|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Никитин Андрей Владимирович  **Отчет по практической работе №14**  **Вариант 6**  студента образовательной программы «Программная инженерия»  по направлению подготовки *09.03.04 Программная инженерия*   |  |  | | --- | --- | |  | Руководитель  О. Л. Викентьева | |

Пермь, 2020 год

**Задание**

1. Сформировать обобщенную коллекцию (лабораторная работа № 11), 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):

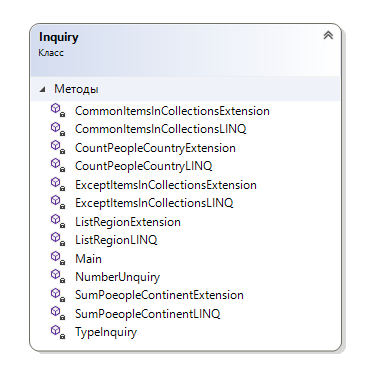
1. На выборку данных.
2. Получение счетчика (количества объектов с заданным параметром).
3. Использование операций над множествами (пересечение, объединение, разность).
4. Агрегирование данных.

Запросы должны быть выполнены двумя способами:

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

Каждый запрос выполняется в отдельной функции.

**Диаграмма классов:**

****

// Запрос LINQ: Название всех Регионов заданной страны (выборка данных)

static void ListRegionLINQ(TestCollections list, string country)

{

var regionsNames = from region in list.Regions // определяем каждый объект из list.Regions как region

where region.Country == country // фильтрация по критерию

orderby region // упорядочиваем по возрастанию

select region; // выбираем объект

if(regionsNames == null)

{

Console.WriteLine("В колекции не существует такой страны как " + country);

return;

}

Console.WriteLine("Список названий регионов страны " + country);

foreach (Region region in regionsNames)

{

Console.WriteLine(region.PlaceName);

}

}

// Запрос LINQ: Количество регионов с данным населением (получение счетчика)

static void CountPeopleCountryLINQ(TestCollections list, int populationSize)

{

var regionlist = from region

in list.Regions

where region.PopulationSize == populationSize

select region;

Console.WriteLine("Количество регионов с населением в " + populationSize + " человек = " + regionlist.Count());

}

// Запрос LINQ: Суммарное количество жителей всех стран на континенте (агрегирование данных)

static void SumPoeopleContinentLINQ(TestCollections list, string continent)

{

int count = (from region in list.Regions where region.Continent == continent select region.PopulationSize).ToArray().Sum();

Console.WriteLine("Суммарное количество жителей всех стран на континенте " + continent + " = " + count);

}

// Запрос LINQ: регионы которые храняться в двух колллекциях (пересечние данных)

static void CommonItemsInCollectionsLINQ(TestCollections list1, TestCollections list2)

{

var collection = (from region in list1.Regions select region).Union(from region1 in list2.Regions select region1);

if (collection == null)

{

Console.WriteLine("Нет общих регионов в коллекциях");

return;

}

Console.WriteLine("Общие регионы:");

foreach(Region item in collection)

{

item.Show();

}

}

// Запрос LINQ: регионы которые есть в одной колллекции, но нет в другой (разность данных)

static void ExceptItemsInCollectionsLINQ(TestCollections list1, TestCollections list2)

{

var collection = (from region in list1.Regions select region).Except(from region1 in list2.Regions select region1);

if (collection == null)

{

Console.WriteLine("В обеих коллекциях храняться одинаовые элементы");

return;

}

Console.WriteLine("Регионы, которых нет во второй коллекции:");

foreach (Region item in collection)

{

item.Show();

}

}

#endregion

#region Методы\_Расширения

// Метод расширения: Название всех Регионов заданной страны (выборка данных)

static void ListRegionExtension(TestCollections list, string country)

{

var regionsNames = list.Regions.Where(region => region.Country == country).OrderBy(region => region);

if (regionsNames == null)

{

Console.WriteLine("В колекции не существует такой страны как " + country);

return;

}

Console.WriteLine("Список названий регионов страны " + country);

foreach (Region region in regionsNames)

{

Console.WriteLine(region.PlaceName);

}

}

// Метод расширения: Количество регионов с данным населением (получение счетчика)

static void CountPeopleCountryExtension(TestCollections list, int populationSize)

{

var regionlist = list.Regions.Where(region => region.PopulationSize == populationSize).Count();

Console.WriteLine("Количество регионов с населением в " + populationSize + " человек = " + regionlist);

}

// Метод расширения: Суммарное количество жителей всех стран на континенте (агрегирование данных)

static void SumPoeopleContinentExtension(TestCollections list, string continent)

{

int count = list.Regions.Where(region => region.Continent == continent).Select(region => region.PopulationSize).ToArray().Sum();

Console.WriteLine("Суммарное количество жителей всех стран на континенте " + continent + " = " + count);

}

// Метод расширения: регионы которые храняться в двух колллекциях (пересечние данных)

static void CommonItemsInCollectionsExtension(TestCollections list1, TestCollections list2)

{

var collection = list1.Regions.Select(region => region).Union(list2.Regions.Select(region1 => region1));

if (collection == null)

{

Console.WriteLine("Нет общих регионов в коллекциях");

return;

}

Console.WriteLine("Общие регионы:");

foreach (Region item in collection)

{

item.Show();

}

}

// Метод расширения: регионы которые есть в одной колллекции, но нет в другой (разность данных)

static void ExceptItemsInCollectionsExtension(TestCollections list1, TestCollections list2)

{

var collection = list1.Regions.Select(region => region).Except(list2.Regions.Select(region1 => region1));

if (collection == null)

{

Console.WriteLine("В обеих коллекциях храняться одинаовые элементы");

return;

}

Console.WriteLine("Регионы, которых нет во второй коллекции:");

foreach (Region item in collection)

{

item.Show();

}

}

**Пример демонстрационной программы:**

TestCollections collections1 = new TestCollections(10);

for(int i = 0; i < collections1.Size; i++)

{

Region region = new Region();

region.RandomCreate();

collections1[i] = region;

}

TestCollections collections2 = new TestCollections(5);

for (int i = 0; i < collections2.Size; i++)

{

Region region = new Region();

region.RandomCreate();

collections2[i] = region;

}

int number = TypeInquiry();

while (number != 0)

{

Console.Clear();

switch (number)

{

case 1:

{

int type = NumberUnquiry(number);

switch (type)

{

case 1:

ListRegionLINQ(collections1, "Россия");

break;

case 2:

CountPeopleCountryLINQ(collections1, 200000);

break;

case 3:

SumPoeopleContinentLINQ(collections1, "Евразия");

break;

case 4:

CommonItemsInCollectionsLINQ(collections1, collections2);

break;

case 5:

ExceptItemsInCollectionsLINQ(collections1, collections2);

break;

}

}

break;

case 2:

{

int type = NumberUnquiry(number);

switch (type)

{

case 1:

ListRegionExtension(collections1, "Китай");

break;

case 2:

CountPeopleCountryExtension(collections1, 100000);

break;

case 3:

SumPoeopleContinentExtension(collections1, "Африка");

break;

case 4:

CommonItemsInCollectionsExtension(collections2, collections1);

break;

case 5:

ExceptItemsInCollectionsExtension(collections2, collections1);

break;

}

}

break;

}

Console.Clear();

number = TypeInquiry();

}

}Ссылка на ГитХаб:

<https://github.com/Anni-HSE/Place-and-Collections>