ЛАБОРАТОРНАЯ РАБОТА №4

ИНВЕРСИЯ УПРАВЛЕНИЯ.

ЧИСТАЯ АРХИТЕКТУРА.

Содержание работы:

1. Спроектировать и реализовать приложение в соответствии с концепцией «Чистой архитектуры» (Clean Architecture), применив принцип инверсии управления. Компоненты приложения должны быть максимально независимыми друг от друга и не должны зависеть от конкретной реализации друг друга.
2. Для классов, взаимодействующих с БД, использовать паттерны "Repository" и "UnitOfWork".
3. Реализовать внедрение зависимостей (DI).
4. Использовать IoC-контейнер Ninject для управления созданием зависимостей.

За основу взять WinForm приложение из предыдущей л.р.

Продемонстрировать работу приложения.

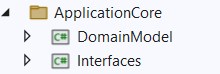
Вопросы для защиты л.р.:

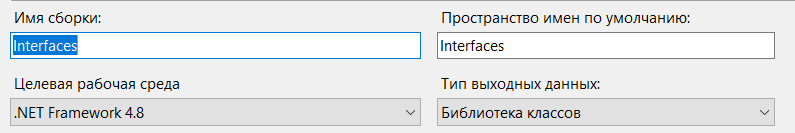
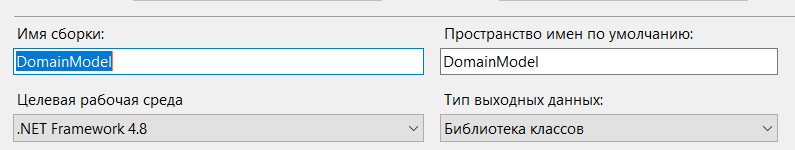
1. Принципы «Чистой архитектуры» (слои в Чистой архитектуре) на примере л.р.
2. Назначение паттернов "Repository" и "UnitOfWork"
3. Смысл инверсии управления и внедрения зависимостей. Привести пример в коде программы.

Основные этапы работы.

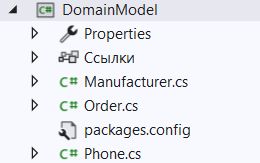
***Часть 1. Создание структуры решения в соответствии с принципами Clean Architecture.***

1. Основа приложения содержит доменную модель и интерфейсы для взаимодействия с данными и выполнения бизнес-логики.

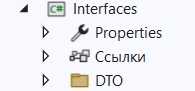




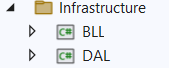
В проекте DomainModel содержатся сущности предметной области (добавить готовые классы сущностей из предыдущей л.р.):



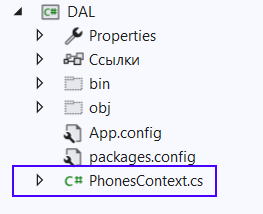
В проект Interfaces добавить модели DTO из предыдущей л.р. Для взаимодействия с данными здесь также будут находиться интерфейсы сервисов интерфейсы репозиториев, интерфейс UnitOfWork для работы с репозиториями из сервисов приложения (создание описано далее в части 2).



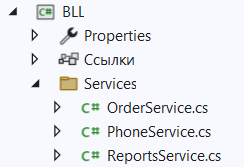
1. Инфраструктуру составляют слои бизнес-логики приложения и доступа к данным БД:



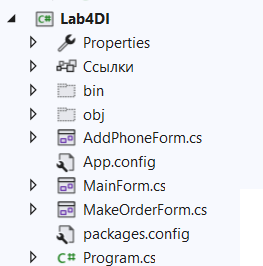
В проект DAL добавить класс контекста данных EF (из предыдущей л.р.)



В проект бизнес логики добавить классы служб приложения (из предыдущей л.р.)

