좋아, "코드 작성"을 **온톨로지**(무엇이 있고 어떻게 얽히는가)로 풀어보면 이렇게 보인다.

---

0) 한 줄 정리

이 프로젝트는 **원칙(Constitution)**이 **작업 흐름(Commands)**을 **산출물(Artifacts)**로 이어 붙이고, 각 흐름은 **게이트(Quality Gates)**로 검증되며, 모든 조각이 **명세–계획–과업–분석–구현**이라는 **유형화된 과정 타입들**로 상호 제약되는 체계다. 헌법 규칙은 상위 규범이며 분석 단계에서 "불가침"으로 집행된다.

---

1) TBox(스키마): 핵심 클래스와 관계

클래스(개념)

- **Project**: 코드 작성 전체를 아우르는 최상위 개체.
- **Principle**: 변경 불가한 규범. *Constitution* 문서로 구체화된다(예: 의사결정·품질 기준).
- **Command(ProcessType)**: /specify·/clarify·/plan·/tasks·/analyze·/implement 같은 실행 가능한 과정 유형. 각 커맨드는 입력/출력 산출물이 정형화되어 있다.
- **Artifact**: spec.md, plan.md, tasks.md, research.md, data-model.md, contracts/, quickstart.md 같은 파일 산출물.
- **QualityGate**: "진행해도 되는가?"를 판단하는 규범/검증 모음. 예: 헌법 준수 체크, TDD/Tidy 루프, 선행 명세/정리 유무.
- **State**: Artifact/Feature의 생애주기 상태(초안 Draft → Clarified → Planned → Tasked → Analyzed → Implemented).
- **Actor**: 참여 주체(개발자/리뷰어/자동 에이전트).
- **Tool**: 테스트/린트·스크립트 등(예: 계획/검증 스크립트, 테스트 러너). 커맨드가 이를 호출한다.

핵심 관계

- governs(Principle, Command/Artifact): 헌법 원칙이 프로세스와 산출물을 규율.
- produces(Command, Artifact): 각 커맨드가 특정 산출물을 생성/갱신.
- requires(Command, Artifact|Principle): 실행 전제조건(예: /plan은 **Clarifications**가

있어야 함).

- validates(QualityGate, Artifact|Process): 분석/게이트가 요구사항 충족 여부를 판정(헌법 충돌은 자동 *CRITICAL*).
- orders(ProcessType, ProcessType): 수행 순서 제약(예: 테스트→코드, 구조→행위).

---

2) 프로세스 타입(Commands)로 본 수명주기

1. **/specify** → *특징*: 브랜치와 스펙 파일을 생성·체크아웃하고 템플릿에 맞춰 **spec.md**를 초기화한다. 산출: SPEC_FILE, 분기: BRANCH_NAME.
2. **/clarify** → *역할*: 최대 5개의 정밀 질문으로 **모호성 해소** 후, 답을 **Clarifications** 섹션과 관련 섹션(요구사항·데이터모델·품질 속성 등)에 **즉시 반영**한다. 상태 전이: Draft → Clarified.
3. **/plan** → *전제*: Clarifications가 존재해야 진행. *산출*: **research.md**, **data-model.md**, **contracts/**, **quickstart.md**, 그리고 최종적으로 **tasks.md**(Phase2).
4. **/tasks** → *규칙*: 의존성 순서와 병렬성([P])을 명시. "**테스트 우선(TDD)**, 모델→서비스→엔드포인트, 코어→통합→폴리시" 등 정형 규칙을 반영해 **실행가능한** 작업 목록을 생성.
5. **/analyze** → *성격*: **읽기 전용** 교차검증. Spec/Plan/Tasks 간 **중복·모호·미커버리지·헌법 충돌**을 표 형태로 보고하고 심각도(CRITICAL/HIGH/...)를 부여. 헌법 위반은 자동 CRITICAL.
6. **/implement** → *집행*: **tasks.md**의 단계/의존/병렬 규칙에 따라 **단계별** 구현. 실패 시(비병렬) 중단·보고, 병렬은 성공분 계속. 완료 시 체크박스를 [X]로 갱신.

---

3) 규범(Principles)과 게이트(Quality Gates)

- **TDD 루프(RED→GREEN→REFACTOR)**: 기능 한 조각의 실패 테스트(RED) → 통과에 필요한 **최소 구현**(GREEN) → **중복 제거/의도 드러내기**(REFACTOR). 단일 테스트는 ms~수백 ms SLA, 느린 I/O는 더블/주입.
- **Tidy First(구조→행위 분리)**: 리네임/추출/이동 등 **구조 변경**은 행위 불변으로 먼저, 그 다음 **행위 변경**(feat/fix). 커밋도 structural: ↔ behavioral:로 분리.
- **진입 게이트**: /plan은 명시적으로 **Clarifications** 유무를 검사(없으면 중단·사전 정

리 권고). /analyze는 헌법 위반을 **CRITICAL**로 지정.

