**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине «ЯПВУ»

на тему: «Коллекции структур данных. Классы-прототипы»

Выполнил: студент гр. ИП-22

Козловский В.В.

Принял: ассистент

Ракицкий А.А.

Гомель 2020

**Цель**: Изучить возможность использования коллекций пространства имен System.Collection, System.Collection.Generic.

**Задание.**

**Вариант 10**

1. Подготовить текстовый файл, каждая строка которого содержит название транспорта, год выпуска, регистрационный номер, у троллейбусов дополнительные поля номер маршрута и пробег.
2. Создать иерархию классов: Транспорт-Троллейбус

Класс «Транспорт» должен содержать следующие элементы: поле вид транспортного средства (автомобиль, автобус, трамвай, трактор и т.д.), поле год выпуска, поле регистрационный номер, Класс троллейбусов должен содержать дополнительные поля номер маршрута и пробег, конструктор с параметрами, свойства для чтения полей класса;

1. Создать класс-прототип для хранения и обработки экземпляров классов, наложить ограничение на параметр типа данных: элементы коллекции должны быть типа «транспорт» или производных от него типов, тип – аргумент должен реализовывать интерфейс IComparable.
2. Написать приложение, выполняющее следующее функции:

• Считывание данных из файла.

• Вывод информации в виде таблицы.

• Сортировку по году выпуска

• Добавление нового троллейбуса

• Формирование и вывод списка троллейбусов, пробег которых не превышает заданный и совпадает с указанным маршрутом (использовать стандартную коллекцию ArrayList).

**Код программы:**

CollectionMapTransport.cs:

using CsvHelper.Configuration;

namespace Task1.FileExtensions

{

public class CollectionMapTransport : ClassMap<Transport>

{

public CollectionMapTransport()

{

Map(m => m.Type).Name(FileExtensions.TransportType);

Map(m => m.ReleaseYear).Name(nameof(Transport.ReleaseYear));

Map(m => m.RegistrationNumber).Name(nameof(Transport.RegistrationNumber));

}

}

}

CollectionMapTrolleybus.cs:

using CsvHelper.Configuration;

namespace Task1.FileExtensions

{

public class CollectionMapTrolleybusType : ClassMap<Trolleybus>

{

public CollectionMapTrolleybusType()

{

Map(m => m.Type).Name(FileExtensions.TrolleybusType);

Map(m => m.ReleaseYear).Name(nameof(Trolleybus.ReleaseYear));

Map(m => m.RegistrationNumber).Name(nameof(Trolleybus.RegistrationNumber));

Map(m => m.RouteNumber).Name(nameof(Trolleybus.RouteNumber));

Map(m => m.Mileage).Name(nameof(Trolleybus.Mileage));

}

}

}

FileExtensions.cs:

using CsvHelper;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Task1.FileExtensions

{

public static class FileExtensions

{

public const string TransportType = "TransportType";

public const string TrolleybusType = "TrolleybusType";

public static IList<Transport> GetTransports(string filePath)

{

using (var reader = new StreamReader(filePath)

{

using (var csv = new CsvReader(reader, System.Globalization.CultureInfo.CurrentCulture))

{

csv.Configuration.IgnoreBlankLines = false;

csv.Configuration.RegisterClassMap<CollectionMapTransport>();

csv.Configuration.RegisterClassMap<CollectionMapTrolleybusType>();

var transportRecord = new List<Transport>();

bool isHeader = true;

while (csv.Read())

{

if (isHeader)

{

csv.ReadHeader();

isHeader = false;

continue;

}

if (string.IsNullOrEmpty(csv.GetField(0)))

{

isHeader = true;

continue;

}

switch (csv.Context.HeaderRecord[0])

{

case TransportType:

transportRecord.Add(csv.GetRecord<Transport>());

break;

case TrolleybusType:

transportRecord.Add(csv.GetRecord<Trolleybus>());

break;

default:

throw new InvalidOperationException("Unknown record type.");

}

}

return transportRecord;