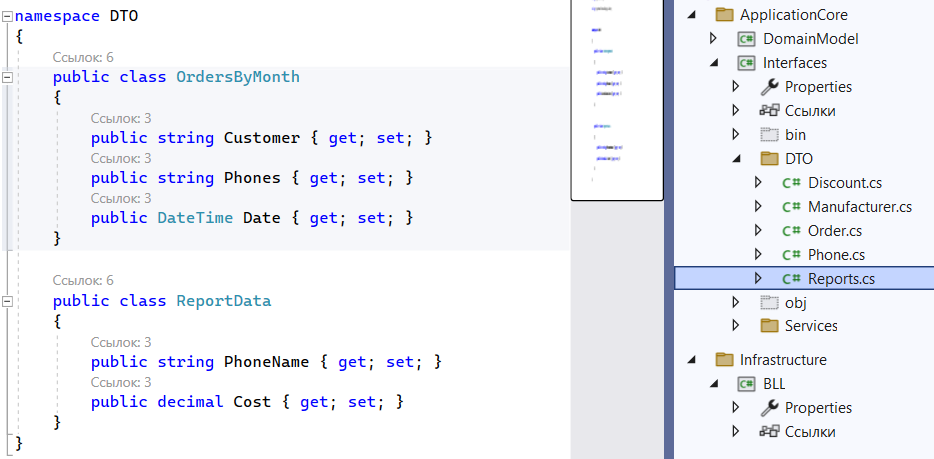


1. Уровень Presentation содержит основной проект WinForms. Здесь находятся все формы из предыдущей л.р. (основное окно, окно редактирования сущности, окно одной функции бизнес-логики):

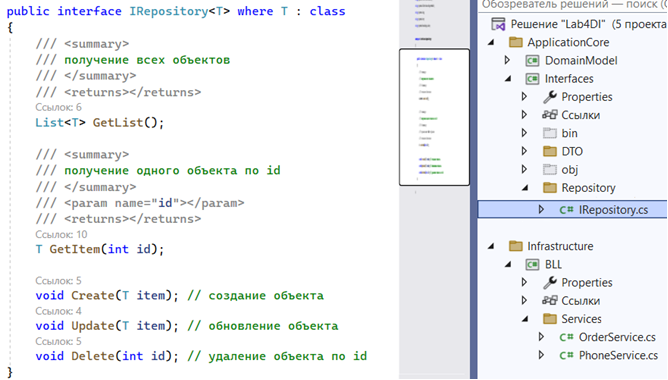


***Часть 2. Создание репозиториев для доменных сущностей.***

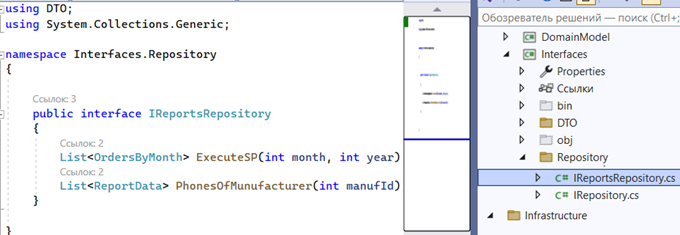
* + - 1. Добавить классы для структур данных, возвращаемых для отчетов, на уровень интерфейсов приложения:



* + - 1. Создать обобщенный интерфейс репозитория для сущностей приложения («Производитель», «Телефон», «Заказ»):

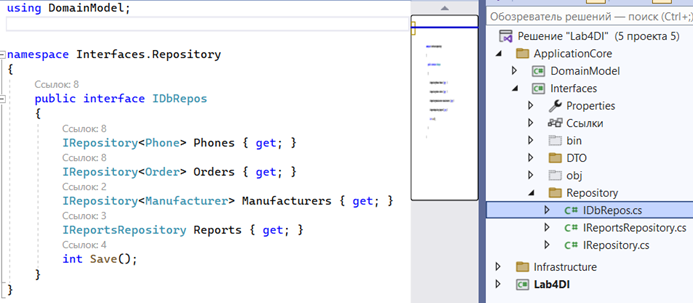


* + - 1. Создать интерфейс репозитория отчетов:

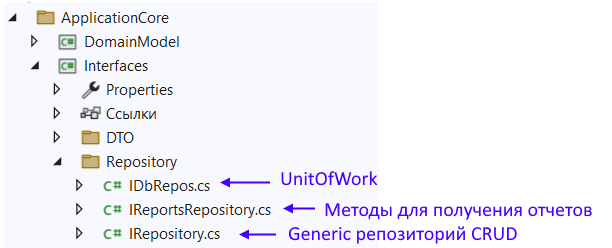


1. Поскольку мы будем использовать несколько репозиториев (для каждой сущности), то для упрощения использования подключения к БД будем использовать паттерн Unit Of Work.

