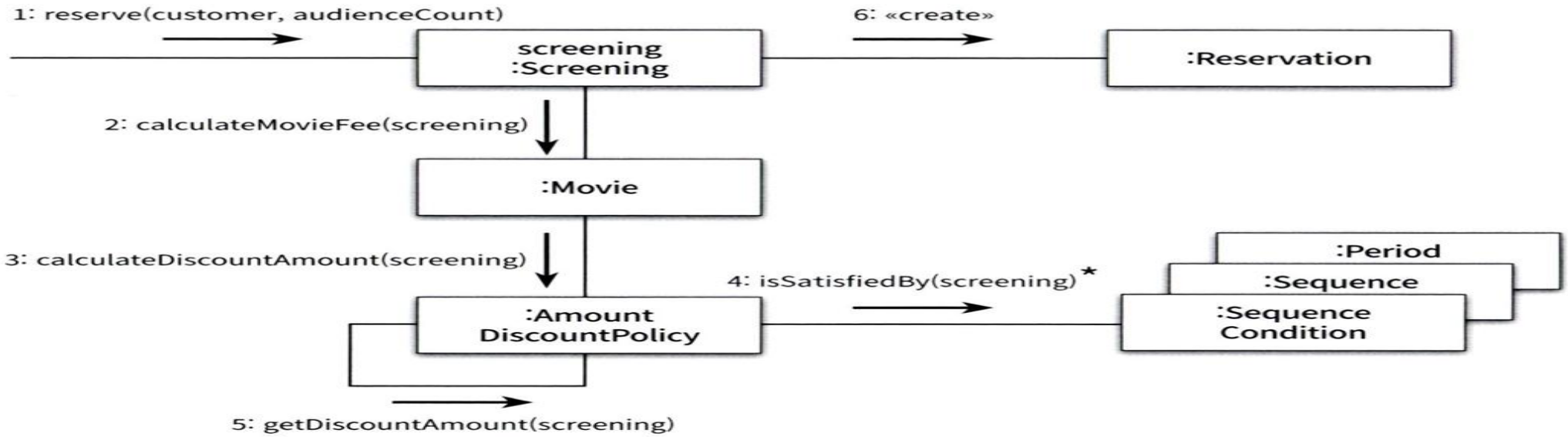
오브젝트 3장

김영록, 황차빈

협력



다양한 객체들이 영화 예매라는 기능을 구현하기 위해 메시지를 주고받으면서 상호작용한다
= 이러한 상호작용을 협력이라고 한다.

협력

객체는 고립된 존재가 아니라 시스템의 기능이라는 더 큰 목표를 달성하기 위해 다른 객체와 협력 해야함

두 객체 사이의 협력

- 메시지 전송을 통해서 타 객체에 자신의 요청사항 전달
- 메시지를 수신한 객체는 메서드를 실행해 요청에 응답

***메시지 수신한 객체가 메시지를 처리할 방법을 스스로 선택한다는 점이 중요**

객체의 자율성

- 자율적인 객체란 자신의 상태를 직접 관리하고 스스로의 결정에 따라 행동하는 객체
- 객체의 자율성을 보장하기 위해서는 필요한 정보와 정보에 기 반한 행동을 같은 객체 안에 모아놓아야 한다
- 자율적인 객체는 자신에게 할당된 책임을 수행하던 중에 필요한 정보를 알지 못하거나 외 부의 도움이 필요한 경우 적절한 객체에게 메시지를 전송해서 협력을 요청

