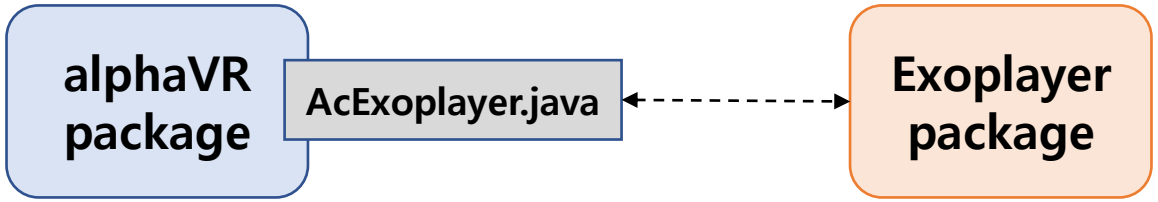
Clean Code

8. 경계

소프트웨어의 경계

- **소프트웨어의 경계란?**
  - 오픈 소스, 패키지 제공자로 부터 구입한 패키지 코드, 사내 다른 팀이 제공한 코드 등 외부 코드와 우리 코드가 만나는 영역 (인터페이스)
- **패키지 제공자와 사용자 간의 문제**
  - 제공자: 더 많은 환경에서 패키지를 구동하기 위해 넓은 적용성의 인터페이스를 제공한다.
  - 사용자: 자신의 요구(우리 코드에 필요한 만큼) 에 집중하는 인터페이스를 제공받기를 원한다.
- **문제의 해결법**
  - 외부 코드를 감싸는 코드를 만들고 우리 코드에 필요한 인터페이스만 제공한다. (**Facade 패턴**)
  - 공개해야 하는 API 의 인수/ 반환 값으로 외부 코드를 사용하지 않는다.
  - example : AVPT/alphaVR/AcExoPlayer는 Exoplayer 객체를 내부에 숨기고 외부에는 필요한 인터페이스만 제공한다.



외부 패키지 익히기

- **외부 패키지를 사용하기 위해 패키지를 학습하는 방법**
  - 학습 테스트 : 기능 구현을 위한 테스트가 아니라 외부 패키지가 어떻게 동작하는지 검증하기 위한 테스트
- **학습 테스트**
  - 다양한 조건에서 기능을 손쉽게 확인할 수 있다.
  - 외부 패키지의 버전 업그레이드 시 호환성 검증을 학습 테스트를 통해 진행할 수 있다.
  - 테스트코드 작성법을 익히는 좋은 훈련이 될 수 있다.
  - Example

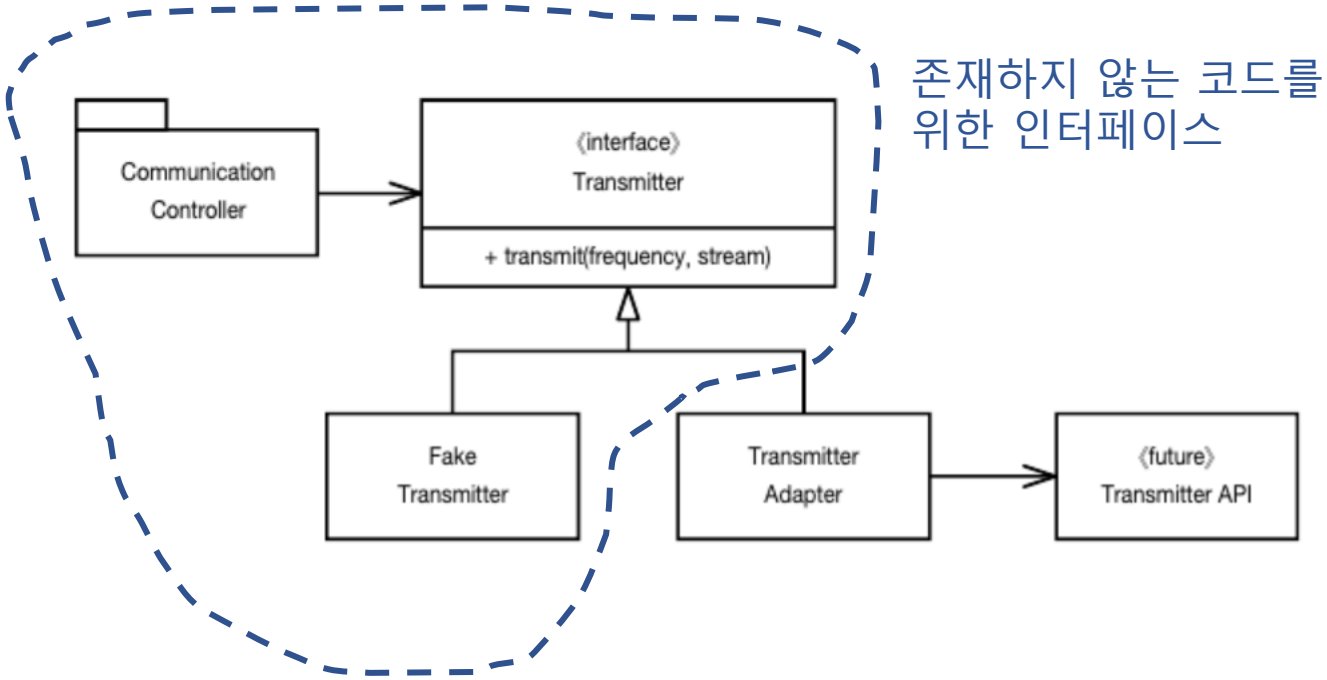
```tsx
import { TextField } from '@mui/material'
import { render } from '@testing-library/react'

describe('MUI TextField Test', () => {
  it('Render Test', () => {
    const { getByLabelText } = render(<TextField label="username" value="june" />)
    const username = getByLabelText('username') as HTMLInputElement
    expect(username.value).toBe('june')
  })
})
```

- MUI 라는 외부 패키지를 익혀야 한다면 react testing library 를 사용해 학습테스트를 작성할 수 있다.
- Storybook으로 UI 컴포넌트를 만들어보는 것도 일종의 학습 테스트라고 볼 수도 있을 것 같다.

