커피매니저와 함께 하는 객체지향 프로그래밍

1. 서비스에서 사용되는 개념을 보편언어로 정의

- 고유명사나 숫자로 표현 될 수 있는 개념은 제외
- 보편언어(Ubiquitous language) : 모든 사람(고객, PM, 디자이너, 개발자, 기획자...)이 같은 의미로 이해하는 언어

CoffeeManager는 **커피**를 **판매**하는 프로그램이다.

커피 - 커피는 판매자가 고객에게 판매하는 사물이다.

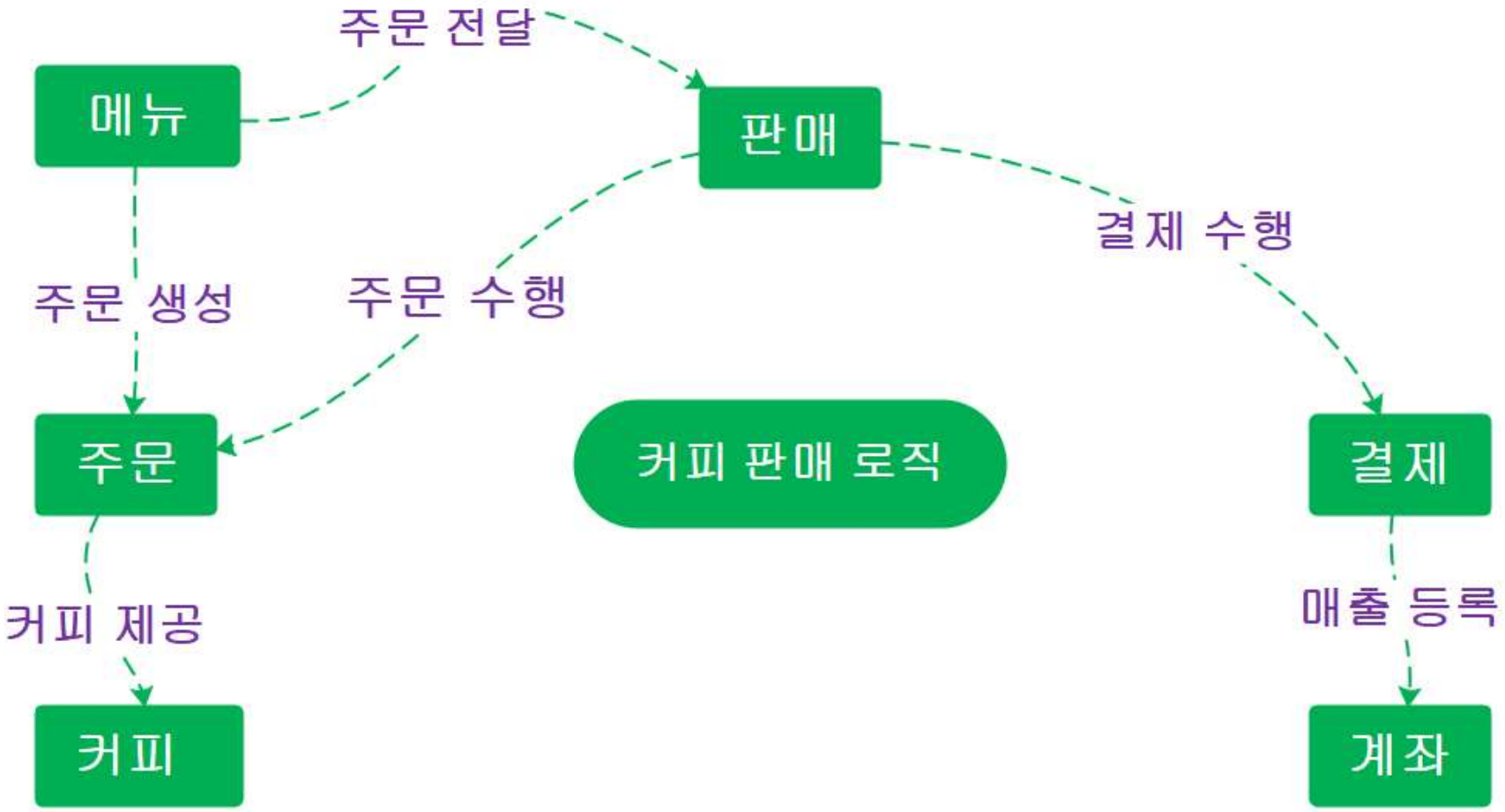
판매 - 판매는 고객이 전달한 **주문**을 수행하고 **결제**를 진행하는 것이다.

