34. 객체지향프로그래밍 ₩ -기초,

2018.12

일병 김재형

객체지향

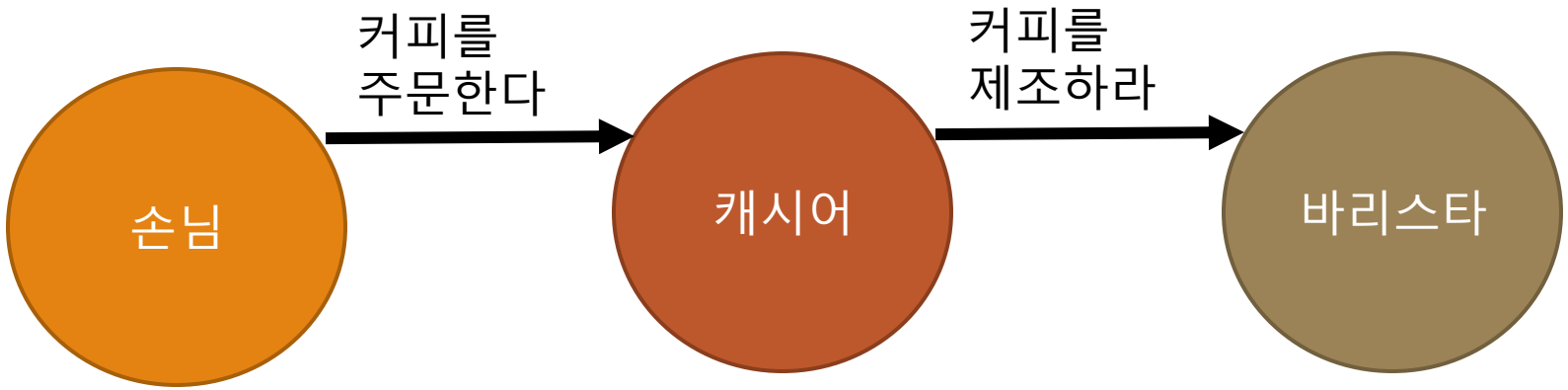
객체지향?

- "실세계를 직접적이고 직관적으로 모델링할 수 있는 패러다임"
- 실제 개발을 할 때 대응되는 실세계의 사물을 발견하기는 어렵다.
- 실세계에 대한 비유-객체지향의 다양한 측면을 이해하고 학습하는데 도움이 된다.

협력, 역할, 책임

협력: 요청과 응답으로 구성

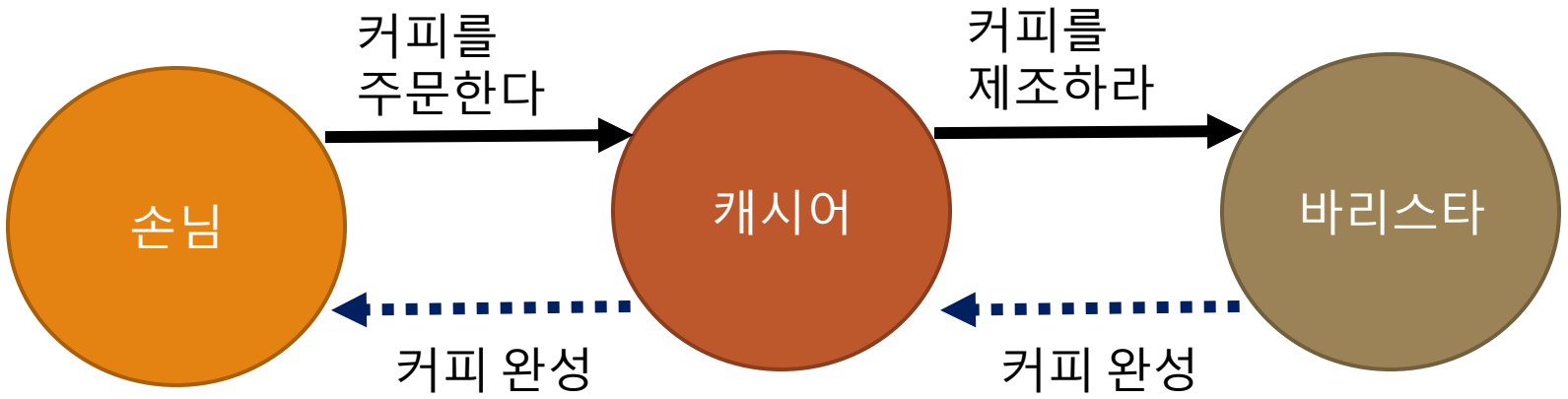
- 현실 사회
- 대부분의 문제는 개인 혼자가 해결하기 어렵다.
- 스스로 해결할 수 없으면 도움을 요청(request)



협력, 역할, 책임

협력: 요청과 응답으로 구성

- 요청을 받으면 응답(response)
- 요청이 연쇄적으로 발생하기 때문에 응답도 연쇄적으로 전달된다.



협력, 역할, 책임

협력: 요청과 응답으로 구성

- 협력(collaboration)을 통해 거대하고 복잡한 문제를 해결할 수 있는 공동체를 형성

협력, 역할, 책임

역할과 책임

- 협력하는 과정에 역할(role)이 부여
- 역할에는 협력에 참여하는 사람이 협력에서 차지하는 책임이나 의무를 의미
  - 손님: 커피 주문 임무
  - 캐시어: 손님에게서 주문 받음
  - 바리스타: 커피 제조 책임

협력, 역할, 책임

역할과 책임

- 특정한 역할은 특정한 책임을 암시

여러 사람이 동일한 역할 수행
- 누가 책임을 수행하든 상관 없다.

역할: 대체가능성 의미
- 동일한 역할을 수행
- 누구든 문제가 없다.

책임 수행 방법은 자율적 선택
- 동일한 요청에 대해 다른 방식으로 응답가능(다형성)

한 사람이 여러 역할 수행
- 둘 이상의 역할을 수행할 수도 있다.

협력, 역할, 책임

역할과 책임

- 객체지향에서도 마찬가지
  - 사람: 객체
  - 요청: 메시지
  - 응답: 메서드

협력과 객체

협력에 참여하는 주체는 객체

- 객체는 '협력적'이여야 한다.
  - 요청에 응답
  - 필요할 경우에는 요청
  - 모든 것을 하려고 하면 복잡해서 자멸
- 객체가 '자율적'이여야 한다.
  - 자율적: '자기 스스로의 원칙에 따라 어떤 일을 하거나 자기 스스로를 통제하여 절제하는 것'
  - 요청에 응답하지만, 스스로의 판단에 따라 결정과 행동
  - ex) 손님이 주문하는 절차를 캐시어가 강제하지 않는다.

