**9.Интерфейсы**

Задание №1 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.

Функции для решения задачи изображены на рисунке 9.1



Рисунок 9.1- функции для решения задачи

Листинг программы:

interface Ix

{ void IxF0(double параметр);

void IxF1();

}

interface Iy

{ void F0(double параметр);

void F1();

}

interface Iz

{ void F0(double параметр);

void F1();

}

class TestClass : Ix, Iy, Iz

{ public double w;

public void F0(double w)

{ Console.WriteLine("Результат неявной реализации метода F0: " + 15 / w);

}

public void F1()

{ Console.WriteLine("Результат неявной реализации метода F1");

}

public void IxF0(double w)

{ Console.WriteLine("Результат неявной реализации метода IxF0: " + (2 \* w - 3));

}

public void IxF1()

{Console.WriteLine("Результат неявной реализации метода IxF1");

}

void Iz.F0(double w)

{Console.WriteLine("Результат явной реализации метода F0: " + (Math.Pow(w,2) - w));

}

void Iz.F1()

{Console.WriteLine("Результат явной реализации метода F1");

class Program

{static void Main(string[] args)

{TestClass obj = new TestClass()

obj.w = 453.0;

Ix objIx = obj;

objIx.IxF0(obj.w);

objIx.IxF1();

Iy objIy = obj;

objIy.F0(obj.w);

objIy.F1();

Iz objIz = obj;

objIz.F0(obj.w);

objIz.F1();

IxF0((double)obj.w);

static void IxF0(double параметр)

{TestClass obj = new TestClass();

Ix objIx = obj;

objIx.IxF0(параметр);

}

}

Таблица 9.1 – Входные и выходные данных задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Результат неявной реализации метода 903  Результат явной реализации : 204756 |

Анализ результатов:

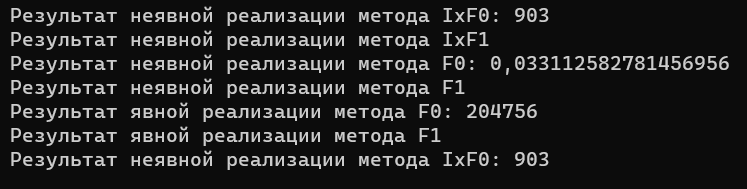


Рисунок 9.2 – Результат работы программы из задания №1

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

название пункта назначения;

шестизначный номер рейса;

время отправления.

Предусмотреть свойства для получения состояния объекта. Описать класс «аэропорт», содержащий закрытый массив самолетов. Обеспечить следующие возможности:

вывод информации о самолете по номеру рейса с помощью индекса;

сортировку самолетов по названию номеру рейса.

Листинг программы:

Airport airport = new Airport(Provider.GetAiroplants());

var airplanes = airport.Airplanes;

Console.WriteLine("Полная информация о самолетах :");

ConsoleUI.Show(airplanes);

Console.WriteLine();

Console.WriteLine("Выберите сортировку: \n1.По пунктам назначения.\n2.По номеру рейса.\n3.По времени.");

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

switch (sort)

{case 1:

Console.WriteLine("Информация посел сортировки по пунктам назначения ");

airplanes.Sort();

ConsoleUI.Show(airplanes);

break;

case 2:

Console.WriteLine("Информация после сорировки по номеру рейса");

airplanes.Sort(new AirplainDepartureTimeComparer());

ConsoleUI.Show(airplanes);

break;

case 3:

Console.WriteLine("Информация после сортироваки по времемени");

airplanes.Sort((f, s) => f.DestinationName.CompareTo(s.DestinationName));

ConsoleUI.Show(airplanes);

break;

default:

break;

}

Console.WriteLine();

Console.WriteLine("Введите пункт назначения");

string destinationName = Console.ReadLine();

Console.WriteLine("Введите номер рейса");

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

Console.WriteLine("Введите время оправления");

TimeOnly departureTime = TimeOnly.Parse(Console.ReadLine());

Console.WriteLine();

Console.WriteLine("Добавления рейса");

var airplain = new Airplane(destinationName, flightNumber, departureTime);

airport.Add(airplain);

ConsoleUI.Show(airplanes);

Console.WriteLine();

