2018

Отчет по лабораторной работе № 7

Базовые компоненты интернет технологий

Зубков андрей, иу5-34б

Описание задания

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс «Сотрудник», содержащий поля:

* ID записи о сотруднике;
* Фамилия сотрудника;
* ID записи об отделе.

1. Создайте класс «Отдел», содержащий поля:

* ID записи об отделе;
* Наименование отдела.

1. Предполагая, что «Отдел» и «Сотрудник» связаны соотношением один-ко-многим разработайте следующие запросы:

* Выведите список всех сотрудников и отделов, отсортированный по отделам.
* Выведите список всех сотрудников, у которых фамилия начинается с буквы «А».
* Выведите список всех отделов и количество сотрудников в каждом отделе.
* Выведите список отделов, в которых у всех сотрудников фамилия начинается с буквы «А».
* Выведите список отделов, в которых хотя бы у одного сотрудника фамилия начинается с буквы «А».

1. Создайте класс «Сотрудники отдела», содержащий поля:

* ID записи о сотруднике;
* ID записи об отделе.

1. Предполагая, что «Отдел» и «Сотрудник» связаны соотношением много-ко-многим с использованием класса «Сотрудники отдела» разработайте следующие запросы:

* Выведите список всех отделов и список сотрудников в каждом отделе.
* Выведите список всех отделов и количество сотрудников в каждом отделе.

Текст программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab7

{

class Program

{

/// <summary>

/// Класс данных

/// </summary>

public class Worker

{

/// <summary>

/// Ключ

/// </summary>

public int id\_;

/// <summary>

/// Для группировки

/// </summary>

public string surname\_;

/// <summary>

/// Значение

/// </summary>

public int idSection\_;

/// <summary>

/// Конструктор

/// </summary>

public Worker(int id, string surname, int idSection)

{

this.id\_ = id;

this.surname\_ = surname;

this.idSection\_ = idSection;

}

/// <summary>

/// Приведение к строке

/// </summary>

public override string ToString()

{

return "(id=" + this.id\_.ToString() + "; surname=" + this.surname\_ + "; idSection=" + this.idSection\_.ToString() + ")";

}

}

/// <summary>

/// Класс для сравнения данных

/// </summary>

public class WorkerEqualityComparer : IEqualityComparer<Worker>

{

public bool Equals(Worker x, Worker y)

{

bool Result = false;

if (x.id\_ == y.id\_ && x.surname\_ == y.surname\_ && x.idSection\_ == y.idSection\_) Result = true;

return Result;

}

public int GetHashCode(Worker obj)

{

return obj.id\_;

}

}

public class Section

{

/// <summary>

/// Ключ

/// </summary>

public int id\_;

/// <summary>

/// Для группировки

/// </summary>

public string name\_;

/// <summary>

/// Конструктор

/// </summary>

public Section(int id, string name)

{

this.id\_ = id;

this.name\_ = name;

}

/// <summary>

/// Приведение к строке

/// </summary>

public override string ToString()

{

return "(id=" + this.id\_.ToString() + "; name=" + this.name\_ + ")";

}

}

public class SectionEqualityComparer : IEqualityComparer<Section>

{

public bool Equals(Section x, Section y)

{

bool Result = false;

if (x.id\_ == y.id\_ && x.name\_ == y.name\_) Result = true;

return Result;

}

public int GetHashCode(Section obj)

{

return obj.id\_;

}

}

public class Staffer

{

/// <summary>

/// Ключ

/// </summary>

public int id\_;

/// <summary>

/// Для группировки

/// </summary>

public int idSection\_;

/// <summary>

/// Конструктор

/// </summary>

public Staffer(int id, int idSection)

{

this.id\_ = id;

this.idSection\_ = idSection;

}

/// <summary>

/// Приведение к строке

/// </summary>

public override string ToString()

{

return "(id=" + this.id\_.ToString() + "; idSection=" + this.idSection\_ + ")";

}

}

public class StafferEqualityComparer : IEqualityComparer<Staffer>

{

public bool Equals(Staffer x, Staffer y)

{

bool Result = false;

if (x.id\_ == y.id\_ && x.idSection\_ == y.idSection\_) Result = true;

return Result;

}

public int GetHashCode(Staffer obj)

{

return obj.id\_;

}

}

//Пример данных

static List<Worker> workerList = new List<Worker>()