협력과 객체

객체의 상태(state)와 행동(behavior)

- 객체가 어떤 행동(behavior)을 한다.
- 행동을 하는데 필요한 상태(state)가 존재한다.
- 스스로 판단하기 위해서는 행동과 상태가 필요

- 자율성
  - 객체의 사적인 부분은 스스로 관리
  - 외부에서는 허락된 수단으로만 객체와 의사소통

협력과 메시지

협력과 메시지

- 메시지(message)
  - 메시지 전송: 객체가 다른 객체에게 요청하는 것
  - 메시지 수신: 다른 객체에서 요청을 받는 것
- 송신자(sender): 메시지를 전송하는 객체
- 수신자(receiver): 메시지를 수신하는 객체

메서드와 자율성

메서드와 자율성

- 메서드(method)
  - 객체가 수신된 메시지를 처리하는 방법
- 외부의 요청이 무엇인지를 표현하는 메시지
  요청을 처리하는 구체적인 방법인 메서드를 분리
- 자율성을 증진
- ex) 커피제조 요청: 메시지, 커피 제조 방법: 메서드

커피 주문

더 깊은 객체 이야기

이상한 나라의 앨리스

- 앨리스는 토끼를 따라 토끼 굴에 들어갔다가 40cm정도 되는 작은 문을 발견한다.
- 아름다운 정원을 본 앨리스는 들어가기로 마음먹고 주변을 둘러보다 "마셔라"라고 적힌 병의 물병을 보고 마셨더니 키가 줄었다.
- 케이크를 먹으니 키가 커졌다가 부채를 부치니 키가 작아졌다. 이를 여러 번 반복하여 문을 통과하게 된다.

더 깊은 객체 이야기

앨리스 객체

- 앨리스의 키는 시간의 흐름에 따라 변한다.
- 앨리스의 행동에 따라 키가 변한다.
- 상태를 결정하는 것은 행동

마신 후

더 깊은 객체 이야기

앨리스 객체

- 행동의 결과를 결정하는 것은 상태이다.
- ex) 얼마만큼 줄어드는가?

140cm

마신 후

100cm

더 깊은 객체 이야기

앨리스 객체

- 어떤 행동의 성공여부는 이전에 어떤 행동들이 발생했는지에 영향을 받는다. (행동간의 순서)
- 행동의 결과가 상태에 영향받기 때문

키가 적당

더 깊은 객체 이야기

객체의 정의

- 객체란 식별 가능한 개체 또는 사물이다.
- 객체는 자동차처럼 만질 수 있는 구체적인 사물일 수도 있고, 시간처럼 추상적인 개념일 수도 있다.
- 객체는 구별가능한 식별자, 특징적인 행동, 변경 가능한 상태를 가진다.
- 소프트웨어 안에서 객체는 저장된 상태와 실행 가능한 코드를 통해 구현된다.

객체의 상태

상태

- 객체가 주변과의 상호작용에 반응하는 방법
  그 시점까지 객체에 어떤 일이 발생했느냐에 좌우
- Ex) 휴가 승인이 안나면 휴가를 가지 못한다.

객체의 상태

상태

- 행동의 결과
  - 과거에 어떤 일이 있었는가?
- 행동의 결과를 판단하기 어렵다.
- 행동의 과정과 결과는 상태를 통해서 알 수 있다.

객체의 상태

상태와 프로퍼티

- 객체의 상태 표현 방법
  - 숫자, 문자열, 시간 등
  - 객체를 사용해, 다른 객체의 상태 표현

    ex) 앨리스가 음료를 들고 있는가?
- 객체의 상태는 단순한 값과 객체의 조합으로 표현

객체의 상태

객체의 상태

- 프로퍼티(property)
  - 객체의 상태를 구성하는 모든 특징
  - 키, 위치 등
  - 정적
- 프로퍼티 값(property value)
  - 시간의 흐름에 따라 변경
  - 키가 몇 cm인가? 현재 위치는 어디인가?
  - 동적

객체의 상태

프로퍼티의 구분

- 속성(attribute): 객체를 구성하는 단순한 값
- 링크(link): 객체와 객체 사이의 연결
  ex) 앨리스가 음료를 들고 있다.
  - 링크는 객체가 다른 객체를 참조할 수 있다는 것을 의미

객체의 행동

상태와 행동

- 객체의 행동에 의해 객체의 상태가 변경된다.
  - Ex) 앨리스가 케이크를 먹는 행위
    앨리스의 키가 줄어든다.
    케이크의 양이 줄어든다.
- 행동을 상태 변화의 관점으로 볼 수 있다.

객체의 행동

상태와 행동사이의 관계

- 행동의 입장에서
  - 객체의 행동은 상태에 영향을 받는다.
  - 객체의 행동은 상태를 변경시킨다.
- 상태의 입장에서
  - 상호작용이 현재의 상태에 어떤 방식으로 의존하는가
  - 상호작용이 어떻게 현재의 상태를 변경시키는가

객체의 행동

상태와 행동사이의 관계

- 앨리스가 통과해야 하는 문의 크기는 40cm
- 앨리스 객체
  - 문을 통과하는 행동수행
- 행동 수행을 위해 키와 위치라는 두 가지 상태 이용
  - 앨리스의 키가 40cm이하라면 문을 통과할 수 있다.
  - 문을 통과하면 앨리스의 위치는 정원으로 바뀐다.

객체의 행동

협력과 행동

- 객체는 다른 객체를 이용하고, 서비스를 제공
- 객체의 행동 트리거
  - 객체가 외부에서 수신한 메시지
- 객체는 자기 자신의 상태 & 다른 객체의 상태 변경
  - 객체 자신의 상태를 변경
  - 행동에서 협력하는 다른 객체에게 메세지 전송

객체의 행동

상태 캡슐화

- 현실세계의 객체
  - 수동적으로 상태가 줄어드는 객체가 있다.
    ex)음료
- 프로그램의 객체
  - 자신의 상태를 스스로 관리하는 자율적인 존재
  - 음료라고 해도, 음료의 상태를 변경시키는 주체는 음료이다.
  - 협력관계에서, 메세지를 보낸 객체는 실제로 원하는 대로 요청이 이뤄지는지 알 수 없다.

