**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

Факультет «Информатика и вычислительная техника»

Кафедра ИУ5 «Системы обработки информации и управления»

**Отчет по лабораторной работе № 7**

Задание: Разработать программу, реализующую работу с LINQ to Objects. В качестве примера используйте проект «SimpleLINQ» из примера «Введение в LINQ»

Студент Левин Михаил Александрович

*фамилия, имя, отчество*

Группа ИУ5-32Б

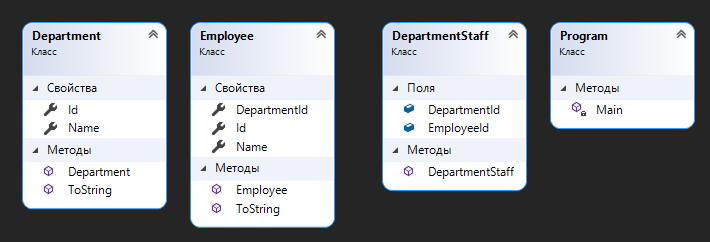
Москва, 2020 г.

# Задание

Разработать программу, реализующую работу с LINQ to Objects. В качестве примера используйте проект «SimpleLINQ» из примера «Введение в LINQ».

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.
3. Создайте класс «Отдел», содержащий поля:
   * ID записи об отделе;
   * Наименование отдела.
4. 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».
5. Создайте класс «Сотрудники отдела», содержащий поля:
   * ID записи о сотруднике;
   * ID записи об отделе.
6. 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.

# Диаграмма классов



# Текст программы

namespace Lab7.Model

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

public Department(string name)

{

Name = name;

}

public override string ToString()

{

return $"DepartmentId: {Id} Name: {Name}";

}

}

}

namespace Lab7.Model

{

class DepartmentStaff

{

public int DepartmentId;

public int EmployeeId;

public DepartmentStaff(int depId, int empId)

{

DepartmentId = depId;

EmployeeId = empId;

}

}

}

namespace Lab7.Model

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int DepartmentId { get; set; }

public Employee(string name)

{

Name = name;

}

public override string ToString()

{

return $"EmployeeId: {Id} Name: {Name}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using Lab7.Model;

namespace Lab7

{

class Program

{

static void Main(string[] args)

{

//Создание отделов предприятия

List<Department> departments = new List<Department>();

departments.Add(new Department("Отдел кадров"));

departments.Add(new Department("Конструкторское бюро"));

departments.Add(new Department("Бухгалтерия"));

for (int i = 0; i < departments.Count; i++)

{

departments[i].Id = i;

}

//Создание сотрудников предприятия

List<Employee> employees = new List<Employee>();

employees.Add(new Employee("Абаканов Э.Э."));

employees.Add(new Employee("Гуляев Г.Ф."));

employees.Add(new Employee("Трофимов А.П."));

for (int i = 0; i < 3; i++)

{

employees[i].DepartmentId = 0;

}

employees.Add(new Employee("Одинцов Е.В."));

employees.Add(new Employee("Пузырьков А.Г."));

employees.Add(new Employee("Зыков В.Е."));

for (int i = 3; i < 6; i++)

{

employees[i].DepartmentId = 1;

}

employees.Add(new Employee("Анишенко М.В."));

employees.Add(new Employee("Антипова А.О."));

employees.Add(new Employee("Абаимова Х.Г."));

for (int i = 6; i < 9; i++)

{

employees[i].DepartmentId = 2;

}

for (int i = 0; i < employees.Count; i++)

{

employees[i].Id = i;

}

Console.WriteLine("Cписок всех сотрудников и отделов, отсортированный по отделам:");

//отделы отсортированные по имени

var q1 = from d in departments orderby d.Name select d;

//цикл по отделам

foreach (Department department in q1)

{

Console.WriteLine(department.ToString());

//цикл по сотрудникам в каждом отделе

var q1in = from e in employees orderby e.Name where e.DepartmentId == department.Id select e;

foreach (Employee employee in q1in)

{

Console.WriteLine(employee);

}

}

Console.WriteLine("\nCписок всех сотрудников, у которых фамилия начинается с буквы «А»:");

var q2 = from e in employees where e.Name.StartsWith("А") select e;

foreach (Employee employee in q2)

{

Console.WriteLine(employee);

}

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

var q3 = from d in departments

join e in employees on d.Id equals e.DepartmentId

group d by d.Name

into grp

select new {key = grp.Key, cnt = grp.Count()};

foreach (var q3var in q3)

{

Console.WriteLine($"{q3var.key} {q3var.cnt}");

}

Console.WriteLine("\nCписок отделов, в которых у всех сотрудников фамилия начинается с буквы «А»:");

var q4 = from e in employees

group e by e.DepartmentId

into grp

select new { key = grp.Key, depaName = (from d in departments where d.Id==grp.Key select d).FirstOrDefault().Name, cnt = grp.Count(), cntA = (from ge in grp where ge.Name.StartsWith("А") select ge).Count() };

var q4in = from emp in q4 where emp.cnt == emp.cntA select emp;

foreach (var q in q4in)

{

Console.WriteLine($"{q.depaName}");

}

Console.WriteLine("\nCписок отделов, в которых хотя бы у одного сотрудника фамилия начинается с буквы «А».");

var q5 = from e in employees

join d in departments on e.DepartmentId equals d.Id

where e.Name.StartsWith("А")

group d by d.Name

into grp

select new {name = grp.Key};

foreach (var q in q5)

{

Console.WriteLine(q.name);

}

List<DepartmentStaff> depStaff = new List<DepartmentStaff>();

foreach (Employee employee in employees)

{

depStaff.Add(new DepartmentStaff(employee.DepartmentId, employee.Id));

}

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

var q6 = from ds in depStaff

join d in departments on ds.DepartmentId equals d.Id

join e in employees on ds.EmployeeId equals e.Id

select new {name = d.Name, emplName = e.Name};

foreach (var q in q6)

{

Console.WriteLine($"Отдел: {q.name}. Кол-во работников: {q.emplName}");

}

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

var q7 = from ds in depStaff

join d in departments on ds.DepartmentId equals d.Id

join e in employees on ds.EmployeeId equals e.Id

group d by d.Name

into grp

select new { name = grp.Key, count = grp.Count() };

foreach (var q in q7)

{

Console.WriteLine($"Отдел: {q.name}. Кол-во работников: {q.count}");

