Пермский филиал федерального государственного автономного   
образовательного учреждения высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Перминов Николай Александрович

**Лабораторная работа №14**

**«Linq»**

*Отчёт*

студента образовательной программы «Программная инженерия»   
по направлению подготовки *09.03.04 Программная инженерия*

Руководитель   
Доцент каф. ИТБ, к.п.н.   
И. В. Ильин

Пермь, 2020 год

Оглавление

[1 Постановка задачи 3](#_Toc73892268)

[2 Диаграмма классов 4](#_Toc73892269)

[3 Код программы 5](#_Toc73892270)

[4 Код unit-тестов 14](#_Toc73892271)

[5 Анализ покрытия тестов 16](#_Toc73892272)

# Постановка задачи

1. Сформировать обобщенную коллекцию, содержащую ссылки на другие коллекции.
2. Заполнить коллекции объектами иерархии классов (лабораторная работа №10).

Выполнить запросы функции (всего должно быть выполнено не менее 5 запросов):

1. На выборку данных.
2. Получение счетчика (количества объектов с заданным параметром).
3. Использование операций над множествами (пересечение, объединение, разность).
4. Агрегирование данных.
5. Группировка данных

Запросы должны быть выполнены двумя способами:

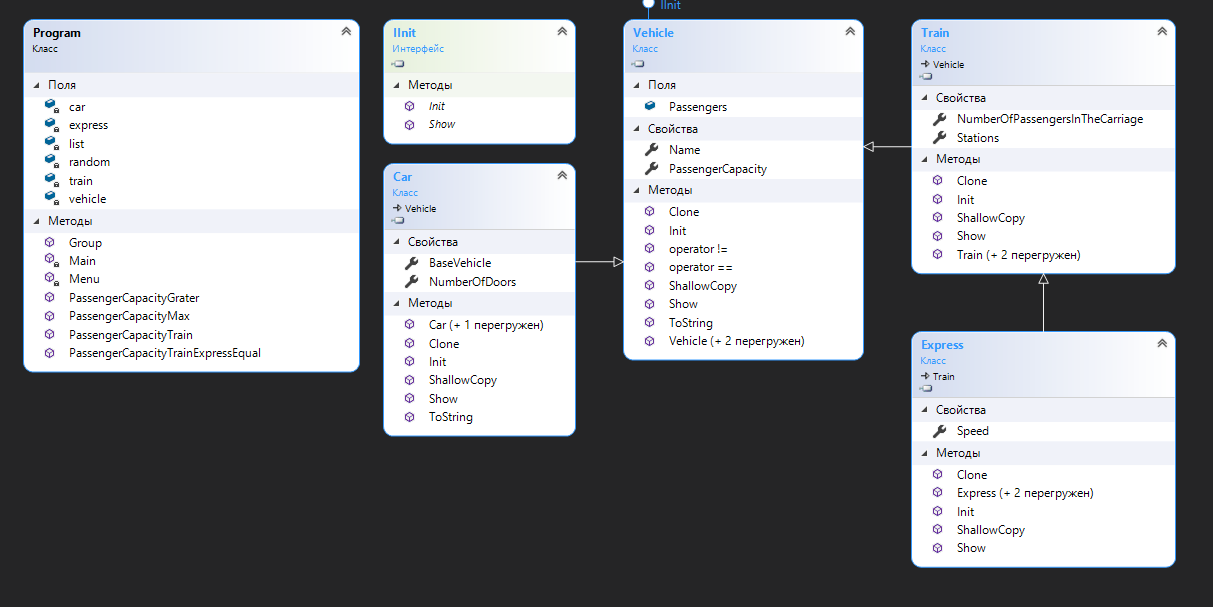
1. С использованием LINQ запросов.
2. С использованием методов расширения.

Каждый запрос выполняется в отдельной функции.

Примеры запросов (лабораторная работа №10).

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | Город (List) | Вокзал (Stack) | автобус, поезд, **транспортное средство**, экспресс; |

# Диаграмма классов



# Код программы

Код программы удобнее смотреть тут: <https://github.com/k0perX-X/1-kurs-labs>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices.ComTypes;

using Laba\_10;

namespace Laba\_14

{

public class Program

{

private static Vehicle vehicle = new Vehicle();

private static Car car = new Car();

private static Train train = new Train();

private static Express express = new Express();

private static List<Vehicle> list = new(10000);

//private static Stack<Vehicle> stack = new(10000);

private static Random random = new Random();

private static void Main(string[] args)

{

for (int i = 0; i < 10000; i++)

{

switch (random.Next(4))

{

case 0:

list.Add((Vehicle)vehicle.Init());

break;

case 1:

list.Add((Car)car.Init());

break;

case 2:

list.Add((Train)train.Init());

break;

case 3:

list.Add((Express)express.Init());

break;

