1. **Интерфейсы**

Задание 1. Интерфейсы (см. пример в папке).

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.

Рассмотреть случай

неявной реализации интерфейсов;

явной реализации интерфейса Iz.

В программе должна выполняться:

неявная неоднозначная реализация методов интерфейсов Iy и Iz;

вызов функций с явным приведением к типу интерфейса;

вызов метода для объекта посредством интерфейсной ссылки.

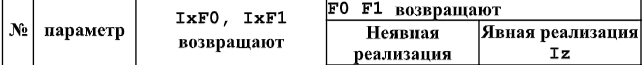




Рисунок 1.1 – Задание

Источник: собственная разработка

Листинг программы:

using Task\_1;

namespace Task\_1

{

// Пара интерфейсов, содержащих объявления одноимённых методов

// с одной и той же сигнатурой.

interface Ix

{

int IxF0(int num);

int IxF1();

}

interface Iy

{

int F0(int num);

int F1();

}

interface Iz

{

int F0(int num);

int F1();

}

class TestClass : Ix, Iy, Iz

{

public int wValue;

public TestClass()

{

wValue = 12;

}

public TestClass(int value)

{

wValue = value;

}

// Реализация неявная и однозначная

public int IxF0(int num)

{

wValue = 7 \* num;

return wValue;

}

public int IxF1()

{

wValue = 7 \* wValue;

return wValue;

}

// Неявная неоднозначная реализация интерфейсов

public int F0(int num)

{

wValue = num \* 8;

return wValue;

}

public int F1()

{

wValue = wValue \* 8;

return wValue;

}

// Явная непосредственная реализация интерфейса

int Iz.F0(int num)

{

wValue = 6 + num;

return wValue;

}

int Iz.F1()

{

wValue = 6 + wValue;

return wValue;

}

}

}

class Program

{

static void Main(string[] args)

{

Console.Write("Введите число для функции F1: ");

int numF1 = int.Parse(Console.ReadLine());

Console.Write("Введите число для функции F0: ");

int numF0 = int.Parse(Console.ReadLine());

TestClass tc = new TestClass();

TestClass tcNum = new TestClass(numF1);

// Неявная неоднозначная реализация интерфейсов Iy и Iz

Console.WriteLine($"Неявная неоднозначная реализация интерфейсов\n" +

$"Iy и Iz = {tc.F0(numF0)}\n" +

$"Iy и Iz = {tcNum.F1()}");

// Явное приведение к типу интерфейса

Console.WriteLine($"Явное приведение к типу интерфейса\n" +

$"Iy = {(tc as Iy).F0(numF0)}\n" +

$"Iy = {(tcNum as Iy).F1()}\n" +

$"Iz = {(tc as Iz).F0(numF0)} \n" +

$"Iz = {(tcNum as Iz).F1()}");

// Ссылки различных типов интерфейсов

Console.WriteLine("Ссылки различных типов интерфейсов");

Ix ix = tcNum;

Console.WriteLine($"IxF0 = {ix.IxF0(numF0)}\n" +

$"IxF1 = {ix.IxF1()}");

Iy iy = tcNum;

Console.WriteLine($"IyF0 = {iy.F0(numF0)}\n" +

$"IyF1 = {iy.F1()}");

Iz iz = tcNum;

Console.WriteLine($"IxF0 = {iz.F0(numF0)}\n" +

$"IxF1 = {iz.F1()}");

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 18 | 144 40 144 320 24 326 126 882 144 1152 24 30 |

Анализ результатов:

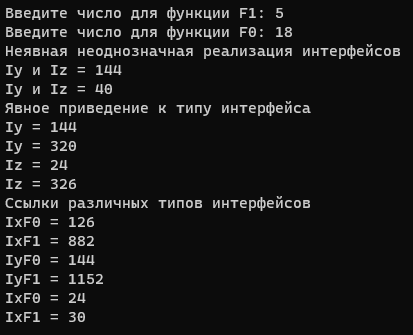


Рисунок 1.2 – Результат работы программы

Источник: собственная разработка

Задание 2. Выполнить задания, используя для хранения экземпляров разработанных классов массивы. Во всех классах реализовать интерфейс IComparable и перегрузить операции отношения для реализации значимой семантики сравнения объектов по какому-либо полю на усмотрение студента.

Описать класс для работы с одномерным массивом целых чисел (вектором). Обеспечить следующие возможности:

задание произвольных целых границ индексов при создании объекта;

обращение к отдельному элементу массива с контролем выхода за пределы массива;

выполнение операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов;

вывод на экран элемента массива по заданному индексу и всего массива.

Написать программу, демонстрирующую. все разработанные элементы класса.

