекта суперкласса можно подставить объект подкласса. *Принцип замещения:* функцию суперкласса можно заменить функцией подкласса.

Ключевое слово *virtual* используется для изменения объявлений методов, свойств, индексаторов и событий и разрешения их переопределения в производном классе. Например, этот метод может быть переопределен любым наследующим его классом: модификатор override требуется для расширения или изменения абстрактной или виртуальной реализации унаследованного метода, свойства, индексатора или события.