협력

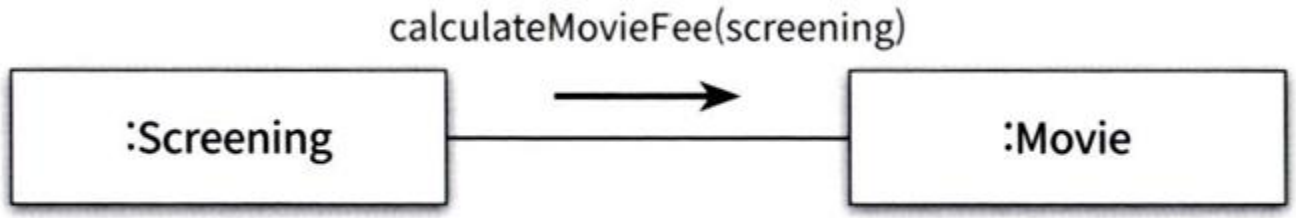


그림 3.2 Screening은 메시지를 전송해서 Movie와 협력한다

요금 계산하는 작업을 만약 Screening이 수행한다면?

객체의 자율성을 보장하기 위해서는 필요한 정보와 정보에 기반한 행동을 같은 객체 안에 모아놓아야 하는데 이 경우는 정보(가격정보)와 행동(요금계산)이 두 객체로 분리가 되어버림
=> Movie는 Screening이 요금계산시 필요한 정보를 주는 그런 수동적인 객체로 전락

그러므로 Movie가 바뀌면 Screening도 변경이 필요함(Movie내부 구현에 접근하니깐)
그래서, Screening과 Movie 사이의 결합도를 느슨하게 유지하기 위해서 캡슐화를 통해 외부객체가 내부구현에 접근못하도록 해야함

협력

상태는 객체가 행동하는 데 필요한 정보에 의해 결정되고 행동은 협력 안에서 객체가 처리할 메시지로 결정
=> 결과적으로 객체가 참여하는 협력이 객체를 구성하는 행동과 상태 모두를 결정
협력은 객체를 설계하는 데 필요한 일종의 문맥(context)을 제공

책임

책임이란 객체에 의해 정의되는 응집도 있는 행위의 집합으로, 객체가 유지해야 하는 정보와 수행 할 수 있는 행동에 대해 개략적으로 서술한 문장

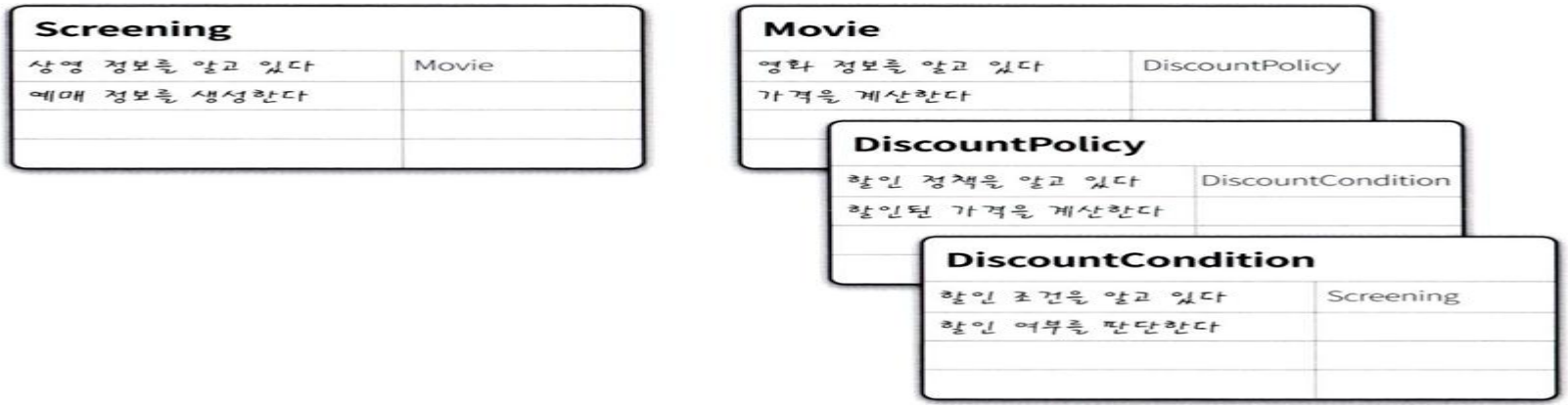
크레이그 라만(CraigLarman)의 책임분류

하는 것	아는 것
- 객체를 생성하거나 계산을 수행하는 등의 스스로 하는 것 - 다른 객체의 행동을 시작시키는 것 - 다른 객체의 활동을 제어하고 조절하는 것	- 사적인 정보에 관해 아는 것 - 관련된 객체에 관해 아는 것 - 자신이 유도하거나 계산할 수 있는 것에 관해 아는 것

Screening의 책임 => 영화를 예매
Movie의 책임 => 요금을 계산
<하는 것과 관련된 책임>

Screening의 책임 => 자신이 상영할 영화를 알고 있는 것
<아는 것과 관련된 책임>

책임



객체는 자신이 맡은 책임을 수행하는 데 필요한 정보를 알고 있을 책임이 있다.
객체는 자신이 할 수 없는 작업을 도와줄 객체를 알고 있을 책임이 있다.

적절한 책임을 적절한 객체에게 할당해야만 단순하고 유연한 설계를 창조할 수 있다

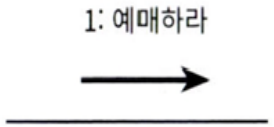
책임

INFORMATION EXPERT(정보 전문가) 패턴

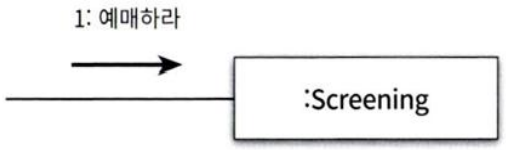
자율적인 객체를 만드는 가장 기본적인 방법은 책임을 수행하는 데 필요한 정보를 가장 잘 알고 있는 전문가에게 그 책임을 할당하는 것

영화 예매 시스템

예매하라 라는 이름의 메시지로 협력을 시작



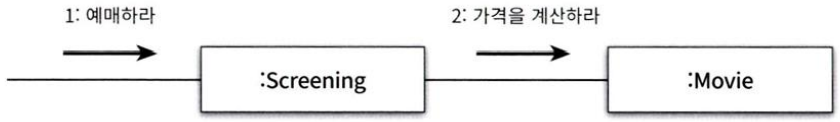
영화를 예매하기 위해서는 상영 시간과 기본 요금을 알아야 한다.



영화를 예매하기 위해서는 예매 가격을 계산 -> Screening이 외부의 객체에게 가격 계산을 요청

책임

메시지를 처리할 적 절한 객체를 선택
- 가격을 계산하는 데 필요한 정보를 가장 많이 알고 있는 정보 전문가를 선택(Movie)



책임주도 설계(RDD)

어떤 책임을 선택하느냐가 전체적인 설계의 방향과 흐름을 결정 => 책임을 찾고 책임을 수행할 적절한 객체를 찾아 책임을 할당하는 방식으로 협력을 설계하는 방법(RDD)

- 시스템이 사용자에게 제공해야 하는 기능인 시스템 책임을 파악한다.
- 시스템 책임을 더 작은 책임으로 분할한다.
- 분할된 책임을 수행할 수 있는 적절한 객체 또는 역할을 찾아 책임을 할당한다.
- 객체가 책임을 수행하는 도중 다른 객체의 도움이 필요한 경우 이를 책임질 적절한 객체 또는 역할을 찾는다.
- 해당 객체 또는 역할에게 책임을 할당함으로써 두 객체가 협력하게 한다.

책임

객체가 메시지를 선택하는 것이 아니라 메시지가 객체를 선택하게 해야함

- 객체가 최소한의 인터페이스(minimal interface)[Weisfeld08]를 가질 수 있게 된다.
- 객체는 충분히 추상적인 인터페이스(abstract interface)[Weisfeld08]를 가질 수 있게 된다.

