Domain Model => Customer need => Java Code.
⇑
Problem to resolve

Ref Architecture: Problem => ~~Ref~~ relation to Ref Arch

System vs. Programming:

System: ~~building~~ outsider requirement.

Ecosystem => ~~inter~~ top-down design pattern.
collaborative & unify system.

Design Principles: Hierarchy: Campus => Building => Room

Modularity: Specialization component

Encapsulation: Data accessability via interface

Algorithm:

logical steps. implicit process with visible input & output

Data Structure:

explicit. focus on the transactive components.

System => components.

Algorithm to navigate data through components to express the requirements.

Domain Model —Java (Language) / OOP technique→ Executing Digital System. (Application)

Java: Classes:

Objective:

Instiation:

Relationship:

Attributes

domain

↓

workflow

↓

concept

↓

interactions.

Mythology:

Problem => Design => Spec => Implementation.

System: organized set of parts + interacting elements

Domain model put human into the system to go beyond the system

Application structure:

Data -> user activity -> roles -> users.

Ecosystem: network of organizations: competition & collaboration.