객체의 행동

상태 캡슐화

- 객체는 상태를 감추고, 행동만을 노출한다.
- 외부에서 객체에 접근하는 방법은 행동만이 가능

객체의 행동

상태 캡슐화

- 객체의 자율성을 높인다.
- 협력을 단순하고 유연하게 만든다.
- 개발 시 변경을 유연하게 한다.
- Ex) 개발 중 개발 내용 변경

객체의 식별자

식별자

- 객체를 구별할 수 있는 특별한 프로퍼티
- 단순한 값은 식별자가 존재하지 않는다.

객체의 식별자

값

- 숫자, 문자열, 시간과 같이 변하지 않는 양
- 불변상태(immutable state)
- 두 인스턴스의 상태가 같다면 같은 것으로 판단
- Ex)숫자 1

동등성(equality)

- 상태를 이용해 두 값이 같은지 판단
- 값의 상태가 변하지 않아, 언제나 동등한 상태

객체의 식별자

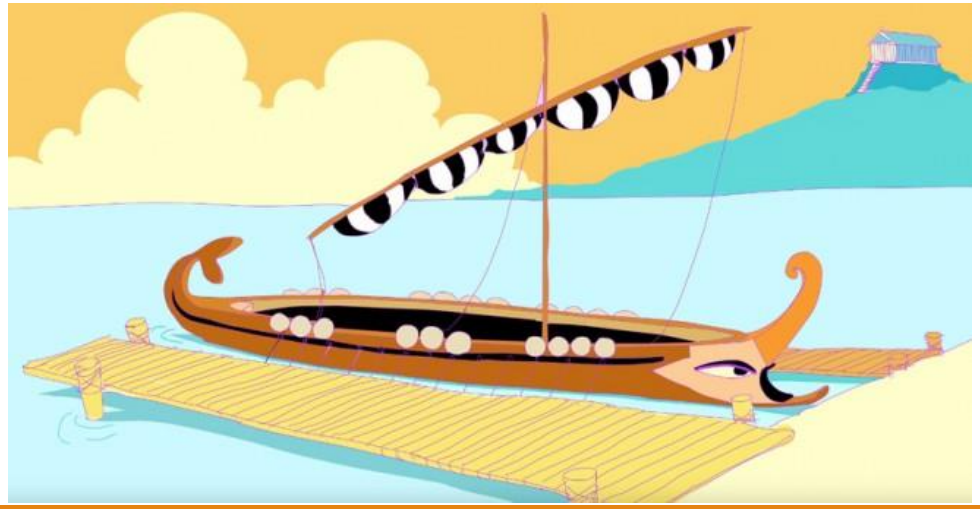
객체

- 시간에 따라 변경되는 상태를 포함
- 행동을 통해 상태를 변경
- 가변상태(mutable state)
- 상태가 완전히 같아도 두 객체는 별개의 객체
- ex) 게임 캐릭터

객체의 식별자

객체

- 테세우스의 배
- cf) 사람은?
- cf) 순간이동은?



객체의 식별자

동일성(identical)

- 두 객체의 상태가 다르더라도 식별자가 같다면
  두 객체를 같은 객체로 판단
- is None
  None은 항상 하나이고, 값이 없다는 뜻임으로
  값을 비교하는 것 보다 동일성을 판단하는게 좋다.

객체의 식별자

- 프로그래밍 언어의 값과 객체
  - 값과 객체를 전부 class를 이용해 구현
    - 정수: Integer클래스
    - 사람: Person클래스
  - 따라서 별도의 용어를 사용하기도 함
    - 식별자가 있는 전통적인 객체
      - 참조객체(reference object)
      - 엔티티(entity)
    - 식별자가 없는 값
      - 값 객체(value object)

행동이 상태를 결정

상태를 중심으로 객체를 보기

- 객체에 필요한 상태를 찾고,
  상태를 변경하거나 조회할 수 있는 행동을 고민
- 설계에 나쁜 영향을 준다.
  - 캡슐화가 저해
  - 객체의 협력성이 저하
  - 객체의 재사용성이 저하

행동이 상태를 결정

책임-주도 설계(Responsibility-Driven Design)

- 협력에서 객체의 행동은 객체가 완수하는 책임
  - 객체는 협력해야 한다.
  - 객체가 협력에 참여하는 방법은 행동
  - 객체의 행동을 결정하고 행동에 적절한 상태를 선택

은유와 객체

객체지향이란 현실세계의 모방?

- 추상화를 하여 현실세계를 간단하게 모방?
- 소프트웨어의 상품
  - 가격 계산, 물품량 변경 등을 함

은유와 객체

의인화(anthropomorphism)

- 현실과 소프트웨어 사이의 가장 큰 차이
  - "수동적인 존재가 구현될 때 능동적으로 변한다."
  - 스스로 물품의 양을 줄이는 음료수..?
- 레베카 워프브룩
  "의인화의 관점에서 소프트웨어를 생물로 생각하자. 모든 생물처럼 태어나고, 삶을 영위하고, 죽는다."

은유와 객체

은유(metaphor)

- 객체지향과 현실과의 유사성
- 은유: 하나의 의미를 다른 것을 이용해 전달
  ex) "그 사람은 사자 같아요."

은유와 객체

은유(metaphor)

- 현실 객체의 의미 일부가 소프트웨어의 객체로 전달
  - 실제 전화기는 전화를 걸 수 없다.
  - 소프트웨어 객체를 '전화기' 개념을 통해 묘사하면 이해와 기억이 쉬워진다.
- 객체지향 지침서
  - 현실 세계인 도메인에서 사용되는 이름을 객체에게 부여하라고 가이드

더 알아보기

- "객체지향의 사실과 오해, 조영호, 위키북스, 2015"를 참고하십시오.
- (심화) 객체지향 프로그래밍에서 자세히 진행

자판기 설계

도메인 모델

- 도메인
  - 사람에 따라 소프트웨어를 다르게 사용(게임, 보안 등)
  - 사용자가 프로그램을 사용하는 대상 분야
- 모델
  - 대상을 단순화해서 표현
- 도메인 모델
  - 소프트웨어 내부를 선택적으로 단순화하여 구조화한 형태
  - 즉, 소프트웨어 개발과 관련된 사람들이 내부에 대해 생각하는 관점

