|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Отчёт**

***Лабораторная работа №7***

Выполнил студент: \_\_***Сафиуллин Антон Павлович\_\_\_\_\_\_\_\_\_\_***

*фамилия, имя, отчество*

Группа: \_\_\_\_***РТ5-31Б\_\_\_\_\_\_\_\_***

**Описание задания**

**Лабораторная работа №7**

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.

**Текст Программы**

**Экранные формы с примерами выполнения программы**

**Department.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Linq;

namespace LabRabota\_\_\_7

{

class Department

{

private int ID\_o;

private string d\_Name;

public List<Workers> D\_Workers;

/// <summary>

///

/// </summary>

/// <param name="Name">Department Name</param>

/// <param name="id">Department ID </param>

public Department(int id,string Name)

{

ID\_o = id;

d\_Name = Name;

D\_Workers = new List<Workers>();

}

public int get\_id\_o() { return this.ID\_o; }

/// <summary>

/// Добавление работника в отдел

/// </summary>

/// <param name="i">Работник</param>

public void addWorker (Workers i)

{

i.set\_id\_o(this.ID\_o);

D\_Workers.Add(i);

Workers[] mas = new Workers[D\_Workers.Count];

int l = 0;

foreach (Workers p in D\_Workers)

{

mas[l] = p;

l++;

}

D\_Workers.Clear();

Sort.BubbleSort(mas);

for (int k = 0; k<l; k++)

{

D\_Workers.Add(mas[k]);

}

}

public void deleteWorker(Workers i)

{

int Num = 0;

foreach (Workers p in D\_Workers)

{

if ( p.get\_id()==i.get\_id() )

{

break;

}

Num++;

}

D\_Workers.RemoveAt(Num);

}

public void cout ()

{

Console.WriteLine("\n\nID отдела: "+this.ID\_o+"\nНазание отдела: " + this.d\_Name+"\nКол-во сотрудников:"+this.D\_Workers.Count );

Console.WriteLine("ID Фамилия");

foreach (Workers p in D\_Workers)

{

if (p.get\_id()<10) Console.WriteLine(p.get\_id() + " " + p.get\_FIO());

else Console.WriteLine(p.get\_id() + " " + p.get\_FIO());

}

}

public int coutName (string Buk)

{

var selectedNames = from w in D\_Workers

where w.FIO.ToUpper().StartsWith(Buk)

orderby w.get\_id()

select w;

if (selectedNames.Count() != 0)

{

Console.WriteLine("\n\nID отдела: " + this.ID\_o + "\nНазание отдела: " + this.d\_Name + "\nКол-во сотрудников:" + this.D\_Workers.Count);

Console.WriteLine("ID Фамилия");

foreach (Workers w in selectedNames)

{

if (w.get\_id() < 10) Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

else if (w.get\_id() < 1000) Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

else Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

}

return selectedNames.Count();

}

else return 0;

}

public int AllcoutName(string Buk)

{

var selectedNames = from w in D\_Workers

where w.FIO.ToUpper().StartsWith(Buk)

orderby w.get\_id()

select w;

if (selectedNames.Count() != 0 && selectedNames.Count()== D\_Workers.Count())

{

Console.WriteLine("\n\nID отдела: " + this.ID\_o + "\nНазание отдела: " + this.d\_Name + "\nКол-во сотрудников:" + this.D\_Workers.Count);

Console.WriteLine("ID Фамилия");

foreach (Workers w in selectedNames)

{

if (w.get\_id() < 10) Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

else if (w.get\_id() < 1000) Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

else Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

}

return selectedNames.Count();

}

else return 0;

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Linq;

namespace LabRabota\_\_\_7

{

class Program

{

static void Main(string[] args)

{

Department D1 = new Department(1, "Юридический отдел");

Department D2 = new Department(2, "IT отдел");

Department D3 = new Department(3, "Отдел разработок");

Department D4 = new Department(4, "Отдел Управления");

Workers w1 = new Workers(1, "Аленин");

Workers w2 = new Workers(2, "Миль");

Workers w3 = new Workers(3, "Васянин");

Workers w4 = new Workers(4, "Перчиков");

Workers w5 = new Workers(5, "Петров");

Workers w6 = new Workers(6, "Полянин");

Workers w7 = new Workers(7, "Иванов");

Workers w8 = new Workers(8, "Аликов");

Workers w9 = new Workers(9, "Азизов");

Workers w10 = new Workers(10, "Абрамян");

