**9.** **Интерфейсы**

Задание №1 Интерфейсы Ix, Iy, Iz, содержат объявления методов с одной и той же сигнатурой следующим образом

interface Ix

{

void IxF0(параметр);

void IxF1();

}

interface Iy

{

void F0(параметр);

void F1();

}

interface Iz

{

void F0(параметр);

void F1();

}

Эти интерфейсы наследуются в классе TestClass, содержащий член w типа параметр и реализуются так, как задано в варианте. В каждом методе задать вывод результата.

Листинг программы:

Ix:

internal interface Ix

{

public string IxF0(string str);

public string IxF1();

}

Iy:

internal interface Iy

{

public string F0(string str);

public string F1();

}

Iz:

public interface Iz

{

public string F0(string str);

public string F1();

}

TestClass:

public class TestClass : Ix, Iy, Iz

{

private string \_str;

public TestClass(string str)

{

\_str = str;

}

public string IxF0(string str)=> str.Remove(str.Length - 2);

public string IxF1()=> \_str.Remove(\_str.Length - 2);

public string F0(string str)=> str.Remove(0, 2);

public string F1()=> \_str.Remove(0, 2);

string Iz.F0(string str)

{

int indexSpace = str.IndexOf(' ');

return str.Remove(indexSpace, 1).Insert(indexSpace, "-");

}

string Iz.F1()

{

int indexSpace = \_str.IndexOf(' ');

return \_str.Remove(indexSpace, 1).Insert(indexSpace, "-");

}

}

Program:

using z1.Interface;

using z1.Model;

TestClass testClass = new TestClass("Тест строчка стол");

Console.WriteLine(testClass.IxF0("Вторая тест строчка стол"));

Console.WriteLine(testClass.IxF1());

Console.WriteLine(testClass.F0("Вторая тест строчка стол"));

Console.WriteLine(testClass.F1());

TestClass testClass2 = new TestClass("Тест строчка стол");

Console.WriteLine((testClass2 as Iz).F0("Вторая тест строчка стол"));

Console.WriteLine((testClass2 as Iz).F1());

Таблица 9.1 – Входные и выходные данные программы задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Тест строчка стол  Вторая тест строчка стол | Вторая тест строчка ст  Тест строчка ст  орая тест строчка стол  ст строчка стол  Вторая-тест строчка стол  Тест-строчка стол |

Анализ результатов:

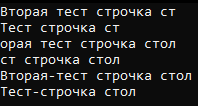


Рисунок 9.1 – Результат выполнения программы задание №1

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 «поезд», содержащий следующие закрытые поля:

- название пункта назначения;

- номер поезда (может содержать буквы и цифры);

- время отправления.

Предусмотреть свойства для получения состояния объекта.

Описать класс «вокзал», содержащий закрытый массив поездов.

Обеспечить следующие возможности:

- вывод информации о поезде по номеру с помощью индекса;

-вывод информации о поездах, отправляющихся после введенного с клавиатуры времени;

- перегруженную операцию сравнения, выполняющую сравнение времени отправления двух поездов;

- вывод информации о поездах, отправляющихся в заданный пункт назначения.

Информация должна быть отсортирована по времени отправления.

Написать программу, демонстрирующую все разработанные элементы классов.

Листинг программы:

Trrain:

internal class Train:IComparable<Train>

{

private string \_destination;

private string \_number;

private TimeOnly \_departureTime;

public string Destination

{

get { return \_destination; }

set { \_destination = value; }

}

public string Number

{

get { return \_number; }

set { \_number = value; }

}

public TimeOnly DepartureTime

{

get { return \_departureTime; }

set { \_departureTime = value; }

}

public Train(string destination, string number, TimeOnly departureTime)

{

\_destination = destination;

\_number = number;

\_departureTime = departureTime;

}

public int CompareTo(Train? train)

{

if (train is null) throw new ArgumentException("Некорректное значение параметра");

return DepartureTime.CompareTo(train.DepartureTime);

