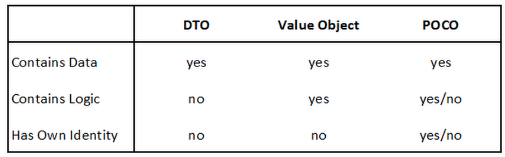
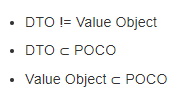
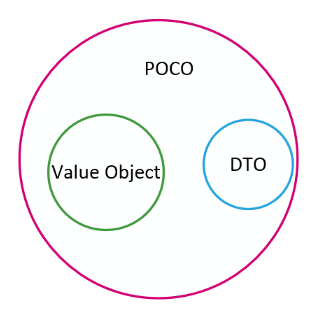
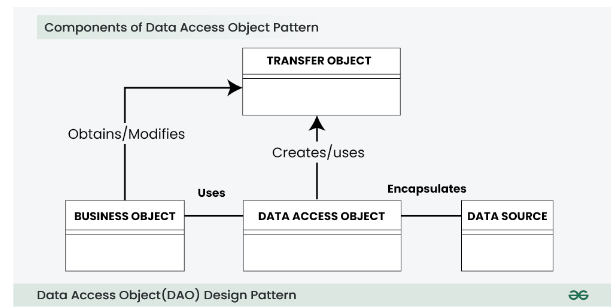
**Паттерны DAO, ORM, Repository**

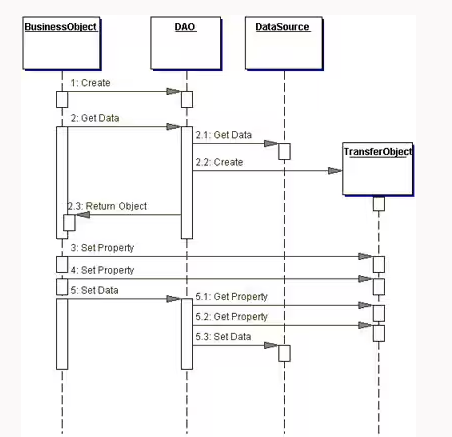
1. **DTO(Data Transfer Object)**: класс, содержащий данные без логики для работы с ними (структуры); применяется для передачи данных между различными приложениями или между слоями одного и того же приложения; рассматривается как хранилище данных, единственная цель которого – передать эти данные.
2. **Value Object:** объект DDD-домена; подчиняется тем же правилам, что и Entity, но у него нет идентичности – два Value-объекта с одинаковыми свойствами не различаются.
3. **POCO (Plain Old CLR Object):** аналогия POJO (Java), могут содержать логику, может быть и Value и Entity.

****

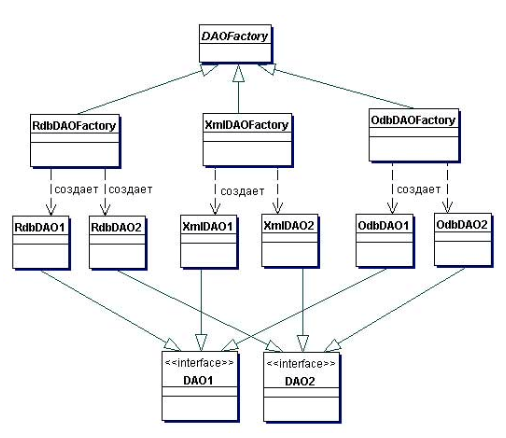
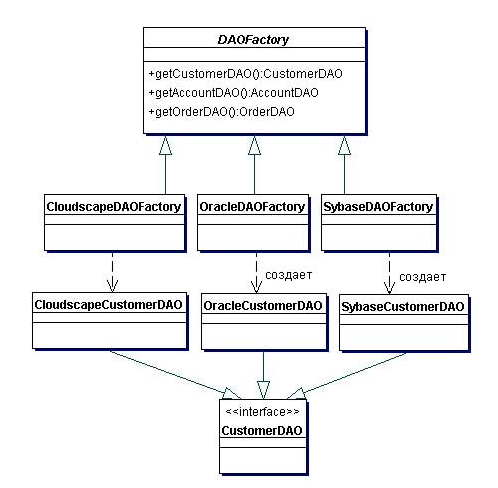
1. **DAO:** Data Object Access; паттерн проектирования; абстрактный интерфейс к БД; класс, содержащий CRUD (Create, Read, Update, Delete) для конкретной сущности БД; отделяет бизнес-логику от механизмов сохранения; четко не описан.

****

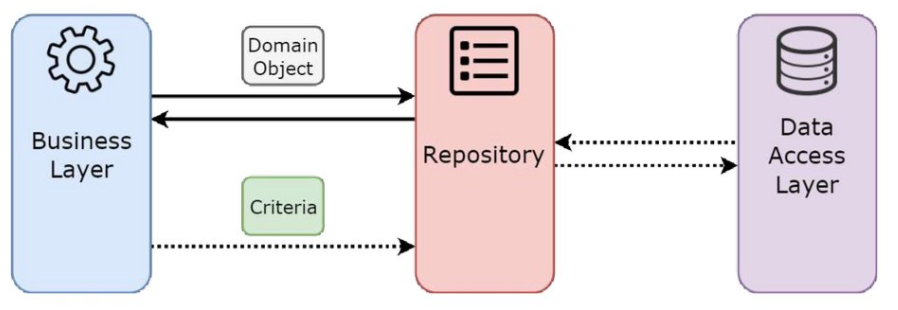
1. **DAO: Business Object** использует DAO-объект для работы с данными; DAO-Объект создает Transfer Object для объекты работы с данными; Data Source – источник данных для подключения.



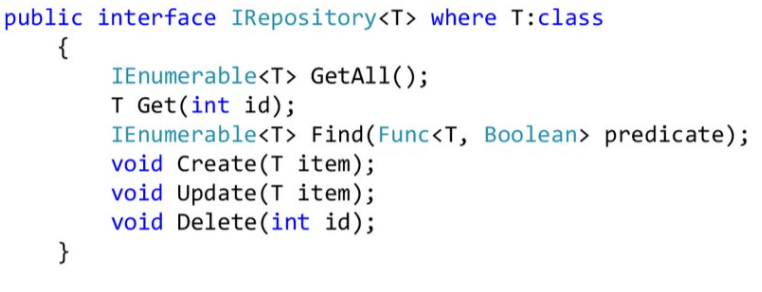
1. **DAO/Abstract Factory:** DAOFactory - абстрактный класс/интерфейс; RdbDAOFactory, XmlDAOFactory, OdbDAOFactory.

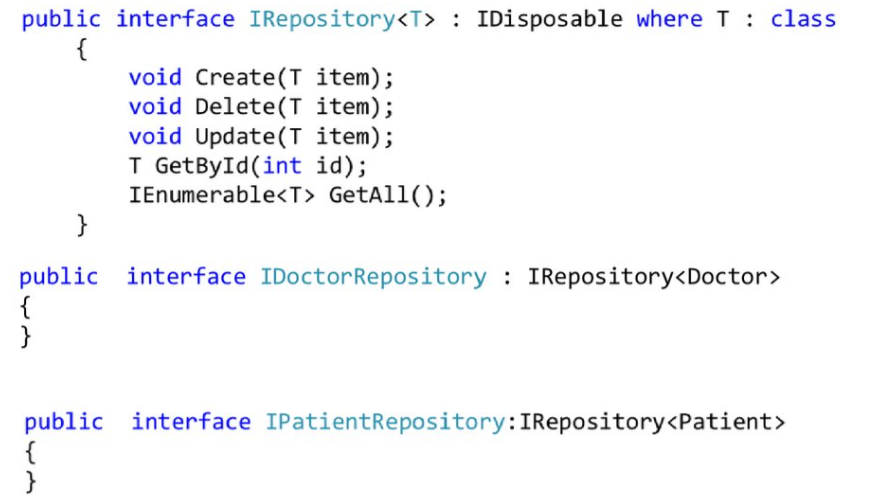


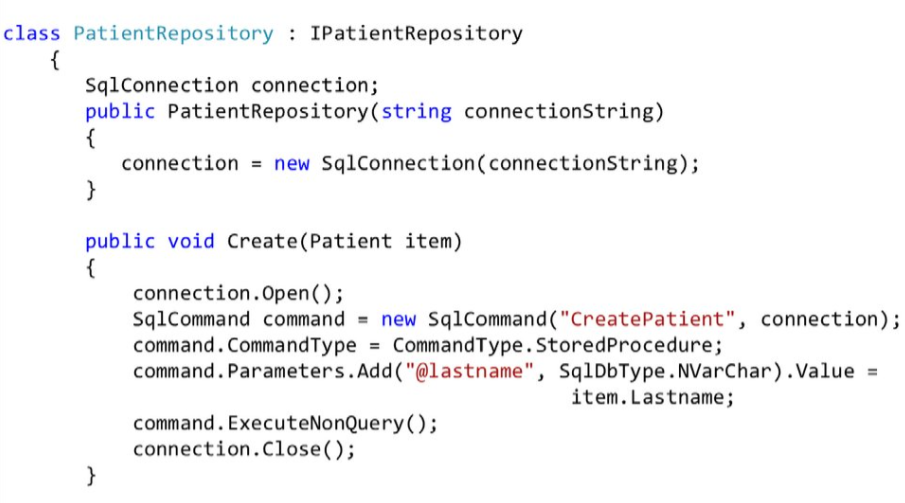
1. **Repository:** паттернпроектирования, применяемый при работе с данными; предназначен для разделения бизнес-логики и от слоя доступа к данным. Два подхода DDD: Database Drive Design (разработка начинается с разработки БД), Domain Drive Design (разработка начинается с проектирования бизнес-логики (домена), Эрик Эванс). Repository ассоциируется с Domain Drive Design.

****

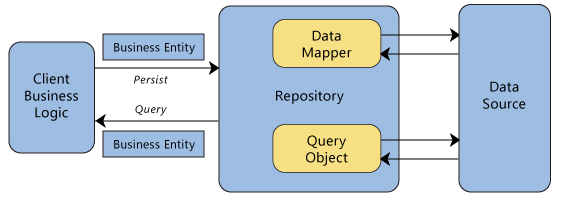
1. **Repository:** посредник между слоем доступа к данным и доменным слоем; in-memory коллекция доменных объектов.
2. **Repository:** клиенты создают декларативные описания запросов и передают их в репозиторий для выполнения. Мартин Фаулер.
3. **Repository:** выполняет CRUD; хорошая практика – отдельный репозиторий для каждого бизнес объекта (POCO – без кода) или контекста.



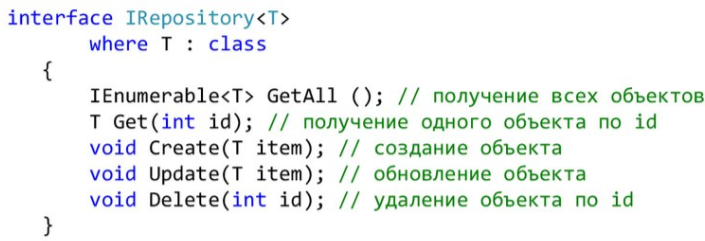


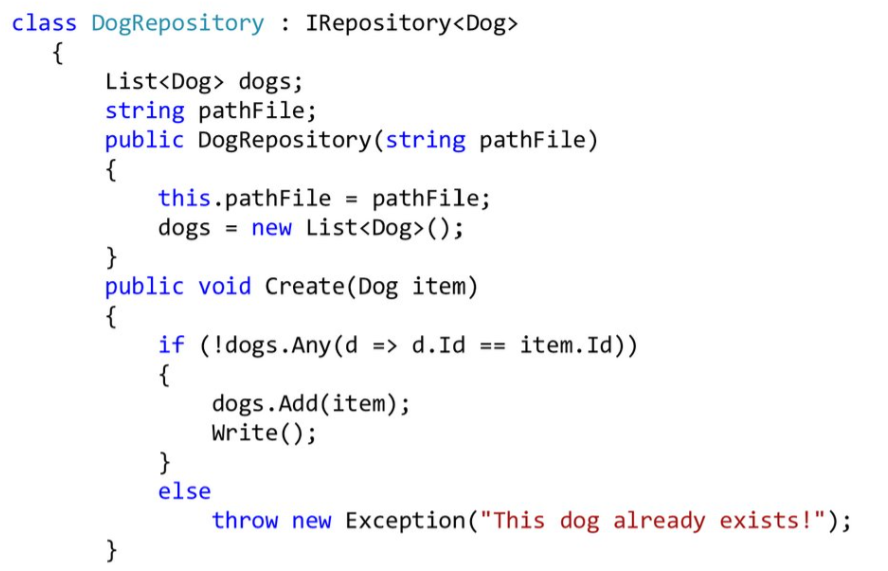


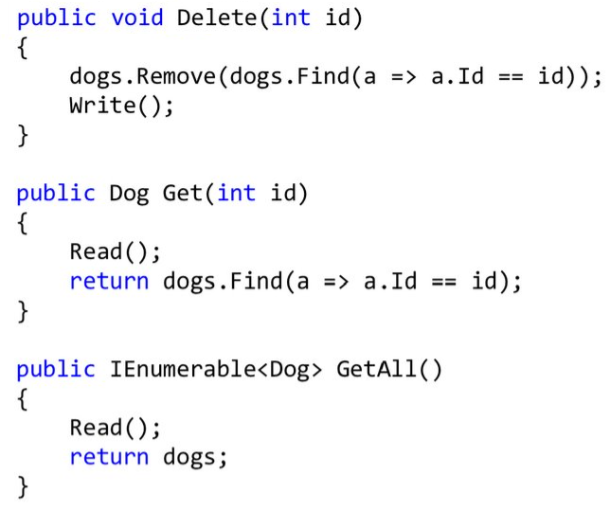
1. **Repository:** Data Mapper, Query Object.

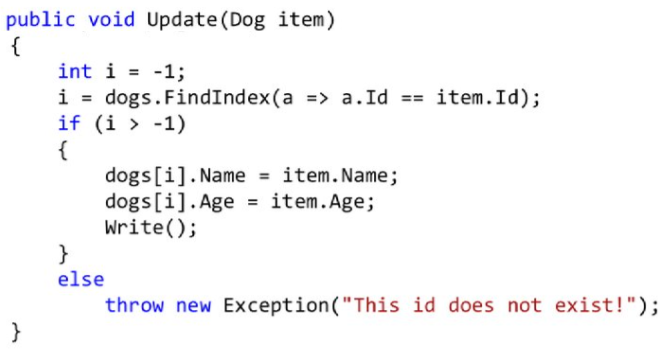


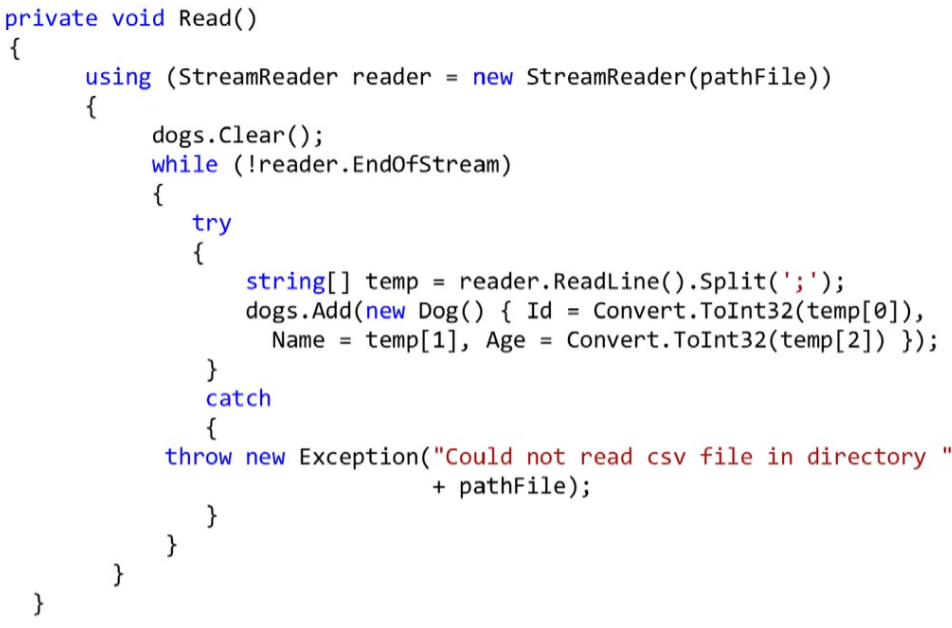
1. **Repository:** независимость бизнес-логики и способа хранения; применение бизнес-объектов вместо daтаbase-related object; возможность использовать разные способы хранения: ORM, Cloud Storage (S3), File System,…; если использовать для репозитория интерфейс, то можно иметь несколько реализаций репозитория.
2. **Repository:** объект Dog в файловом хранилище

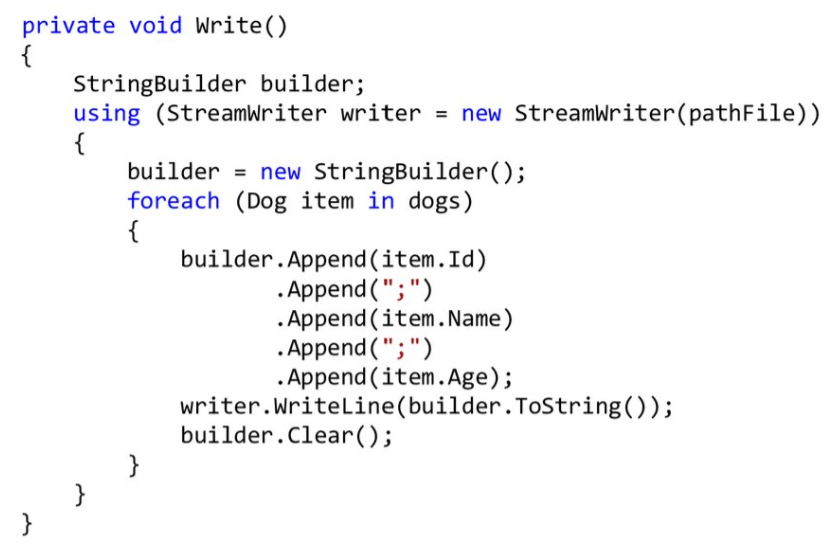
****

****

****

****

****

****

1. **Entity**
2. **ORM:** Object-Relation Mapping, паттерн проектирования, применяется для работы с данными, концепция БД/ООП, ООП-интерфейс с БД, стандарт IT-индустрии.



1. **ORM/Java**: Hibernate, JPA
2. **ORM/Python/Django**: Django ORM, SQLAlchemy
3. **ORM/JavaScript/Node.js:** Sequelize, TypeORM
4. **ORM/Ruby:** ActiveRecord (on Rails)**,** Sequel
5. **ORM/PHP:** Doctrine
6. **ORM/C#:** Entity Framework
7. **ORM:** БД, таблица, строка таблицы, структура таблицы, первичные ключи, внешние ключи.
8. Системные свойства реализаций:

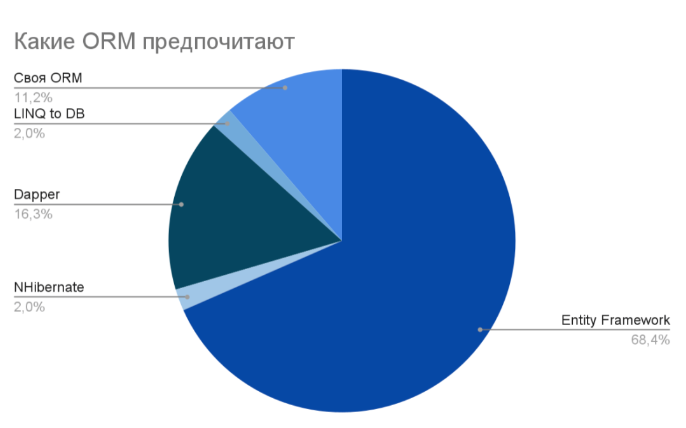
*Eager Loading* — жадная загрузка связанных объектов сущности одновременно с основным объектом; предотвращает множество дальнейших запросов, плохо сказывается на быстроте выполнения чтения;

*Lazy Loading* — ленивая загрузка, позволяющая оптимизировать процесс чтения и получать связанные объекты только при необходимости;

*Identity Map* — жадное получение объекта из базы данных 1 раз в рамках сессии; позволяет избегать возможного противоречия в данных, возникающих из-за нескольких запросов;

*Unit of Work* — собирает все изменения во всех объектах и позволяет выполнить их в рамках одной транзакции; благоприятно сказывается на консистентности данных и количестве запросов.

1. **ORM**



1. ORM: https://metanit.com/sharp/adonetcore/2.4.php
2. <https://habr.com/ru/articles/667078/>
3. Data Mapper
4. **ADO.NET**: <https://metanit.com/sharp/adonet/1.1.php>
6. **Entity-Component-System(ECS) pattern:** [**https://habr.com/ru/articles/665276/**](https://habr.com/ru/articles/665276/)
8. Hh
9. Hh
10. Hh
11. Hh
12. Hh
13. Hh