주문 - 주문은 고객이 제공받고 싶은 커피의 종류와 수량을 기록한 데이터이다.

결제 - 결제는 주문을 바탕으로 결제금액을 산정하고, 해당 결제금액을 **계좌**에 매출로 등록하는 것이다.

계좌 - 계좌는 매출과 지출, 잔고를 기록한 데이터이다.

CoffeeManager의 객체 간 메시지



2. 각 개념에 대한 DOMAIN을 정리

- DOMAIN : 실제 세계에서 발생하는 사건들의 집합

커피 - 커피는 판매자가 고객에게 판매하는 사물이다.

- 커피를 제공할 때 커피의 누적판매량을 기록한다.
- 커피를 제공한 뒤 재고가 안전재고 기준에 미달한다면 안전재고의 두 배만큼 재고를 매입한다.
- 만약 잔고가 부족하다면 커피의 안전재고 매입을 취소한다.

- 시즌커피는 지정된 시즌달 에만 판매하는 커피이다.

판매 - 판매는고객이 메뉴를 보고 커피를 주문하면 해당 주문을 처리하고 결제를 진행하는 것이다.

주문 - 주문은 고객이 제공받고 싶은 커피의 종류와 수량을 기록한 데이터이다.

- 고객이 주문한 수량이 커피의 재고보다 많다면 주문을 취소한다.
- 만약 시즌음료의 시즌이 아닐 때 주문이 들어오면 주문을 취소처리한다.
- 주문 내용대로 고객에게 커피를 제공한다.

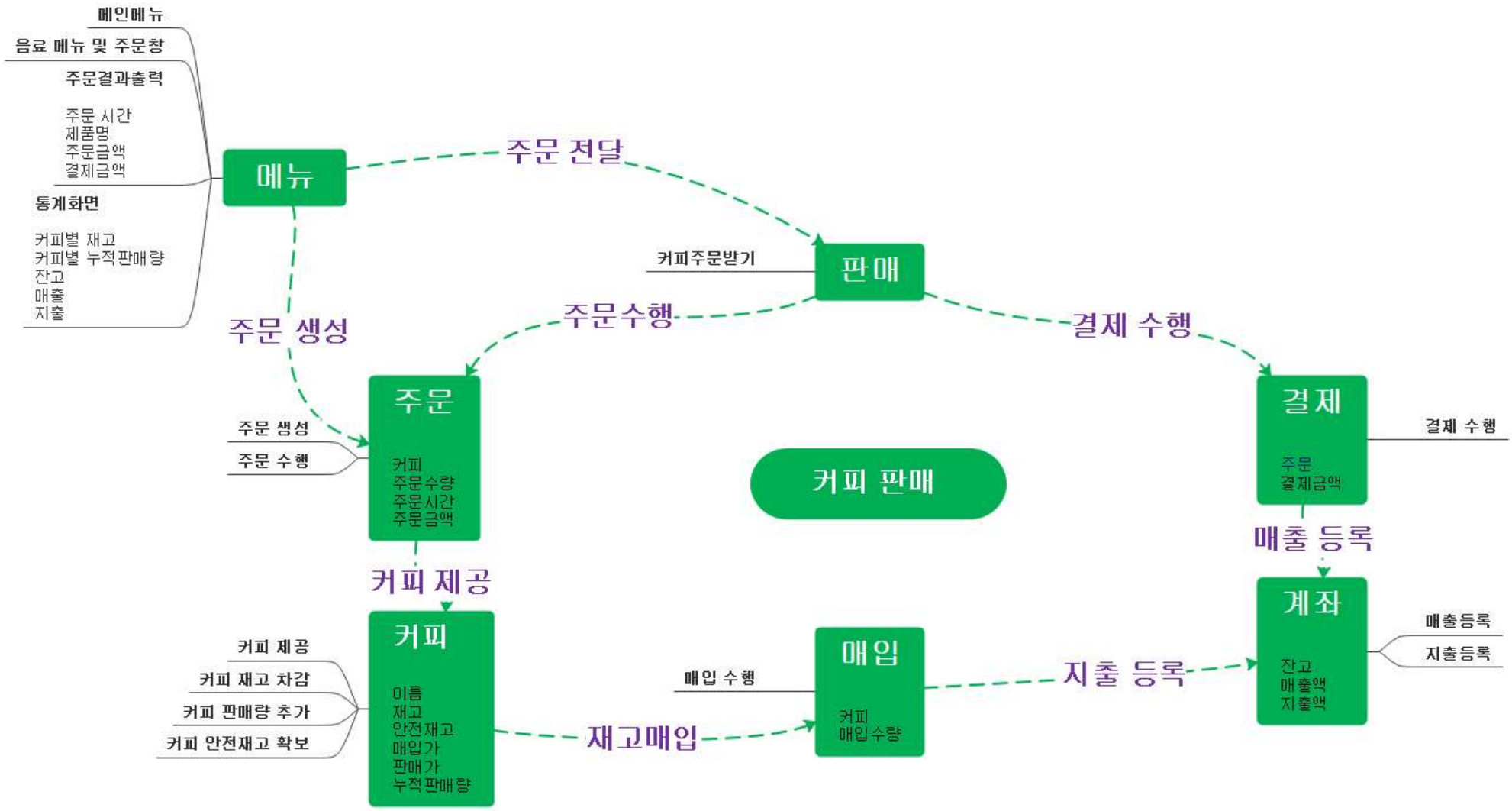
결제 - 결제는 주문을 바탕으로 결제금액을 산정하고, 해당 결제금액을 계좌에 매출로 등록하는 것이다.