자판기 설계

도메인 모델

- 초기 자판기
  - 고객과 상호작용하는 자판기
  - 돈과 관련된 작업을 하는 돈 통
  - 음료와 관련된 작업을 하는 음료
  - 음료 목록을 가지고 있는 메뉴판

자판기 설계

도메인 모델

자판기 설계

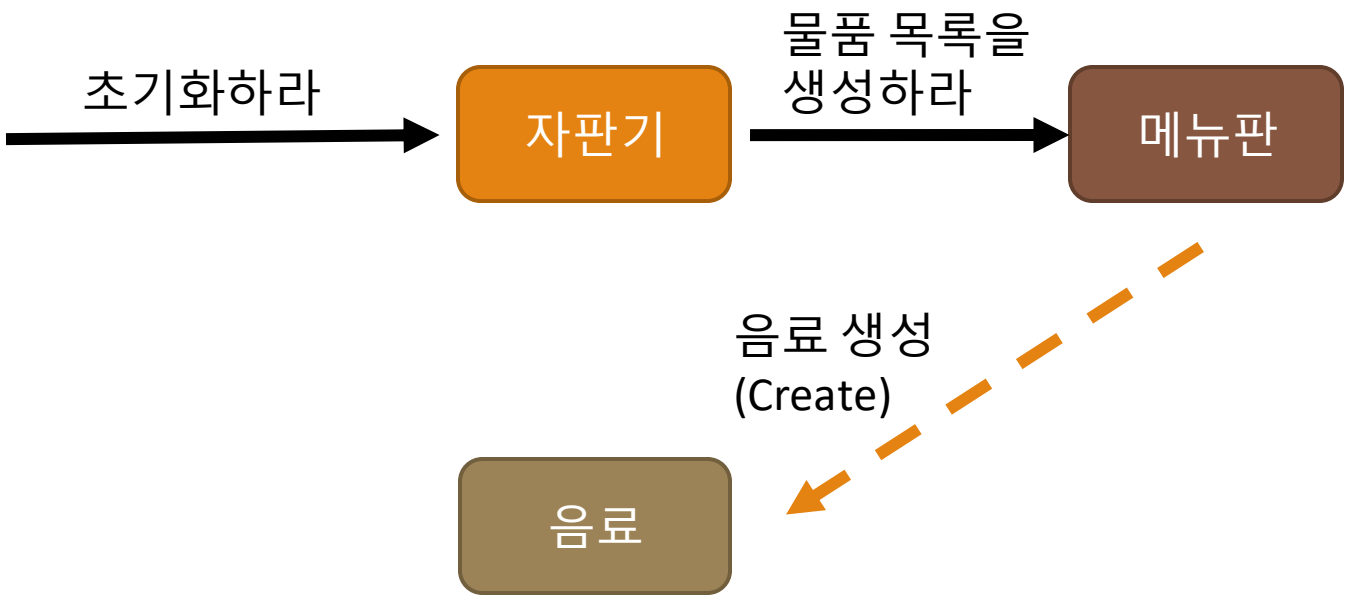
협력 설계

- 초기 자판기
  - 시작 시 물품을 초기화
  - 물품을 보여줌
  - 돈을 넣음
  - 물품을 선택하면 돈을 확인하여 물품을 출력
  - 입력된 돈을 확인
  - 돈을 반환

자판기 설계

협력 설계

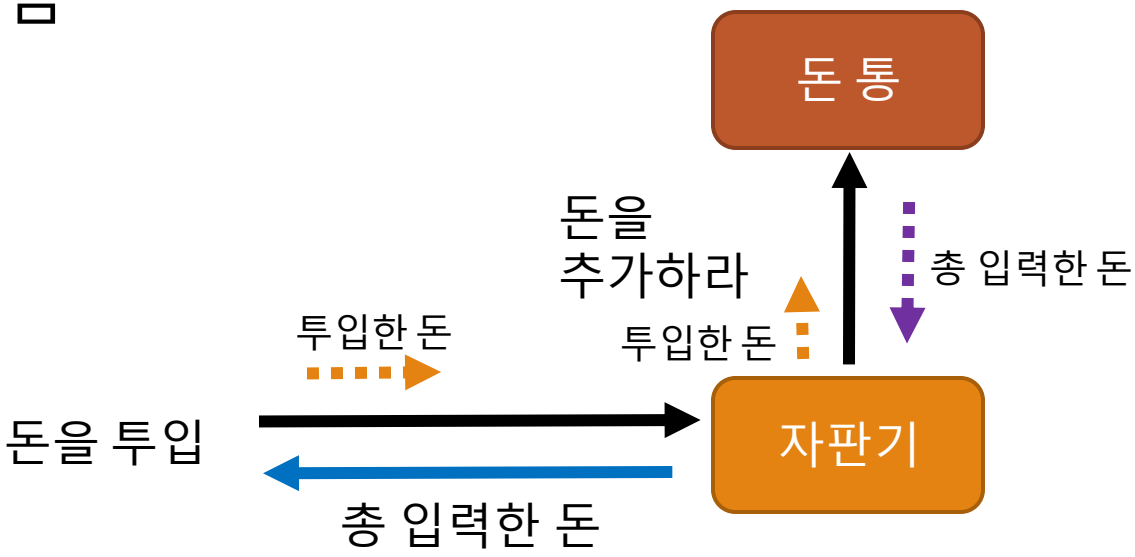
- 시작 시 물품을 초기화



자판기 설계

협력 설계

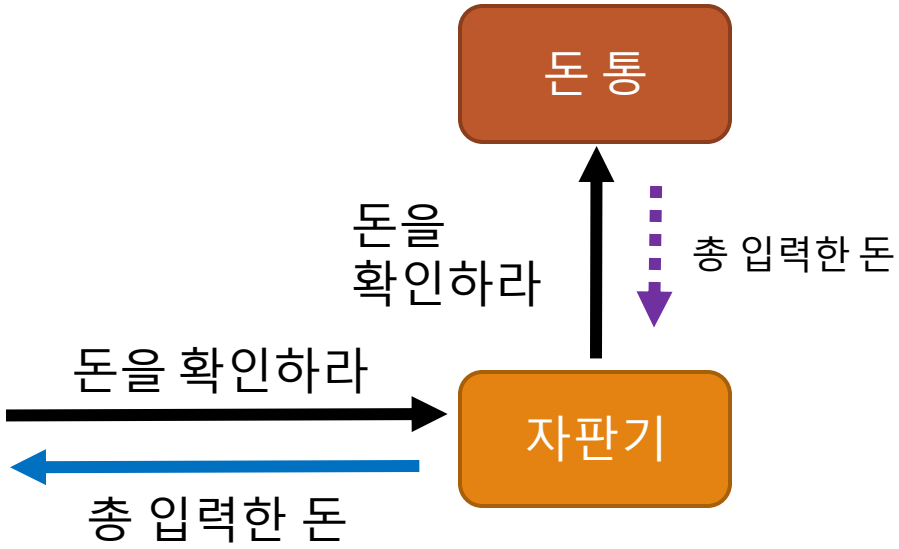
- 돈을 넣음



자판기 설계

협력 설계

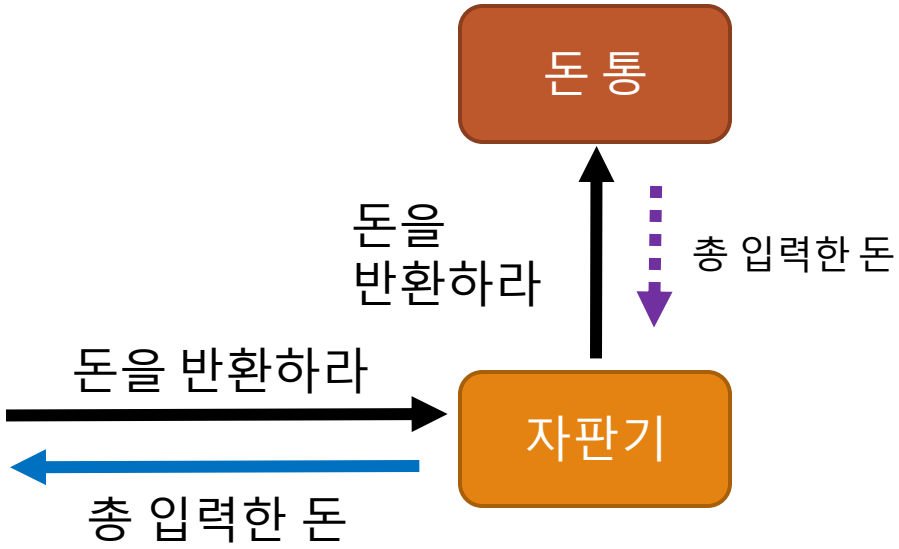
- 돈 확인



자판기 설계

협력 설계

- 돈 반환



자판기 설계

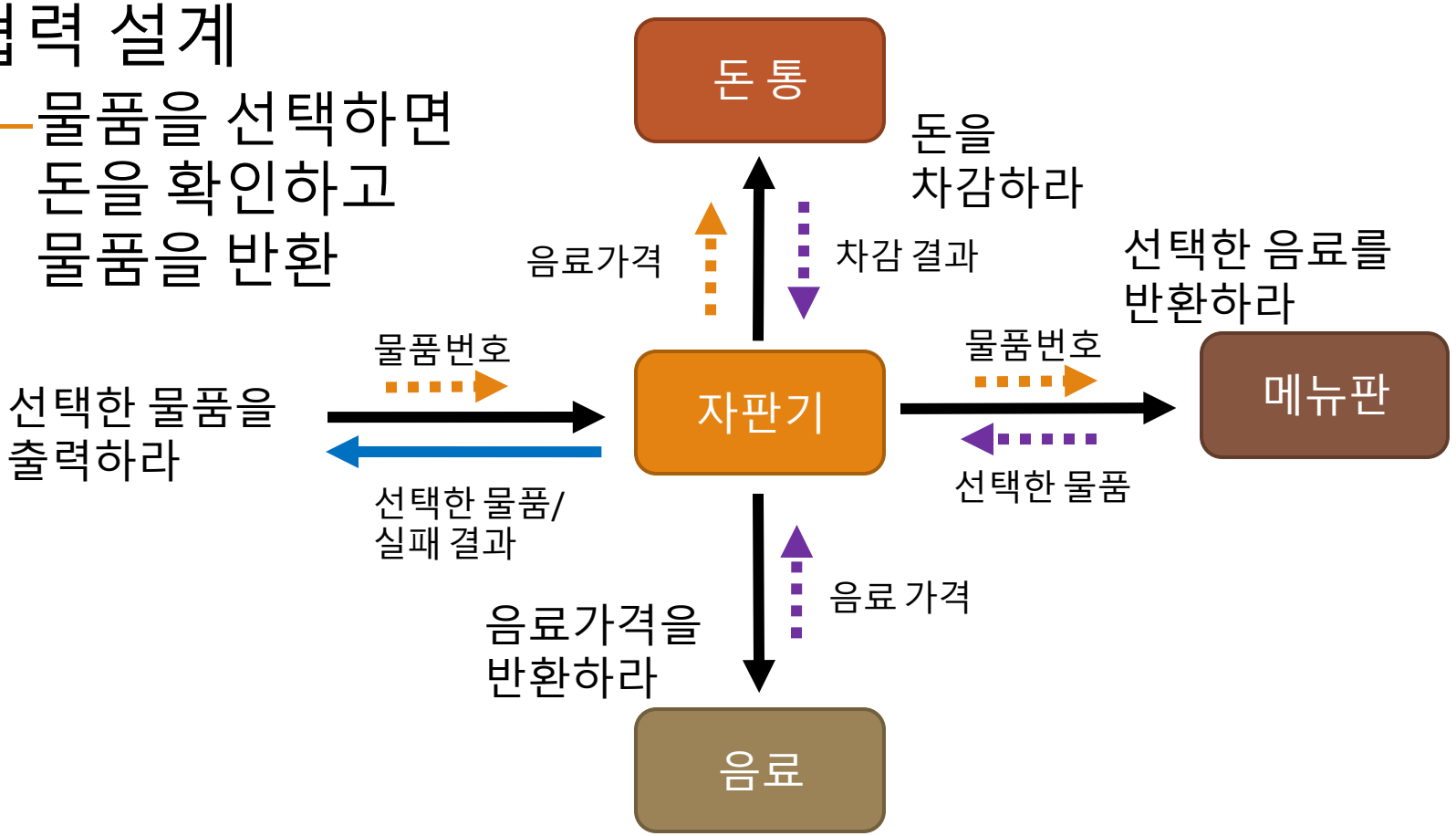
협력 설계

- 물품을 보여줌

자판기 설계

협력 설계

- 물품을 선택하면 돈을 확인하고 물품을 반환



자판기 설계

인터페이스 정리

- 인터페이스(interface)
  - 두 사물이 마주치는 경계 지점에서 상호작용할 수 있게 이어주는 방법이나 장치
  - 사용자가 기기를 쉽게 동작시키는 것에 도움을 줌