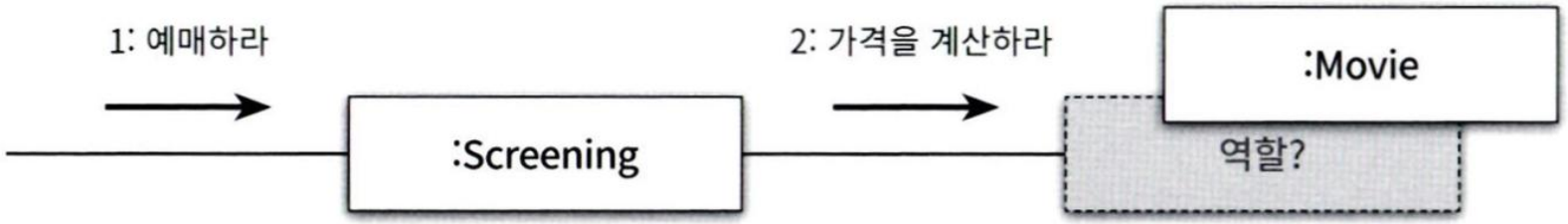
행동이 상태 결정

- 객체의 존재 이유는 협력
- 상태보다 행동이 우선

역할

객체가 특정한 협력 안에서 수행하는 책임의 집합

- Screening이 전송하는 `가격을 계산하라` 라는 메시지를 수신한 객체는 Movie 인스턴스이다.
- 그러나 (가격을 계산할)역할에 관해 먼저 고민하고, 이를 수행할 객체로 Movie를 를 선택한 것이다.
- 역할에 특별한 이름이 붙지는 않았으나, 객체를 수용할 수 있는 **위치로써 역할이라는 개념**은 여전히 유효하다.
- 왜 협력이라는 개념을 도입한 걸까?

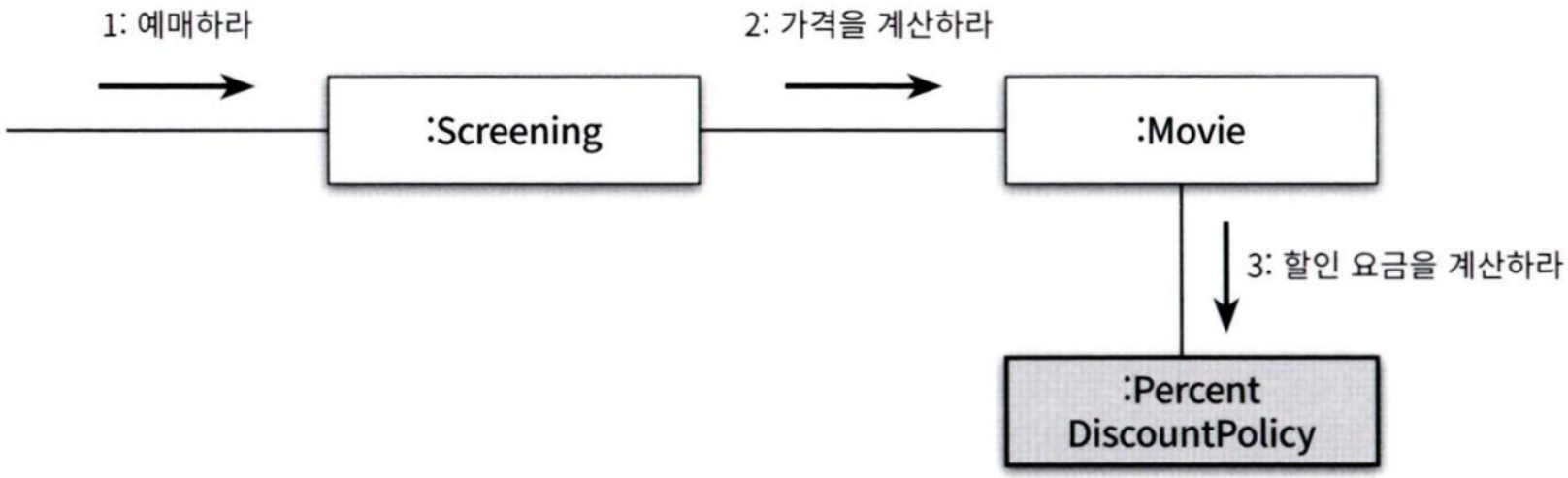


유연하고 재사용 가능한 협력

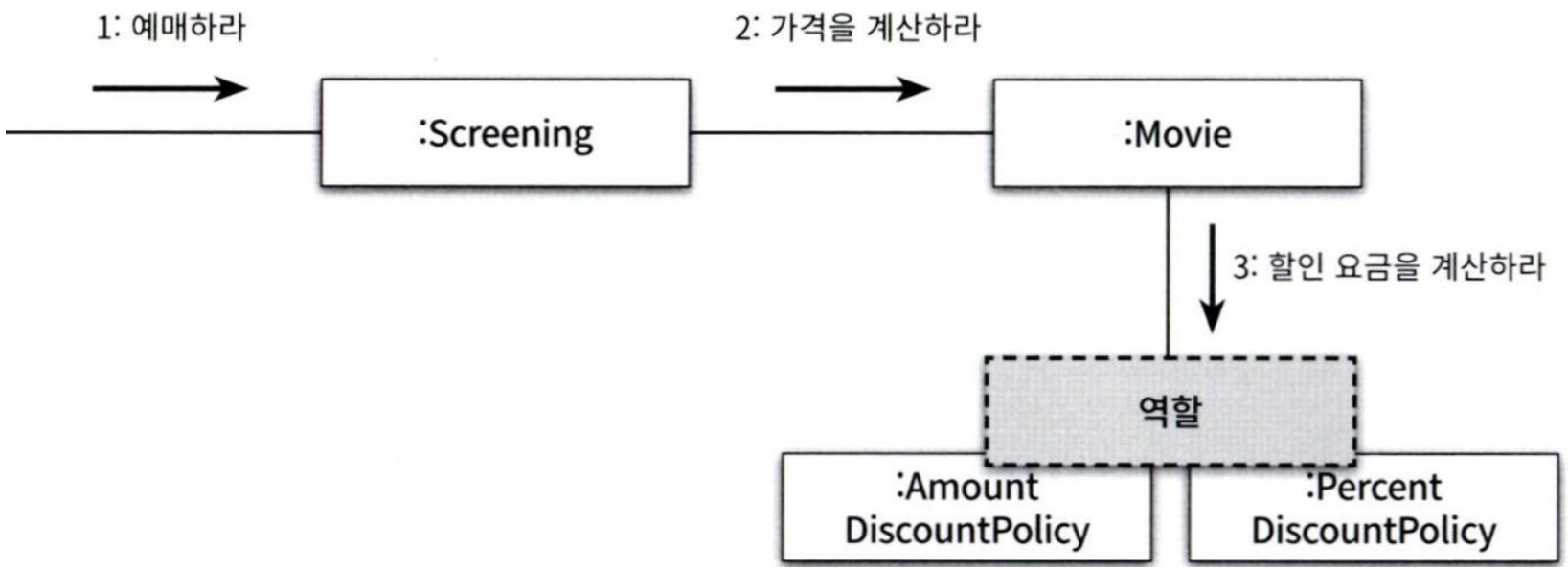
역할이 중요한 이유는 역할을 통해 유연하고 재사용 가능한 협력을 얻을 수 있기 때문이다

- 만일 역할이라는 개념 없이 객체에게 책임을 할당한다고 생각해보자. Movie 는 가격을 계산하기 위해 `할인 요금` 정보가 필요하다. 이를 위해 **할인요금을 계산하라** 라는 명령을 전송해서 외부의 객체에게 도움을 요청한다.
- 이때, `금액 할인 정책` 과 `비율 할인 정책` 두개라면? 전부 개별적으로 생각해야 할까? 개별 영화마다 정책이 바뀌면 이를 계속 수정하거나, 개별적으로 두 종류의 객체가 참여하는 협력을 만들어야 한다.

