1. **Механизм наследования. Отношения между классами**

Задание 1. Наследование

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 объект базового и производного типов. В программе должна выполняться проверка всех разработанных элементов класса.

Базовый класс: Товар (поле – количество, цена.) Метод: Стоимость товара. Потомок: Фломастеры (поле – название, сорт – s) Изменения в потомках: Изменить стоимость фломастеров с учетом сорта в 1/√s раз.

Листинг программы:

namespace Task\_1

{

class Program

{

public static void Main(string[] args)

{

try

{

Marker markers = new Marker();

markers.ProductCost();

markers.GetInfo();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

class Product

{

private int count;

private double cost;

private double result;

public int Count

{

get { return count; }

set { count = value; }

}

public double Cost

{

get { return cost; }

set { cost = value; }

}

public double Result

{

get { return result; }

set { result = value; }

}

public Product()

{

this.count = 0;

this.cost = 0;

this.result = 0;

Input();

}

public virtual void GetInfo()

{

Console.WriteLine($"Кол-во: {count}\n" +

$"Цена: {cost}");

}

public virtual void ProductCost()

{

result = cost \* count;

}

public void Input()

{

Console.Write("Введите цену: ");

cost = double.Parse(Console.ReadLine());

Console.Write("Введите количество: ");

count = int.Parse(Console.ReadLine());

}

}

class Marker : Product

{

private string name;

private int sort;

public Marker() : base()

{

this.name = "Фломастерс";

this.sort = 1;

Input();

}

public override void ProductCost()

{

Result = Count \* Cost \* (1 / Math.Sqrt(sort));

}

public void GetInfo()

{

Console.WriteLine($"Кол-во: {Count}\n" +

$"Цена: {Cost}\n" +

$"Название фломастеров: {name}\n" +

$"Сорт: {sort}\n" +

$"Новая стоимость: {Result}");

}

public void Input()

{

Console.Write("Введите название: ");

name = Console.ReadLine();

Console.Write("Введите сорт: ");

sort = int.Parse(Console.ReadLine());

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 23 10 Flaster 2 | 10 23 Flaster 2 162,634556… |

Анализ результатов:

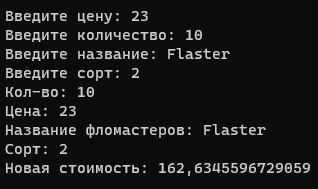


Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

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. Управляющий оператор Switch

Листинг программы:

using Task\_2;

namespace Task\_2

{

class Program

{

public static void Main(string[] args)

{

try

{

A objectMy = new A();

Console.WriteLine($"Класс A без параметров\n{objectMy.c}");

Console.WriteLine($"Класс A с параметрами");

Console.Write($"Введите значение: ");

int numOne = int.Parse(Console.ReadLine());

Console.Write($"Введите значение: ");

int numTwo = int.Parse(Console.ReadLine());

A objectMy1 = new A(numOne, numTwo);

Console.WriteLine(objectMy1.c);

B objectMy2 = new B();

Console.WriteLine($"Класс B без параметров\n{objectMy2.c2}");

Console.WriteLine($"Класс B с параметрами");

Console.Write("Введите действие (сложение-1 | вычитание-2): ");

int operation = int.Parse(Console.ReadLine());

Console.Write($"Введите значение: ");

int numThree = int.Parse(Console.ReadLine());

B objectMy3 = new B(operation, numOne, numTwo, numThree);

Console.WriteLine(objectMy3.c2);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

class A

{

protected int a;

protected int b;

public int c

{

get

{

if (a > b)

{

return a - b;

}

else

{

return b - a;

}

}

set { a = value; b = value; }

}

public A()

{

a = 1;

b = 1;

}

public A(int a, int b)

{

this.a = a;

this.b = b;

}

}

class B : A

{

private int d;

private int operation;

public int c2

{

get

{

switch (operation)

{

case 1:

return a + b + d;

case 2:

return a - b - d;

default:

return 0;

}

}

set

{ a = value; b = value; d = value; }

}

public B()

{

a = 1;

b = 1;

d = 1;

}

public B(int operation, int a, int b, int d) : base(a, b)

{

this.d = d;

this.operation = operation;

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 5 1 4 | 0 3 0 11 |

Анализ результатов:

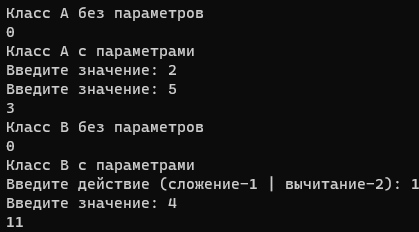


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка