**Лабораторная работа №3. Принцип подстановки.**

**Диаграмма Классов**:

A

Public

Private

Protected

Public

C

Protected

Private

Public

B

Protected

Private

Private

Public

J

Protected

Public

F

Protected

Private

Public

E

Protected

Private

Private

Public

K

Protected

**Диаграмма объектов:**

new A

a

m()

~A()

new C

Аналогично для объектов J и B.----------🡪

A()

a

c

m()

a

c

~A() ~C()

new F

<------------------Аналогично для объектов E и K.

C()

A()

a

c

f

m()

a

c

~C()

~A() f

v ~F()

**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 = 5;

}

~A() { }

public virtual int FA()

{

Console.WriteLine("Func class A\t a=5");

return this.varA + this.varA;

}

protected int varA { get; set; }

}

public class J : A

{

public J()

{

Console.WriteLine("Constructor J");

this.varJ = 20;

}

~J() { }

public override int FA()

{

Console.WriteLine("Func of class J\tj=20");

return this.varA + this.varJ;

}

protected int varJ { get; set; }

}

public class B : A

{

public B()

{

Console.WriteLine("Constructor B");

//this.varA = 10;

this.varB = 100;

}

~B() { }

//расширяю функцию FA, наследуемую классом B из класса А

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 +200");

return varB + 200;

}

protected int varB { get; set; }

}

public class C : A

{

public C()

{

Console.WriteLine("Constructor C");

//this.varA = 15;

this.varC = 2;

}

~C() { }

//расширяю функцию FA, наследуемую классом C из класса А

public override int FA()

{

Console.WriteLine("Func of class C\tc=2");

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; }

}

class E : C

{

public E()

{

Console.WriteLine("Constructor E");

this.varE = 45;

}

~E() { }

public override int FC()

{

Console.WriteLine("Func of class E\te=45");

return (this.varC \* this.varA \* 2 + this.varE);

}

protected int varE { get; set; }

}

class F : C

{

public F()

{

//Console.WriteLine("Constructor F");

this.varF = 11;

}

~F() { }

public override int FC()

{

Console.WriteLine("Func of class F\tf=11");

return (this.varC \* this.varA \* 4 + this.varF);

}

protected int varF { get; set; }

}

public class K : B

{

public K()

{

Console.WriteLine("Constructor K");

this.varK = 75;

}

~K() { }

public override int FA()

{

return this.varA + this.varK;

}

public override int FB()

{

Console.WriteLine("Func of class K\tk=75");

return this.varK + this.varB;

}

protected int varK { get; set; }

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Подстановка и расширение функции FA класса А:");

A a1 = new A();

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

a1 = new J();

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

Console.WriteLine();

{

A a = new K();

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 С:");

//проверка, явл-ся ли a1 объектом класса C

//какой объект в этой ссылке

if (a1 is C)

{

Console.WriteLine(" a1 объект класса C");

}

else Console.WriteLine("а1 не является объектом класса C");

Console.WriteLine();

Console.WriteLine("Расширение функции FB класса B:");

B b1 = new B();

b1 = new K();

Console.WriteLine("b1.FB() = {0}", b1.FB());

Console.WriteLine();

Console.WriteLine();

Console.WriteLine("Расширение функции FC класса C:");

C c1 = new C();

Console.WriteLine("c1.FC() = {0}\t(5\*2)", 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");

}

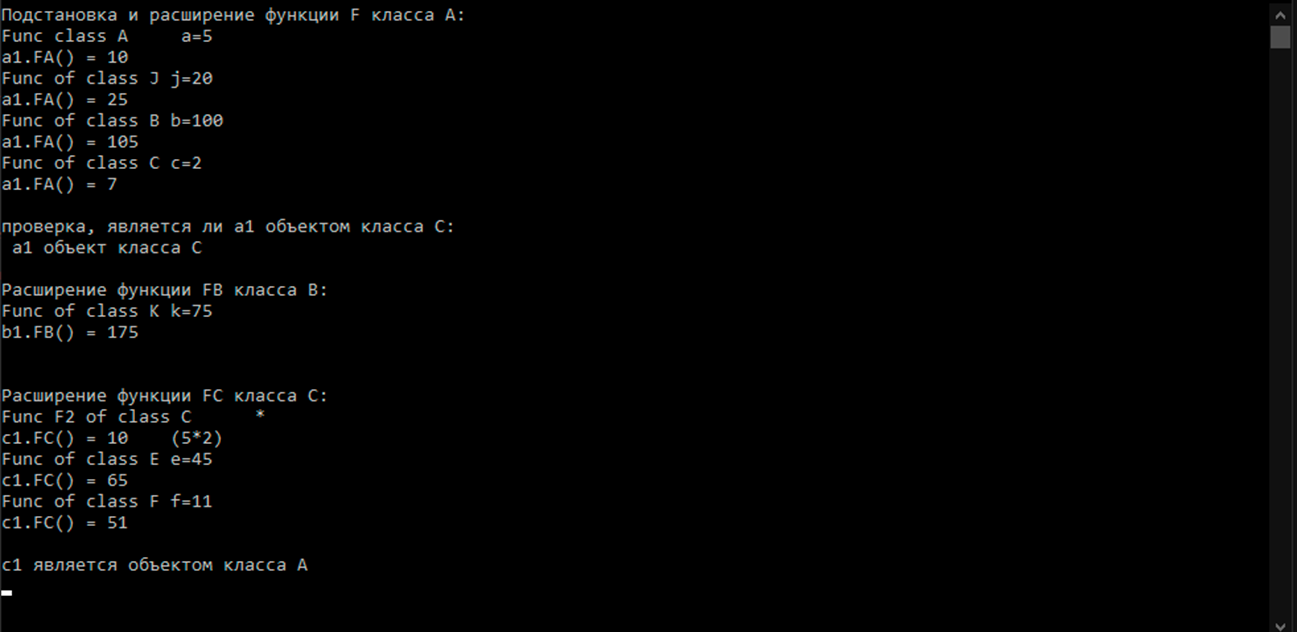
else Console.WriteLine("c1 не является объектом класса A");

Console.ReadKey();

}

}

}



**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.