- 이러한 구현의 결과는 **코드중복**이다. 이러한 방법은 지양해야 한다.



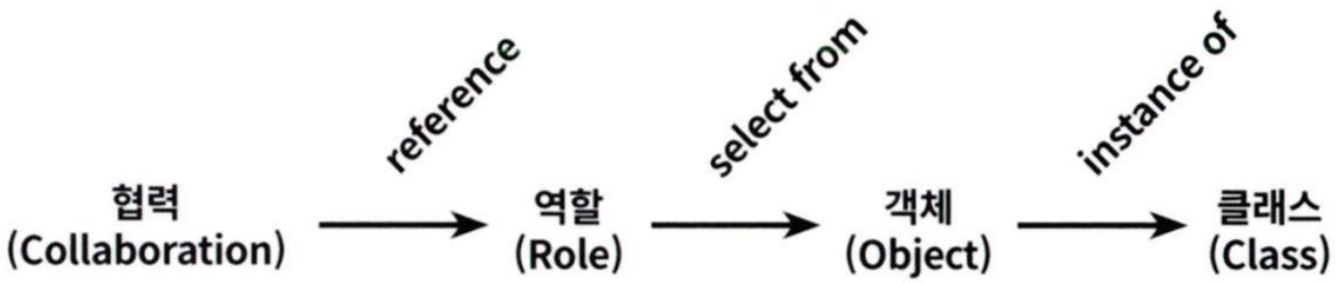
- 이를 해결하기 위해서는, 객체가 아닌 **책임** 에 집중해야 한다.
- 두 정책 모두 **할인 요금 계산** 이라는 동일한 **책임**을 수행한다는 것을 알 수 있다.
- 객체라는 존재를 지우고 **책임**을 강조하자.



추상화
두 객체를 포괄하는 역할

- 역할을 이용하면 불필요한 중복 코드를 제거할 수 있다.
- 새로운 할인 정책을 추가하기 위해 새로운 협력(3번 할인 정책을 적용하시오 등)을 추가할 필요가 없다.

협력에 참여하는 후보가 여러 종류의 객체에 의해 수행된다면 그 후보는 역할이지만, 단지 한 종류의 객체만이 협력에 참여하면 후보는 객체가 된다. - 레베카 워프스브록



추상화의 장점

1. 세부사항에 종속되지 않고 상위 수준의 역할을 쉽고 간단하게 표현할 수 있다
   - 할인 정책 각각의 할인 계산 방법에 **종속**되지 않는다
2. 설계를 유연하게 만들 수 있다
   - 해당 역할을 수행하는 할인 종류 또한 신경 쓸 필요 없다

배우와 배역을 통한 추상화 이해

연극의 막이 오르면, 배우 개인은 없어지고 해당 배역들만 남게 된다. 배우들은 배역을 충실히 수행하며 해당 배역의 `역할`을 책임진다.

연극의 막이 내리면, `배역` 은 사라지고 다시 본래 `배우`가 된다.
- 배역은 연극 배우가 특정 연극에서 연기하는 역할이다.
    - 인터페이스(배역)과 클래스(배우)의 관계에 대한 설명
- 배역은 연극이 상영되는 동안에만 존재하는 일시적인 개념이다.
    - 인터페이스는 추상적(연극 상영 중)이다.
- 서로 다른 배우들이 동일한 배역을 맡을 수 있다.
    - 여러 클래스가 한 인터페이스를 구현해낼 수 있다.

- 하나의 배우가 다양한 연극에서 서로 다른/같은 배역을 연기할 수 있다
- 한 클래스(객체)가 다양한 협력 속에서 다양한 역할을 맡게 된다.