}

public static bool operator >(Train t1, Train t2)=> t1.\_departureTime > t2.\_departureTime;

public static bool operator <(Train t1, Train t2)=> t1.\_departureTime < t2.\_departureTime;

public static bool operator ==(Train t1, Train t2) => t1.\_departureTime == t2.\_departureTime;

public static bool operator !=(Train t1, Train t2) => t1.\_departureTime != t2.\_departureTime;

}

RailwayStation:

internal class RailwayStation

{

private Train[] \_trains;

public RailwayStation(int countTrain)

{

\_trains = GetRandom(countTrain);

Array.Sort(\_trains);

}

public void WriteAllsTrains()

{

foreach (Train train in \_trains)

{

Console.WriteLine($"Destination: {train.Destination}");

Console.WriteLine($"Number: {train.Number}");

Console.WriteLine($"Departure Time: {train.DepartureTime}");

Console.WriteLine();

}

}

public void WriteTrains(Train[] trains)

{

foreach (Train train in trains)

{

Console.WriteLine($"Destination: {train.Destination}");

Console.WriteLine($"Number: {train.Number}");

Console.WriteLine($"Departure Time: {train.DepartureTime}");

Console.WriteLine();

}

}

public void WriteTrainByIndex(int i)

{

Console.WriteLine($"Destination: {\_trains[i].Destination}");

Console.WriteLine($"Number: {\_trains[i].Number}");

Console.WriteLine($"Departure Time: {\_trains[i].DepartureTime}");

Console.WriteLine();

}

public Train[] GetArrayTrainByTime(TimeOnly time)

{

var sortTrain = from t in \_trains

where t.DepartureTime >= time

select t;

return sortTrain.ToArray();

}

public Train[] GetArrayTrainByDestination(string destination)

{

var sortTrain = from t in \_trains

where t.Destination == destination

select t;

return sortTrain.ToArray();

}

private Train[] GetRandom(int countTrain)

{

Train[] trains = new Train[countTrain];

Faker faker = new Faker();

Random random = new Random();

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

{

Train train = new Train(faker.Address.City(),

random.Next(100, 999).ToString(),

new TimeOnly(random.Next(0,13), random.Next(0, 60)));

trains[i] = train;

}

return trains;

}

}

Programm:

using z2.Model;

RailwayStation rs = new RailwayStation(10);

ConsoleWriteLineGreen("List of trains: ");

rs.WriteAllsTrains();

ConsoleWriteLineGreen("Enter the index: ");

int index = Convert.ToInt32(Console.ReadLine());

rs.WriteTrainByIndex(index);

ConsoleWriteLineGreen("Enter the time: ");

Console.Write("Рour: ");

int hour = Convert.ToInt32(Console.ReadLine());

Console.Write("Minute: ");

int minute = Convert.ToInt32(Console.ReadLine());

WriteArrayTrainByTime(rs,new TimeOnly(hour, minute));

ConsoleWriteLineGreen("Enter your destination: ");

string destination = Console.ReadLine();

WriteArrayTrainByDestination(rs, destination);

void ConsoleWriteLineGreen(string mes)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine(mes);

Console.ForegroundColor = ConsoleColor.White;

}

void WriteArrayTrainByTime(RailwayStation rs, TimeOnly time)

{

rs.WriteTrains(rs.GetArrayTrainByTime(time));

}

void WriteArrayTrainByDestination(RailwayStation rs, string destination)

{

rs.WriteTrains(rs.GetArrayTrainByDestination(destination));

}

Таблица 9.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| index: 1  hour: 11  minute: 0  destination: Gusikowskiton | Destination: North Israel  Number: 616  Departure Time: 3:11 |

Анализ результатов:

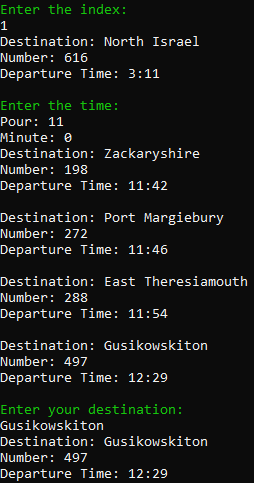


Рисунок 9.2 – Результат выполнения программы задание №2