}

}

//for (int i = 0; i < 10000; i++)

//{

// switch (random.Next(4))

// {

// case 0:

// stack.Push((Vehicle)vehicle.Init());

// break;

// case 1:

// stack.Push((Car)car.Init());

// break;

// case 2:

// stack.Push((Train)train.Init());

// break;

// case 3:

// stack.Push((Express)express.Init());

// break;

// }

//}

while (Menu()) { }

}

public static (int zapros, int metod) PassengerCapacityTrain(IEnumerable<Vehicle> s)

{

var t = from l in s where l.GetType() == typeof(Train) select l.PassengerCapacity;

return (t.Sum(), s.Where(vehicle1 => vehicle1.GetType() == typeof(Train)).Select(x => x.PassengerCapacity).ToList().Sum());

}

public static (int zapros, int metod) PassengerCapacityGrater(IEnumerable<Vehicle> s, int gr)

{

var t = (from l in s where l.PassengerCapacity > gr select l).Count();

return (t, s.Where(l => l.PassengerCapacity > gr).Count());

}

public static (int zapros, int metod) PassengerCapacityTrainExpressEqual(IEnumerable<Vehicle> s)

{

var t1 = from l in s where l.GetType() == typeof(Train) select l.PassengerCapacity;

var t2 = from l in s where l.GetType() == typeof(Express) select l.PassengerCapacity;

var t = t1.Except(t1.Except(t2)).Count();

t1 = s.Where(vehicle1 => vehicle1.GetType() == typeof(Train)).Select(x => x.PassengerCapacity);

t2 = s.Where(vehicle1 => vehicle1.GetType() == typeof(Express)).Select(x => x.PassengerCapacity);

return (t, t1.Except(t1.Except(t2)).Count());

}

public static (int zapros, int metod) PassengerCapacityMax(IEnumerable<Vehicle> s)

{

return ((from l in s select l.PassengerCapacity).Max(), s.Select(x => x.PassengerCapacity).Max());

}

public static ((IEnumerable<Vehicle> vehicles, IEnumerable<Car> cars, IEnumerable<Train> trains, IEnumerable<Express> expresses) zapros,

(IEnumerable<Vehicle> vehicles, IEnumerable<Car> cars, IEnumerable<Train> trains, IEnumerable<Express> expresses) metod)

Group(IEnumerable<Vehicle> s)

{

var t1 = from l in s where l.GetType() == typeof(Car) select (Car)l;

var t2 = from l in s where l.GetType() == typeof(Train) select (Train)l;

var t3 = from l in s where l.GetType() == typeof(Express) select (Express)l;

var t4 = from l in s where l.GetType() == typeof(Vehicle) select l;

return ((t4, t1, t2, t3), (s.Where(x => x.GetType() == typeof(Vehicle)),

s.Where(x => x.GetType() == typeof(Car)).Select(x => (Car)x),

s.Where(x => x.GetType() == typeof(Train)).Select(x => (Train)x),

s.Where(x => x.GetType() == typeof(Express)).Select(x => (Express)x)));

}

private static bool Menu()

{

try

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nМЕНЮ\n");

Console.ResetColor();

Console.Write(" 1. Вместительность всех поездов\n" +

" 2. Количество транспортных средств с вместимостью больше 10\n" +

" 3. Количество поездов и экспрессов с одинаковой вместимостью\n" +

" 4. Максимальная вместимость среди всех\n" +

" 5. Сгруппировать по типу транспортного средства\n" +

" 6. Выход из программы\n" +

"\nВыберите задание: ");

string str;

switch (int.Parse(Console.ReadLine()))

{

case 1:

str = "=";

str = str.PadRight(Console.WindowWidth, '=');

Console.WriteLine(str);

Console.SetCursorPosition(0, Console.CursorTop + Console.WindowHeight + 2);

Console.SetCursorPosition(0, Console.CursorTop - Console.WindowHeight);

{

(int zapros, int metod) t = PassengerCapacityTrain(list);

Console.WriteLine($"Операции запросов {t.zapros}");

Console.WriteLine($"Операции методов {t.metod}");

}

return true;

case 2:

str = "=";

str = str.PadRight(Console.WindowWidth, '=');

Console.WriteLine(str);

Console.SetCursorPosition(0, Console.CursorTop + Console.WindowHeight + 2);

Console.SetCursorPosition(0, Console.CursorTop - Console.WindowHeight);

{

(int zapros, int metod) t = PassengerCapacityGrater(list, 10);

Console.WriteLine($"Операции запросов {t.zapros}");

Console.WriteLine($"Операции методов {t.metod}");

}

return true;

case 3:

str = "=";

str = str.PadRight(Console.WindowWidth, '=');