아직 존재하지 않는 코드 사용하기

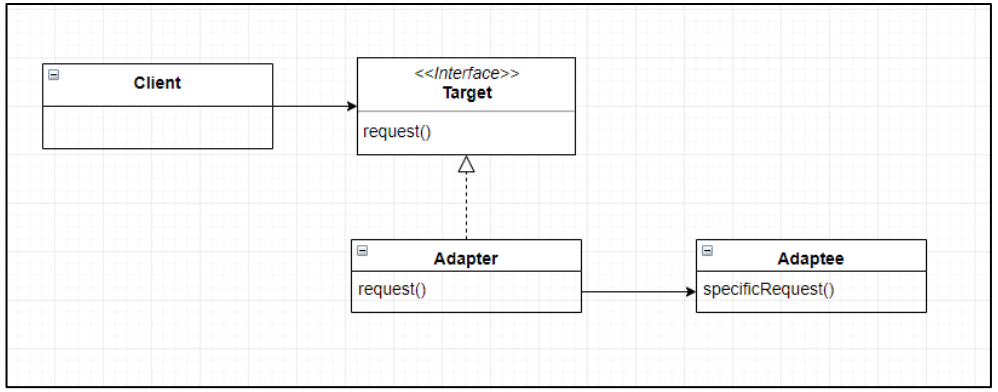
- **경계는 아는 코드와 모르는 코드를 분리하는 영역일 수 있다.**
  - 다른 팀과 협업 시 다른 팀이 아직 개발 중인 API를 사용해야하는 부분을 우리 코드에서 사용해야 할 수 도 있다.
- **경계 영역에 인터페이스를 만들고 해당 인터페이스를 구현하는 Fake API class 를 구현해 개발을 하고 추후 다른 팀의 API 개발이 완료되면 해당 코드를 사용한 class를 구현한다.**
  - 이 때 Adapter 패턴을 사용해 API 사용을 캡슐화 한다.



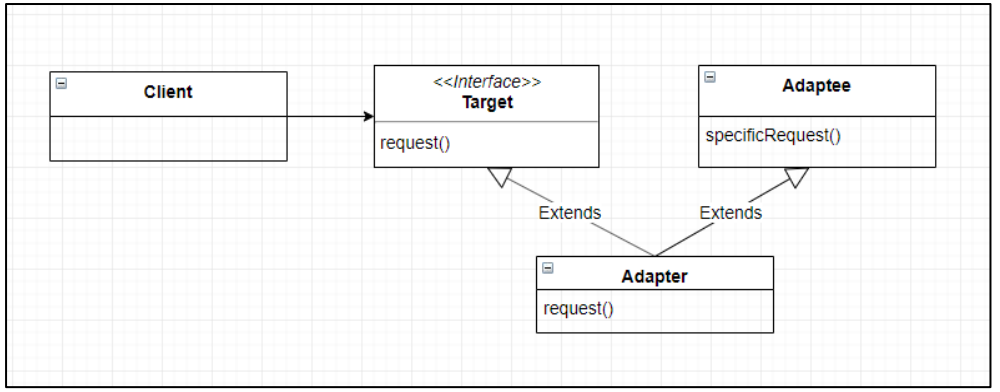
Adapter 패턴

- 한 클래스의 인터페이스를 클라이언트에서 사용하고자 하는 다른 인터페이스로 변환한다.

Object Adapter



Class Adapter



	Object Adapter	Class Adapter
장점	• 상속이 아닌 Composition을 사용하기 때문에 더 유연하다. (Adaptee의 모든 서브클래스에 대해 Adapter 사용가능)	• Adapter가 Adaptee의 sub-class이기 때문에 Adaptee의 행동을 Override 할 수 있다. • Adaptee 객체를 만들지 않아도 된다.
단점	• Adaptee 객체를 만들어야 사용 가능하다.	• 상속을 이용하기 때문에 한 Adapter 클래스가 특정 Adatee 클래스에만 적용 가능하다. • 다중상속이 지원되는 언어에서만 사용 가능하다.

참고 : https://jihyehwang09.github.io/2019/11/17/design-pattern-adapter-pacade/

정리

- **통제하지 못하는 코드를 사용할 때는 너무 많은 투자를 하거나 향후 변경 비용이 지나치게 커지지 않도록 각별히 주의해야한다.**
- **경계에 위치하는 코드는 깔끔히 분리하고 테스트 케이스를 작성한다.**
- **외부 패키지를 호출하는 코드를 가능한 줄여 경계를 관리해야 한다.**
  - 경계를 감싸는 Wrapper class
  - Adapter 패턴