**9 Интерфейсы**

Задание №1. Интерфейсы Ix, Iy, Iz, содержат объявления методов с одной и той же сигнатурой, эти интерфейсы наследуются в классе TestClass, содержащий член w типа параметр и реализуются так, как задано в варианте. В каждом методе задать вывод результата. Задаётся параметр – double, IxF0, IxF1 возвращают Log(w), F0 и F1 возвращают: неявная реализация – 2 / w, явная реализация Iz – w3.

Листинг программы:

TestClass testClass = new TestClass();

TestClass testClass1 = new TestClass(5);

Console.WriteLine($"IxF0 - "); testClass.IxF0(5);

Console.WriteLine($"IxF1 -"); testClass1.IxF1();

Console.WriteLine("неявную НЕОДНОЗНАЧНУЮ реализацию");

testClass.F0(5);

testClass1.F1();

Console.WriteLine("C явным приведением к типу интерфейса");

(testClass as Iy).F0(7);

(testClass1 as Iz).F1();

Console.WriteLine("==========Ix==========");

Ix ix = testClass1;

ix.IxF0(5);

ix.IxF1();

Console.WriteLine("==========Iy==========");

Iy iy = testClass1;

iy.F0(5);

iy.F1();

Console.WriteLine("==========Iz==========");

Iz iz = testClass1;

iz.F0(5);

iz.F1();

interface Ix

{

void IxF0(double w);

void IxF1();

}

interface Iy

{

void F0(double w);

void F1();

}

interface Iz

{

void F0(double w);

void F1();

}

class TestClass : Ix, Iy, Iz

{

double w;

public TestClass()

{

w = 125;

}

public TestClass(double W)

{

w = W;

}

public void IxF0(double w)

{

Console.WriteLine(Math.Log(w));

}

public void IxF1()

{

Console.WriteLine(Math.Log(w));

}

public void F0(double w)

{

Console.WriteLine(2 / w);

}

public void F1()

{

Console.WriteLine(2 / w);

}

void Iz.F0(double w)

{

Console.WriteLine(w + 2);

}

void Iz.F1()

{

Console.WriteLine(w + 2);

}

}

Таблица 9.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5, 6, 5, 5, 5 | 1.6094, 1.6094, 0.4, 0.4, 0.28571, 7 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 9.1.

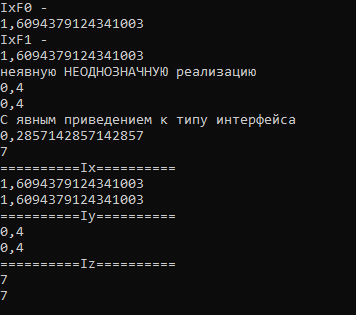


Рисунок 9.1 – Результат работы программы

Источник: собственная разработка

Задание №2. Описать класс «товар», содержащий следующие закрытые поля:

название товара;

название магазина, в котором продается товар;

стоимость товара в рублях.

Предусмотреть свойства для получения состояния объекта.

Описать класс «склад», содержащий закрытый массив товаров. Обеспечить следующие возможности:

вывод информации о товаре по номеру с помощью индекса;

сортировку товаров по названию магазина, по наименованию и по цене;

Написать программу, демонстрирующую все разработанные элементы классов.

Листинг программы:

class Stock

{

List<Product> products = new List<Product>();

public Stock(List<Product> porductsList)

{

this.products = porductsList;

}

public void WritePlane(int index)

{

if (index > products.Count - 1)

throw new Exception("Продукта не существует");

Console.WriteLine("Название продукта: " + products[index].productName);

Console.WriteLine("Название магазина: " + products[index].shopName);

Console.WriteLine("Стоимость товара в рублях: " + products[index].cost);

}

public void ProductAdd(Product addProduct)

{

products.Add(addProduct);

}

public void AddProductsRange(params Product[] Product)

{

products.AddRange(Product);

}

}

class Product : IComparable<Product>

{

private string \_productName;

private string \_shopName;

private decimal \_cost;

public Product(string productName, string shopName, decimal cost)

{

this.\_productName = productName;

this.\_shopName = shopName;

this.\_cost = cost; }

public string productName { get { return \_productName; } set { \_productName = value; } }

public string shopName { get { return \_shopName; } set { \_shopName = value; } }

public decimal cost { get { return \_cost; } set { \_cost = value; } }

public int CompareTo(Product obj)

{

if (shopName.CompareTo(obj.shopName)!=0)

{

return shopName.CompareTo(obj.shopName);

}

if (productName.CompareTo(obj.productName)!=0)

{

return productName.CompareTo(obj.productName);

}

return cost.CompareTo(obj.cost);

}

}

internal class Program

{

static void Main(string[] args)

{

List<Product> products = new List<Product>();

products.Add(new Product("Юбка", "ExPz", 25));

products.Add(new Product("Шорты", "ExPу", 55));

products.Add(new Product("Юбка", "ExPу", 11));

products.Add(new Product("Бита", "ExPzz", 99));

products.Sort();

Stock product = new Stock(products);

for (int i = 0; i < products.Count; i++)

{

product.WritePlane(i);

}

Console.ReadLine();

}

}

Таблица 9.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| ExPz, Юбка, Шорты, Бита | List<Product> |

Источник: собственная разработка

Анализ результатов представлен на рисунке 9.2.

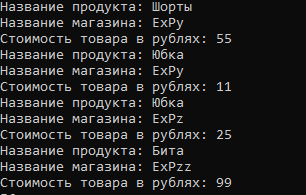


Рисунок 9.2 – Результат работы программы

Источник: собственная разработка