Что такое доменно-ориентированный подход (Domain-Driven Design, DDD) в разработке программного обеспечения.

**1. Оригинальное название:**

Оригинальное название этого подхода на английском языке — **Domain-Driven Design (DDD)**.

**2. Суть доменно-ориентированного подхода (DDD):**

Суть DDD заключается в том, чтобы *сделать модель предметной области (домена) центральной частью разработки программного обеспечения*. Вместо того чтобы начинать разработку с технических аспектов (базы данных, интерфейсы и т.п.), DDD призывает начать с *понимания* и *моделирования* *бизнес-области* (домена), для которой создается ПО.

**Основные идеи DDD:**

* **Углубленное понимание предметной области (домена):**
  + DDD предполагает глубокое изучение *бизнес-процессов*, *правил*, *сущностей* и *контекста*, в которых работает программное обеспечение.
  + Это требует тесного взаимодействия между разработчиками и экспертами в предметной области.
* **Общий язык (Ubiquitous Language):**
  + DDD призывает разработчиков и экспертов предметной области использовать *один и тот же язык*, чтобы избежать недопонимания и двусмысленностей.
  + Этот язык формируется на основе терминологии и понятий из предметной области.
* **Разработка на основе моделей:**
  + DDD неразрывно связан с созданием *моделей предметной области*, которые представляют *бизнес-сущности* и *взаимодействия* между ними.
  + Модели должны быть понятными для всех участников процесса разработки.
* **Итеративная разработка:**
  + DDD обычно применяется в рамках гибких методологий, где разработка ведется итеративно.
  + Модель домена постоянно уточняется и дополняется по мере углубления понимания предметной области.
* **Стратегический и тактический DDD:**
  + DDD может применяться на *стратегическом уровне* для *определения границ предметных областей* и *контекстов*, а также на *тактическом уровне* для *моделирования конкретных сущностей* и *бизнес-процессов*.

**3. Структура какой части решения:**

DDD в первую очередь касается **бизнес-логики** и **структуры предметной области (домена)**. Это означает, что DDD влияет на следующие части программного решения:

* **Модель предметной области (Domain Model):**
  + Это *основная часть* решения в DDD. Она включает *сущности* (например, Customer, Order, Product), *агрегаты* (группы связанных сущностей), *сервисы домена* (бизнес-логика, которая не принадлежит конкретной сущности) и другие *концепции предметной области*.
  + Модель предметной области *определяет*, как *бизнес работает*, и *является основой* для написания кода.
* **Слой приложения (Application Layer):**
  + Этот слой отвечает за *координацию действий* и *выполнение сценариев использования* (*use cases*) приложения.
  + Он *обращается* к *модели предметной области* для выполнения *бизнес-логики*, но *не содержит* её.
* **Слой инфраструктуры (Infrastructure Layer):**
  + Этот слой отвечает за взаимодействие с *внешними системами* (базы данных, сервисы и т.д.).
  + *Используется* для *реализации механизмов хранения* данных, *взаимодействия* с внешними сервисами, но *не содержит* логики домена.

**DDD не фокусируется на:**

* **Пользовательских интерфейсах (UI):** DDD не определяет, как должны выглядеть интерфейсы, а только то, как должны взаимодействовать объекты внутри домена.
* **Конкретных технологиях:** DDD не привязан к конкретным языкам программирования, базам данных или фреймворкам. Это *архитектурный подход*, который можно использовать в различных контекстах.

**В итоге:**

* **Domain-Driven Design (DDD)** – это *подход к разработке* программного обеспечения, где *модель предметной области* является *ядром проекта*.
* DDD *помогает разработчикам и бизнес-экспертам* *понимать и моделировать* *бизнес-процессы*, *создавать эффективное ПО* и *разговаривать на одном языке*.
* DDD влияет на *структуру бизнес-логики* (*модели домена*) и *её взаимодействие* с *другими слоями приложения*.
* DDD *не задает* конкретных *технических решений* и *не касается* UI.