**10.** **Механизм наследования. Отношения между классами. Делегаты**

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: Поле (поля: название, r – вес посеянных семян на единицу площади) Метод: Количество урожая с единицы площади: k\*r, где k – коэффициент. Потомок: Картофельное (поле S – площадь поля) Изменения в потомках: Найти урожай со всего поля.

Листинг программы:

Program:

using z1.Model;

Field field = new Field("Базовый класс", 3.4);

Console.WriteLine("Базовый класс: ");

Console.WriteLine($"Название: {field.Name}");

Console.WriteLine($"R: {field.R}");

PotatoField potatoField = new PotatoField("Наследник Field", 4.4, 20);

Console.WriteLine("Наследник класса Field: ");

Console.WriteLine($"Название: {potatoField.Name}");

Console.WriteLine($"R: {potatoField.R}");

Console.WriteLine($"S: {potatoField.Square}");

Console.WriteLine($"Количество урожая с единицы площади(базовый класс):" +

$" {field.GetCountHarvestPerUnitSquare(2)}");

Console.WriteLine($"Урожай со всего поля(класс наследник):" +

$" {potatoField.GetCountHarvest(2)}");

Field:

namespace z1.Model

{

internal class Field

{

private string \_name;

private double \_r;

public string Name { get => \_name; }

public double R { get => \_r; }

public Field(string name, double r)

{

\_name = name;

\_r = r;

}

public double GetCountHarvestPerUnitSquare(double k) => \_r \*k;

}

}

PotatoField:

namespace z1.Model

{

internal class PotatoField : Field

{

private double \_square;

public double Square { get => \_square; }

public PotatoField(string name, double r, double square) : base(name, r)

{

\_square = square;

}

public double GetCountHarvest(double k)=> GetCountHarvestPerUnitSquare(k) \* \_square;

}

}

Таблица 10.1 – Входные и выходные данные программы задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Класс Field \_name = «Базовый класс» \_r = 3.4  Класс PotatoField \_name = «Наследник Field»\_\_r = 4.4\_square = 20 | Базовый класс:  Название: Базовый класс  R: 3,4  Наследник класса Field:  Название: Наследник Field  R: 4,4  S: 20  Количество урожая с единицы площади(базовый класс): 6,8  Урожай со всего поля(класс наследник): 176 |

Анализ результатов:

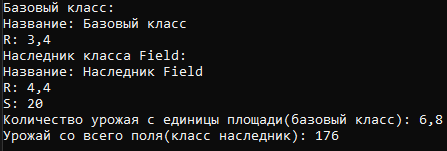


Рисунок 10.1 – Результат выполнения программы задание №1

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:

Program:

using z2.Model;

A a = new A();

Console.WriteLine($"Свойства C у класса A {a.C}");

B b = new B(3, 4, 2);

Console.WriteLine($"Свойства C у класса B {b.C}");

Console.WriteLine($"Свойства C2 у класса B {b.C2}");

A:

namespace z2.Model

{

internal class A

{

protected int \_a;

protected int \_b;

public int C

{

get

{

int a = \_a;

int b = \_b;

do

{

a--;

b++;

} while (a > 0);

return b;

}

}

public A()

{

\_a = 3;

\_b = 5;

}

}

}

B:

namespace z2.Model

{

internal class B : A

{

private int \_d;

public int C2 { get =>\_a + \_b + \_d; }

public B() : base()

{

}

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

{

\_a = a;

\_b = b;

\_d = d;

}

}

}

Таблица 10.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Класс A \_a = 3 \_b = 5  Класс B \_a = 3\_b= 4\_d = 2 | Свойства C у класса A: 8  Свойства C у класса B: 7  Свойства C2 у класса B: 9 |

Анализ результатов:



Рисунок 10.2 – Результат выполнения программы задание №2