|  |  |
| --- | --- |
| Розробка програм  На платформі .NET | ІПЗ-22-2  Бабенко Дмитро Андрійович |

**Лабораторна робота №7**

**Тема: Методи розширення, LINQ**

Комплект LINQ

1. Знайдіть всі оплачені доставки (0.5)

2. Знайдіть усі доставки, що зараз опрацьовуються системою (тобто не

скасовані і не виконані) (0.5)

3. Сформуйте DeliveriesShortInfo з усіх доставок певного клієнта (1)

4. Поверніть перших 10 доставок певного типу, що починаються з певного

міста. (1)

5. Відсортуйте записи по їх статусу, якщо статуси однакові, відсортуйте за

часом початку завантаження (за зростанням) (1)

6. Підрахуйте кількість унікальних типів вантажів (1)

7. Згрупуйте доставки за їх статусом та підрахуйте кількість доставок в

кожній групі. Результат подати у вигляді словника

Dictionary<DeliveryStatus, int> (1.5)

8. Згрупуйте доставки за парами «місто старту-місто фінішу» та порахуйте

середній проміжок між кінцем часу завантаження та початком часу

прибуття. (1.5)

9. Створіть метод, що буде віддавати певну сторінку даних. Він приймає

делегат, що описує як треба фільтрувати дані, делегат, як треба

сортувати дані, кількість записів на одній сторінці, номер поточної

сторінки. Номер сторінки починається з 1. Якщо делегат фільтрації не

надано, то пропускати всі записи. (2)

**Код програми:**

using ConsoleApp.Model;

using ConsoleApp.Model.Enum;

using ConsoleApp.OutputTypes;

namespace ConsoleApp;

public class QueryHelper : IQueryHelper

{

/// <summary>

/// Get Deliveries that has payed

/// </summary>

public IEnumerable<Delivery> Paid(IEnumerable<Delivery> deliveries) //TODO: Завдання 1

{

return deliveries.Where(d => d.PaymentId != null);

}

/// <summary>

/// Get Deliveries that now processing by system (not Canceled or Done)

/// </summary>

public IEnumerable<Delivery> NotFinished(IEnumerable<Delivery> deliveries) //TODO: Завдання 2

{

return deliveries.Where(delivery => delivery.Status != DeliveryStatus.Cancelled && delivery.Status != DeliveryStatus.Done);

}

/// <summary>

/// Get DeliveriesShortInfo from deliveries of specified client

/// </summary>

public IEnumerable<DeliveryShortInfo> DeliveryInfosByClient(IEnumerable<Delivery> deliveries, string clientId) //TODO: Завдання 3

{

return deliveries

.Where(delivery => delivery.ClientId == clientId)

.Select(delivery => new DeliveryShortInfo

{

Id = delivery.Id,

ClientId = delivery.ClientId,

StartCity = delivery.Direction?.Origin?.City,

EndCity = delivery.Direction?.Destination?.City,

Type = delivery.Type,

LoadingPeriod = delivery.LoadingPeriod,

ArrivalPeriod = delivery.ArrivalPeriod,

Status = delivery.Status,

CargoType = delivery.CargoType

});

}

/// <summary>

/// Get first ten Deliveries that starts at specified city and have specified type

/// </summary>

public IEnumerable<Delivery> DeliveriesByCityAndType(IEnumerable<Delivery> deliveries, string cityName, DeliveryType type) => new List<Delivery>();//TODO: Завдання 4

/// <summary>

/// Order deliveries by status, then by start of loading period

/// </summary>

public IEnumerable<Delivery> OrderByStatusThenByStartLoading(IEnumerable<Delivery> deliveries) => new List<Delivery>();//TODO: Завдання 5

/// <summary>

/// Count unique cargo types

/// </summary>

public int CountUniqCargoTypes(IEnumerable<Delivery> deliveries) //TODO: Завдання 6

{

return deliveries.Select(delivery => delivery.CargoType).Distinct().Count();

}

/// <summary>

/// Group deliveries by status and count deliveries in each group

/// </summary>

public Dictionary<DeliveryStatus, int> CountsByDeliveryStatus(IEnumerable<Delivery> deliveries) //TODO: Завдання 7

{

return deliveries.GroupBy(delivery => delivery.Status).ToDictionary(group => group.Key, group => group.Count());

}

/// <summary>

/// Group deliveries by start-end city pairs and calculate average gap between end of loading period and start of arrival period (calculate in minutes)

/// </summary>

public IEnumerable<AverageGapsInfo> AverageTravelTimePerDirection(IEnumerable<Delivery> deliveries) => new List<AverageGapsInfo>();//TODO: Завдання 8

/// <summary>

/// Paging helper

/// </summary>

public IEnumerable<TElement> Paging<TElement, TOrderingKey>(IEnumerable<TElement> elements,

Func<TElement, TOrderingKey> ordering,

Func<TElement, bool>? filter = null,

int countOnPage = 100,

int pageNumber = 1) //TODO: Завдання 9

{

// Фільтрація

if (filter != null)

{

elements = elements.Where(filter);

}

// Сортування

elements = elements.OrderBy(ordering);

// Пагінація

return elements.Skip((pageNumber - 1) \* countOnPage).Take(countOnPage);

}

}

**Скріншот програми:**