Листинг программы:

namespace MainProgram

{

class Program

{

public static void Main(string[] args)

{

try

{

Massiv mas = new Massiv();

mas.list = mas.GetCreateArray(GetSizeArray());

mas.list = mas.GetCreateArray(GetSizeArray());

Console.WriteLine($"Перечень массивов");

PrintListArray(mas.list);

Console.WriteLine("Введите номер массива и его значение к которому хотите обратится");

Console.Write("Номер массива:");

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

Console.Write("Значение:");

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

Console.WriteLine($"Вы обратились к элементу под индексом: {mas.GetElementIndex(indexArray, number)}");

Console.Write($"Выберите массивы для операций\n" +

$"Введите номер массива в списке: ");

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

Console.Write("Введите номер массива в списке: ");

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

Console.WriteLine("Сложение массивов");

PrintArray(mas.GetResultOperationSum(mas.list[one], mas.list[two]));

Console.WriteLine("Вычитание массивов");

PrintArray(mas.GetResultOperationSub(mas.list[one], mas.list[two]));

Console.Write($"Выберите массив\n" +

$"Введите номер массива в списке: ");

int[] array = mas.GetArray(mas.list, int.Parse(Console.ReadLine()));

Console.Write("Выберите номер элемента массива: ");

Console.WriteLine($"Вы выбрали: {mas.GetElement(array, int.Parse(Console.ReadLine()))}");

Console.Write($"Из массива: ");

PrintArray(array);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static int GetSizeArray()

{

try

{

Console.Write("Введите размерность массива: ");

return int.Parse(Console.ReadLine());

}

catch (Exception e)

{

Console.WriteLine(e.Message);

return -1;

}

}

public static void PrintArray(int[] array)

{

try

{

for (int i = 0; i < array.Length; i++)

{

Console.Write($"{array[i]} ");

}

Console.WriteLine();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static void PrintListArray(List<int[]> list)

{

try

{

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

{

int[] array = list[i];

Console.WriteLine($"{i} массив");

PrintArray(array);

}

Console.WriteLine();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

class Massiv : IComparable

{

public List<int[]> list;

public Massiv()

{

list = new List<int[]>();

}

public List<int[]> GetCreateArray(int Length)

{

try

{

int[] array = new int[Length];

Random R = new Random();

for (int i = 0; i < Length; i++)

{

array[i] = R.Next(100);

}

list.Add(array);

return list;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

return list;

}

}

public int[] GetArray(List<int[]> list, int index)

{

try

{

return list[index];

}

catch (Exception e)

{

Console.WriteLine(e.Message);

return new int[] { 0 };

}

}

public int GetElement(int[] array, int index)

{

try

{

return array[index];

}

catch (Exception e)

{

Console.WriteLine(e.Message);

return -1;

}

}

public int GetElementIndex(int indexArray, int num)

{

try

{

int[] array = GetArray(list, indexArray);

for (int i = 0; i < array.Length; i++)

{

if (array[i] == num)

{

return i;

}

}

return -1;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

return -1;

}

}

public int[] GetResultOperationSum(int[] ferstArray, int[] secondArray)

{

if (GetBoolEqualLengthArray(ferstArray, secondArray))

{

int length = ferstArray.Length;

int[] resultArray = new int[length];

for (int i = 0; i < length; i++)

{

resultArray[i] = ferstArray[i] + secondArray[i];

}

return resultArray;

}

else

{

return new int[] { 0 };

}

}

public int[] GetResultOperationSub(int[] ferstArray, int[] secondArray)

{

if (GetBoolEqualLengthArray(ferstArray, secondArray))

{

int length = ferstArray.Length;

int[] resultArray = new int[length];

for (int i = 0; i < length; i++)

{

resultArray[i] = ferstArray[i] - secondArray[i];

}

return resultArray;

}

else

{

return new int[] { 0 };

}

}

public bool GetBoolEqualLengthArray(int[] ferstArray, int[] secondArray)

{

if (ferstArray.Length == secondArray.Length) return true;

else return false;

}

public int CompareTo(object? other)

{

throw new NotImplementedException();

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 5 1 93 1 0 0 2 | 0 (83 11 54 10 94)  1 (29 61 42 93 41)  3  112 72 96 103 135  -54 50 -12 83 -53  54 (83 11 54 10 94) |

Анализ результатов:

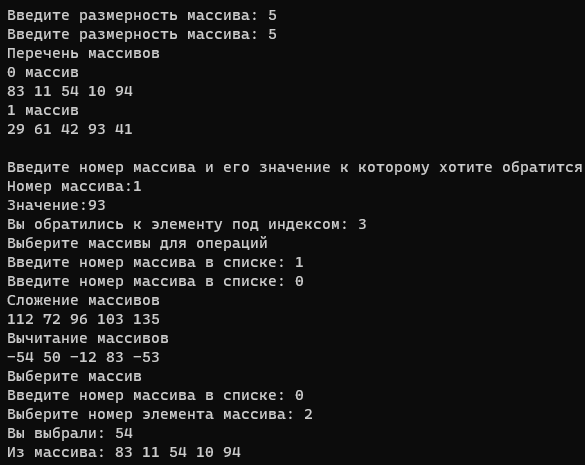


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка