DDD（Domain-Driven Design）脑图

战术设计

战术设计则是在战略设计的基础上，对领域中的具体问题进行具体的解决方案设计。**战术设计关注的是领域中的具体情境和场景，需要针对具体的问题进行具体的分析和设计，以满足业务需求。实体、值对象、聚合、工厂、资源库、领域服务和领域事件就属于战术设计的范畴。**

聚合根

聚合根（Aggregate Root）是聚合的唯一入口和核心管理者，负责维护聚合内部的一致性和业务规则，外部只能通过它访问和操作聚合内的其他对象，例如：电商系统中的"订单"作为聚合根，控制订单项、支付信息等子对象的生命周期和状态变更。

实体

具有唯一标识符（ID）且承载业务逻辑的领域对象，其身份标识在生命周期内保持不变（如订单、用户），即使属性变化仍视为同一对象。

值对象

通过属性值定义且无唯一标识的不可变对象，用于描述业务中的某个概念整体（如金额、地址），Domain Primitive 在值对象的基础上增加了自我验证、行为封装和业务语义完整性的特性。

领域服务

领域服务（Domain Service）是用于封装跨实体或值对象的复杂业务逻辑的无状态组件，当某个操作无法自然地归属于单一领域对象时，便由领域服务协调处理（如银行转账、订单折扣计算）。

领域事件

DDD的领域事件是业务领域中已发生的、具有重要业务意义的状态变化（如"订单已创建"），用于解耦系统组件并通过发布/订阅机制驱动后续业务逻辑。

仓储

仓储（Repository）是领域层与数据存储层之间的抽象中介，它像内存集合一样管理聚合根的持久化与检索，隐藏底层数据库细节，使领域模型保持技术无关性。

关键点：

接口定义在领域层，实现放在基础设施层，确保业务逻辑不依赖具体存储技。

仅通过聚合根操作数据，维护聚合内的一致性。

提供统一语言，如 `GetAllActiveUsers()` 方法名直接体现业务意图。

类比：仓储如同仓库管理员，业务层只需说"取订单ID=123"，无需关心订单存在MySQL还是Redis中。

工厂

工厂（Factory）是封装复杂领域对象创建逻辑的模式，将对象的构造与初始化职责从聚合根中剥离，确保聚合内依赖对象的一致性和生命周期管理。

模块

模块（Module）是按业务能力划分的高内聚代码单元，用于组织领域模型中的相关类（如实体、值对象），确保语义清晰且减少耦合。

战略设计

战略设计指的是对整个领域进行分析和规划，确定领域中的概念、业务规则和领域边界等基础性问题。在战略设计中，需要对领域进行全面的了解和分析，探究业务的规则和本质，并且需要考虑到领域的未来发展趋势和可能的变化。**领域、子域和限界上下文属于战略设计的范畴。**

统一语言

开发团队与业务专家共同定义的、无歧义的业务术语体系，确保从需求到代码的全程概念一致性。

领域

领域(Domain)其实就是一个组织所要做的整个事情，以及这个事情下所包含的一切内容。这是一个范围概念，而且是面向业务的(注意这里不是面向技术的，更不是面向数据库的持久化)，每个组织都有自己的人员、规则和流程，当你为该组织开发软件的时候，你面对的就是这个组织的领域。如:知识付费领域。

限界上下文

限界上下文就是业务边界的划分，这个边界**可以是一个子域或者多个子域的集合。**如何进行划分，一个行之有效的方法是一个界限上下文必须支持一个完整的业务流程，保证这个业务流程所涉及的领域都在一个限界上下文中。**限界上下文是微服务拆分的依据，即每个限界上下文对应一个微服务。**

核心域

决定公司和产品核心竞争力的子域就是核心域，它是业务成功的主要因素。核心域直接对业务产生价值。

通用域

没有太多个性化的诉求，同时被多个子域使用的、**具有通用功能的子域就是通用域**。通用域间接对业务产生价值。

支撑域

支撑其他领域业务，**具有企业特性，但不具有通用性。**支撑域间接对业务产生价值。