자판기 설계

인터페이스 정리

- 인터페이스(interface)-리모콘
  - 내부 구조나 동작 방식을 몰라도 조작 가능
  - 내부 구성이나 동작 방식이 바뀌어도 사용자에게 영향을 미치지 않음
  - 대상이 변경되어도 동일한 인터페이스를 제공하면 상호작용할 수 있음

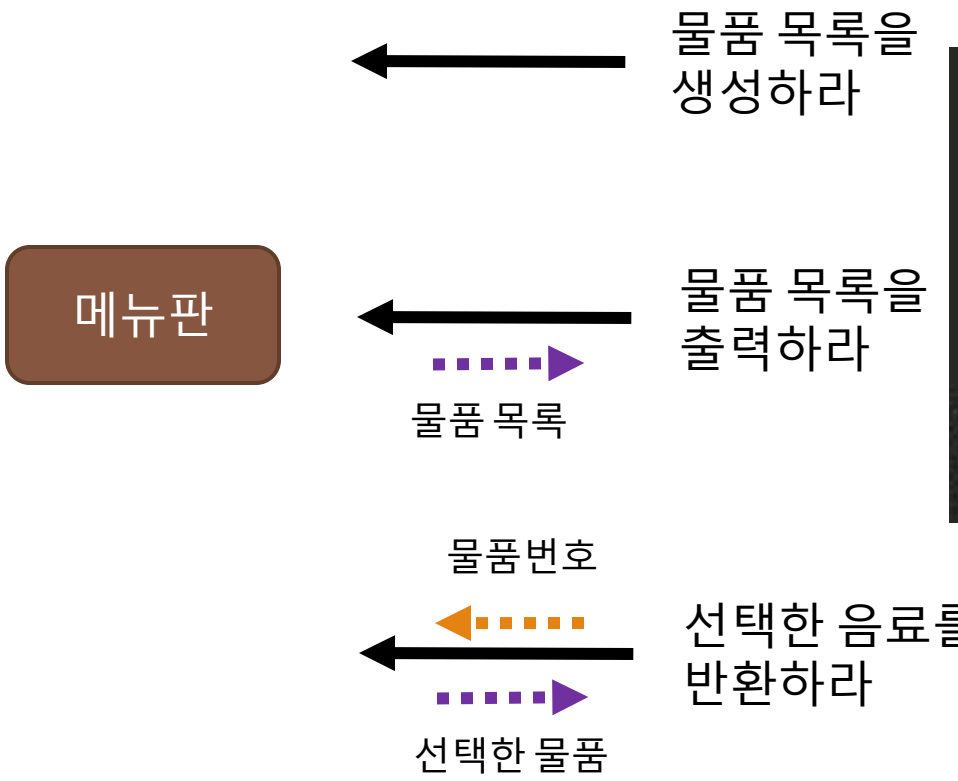
자판기 설계

인터페이스 정리

- 인터페이스(interface)-객체
  - 어떤 객체가 메시지를 수신하여 작동하는 것을 의미
  - 객체 메시지를 수신할 수 있다는 것
    - 메시지에 해당하는 작동(operation)이 있다
- 수신 가능한 메시지만 추려내어 인터페이스를 정리

자판기 설계

인터페이스 정리




```
1   class Menu:
2       def __init__(self):
3           pass
4
5       def create_list(self):
6           pass
7
8       def print_list(self):
9           pass
10
11      def return_item(self, item_num):
12          pass
```

자판기 설계

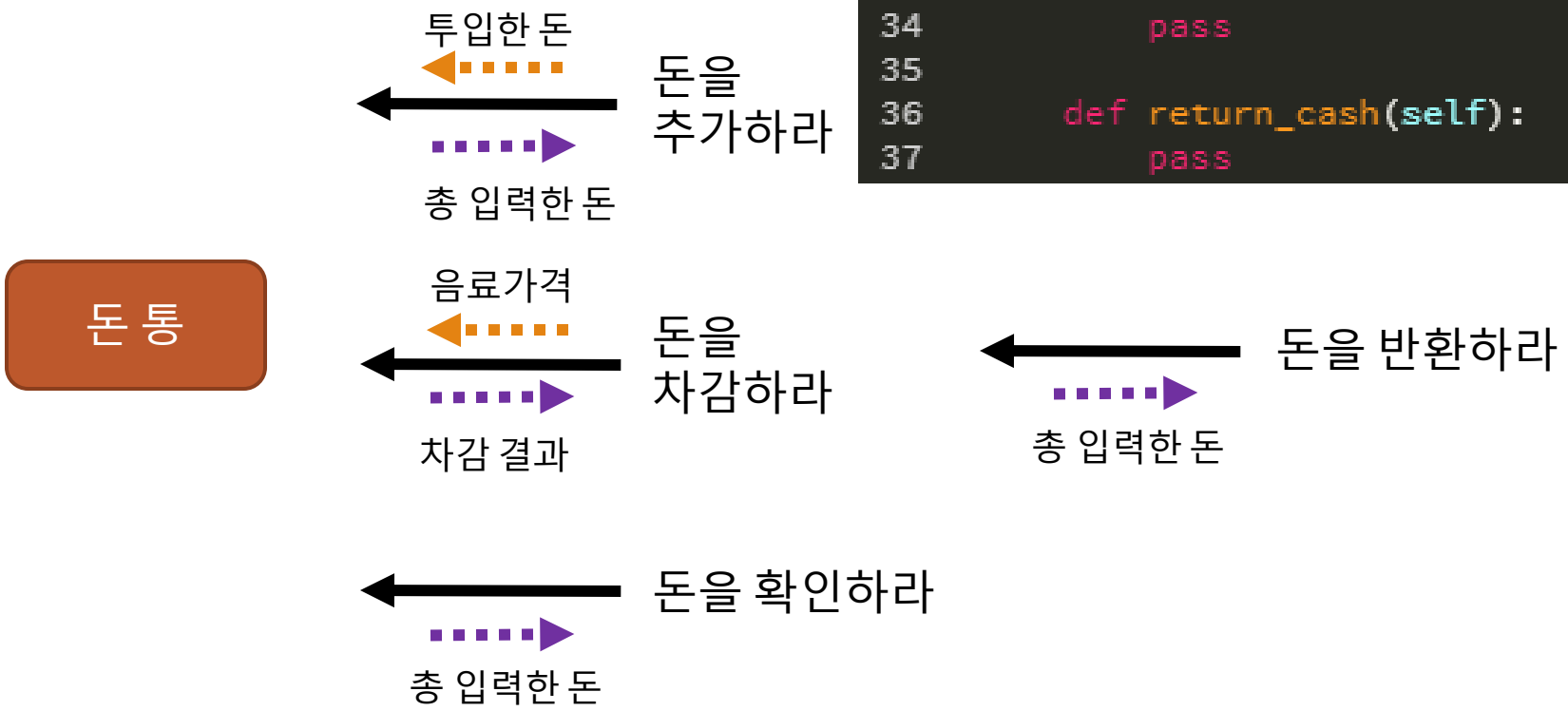
인터페이스 정리


```
15  class Beverage:
16      def __init__(self):
17          pass
18
19      def return_price(self):
20          pass
```