{

new Worker(1, "zubkov", 1),

new Worker(2, "smirnov", 2),

new Worker(3, "ivanov", 1),

new Worker(4, "petrov", 2),

new Worker(5, "ivanova", 2),

new Worker(6, "petrova", 1),

new Worker(7, "alexandrov", 3),

new Worker(8, "alekseev", 1)

};

static List<Section> sectionList = new List<Section>()

{

new Section(1, "otdel1"),

new Section(2, "otdel2"),

new Section(3, "otdel3")

};

static List<Staffer> stafferList = new List<Staffer>()

{

new Staffer(1, 1),

new Staffer(1, 3),

new Staffer(2, 1),

new Staffer(3, 2),

new Staffer(3, 1)

};

static void Main(string[] args)

{

Console.WriteLine("Вывод сотрудников с сортировкой по отделам");

var q1 = from x in workerList

join y in sectionList on x.idSection\_ equals y.id\_ into temp

orderby x.idSection\_ ascending, x.surname\_ ascending

select new { tmpWorker = x, tmpSection = temp };

foreach (var x in q1)

{

Console.Write(x.tmpWorker);

foreach (var y in x.tmpSection)

Console.WriteLine(" section name: " + y.name\_);

}

Console.WriteLine("\nВывод сотрудников, у которых фамилия начинается с А");

var q2 = from x in workerList

where x.surname\_[0] == 'a'

select x;

foreach (var x in q2) Console.WriteLine(x);

Console.WriteLine("\nВывод всех отделов и количество работающих сотрудников");

var q3 = from x in sectionList

join y in workerList on x.id\_ equals y.idSection\_ into tmp

select new { tmpSection = x, countWorkers = tmp.Count() };

foreach (var x in q3)

{

Console.WriteLine("{0}; count workers = {1}", x.tmpSection, x.countWorkers);

}

Console.WriteLine("\nСписок отделов, у которых фамилии всех сотрудников начинаются с а");

var q4 = from x in sectionList

join y in workerList on x.id\_ equals y.idSection\_ into tmp

where tmp.All(y => y.surname\_[0] == 'a')

select new { tmpSection = x, workers = tmp };

foreach (var x in q4)

{

Console.WriteLine("{0}:", x.tmpSection);

foreach (var y in x.workers)

{

Console.WriteLine(y);

}

}

Console.WriteLine("\nСписок отделов, у которых фамилия хотя бы одного сотрудника начинаются с а");

var q5 = from x in sectionList

join y in workerList on x.id\_ equals y.idSection\_ into tmp

where tmp.Any(y => y.surname\_[0] == 'a')

select new { tmpSection = x, workers = tmp };

foreach (var x in q5)

{

Console.WriteLine("{0}:", x.tmpSection);

foreach (var y in x.workers)

{

Console.WriteLine(y);

}

}

Console.WriteLine("\nСписок всех отделов и список всех сотрудников в каждом отделе");

var q6 = from x in sectionList

join y in stafferList on x.id\_ equals y.idSection\_ into tmp

select new { tmpSection = x, tmpWorkers = tmp };

foreach (var x in q6)

{

Console.WriteLine("otdel: {0}", x.tmpSection);

foreach (var y in x.tmpWorkers)

{

Console.WriteLine("staffer: {0}", y);

}

Console.WriteLine();

}

Console.WriteLine("\nСписок всех отделов и количество сотрудников в каждом отделе");

var q7 = from x in sectionList

join y in stafferList on x.id\_ equals y.idSection\_ into tmp

select new { tmpSection = x, countWorkers = tmp.Count() };

foreach (var x in q7)

{

Console.WriteLine("otdel: {0}; countWorkers = {1}", x.tmpSection, x.countWorkers);

}

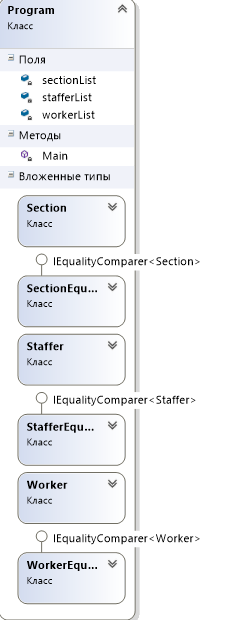
Console.ReadKey();

}

}

}

Диаграмма классов



Пример работы программы

