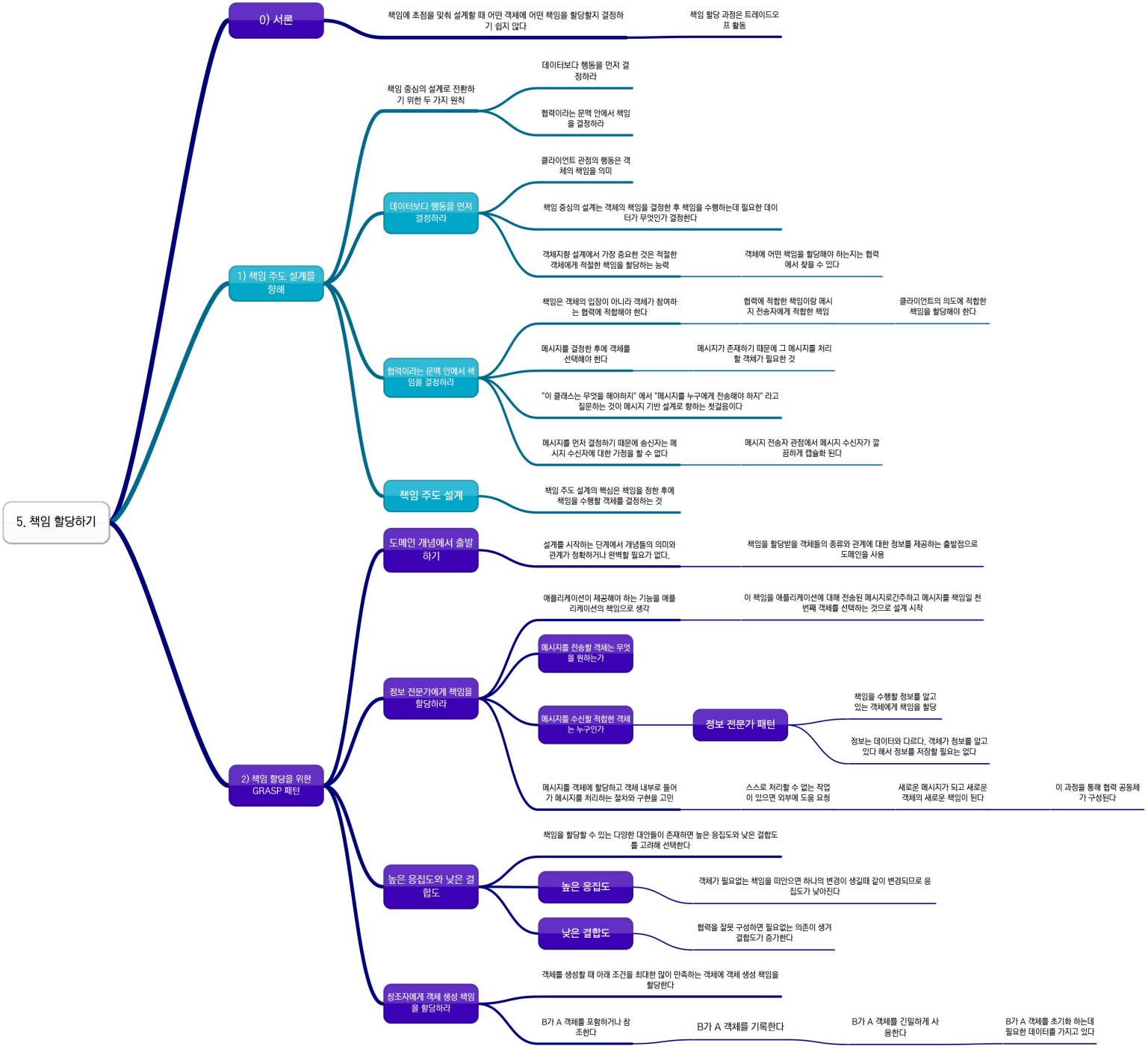
5. 책임 할당하기
0) 서론
책임에 초점을 맞춰 설계할 때 어떤 객체에 어떤 책임을 할당할지 결정하기 쉽지 않다
책임 할당 과정은 트레이드오프 활동
1) 책임 주도 설계를 향해
책임 중심의 설계로 전환하기 위한 두 가지 원칙
데이터보다 행동을 먼저 결정하라
협력이라는 문맥 안에서 책임을 결정하라
데이터보다 행동을 먼저 결정하라
클라이언트 관점의 행동은 객체의 책임을 의미
책임 중심의 설계는 객체의 책임을 결정한 후 책임을 수행하는데 필요한 데이터가 무엇인가 결정한다
객체지향 설계에서 가장 중요한 것은 적절한 객체에게 적절한 책임을 할당하는 능력
객체에 어떤 책임을 할당해야 하는지는 협력에서 찾을 수 있다
협력이라는 문맥 안에서 책임을 결정하라
책임은 객체의 입장이 아니라 객체가 참여하는 협력에 적합해야 한다
협력에 적합한 책임이랑 메시지 전송자에게 적합한 책임
클라이언트의 의도에 적합한 책임을 할당해야 한다
메시지를 결정한 후에 객체를 선택해야 한다
메시지가 존재하기 때문에 그 메시지를 처리할 객체가 필요한 것
"이 클래스는 무엇을 해야하지" 에서 "메시지를 누구에게 전송해야 하지" 라고 질문하는 것이 메시지 기반 설계로 향하는 첫걸음이다
메시지를 먼저 결정하기 때문에 송신자는 메시지 수신자에 대한 가정을 할 수 없다
메시지 전송자 관점에서 메시지 수신자가 깔끔하게 캡슐화 된다
책임 주도 설계
책임 주도 설계의 핵심은 책임을 정한 후에 책임을 수행할 객체를 결정하는 것
2) 책임 할당을 위한 GRASP 패턴
도메인 개념에서 출발하기