계좌 - 계좌는 매출과 지출, 잔고의 결과를 기록한 데이터이다.

- 매출등록은 계좌에 전달된 금액을 잔고와 매출액에 추가하는 것이다.
- 지출등록은 계좌에 전달된 금액을 잔고에서 차감하고 지출액에 추가하는 것이다.

매입 - 매입은 매입에 필요한 금액을 지출로 등록하고 판매자가 필요한 커피 수량을 확보하는 것이다.

도메인 작성 이후 객체 간 메시지 + 책임



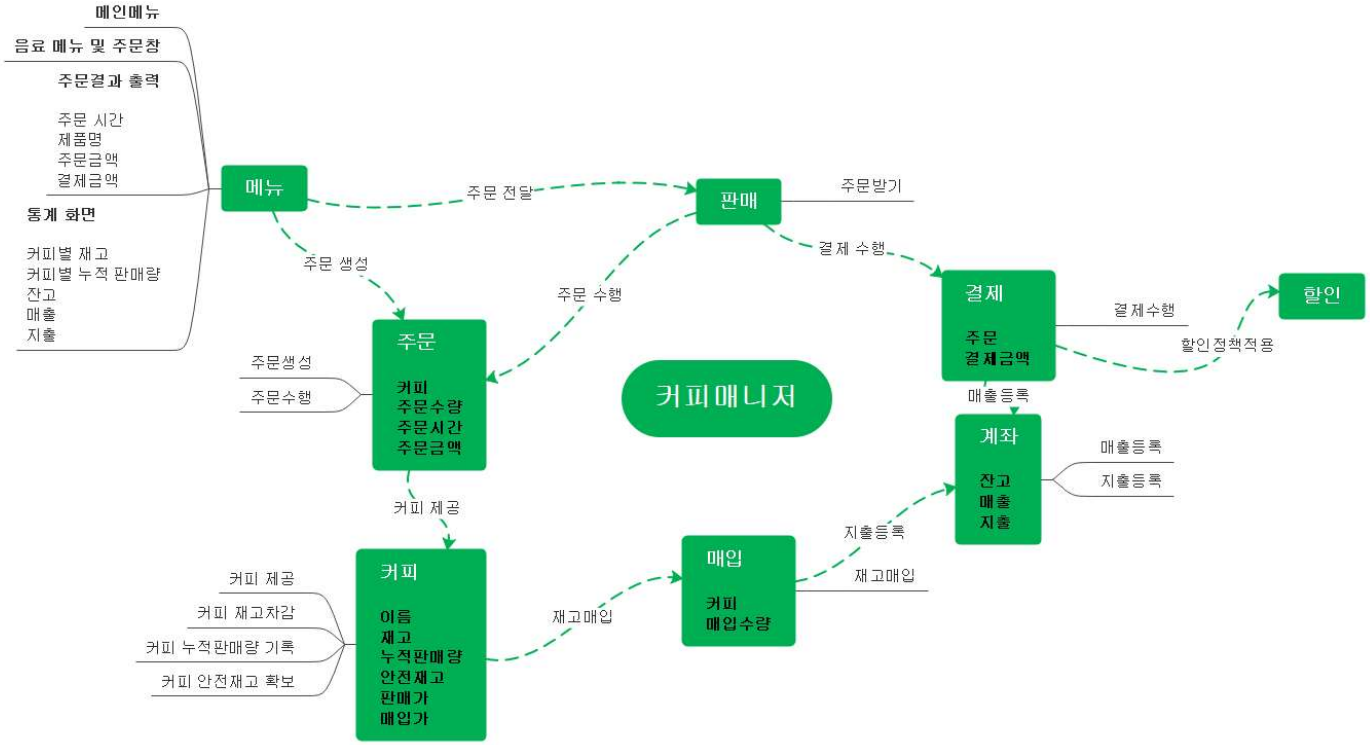
기능 확장 해보기

주문 금액이 10000원 이상인 주문은 10% 할인하는 정책을 추가하세요.

기능 확장 해보기

할인 개념 추가

할인 - 할인은 주문에 할인 정책에 해당하는 내용이 있다면 주문금액에서 결제금액을 산정할 때 일정 금액을 차감하는 것이다.



객체 지향 프로그래밍의 장점

1. 객체 간의 메시지만 파악해도 프로그램의 전체 구조를 파악할 수 있다.

2. 객체간 메시지를 주고 받는 메서드를 수정하지 않는 이상, 한 클래스의 변경이 다른 클래스에 영향을 미칠 일이 없다. (낮은 결합도)

3. 기능추가를 위한 개념이 명시적으로 드러난다.
   이전 CoffeeManager에도 주문, 판매, 할인 등의 개념이 있지만 해당 개념들은 모두 코드의 동작 안에 숨어있다.
   따라서 판매와 관련된 코드를 수정하기 위해 전체 코드를 읽어야 하는 불편함이 있지만
   객체지향적으로 설계한 CoffeeManager는 개념이 명시적으로 드러나기 때문에 Coffee클래스만 읽어도 된다. (높은 응집도)

4. 새로운 기능을 추가하기 위해 기존의 코드를 수정하지 않아도 된다.
   * OCP 원칙 : 수정에는 닫혀있고 확장에는 열려있다.

객체지향적설계를 따르면
=> 메시지만으로 객체간의 구조가 드러나고 높은 응집도를 갖추고 있다 : 빠르고 적확하게 프로그램을 파악할 수 있다.
낮은 결합도를 가지게 되어 OCP원칙을 준수하게 된다 : 기능을 수정할 때 에러가 적게 발생한다.