자판기 설계

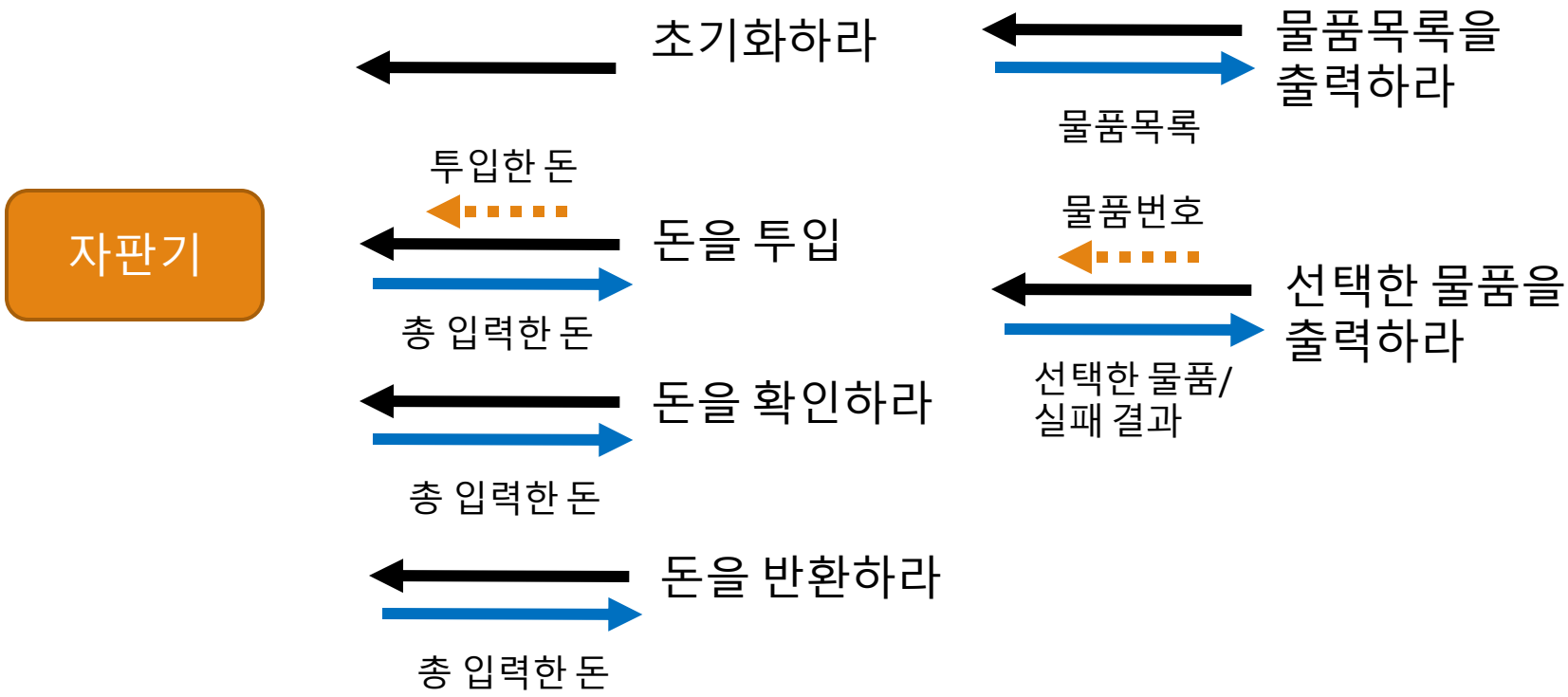
인터페이스 정리

```
23  class Cash:
24      def __init__(self):
25          pass
26
27      def add_cash(self, money):
28          pass
29
30      def sub_cash(self, price):
31          pass
32
33      def print_cash(self):
34          pass
35
36      def return_cash(self):
37          pass
```



자판기 설계

인터페이스 정리



자판기 설계

인터페이스 정리

```
40  class VendingMachine:
41      def __init__(self):
42          pass
43
44      def add_cash(self, money):
45          pass
46
47      def print_cash(self):
48          pass
49
50      def return_cash(self):
51          pass
52
53      def print_itemlist(self):
54          pass
55
56      def select_item(self, item_num):
57          pass
```

자판기 설계

구현

- 내부 구조와 작동 방식을 가리키는 고유의 용어
- 상태
  - 객체는 상태를 가진다.
  - 객체 외부(인터페이스)에 노출되지 않는다.
- 행동(메서드)
  - 메시지 처리방법
  - 마찬가지로, 외부에 노출되지 않는다.

자판기 설계

구현

- 메뉴와 음료, 자판기와 돈 통
  - 포함관계-has a 관계
- 자판기가 메뉴판에 접근
  - 접근이 가능해야 메시지를 전달
  - print_itemlist와 select_item에 메뉴판을 매개변수로 전달

  - ※ 구현 도중에 인터페이스가 변경될 수 있다.