}

}

}

public static void SetProducts(IReadOnlyCollection<Transport> transports, string filePath)

{

using (var writer = new StreamWriter(filePath)

{

using (var csv = new CsvWriter(writer, System.Globalization.CultureInfo.CurrentCulture))

{

csv.WriteRecords(transports);

}

}

}

}

}

Airplane.cs:

using System.Collections.Generic;

namespace Task1

{

public class Airplane : Transport

{

public Airplane() : base()

{ }

public int FreeSeats { get; set; }

public int FlightNumber { get; set; }

public string Destination { get; set; }

public IReadOnlyList<decimal> CostTickets { get; set; }

public decimal this[SeatType place]

{

get => CostTickets[(int)place];

}

public override string ToString() => base.ToString() + "\t" + string.Format($"{FreeSeats}\t{FlightNumber}\t{Destination}");

}

}

Program.cs:

using System;

namespace Task1

{

class Program

{

static void Main(string[] args)

{

const string filePath = "Transports.csv";

Trolleybus trolleybus = new Trolleybus();

trolleybus.Type = "Trolleybus";

trolleybus.ReleaseYear = 2018;

trolleybus.RegistrationNumber = "n8ji52";

trolleybus.RouteNumber = 478;

trolleybus.Mileage = 59;

TransportCollection<Transport> transports;

try

{

transports = new TransportCollection<Transport>(FileExtensions.FileExtensions.GetTransports(filePath));

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return;

}

foreach (var transport in transports.Transports)

{

Console.WriteLine(transport);

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

transports.Sort();

transports.Add(trolleybus);

foreach (var transport in transports.Transports)

{

Console.WriteLine(transport);

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

foreach (var transport in transports.GetTrolleybus(52, 78))

{

Console.WriteLine(transport);

}

}

}

}

Trolleybus.cs:

namespace Task1

{

public class Trolleybus : Transport

{

public Trolleybus() : base()

{ }

public int RouteNumber { get; set; }

public int Mileage { get; set; }

public override string ToString() => base.ToString() + "\t" + ($"{RouteNumber}\t{Mileage}");

}

}

Transport.cs:

using System;

namespace Task1

{

public class Transport : IComparable<Transport>

{

public Transport()

{ }

public string Type { get; set; }

public int ReleaseYear { get; set; }

public string RegistrationNumber { get; set; }

public int CompareTo(Transport transport)

{

return Type.CompareTo(transport.Type);

}

public override string ToString() => ($"{Type}\t{ReleaseYear}\t{RegistrationNumber}");

}

}

TransportCollection.cs:

using System.Collections;

using System.Collections.Generic;

using System.Linq;

namespace Task1

{

public class TransportCollection<T> where T : Transport

{

public TransportCollection(IList<T> transports)

{

Transports = transports.ToList();

}

public List<T> Transports { get; }

public void Add(T transport) => Transports.Add(transport);

public void Remove(T transport) => Transports.Remove(transport);

public void Sort()

{

Transports.OrderBy(obj => obj).ToList();

}

public ArrayList GetTrolleybus(int milleage, int routeNumber)

{

var list = new ArrayList();

foreach (object transport in Transports)

{

if (transport.GetType() == typeof(Trolleybus))

{

if (((Trolleybus)transport).Mileage <= milleage && ((Trolleybus)transport).RouteNumber == routeNumber)

{

list.Add(transport);

}

}

}

return list;

}

}

}

**Результат выполнения программы:**

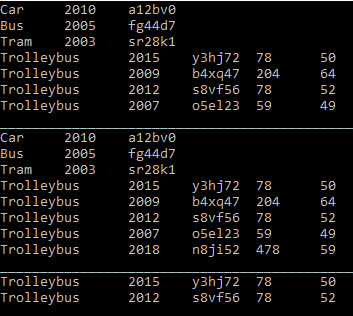


Рисунок 1 – Результат выполнения задания 1

**Вывод:** Изучил возможность использования коллекций пространства имен System.Collection, System.Collection.Generic.