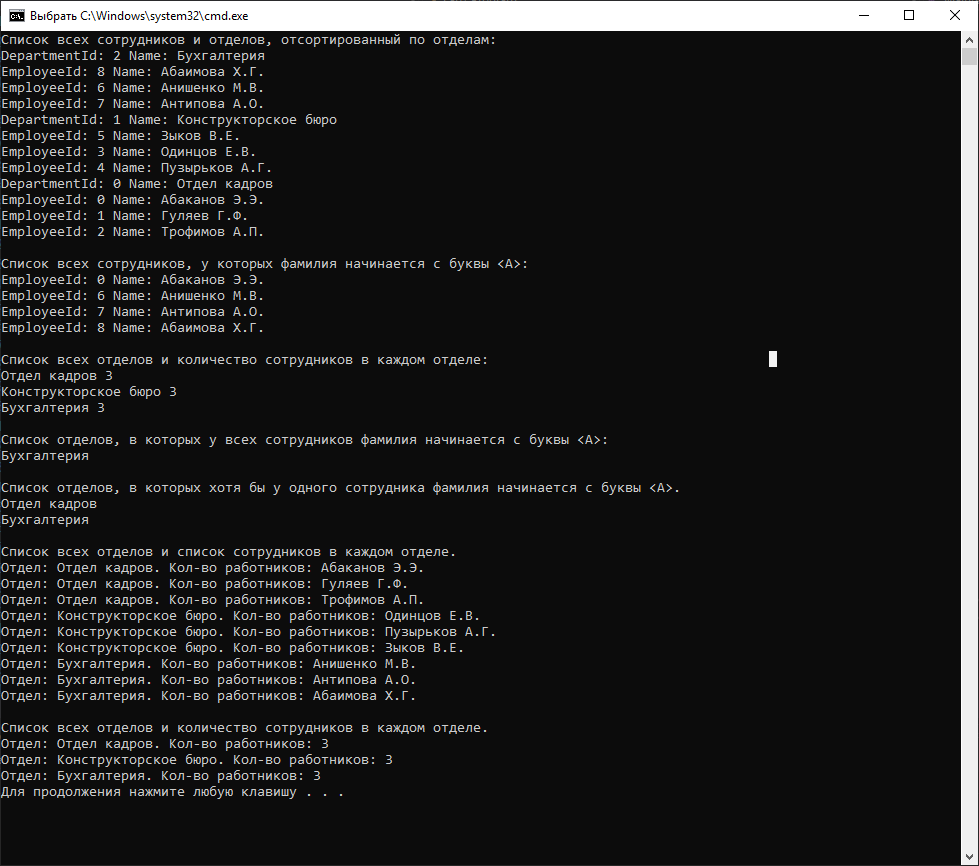
}

}

}

}

# Экранные формы



# Контрольные вопросы:

1. Для чего используется технология LINQ?

Language Integrated Query (LINQ) — проект Microsoft по добавлению синтаксиса языка запросов, напоминающего SQL, в языки программирования платформы .NET Framework. Язык LINQ используется для манипуляций с объектами запросами похожими на язык SQL.

1. Что такое провайдер в технологии LINQ и для чего он используется?

Провайдеры LINQ представляют собой кодовую прослойку, преобразующую запросы на языке LINQ в запросы понятные источнику данных, например, если источник данный СУБД, то запросы будут преобразованы в запросы к этой СУБД, а результат, который возвращается в СУБД будет преобразован в объектный вид.

1. Для каких видов данных существуют провайдеры в технологии LINQ?

Существуют провайдеры для популярных систем управления базами данных, таких как MS SQL Server, Oracle, SQLite и т. д. Провайдеры могут быть написаны для любых источников данных, не являющихся реляционными базами данных, таким как Excel.

1. В чем особенность технологии LINQ to Objects?

LINQ to Object предназначена для манипуляциями с массивами и коллекциями объектов.

1. Как осуществляется сортировка данных в технологии LINQ?

Для сортировки в LINQ используется два ключевых слова:

Ascending – для сортировки по возрастанию;

Descending – для сортировки по убыванию.

1. Как осуществляется группировка данных в технологии LINQ?

Группировка результата запроса используется ключевое слово group by с указанием полей объектов по которым осуществляется группировка. Данные поля являются ключами группы.

1. Как осуществляется соединение данных (join) в технологии LINQ?

Для связывания двух наборов данных по определенным условиями используется ключевое слово join, с указанием связываемой коллекции и (чаще всего) поля из двух коллекций которые разделяются ключевым словом

1. Как осуществляется соединение данных (join) для связи много-ко-многим в технологии LINQ?

Многие-ко-многим соединение используется с участием класса, которые имеет в своём составе сведения о ключах о двух связываемых объектах. Используя соединение трех таблиц участников получится результат многие-ко-многим.

1. Как работают операции над множествами в технологии LINQ?

В linq доступны следующий операции над множествами: объединение, пересечение и разность. Выполняются операции с помощь ключевых слов, соответственно, except, intersect, union.

1. Что такое анонимные типы и как они используются в технологии LINQ?

Анонимные типы в запросах создаются с помощь ключевых слов select new, с указанием полей, которые берутся из свойств участвующих в запросе наборов, либо ключей групп. В результате запроса может быть сформирован анонимный тип, который создается в момент выполнения запроса. Далее возможно обращение к полям полученных экземпляров анонимного типа для получения данных.