이 세 가지는 온톨로지 상에서 QualityGate 인스턴스들로 모델링되어 각 Process 인스턴스에 연결된다. 실패 시 **State**는 다음 단계로 전이되지 않는다(불합격 전이 차단).

---

4) ABox(예시 인스턴스) — 한 기능이 지나가는 이야기

- Process(/specify) **produces** Artifact(spec.md) & **creates** Branch(feature/x).
- Process(/clarify) **updates** Artifact(spec.md#Clarifications); 상태 Draft→Clarified.
- Process(/plan) **produces** research.md, data-model.md, contracts/, quickstart.md, tasks.md.
- Process(/tasks) **orders** 작업: **Setup→Tests→Core→Integration→Polish**, 파일 충돌 시 **순차**, 파일 분리 시 **[P] 병렬**.
- Process(/analyze) **validates** (dup/ambiguity/coverage/constitution) ⇒ **OK**면 다음.
- Process(/implement) **executes** tasks; 실패 시 중단·리포트, 성공은 [X] 체크.

---

5) 소형 온톨로지 스케치(의미 보존용 의사-Turtle)

Classes

Project, Principle, Command, Artifact, QualityGate, State .

Object properties

governs, produces, requires, validates, orders, updates, creates, blocksTransition .

Instances (축약 예)

:Constitution a Principle .

:TDD_Gate a QualityGate .

:TidyFirst_Gate a QualityGate .

:CodeWriting a Project .

:Specify a Command ; produces :SpecMD ; creates :FeatureBranch .

:Clarify a Command ; requires :SpecMD ; updates :SpecMD .

:Plan a Command ; requires :SpecMD ;
produces :Research, :DataModel, :Contracts, :Quickstart, :TasksMD .

:Tasks a Command ; requires :Plan ; produces :TasksMD .

:Analyze a Command ; validates :SpecMD, :PlanMD, :TasksMD ;
governedBy :Constitution .

:Implement a Command ; requires :TasksMD ; validates :TDD_Gate, :TidyFirst_Gate .

규칙 예시

:Analyze --governedBy--> :Constitution . # 헌법 불가침

:TDD_Gate --blocksTransition--> :ImplementWhenFailed .

- 헌법 규칙의 "불가침"과 분석의 읽기 전용·심각도 체계는 /analyze 문서의 규정으로 근거화된다.
- Clarify의 "최대 5문항"·분류 택소노미·즉시 반영 규칙은 /clarify 흐름 정의로부터 온다.
- Plan의 단계 산출물 세트와 Tasks의 의존·병렬 규칙은 각 명세에 기재되어 있다.
- TDD/Tidy 루프와 구조↔행위 분리는 Tidy 지침에 명시되어 있다.

---

6) 이 온톨로지가 답해야 하는 Competency Questions

- 현재 Feature X는 **어떤 상태**이며(Clarified? Planned?) 다음 단계로 **전이 가능한가?** 전이를 막는 **게이트 실패**는 무엇인가? (예: Clarifications 없음, Analyze CRITICAL)
- spec/plan/tasks 사이에 **중복·충돌·미커버리지**가 있는가? 있다면 **심각도**와 **수정 후보**는?
- 특정 과업 T###는 **어떤 요구사항**을 충족하며, **병렬 실행** 가능한가(파일 경합 여부)?

- 이번 커밋은 **구조 변경**인가 **행위 변경**인가? (분리 커밋/리뷰 기준)

---

7) 함께 볼 메타-원칙(현실과의 접점)

- **plan.md 최우선 & RED→GREEN→REFACTOR & 구조→행위**: 설계→검증→집행 순서가 지식 구조로서 먼저 서야, 실제 코드 변화가 안정적으로 수렴한다.
- **/plan 전 Clarifications 필수**: 모호성은 가장 비싼 결함. 전방에서 줄인다.
- **헌법 우선권**: 원칙을 바꾸려면 별도 절차(버전/개정 내역/동기)가 필요하다.

다음 선택지(가벼운 옵션)

1. 위 스키마를 **TTL/OWL 초안**으로 뽑아드릴 수 있다(프로세스·산출물·게이트·상태 포함).
2. 현재 진행 중인 기능 하나를 골라 **ABox 인스턴스**(실제 삼중)로 채워 시각화해보자.
3. /analyze 리포트 항목을 **온톨로지 질의**(competency questions)로 바로 매핑하는 규칙표를 만들어 드릴 수도 있다.

원하면 바로 골라줘. 난 틀을 잡아주고, 너는 도메인 언어를 더해 주면 된다.
이 구조는 네가 원하는 "온톨로지 관점의 설계—운영—검증"을 같은 축 위에 올려준다.