설계를 시작하는 단계에서 개념들의 의미와 관계가 정확하거나 완벽할 필요가 없다.
책임을 할당받을 객체들의 종류와 관계에 대한 정보를 제공하는 출발점으로 도메인을 사용
정보 전문가에게 책임을 할당하라
애플리케이션이 제공해야 하는 기능을 애플리케이션의 책임으로 생각
이 책임을 애플리케이션에 대해 전송된 메시지로간주하고 메시지를 책임일 첫 번째 객체를 선택하는 것으로 설계 시작
메시지를 전송할 객체는 무엇을 원하는가
메시지를 수신할 적합한 객체는 누구인가
정보 전문가 패턴
책임을 수행할 정보를 알고 있는 객체에게 책임을 할당
정보는 데이터와 다르다. 객체가 정보를 알고 있다 해서 정보를 저장할 필요는 없다
메시지를 객체에 할당하고 객체 내부로 들어가 메시지를 처리하는 절차와 구현을 고민
스스로 처리할 수 없는 작업이 있으면 외부에 도움 요청
새로운 메시지가 되고 새로운 객체의 새로운 책임이 된다
이 과정을 통해 협력 공동체가 구성된다
높은 응집도와 낮은 결합도
책임을 할당할 수 있는 다양한 대안들이 존재하면 높은 응집도와 낮은 결합도를 고려해 선택한다
높은 응집도
객체가 필요없는 책임을 떠안으면 하나의 변경이 생길때 같이 변경되므로 응집도가 낮아진다
낮은 결합도
협력을 잘못 구성하면 필요없는 의존이 생겨 결합도가 증가한다
창조자에게 객체 생성 책임을 할당하라
객체를 생성할 때 아래 조건을 최대한 많이 만족하는 객체에 객체 생성 책임을 할당한다
B가 A 객체를 포함하거나 참조한다
B가 A 객체를 기록한다
B가 A 객체를 긴밀하게 사용한다
B가 A 객체를 초기화 하는데 필요한 데이터를 가지고 있다