Console.WriteLine("Введите номер рейса самолета, о котором хотите знать информацию: ");

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

var foundedAirplan = airplanes.Where(a => a.FlightNumber == num).ToList();

if (foundedAirplan != null)

{ ConsoleUI.Show(foundedAirplan);

}

else

{ Console.WriteLine("Самолета с таким рейсом не существует");

}

Console.WriteLine("Введите время отправления самолета, о котором хотите знать информацию: ");

var time = TimeOnly.Parse(Console.ReadLine());

var foundedAirplan2 = airplanes.Where(a => a.DepartureTime == time).ToList();

if (foundedAirplan2 != null)

{ConsoleUI.Show(foundedAirplan2);

}

else

{Console.WriteLine("Самолета с таким рейсом не существует");

}

internal class Airport : ICollection<Airplane>

{private readonly List<Airplane> \_airplanes;

public Airport(List<Airplane> airplanes)

{ \_airplanes = airplanes;

}

public List<Airplane> Airplanes=>\_airplanes;

public Airport()

{IList<Airplane> airplanes = new List<Airplane>();

}

public int Count => \_airplanes.Count ;

public bool IsReadOnly => true;

public void Add(Airplane plane)

{\_airplanes.Add(plane);

}

public void Clear()

{\_airplanes.Clear();

}

public bool Contains(Airplane item)

{

return \_airplanes.Contains(item) ;

}

public void CopyTo(Airplane[] array, int arrayIndex)

{\_airplanes.CopyTo(array, arrayIndex);

}

public IEnumerator<Airplane> GetEnumerator()

{return \_airplanes.GetEnumerator();

}

public bool Remove(Airplane item)

{return \_airplanes.Remove(item);

}

IEnumerator IEnumerable.GetEnumerator()

{return \_airplanes.GetEnumerator();

internal class Airplane:IComparable<Airplane>

{private string \_destinationName

private int \_flightNumber;

private TimeOnly \_departureTime;

public Airplane()

{

}

public Airplane(string destinationName, int flightNumber, TimeOnly departureTime)

{\_destinationName = destinationName;

\_flightNumber = flightNumber;

\_departureTime = departureTime;

}

public string DestinationName

{ get { return \_destinationName; }

set { \_destinationName = value; }

}

public int FlightNumber

{ get { return \_flightNumber; }

set { \_flightNumber = value; }

}

public TimeOnly DepartureTime

{ get { return \_departureTime; }

set { \_departureTime = value; }

}

public int CompareTo(Airplane? other)

{ var flightNumerComparer = \_flightNumber.CompareTo(other.FlightNumber);

if (flightNumerComparer != 0)

{return flightNumerComparer;

}

var departureTimeComparer = \_departureTime.CompareTo(other.DepartureTime);

if (departureTimeComparer != 0)

{ return departureTimeComparer;

}

return \_destinationName.CompareTo(other.DestinationName);

}

public override string ToString()

{ return $"{DestinationName} | {FlightNumber} | {DepartureTime.ToShortTimeString()} ";

}

}

Таблица 9.2 – Входные и выходные данных задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Полная информация о самолетах  Минск 581558 20:30  Стамбул 562545 19:45  Лондон 526485 21:00 |

Анализ результатов:

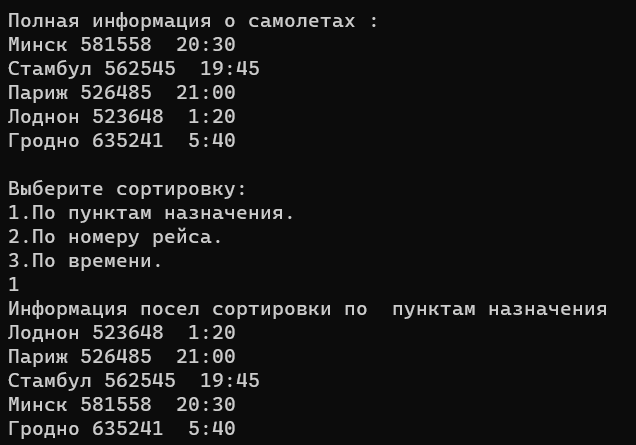


Рисунок 9.3 – Результат работы программы из задания №2