D1.addWorker(w8);

D1.addWorker(w2);

D1.addWorker(w1);

D1.addWorker(w7);

D1.addWorker(w3);

D3.addWorker(w4);

D2.addWorker(w5);

D2.addWorker(w6);

D1.addWorker(w10);

D4.addWorker(w9);

Console.WriteLine("\n");

List<Workers> AllWorkers = new List<Workers>() { w1, w2, w3, w4, w5, w6, w7, w8, w9, w10 };

Department[] Company = new Department[] { D4, D2, D3, D1 };

Console.WriteLine("\nВыведите список всех сотрудников и отделов, отсортированный по отделам.");

var selectedName = from k in Company

where k.get\_id\_o() != 0

orderby k.get\_id\_o()

select k;

foreach (Department k in selectedName)

{

k.cout();

}

Console.WriteLine("\n");

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

var selectedNames = from w in AllWorkers where w.FIO.ToUpper().StartsWith("А") orderby w.get\_id() select w;

Console.WriteLine("ID Фамилия");

foreach (Workers w in selectedNames)

{

if (w.get\_id() < 10) Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

else if (w.get\_id() < 1000) Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

else Console.WriteLine(w.get\_id() + " " + w.get\_FIO());

}

Console.WriteLine("\n");

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

foreach (Department t in Company)

{

t.AllcoutName("А");

}

Console.WriteLine("\n");

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

foreach (Department f in Company)

{

f.coutName("А");

}

Console.WriteLine("\n");

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

List<Workers\_and\_Department> WD = new List<Workers\_and\_Department>();

foreach (Department Dep in Company)

{

foreach (Workers wor in Dep.D\_Workers)

{

Workers\_and\_Department k = new Workers\_and\_Department(wor, Dep);

WD.Add(k);

}

}

var selectedWD\_All = from wd in WD

where wd.get\_W().get\_id() != 0 && wd.get\_D().get\_id\_o() != 0

orderby wd.get\_D().D\_Workers.Count()

select wd;

List<int> Depu = new List<int>();

Depu.Add(0);

foreach (Workers\_and\_Department Wdk in selectedWD\_All)

{

if (!Depu.Contains(Wdk.get\_D().get\_id\_o() ))

{

Wdk.get\_D().cout();

Depu.Add(Wdk.get\_D().get\_id\_o());

}

}

Console.WriteLine("\n");

}

}

}

**Sort.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace LabRabota\_\_\_7

{

class Sort

{

public static Workers[] BubbleSort(Workers[] mas)

{

Workers temp;

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

{

for (int j = i + 1; j < mas.Length; j++)

{

if (mas[i].get\_id() > mas[j].get\_id())

{

temp = mas[i];

mas[i] = mas[j];

mas[j] = temp;

}

}

}

return mas;

}

}

}

**Workers and Departments.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace LabRabota\_\_\_7

{

class Workers\_and\_Department

{

Department Dep; //Department id

Workers Wor;// Worker id

public Workers\_and\_Department(Workers W, Department De)

{

Wor = W;

Dep = De;

}

public Workers get\_W() { return this.Wor; }

public Department get\_D() { return this.Dep; }

}

}

**Workers.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace LabRabota\_\_\_7

{

class Workers

{

private int id;

public string FIO;

private int id\_o = 0;

/// <summary>

///

/// </summary>

/// <param name="i">ID</param>

/// <param name="f">Familia</param>

/// <param name="io">Department ID</param>

public Workers(int i, string f)

{

id = i;

FIO = f;

}

/// <summary>

/// Чтение Id

/// </summary>

/// <returns></returns>

public int get\_id() { return this.id; }

/// <summary>

/// Чтение Id отдела

/// </summary>

/// <returns></returns>

public int get\_id\_o() { return this.id\_o; }

/// <summary>

/// Чтение FIO

/// </summary>

/// <returns></returns>

public string get\_FIO() { return this.FIO; }

/// <summary>

/// смена отлела

/// </summary>

/// <param name="o">ID отела</param>

public void set\_id\_o(int o) { this.id\_o = o; }

public string cout ()

{

if (this.id < 10) return "ID: " + this.get\_id() + " FIO: " + this.get\_FIO() + " ID\_O: " + this.get\_id\_o();

else return "ID: " + this.get\_id() + " FIO: " + this.get\_FIO() + " ID\_O: " + this.get\_id\_o();

}

}

}