**10 «МЕХАНИЗМ НАСЛЕДОВАНИЯ. ОТНОШЕНИЯ МЕЖДУ КЛАССАМИ».**

Задание 1. Составить программу с одним родительским классом и потомком. Все поля должны быть закрытыми. Базовый класс должен содержать конструкторы с параметрами, методы доступа к закрытым полям, вывод полей и указанный метод. Производный класс содержит дополнения и изменения, организовать вывод новых полей потомка, при этом имена методов совпадают с именами методов базового класса. Составить тестирующую программу с выдачей результатов. Создать объекты базового и производного типов. В программе должна выполняться проверка всех разработанных элементов класса.

Базовый класс: Квартира (поля: название, стоимость 1м 2 ,площадь) Метод: Стоимость квартиры. Потомок: Квартира в центре (поле название района) Изменения в потомках: Увеличить стоимость с учетом надбавки за расположение на 0.01 стоимости квартиры.

Листинг программы:

using System;

class ParentClass

{

private int parentField1;

private string parentField2;

public ParentClass(int field1, string field2)

{

parentField1 = field1;

parentField2 = field2;

}

public int GetParentField1()

{

return parentField1;

}

public string GetParentField2()

{

return parentField2;

}

public void DisplayParentFields()

{

Console.WriteLine("ParentField1: " + parentField1);

Console.WriteLine("ParentField2: " + parentField2);

}

public void ParentMethod()

{

Console.WriteLine("This is a method from the parent class.");

}

}

class ChildClass : ParentClass

{

private double childField;

public ChildClass(int field1, string field2, double field3) : base(field1, field2)

{

childField = field3;

}

public void DisplayChildField()

{

Console.WriteLine("ChildField: " + childField);

}

public new void ParentMethod()

{

Console.WriteLine("This is a method from the child class.");

}

}

class Program

{

static void Main(string[] args)

{

ParentClass parentObj = new ParentClass(10, "ParentField");

ChildClass childObj = new ChildClass(20, "ParentField", 30.5);

Console.WriteLine("Base Class:");

parentObj.DisplayParentFields();

Console.WriteLine("ParentField1: " + parentObj.GetParentField1());

Console.WriteLine("ParentField2: " + parentObj.GetParentField2());

parentObj.ParentMethod();

Console.WriteLine();

Console.WriteLine("Derived Class:");

childObj.DisplayParentFields();

childObj.DisplayChildField();

Console.WriteLine("ParentField1: " + childObj.GetParentField1());

Console.WriteLine("ParentField2: " + childObj.GetParentField2());

childObj.ParentMethod();

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Base Class:  ParentField1: 10  ParentField2: ParentField  ParentField1: 10  ParentField2: ParentField  This is a method from the parent class.  Derived Class:  ParentField1: 20  ParentField2: ParentField  ChildField: 30,5  ParentField1: 20  ParentField2: ParentField  This is a method from the child class. |

Анализ результатов:

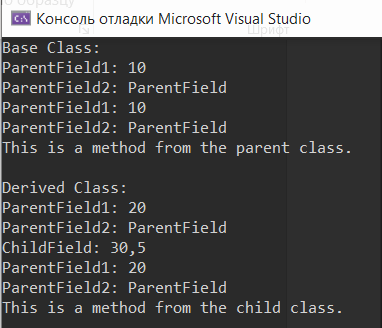


Рисунок 1.1 – Результат работы программы

Задание 2. В рамках консольного приложения разработать класс В- наследник класса А (класс А с полями а и b и свойством с. Свойство – значение выражения над полями а и b (выражение и типы полей – см.вариант в таблице 1). Поля инициализировать при объявлении класса. Конструктор оставить по умолчанию. Проследить, чтобы поля а и b напрямую в других классах были недоступны.) с полем d и свойством с2. Свойство с2 – результат вычисления выражения над полями a, b, d. В теле свойства использовать управляющий оператор (см. вариант в таблице 1) Для класса В определить 2 конструктора: один –наследуется от конструктора класса А, второй –собственный. В теле программы создать объекты классов А и В, продемонстрировав работу всех конструкторов. Вывести значения свойства на экран.

Листинг программы:

using System;

class A

{

protected int a = 5;

protected int b = 10;

public int C

{

get { return a + b; }

}

}

class B : A

{

private int d = 20;

public B() { }

public B(int a, int b, int d)

{

this.a = a;

this.b = b;

this.d = d;

}

public int C2

{

get

{

if (a > b)

return a + b + d;

else

return a - b + d;

}

}

}

class Program

{

static void Main(string[] args)

{

A objA = new A();

B objBDefault = new B();

B objBCustom = new B(7, 3, 25);

Console.WriteLine("Значение свойства C для объекта класса A: " + objA.C);

Console.WriteLine("Значение свойства C для объекта класса B (конструктор по умолчанию): " + objBDefault.C);

Console.WriteLine("Значение свойства C для объекта класса B (собственный конструктор): " + objBCustom.C);

Console.WriteLine("Значение свойства C2 для объекта класса B (конструктор по умолчанию): " + objBDefault.C2);

Console.WriteLine("Значение свойства C2 для объекта класса B (собственный конструктор): " + objBCustom.C2);

}

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Значение свойства C для объекта класса A: 15  Значение свойства C для объекта класса B (конструктор по умолчанию): 15  Значение свойства C для объекта класса B (собственный конструктор): 10  Значение свойства C2 для объекта класса B (конструктор по умолчанию): 15  Значение свойства C2 для объекта класса B (собственный конструктор): 35 |

Анализ результатов:

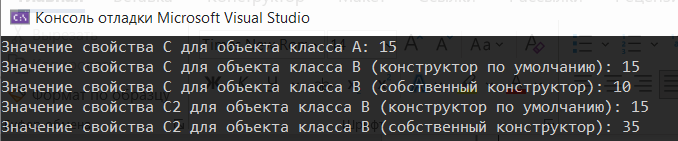


Рисунок 1.2 – Результат работы программы