Domain Driven Design (DDD)

DDD is an object-oriented software design practice that you can find lots of in-depth reading on, and work on practicing in your own programming. If you search the web or AI you can find good reading on it from Martin Fowler and Microsoft's Learn pages, or you can get the original book Domain-Driven Design: Tackling Complexity in the Heart of Software, by Eric Evans. This poster briefly describes the basics, with my own take on it.

The Problem Domain Provides Language and Objects (Classes)

Understanding your problem domain comes first. Figure out the *language* of your domain, create a domain model of entities, their attributes, actions, and other concepts. This domain language should show up in your code.

Focus on Entities, Attributes, and Relationships First

Both Fred Brooks and Linus Torvalds (Linux creator) have said, essentially, "data design comes first". So first figure out your domain entities, their types, and their attributes. Entity types will become classes, entities objects, and their attributes are data members. Ignore code functionality design for now.

Design Domain-Level Computational Flows

Identify and then describe activity flows, workflows, and computational flows that happen in your problem domain. Then make these happen across *cooperating objects*. Create class methods so that objects encapsulate their own part of the computation flow; use object parameters or existing relationships (as reference attributes) where multiple objects must be involved in a computation. Avoid having a God Class that does everything.

Identify Solution-Side Concepts and Entities

You will need classes and objects that simply manage the program itself and interact with the computer and outside world. You can use the same ideas above to understand your solution-side entities and concepts.

Put It All Together

The solution side objects control the program, interact with the user and world, and manage and invoke the domain objects. OOP == making a bunch of active, cooperating objects!

Copyright 2026 Jonathan Cook. Domain Driven Design is most definitely NOT my original idea.