유지보수와 기능확장이 용이한 유연한 프로그램 설계를 할 수 있게 된다.

기능 확장 해보기

1. 특정 계절에만 판매할 수 있는 계절음료를 추가합니다.

2. 계절음료는 해당 음료의 판매기간이 아니라면
   “해당 음료는 계절상품입니다.”
   를 출력하고 주문을 취소합니다.

좋은 코드란

모든 소프트웨어 모듈에는 세 가지 목적이 있다.

첫 번째 목적은 실행 중에 제대로 동작하는 것이다.
이것은 모듈의 존재 이유라고 할 수 있다.

두 번째 목적은 변경을 위해 존재하는 것이다.
대부분의 모듈은 생명주기 동안 변경되기 때문에 간단한 작업만으로도 변경이 가능해야 한다.
변경하기 어려운 모듈은 제대로 동작하더라도 개선해야 한다.

세 번째 목적은 코드를 읽는 사람과 의사소통하는 것이다.
모듈은 특별한 훈련 없이도 개발자가 쉽게 읽고 이해할 수 있어야 한다.
읽는 사람과 의사소통 할 수 없는 모듈은 개선해야 한다.

– 로버트 C. 마틴 <클린 소프트웨어>

객체지향 SOLID 원칙

S (SRP : Single Responsibility Principle)
한 클래스는 하나의 책임만 가져야 한다.

O (OCP : Open/Closed Principle)
확장에는 열려(Open) 있으나, 변경에는 닫혀(Closed)있어야 한다.

L (LSP : Liskov's Substitution Principle)
프로그램의 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 한다.
-> 호출부에서 기대하는 대로 동작하도록 메서드를 오버라이딩 해야한다.

I (ISP : Interface Segregation Principle)
특정 클라이언트를 위한 인터페이스 여러 개가 범용 인터페이스 하나보다 낫다.

D (DIP : Dependency Inversion Principle)
추상화에 의존한다. 구체화에 의존하면 안된다.