Console.WriteLine(str);

Console.SetCursorPosition(0, Console.CursorTop + Console.WindowHeight + 2);

Console.SetCursorPosition(0, Console.CursorTop - Console.WindowHeight);

{

(int zapros, int metod) t = PassengerCapacityTrainExpressEqual(list);

Console.WriteLine($"Операции запросов {t.zapros}");

Console.WriteLine($"Операции методов {t.metod}");

}

return true;

case 4:

str = "=";

str = str.PadRight(Console.WindowWidth, '=');

Console.WriteLine(str);

Console.SetCursorPosition(0, Console.CursorTop + Console.WindowHeight + 2);

Console.SetCursorPosition(0, Console.CursorTop - Console.WindowHeight);

{

(int zapros, int metod) t = PassengerCapacityMax(list);

Console.WriteLine($"Операции запросов {t.zapros}");

Console.WriteLine($"Операции методов {t.metod}");

}

return true;

case 5:

str = "=";

str = str.PadRight(Console.WindowWidth, '=');

Console.WriteLine(str);

Console.SetCursorPosition(0, Console.CursorTop + Console.WindowHeight + 2);

Console.SetCursorPosition(0, Console.CursorTop - Console.WindowHeight);

{

((IEnumerable<Vehicle> vehicles, IEnumerable<Car> cars, IEnumerable<Train> trains, IEnumerable<Express> expresses) zapros,

(IEnumerable<Vehicle> vehicles, IEnumerable<Car> cars, IEnumerable<Train> trains, IEnumerable<Express> expresses) metod) t = Group(list);

Console.WriteLine($"Операции запросов Vehicles: {t.zapros.vehicles.Count()} Cars: {t.zapros.cars.Count()} Trains: {t.zapros.trains.Count()} Expresses: {t.zapros.expresses.Count()}");

Console.WriteLine($"Операции методов Vehicles: {t.metod.vehicles.Count()} Cars: {t.metod.cars.Count()} Trains: {t.metod.trains.Count()} Expresses: {t.metod.expresses.Count()}");

}

return true;

case 6:

Console.Write("Нажмите любую клавишу для выхода...");

Console.ReadKey();

return false;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный формат числа");

Console.ResetColor();

Console.WriteLine();

return true;

}

}

catch (FormatException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Неверный формат числа");

Console.ResetColor();

Console.WriteLine();

return true;

}

}

}

}

# Код unit-тестов

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using Laba\_10;

using static Laba\_14.Program;

namespace UnitTestsLaba14

{

[TestClass]

public class Tests

{

private static Vehicle vehicle = new Vehicle();

private static Car car = new Car();

private static Train train = new Train();

private static Express express = new Express();

private static Random random = new Random();

private static int number = 1;

private List<Vehicle> createList()

{

List<Vehicle> list = new(1000);

for (int i = 0; i < 100000; i++)

{

switch (random.Next(4))

{

case 0:

list.Add((Vehicle)vehicle.Init());

break;

case 1:

list.Add((Car)car.Init());

break;

case 2:

list.Add((Train)train.Init());

break;

case 3:

list.Add((Express)express.Init());

break;

}

}

return list;

}

[TestMethod]

public void PassengerCapacityTrainTest()

{

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

{

var l = createList();

var t = PassengerCapacityTrain(l);

if (t.metod != t.zapros)

throw new Exception($"{t.zapros} {t.metod}");

}

}

[TestMethod]

public void PassengerCapacityGraterTest()

{

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

{

var l = createList();

var t = PassengerCapacityGrater(l, i / 100);

if (t.metod != t.zapros)

throw new Exception($"{t.zapros} {t.metod}");

}

}

[TestMethod]

public void PassengerCapacityTrainExpressEqualTest()

{

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

{

var l = createList();

var t = PassengerCapacityTrainExpressEqual(l);

if (t.metod != t.zapros)

throw new Exception($"{t.zapros} {t.metod}");

}

}

[TestMethod]

public void PassengerCapacityMaxTest()

{

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

{

var l = createList();

var t = PassengerCapacityMax(l);

if (t.metod != t.zapros)

throw new Exception($"{t.zapros} {t.metod}");

}

}

[TestMethod]

public void GroupTest()

{

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

{

var l = createList();

var t = Group(l);

if (t.metod.vehicles.Except(t.zapros.vehicles).Count() > 0)

throw new Exception($"vehicles");

if (t.metod.cars.Except(t.zapros.cars).Count() > 0)

throw new Exception($"cars");

if (t.metod.trains.Except(t.zapros.trains).Count() > 0)

throw new Exception($"trains");

if (t.metod.expresses.Except(t.zapros.expresses).Count() > 0)

throw new Exception($"expresses");

}

}

}

}

# Анализ покрытия тестов