자판기 설계

구현

- VendingMachine

```
40  class VendingMachine:
41      def __init__(self):
42          self.cash_bucket = Cash(0)
43
44      def control_cash(self, string, money=0):
45          if string == "add":
46              self.cash_bucket.add_cash(money)
47          elif string == "print":
48              self.cash_bucket.print_cash()
49          elif string == "return":
50              self.cash_bucket.return_cash()
51          else:
52              print("잘못된 입력입니다.")
53
54      def print_itemlist(self, Menu):
55          Menu.print_list()
56
57      def return_item(self, Menu, item_num):
58          item = Menu.retun_item(item_num)
59          item_value = item.return_price()
60          if self.cash_bucket.sub_cash(item_value):
61              print("%s가 나왔습니다." % item.name)
62          else:
63              print("금액이 부족합니다.")
64
```

자판기 설계

구현

- 머리속으로 구상한 코드
  => 코드로 구현 시 변경 가능
- 협력설계에 치중하지 말 것
- 실제 구현 가능한지,
  구현하여 피드백 받아야 함

자판기 설계

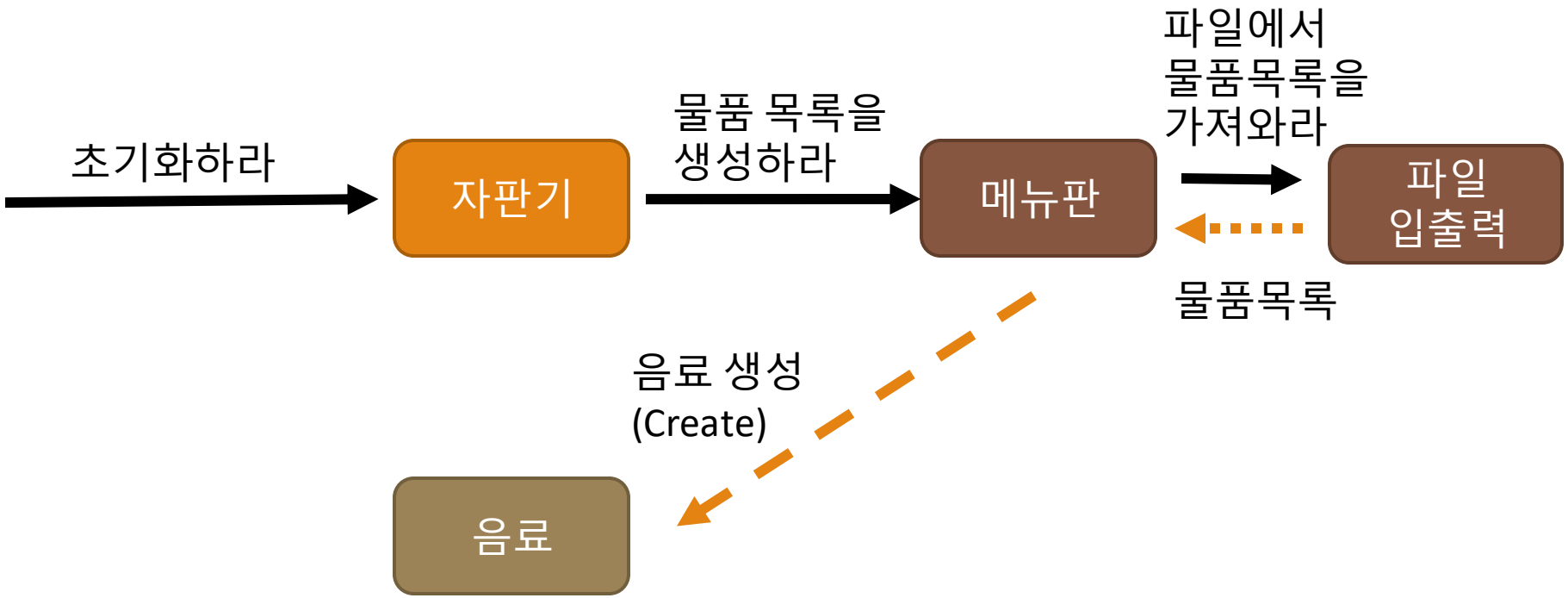
확장

- 파일에서 데이터 입출력을 하고 싶다면?
  - 데이터 입출력 class를 새로 제작
  - 메뉴판의 파일 생성 부분의 함수 변경

자판기 설계

협력 설계

- 시작 시 물품을 초기화



기본과제-자판기7

자판기(vending_machine_class.py)

- 앞에서 구현한 것을 보고 남은 부분을 구현해보자.

기본과제-369

three_six_nine.py

- 고객이 369를 할 때, 짝을 몇 번하는지 알고 싶다고 프로그램을 주문하였다.
- 1. 입력되는 값은 정수이고, 값의 제한은 없다.
- 2. 1부터 입력받은 수까지 한 칸씩 띄우며 출력하며, 20번 째 수마다 줄바꿈을 한다.
- 3. 숫자에 3, 6, 9가 들어가면 그 횟수만큼 짝을 출력한다.

기본과제-369

three_six_nine.py

- 예시

```
마지막 숫자를 입력하세요: 80
1 2 짝 4 5 짝 7 8 짝 10 11 12 짝 14 15 짝 17 18 짝 20
21 22 짝 24 25 짝 27 28 짝 짝 짝 짝 짝짝 짝 짝 짝짝 짝 짝 짝짝 40
41 42 짝 44 45 짝 47 48 짝 50 51 52 짝 54 55 짝 57 58 짝 짝
짝 짝 짝짝 짝 짝 짝짝 짝 짝 짝짝 70 71 72 짝 74 75 짝 77 78 짝 80
```

심화과제-369

three_six_nine.py

- 확장 문제
- 1. 2의 배수는 '뽁'을 출력한다.
  단, 출력 순서가 맞아야 되며, 공배수일 경우 3을 우선적으로 출력한다.
  예시)
  346 '짝''뽁''짝뽁'

```
마지막 숫자를 입력하세요: 40
1 뽁 짝 뽁 5 뽁짝 7 뽁 짝 10 11 뽁 짝 뽁 15 뽁짝 17 뽁 짝 뽁
뽁 뽁뽁 뽁짝 뽁뽁 뽁 뽁뽁짝 뽁 뽁뽁 뽁짝 짝 짝 짝뽁 짝짝 짝뽁 짝 짝뽁짝 짝 짝뽁 짝짝 뽁
```