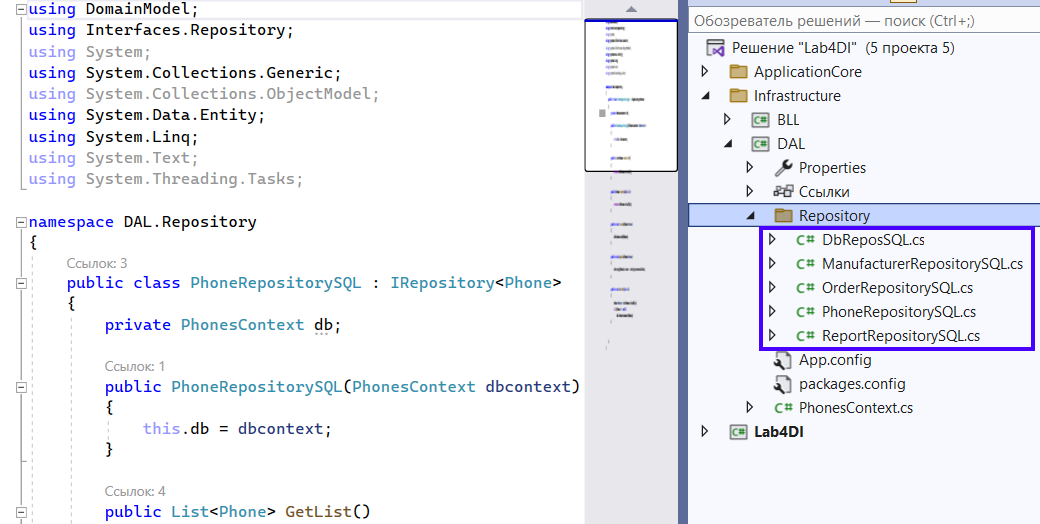
Создать интерфейс для взаимодействия с репозиториями:



Итоговый состав интерфейсов репозиториев:

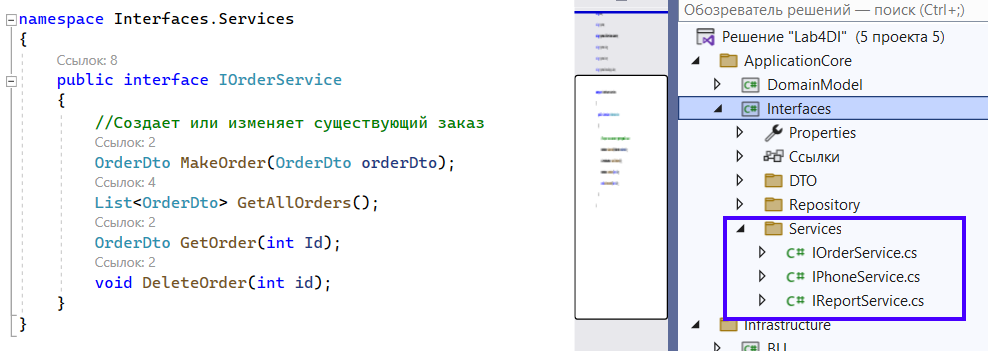


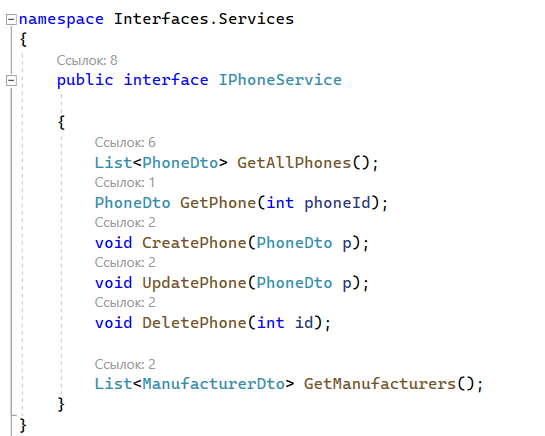
1. Создать реализации репозиториев и UnitOfWork:

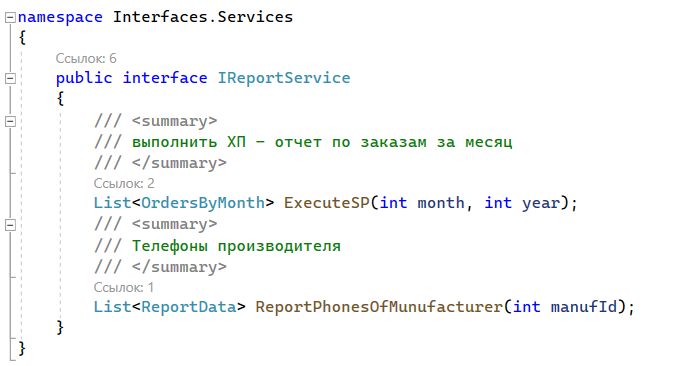


***Часть 3. Применение принципа инверсии управления к слою бизнес-логики и слою взаимодействия с пользователем.***

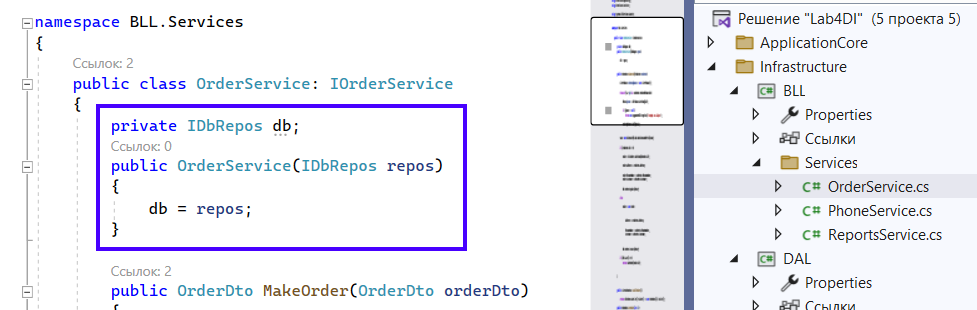
1. Для каждого сервиса приложения создадим интерфейс.

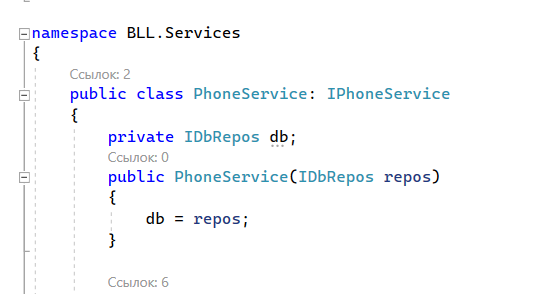


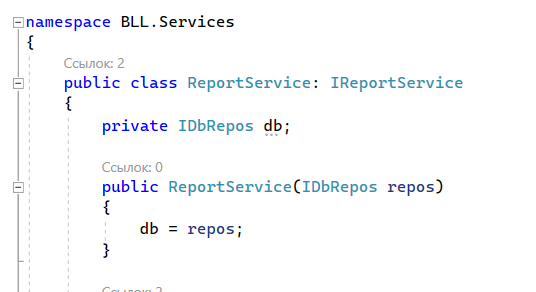




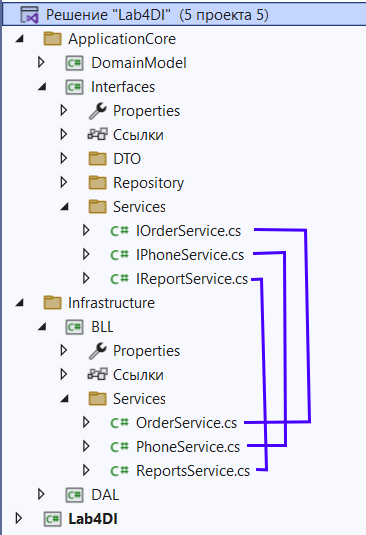
1. Избавимся от жесткой зависимости классов бизнес-логики от класса контекста данных с помощью промежуточных абстракций. Необходимо вынести создание контекста данных во вне. Классы сервисов реализуют интерфейсы приложения.





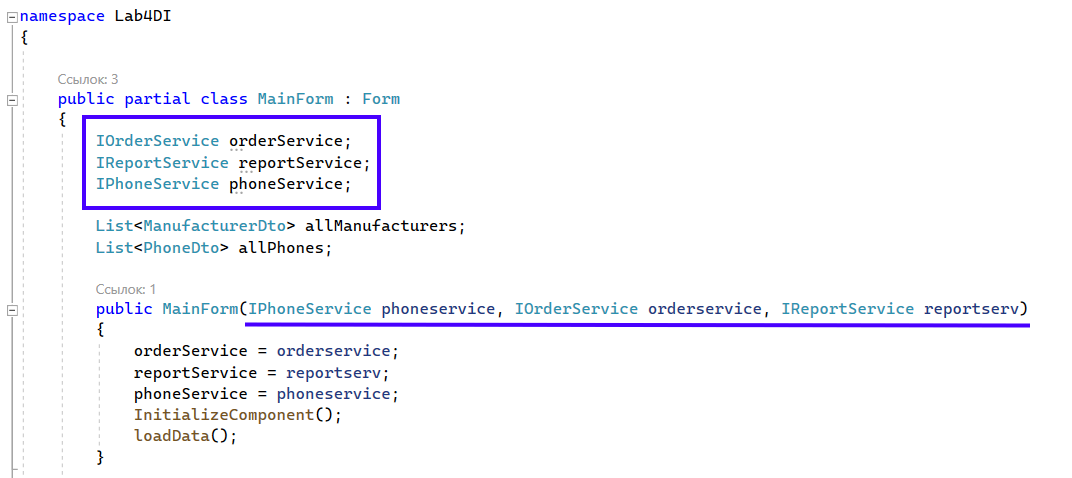


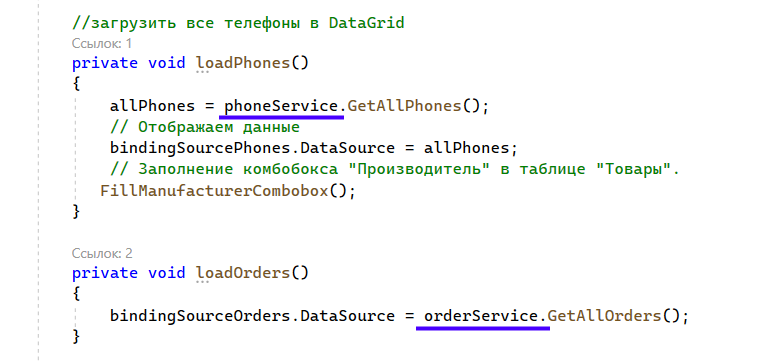
Таким образом, получаем интерфейсы сервисов и их реализации:



1. Избавимся от жесткой зависимости классов уровня представления (WinForm) от классов BLL: взаимодействие с бизнес-логикой будет происходить через интерфейс.

Замена взаимодействия UI формы с методами бизнес-логики на методы интерфейсов:





***Часть 3. Внедрение зависимостей с помощью IoC контейнера Ninject***

Форма теперь не имеет жесткой связи с BLL. При выполнении приложения необходимо передать форме ее зависимости. Для инстанцирования зависимостей воспользуемся библиотекой Ninject.

1. Установить Ninject на уровне представления в проект WinForms через NuGet.
2. Сопоставление с реализациями репозитория UnitOfWork и сервисов сделаем через Ninject-модули.





1. Добавить внедрение зависимостей для формы приложения.

