

INFCON 2023

점진적 추상화

추상화 방향, 범위, 시기 그리고 점진적 추상화에 대한 이야기

토스증권 이승천



연사 소개

커리어

- 21.05 ~ 토스증권 해외주식 원장 서버 개발
- 20.05 ~ 21.05 카사코리아 (부동산 유동증권 거래소) 서버 개발

그 외

- 토스 SLASH 22 - 애플 한 주가 고객에게 전달되기까지

오늘 발표는

하나의 예제에서 추가되는 요구사항을 여러분과 같이 대응해보면서
그 동안 제가 추상화에 대해 느낀점들을 매우 주관적인 관점에서 공유드립니다.

목차

1. 추상화 방향
2. 추상화 범위
3. 추상화 시기
4. 변화무쌍한 요구사항을 유연하게

1. 추상화 방향

요구사항

원화/달러 입출금을 관리하는 시스템을 구축중인 스타트업의 개발자

요구사항 – 픽션입니다. (다소 억지스러운 면이 있음)

요구사항

목적 - 내부 직원이 사용할 백오피스 입출금 API 개발

| 달러 입금/출금, 원화 입금/출금 지원



입출금 요청

- 입력값: 계좌번호, 원화/달러 입출금 타입, 금액
- 기대동작: 요청내역 저장



입출금 승인

- 입력값: 입출금 요청 객체 식별자
- 기대동작: 타입별 입출금 실행

입출금 요청 객체

```
01 class TrxRequest(  
02     val account: String,  
03     val amount: Long,  
04     val type: Type,  
05 ) {  
06     val id: String = UUID.randomUUID().toString()  
07     var status: Status = Status.CREATED  
08  
09     enum class Type(name: String) {  
10         DEPOSIT_KRW("원화입금"),  
11         WITHDRAW_KRW("원화출금"),  
12         DEPOSIT_USD("달러입금"),  
13         WITHDRAW_USD("달러출금"),  
14     }  
15  
16     enum class Status(name: String) {  
17         CREATED("생성"),  
18         ACCEPTED("승인")  
19     }  
20 }
```


IF 문 구현

```
01 class TrxRequestService {
02     fun create(account: String, amount: Long, type: TrxRequest.Type) {
03         save(TrxRequest(account, amount, type))
04     }
05
06     fun accept(id: Long) {
07         val trxRequest = getTrxRequest(id)
08         trxRequest.accept()
09
10         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
11             depositKrw(trxRequest.account, trxRequest.amount)
12         }
13
14         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
15             withdrawKrw(trxRequest.account, trxRequest.amount)
16         }
17
18         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
19             depositUsd(trxRequest.account, trxRequest.amount)
20         }
21
22         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
23             withdrawUsd(trxRequest.account, trxRequest.amount)
24         }
25     }
26 }
```

IF 문 구현

```
01 class TrxRequestService {
02     fun create(account: String, amount: Long, type: TrxRequest.Type) {
03         save(TrxRequest(account, amount, type))
04     }
05
06     fun accept(id: Long) {
07         val trxRequest = getTrxRequest(id)
08         trxRequest.accept()
09
10         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
11             depositKrw(trxRequest.account, trxRequest.amount)
12         }
13
14         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
15             withdrawKrw(trxRequest.account, trxRequest.amount)
16         }
17
18         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
19             depositUsd(trxRequest.account, trxRequest.amount)
20         }
21
22         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
23             withdrawUsd(trxRequest.account, trxRequest.amount)
24         }
25     }
26 }
```



koogk7 now

Owner

...

OCP(open closed principle)에 위배되는 코드로 보이네요.
확장에 열려있는 코드면 더욱 좋겠네요 🙏



개방-폐쇄 원칙(OCP, Open-Closed Principle)은 '소프트웨어 개체(클래스, 모듈, 함수 등등)는 확장에 대해 열려 있어야 하고, 수정에 대해서는 닫혀 있어야 한다'는 프로그래밍 원칙이다.

상세설명 [\[편집 \]](#)

소프트웨어 개발 작업에 이용된 많은 모듈 중에 하나에 수정을 가할 때 그 모듈을 이용하는 다른 모듈을 줄줄이 고쳐야 한다면, 이와 같은 프로그램은 수정하기가 어렵다. 개방-폐쇄 원칙은 시스템의 구조를 올바르게 재조직(리팩토링)하여 나중에 이와 같은 유형의 변경이 더 이상의 수정을 유발하지 않도록 하는 것이다. 개방-폐쇄 원칙이 잘 적용되면, 기능을 추가하거나 변경해야 할 때 이미 제대로 동작하고 있던 원래 코드를 변경하지 않아도, 기존의 코드에 새로운 코드를 추가함으로써 기능의 추가나 변경이 가능하다.

개방-폐쇄 원칙의 두 가지 속성 [\[편집 \]](#)

확장에 대해 열려 있다. [\[편집 \]](#)

이것은 모듈의 동작을 확장할 수 있다는 것을 의미한다. 애플리케이션의 요구 사항이 변경될 때, 이 변경에 맞게 새로운 동작을 추가해 모듈을 확장할 수 있다. 즉, 모듈이 하는 일을 변경할 수 있다.

수정에 대해 닫혀 있다 [\[편집 \]](#)

모듈의 소스 코드나 바이너리 코드를 수정하지 않아도 모듈의 기능을 확장하거나 변경할 수 있다. 그 모듈의 실행 가능한 바이너리 형태나 링크 가능한 라이브러리(예를 들어 윈도우의 DLL이나 자바의 .jar)를 건드릴 필요가 없다.

SOLID

원칙

단일 책임 원칙

개방-폐쇄 원칙

리스코프 치환 원칙

인터페이스 분리 원칙

의존관계 역전 원칙

V · T · E

타입을 축으로 추상화

```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5 }
```

타입을 축으로 추상화

```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5 }
```

```
01 class DepositKrwTypeService: TrxTypeService {  
02     override fun create(account: String, amount: Long) {  
03         save(TrxRequest(account, amount,  
TrxRequest.Type.DEPOSIT_KRW))  
04     }  
05  
06     override fun accept(id: Long) {  
07         val trxRequest = getTrxRequest(id)  
08         trxRequest.accept()  
09         depositKrw(trxRequest.account, trxRequest.amount)  
10     }  
11 }  
12
```

타입을 축으로 추상화

```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5 }
```

```
01 class DepositKrwTypeService: TrxTypeService {  
02     override fun create(account: String, amount: Long) {  
03         save(TrxRequest(account, amount,  
TrxRequest.Type.DEPOSIT_KRW))  
04     }  
05  
06     override fun accept(id: Long) {  
07         val trxRequest = getTrxRequest(id)  
08         trxRequest.accept()  
09         depositKrw(trxRequest.account, trxRequest.amount)  
10     }  
11 }  
12  
13 class WithdrawKrwTypeService: TrxTypeService {  
14     override fun create(account: String, amount: Long) {  
15         save(TrxRequest(account, amount,  
TrxRequest.Type.WITHDRAW_KRW))  
16     }  
17  
18     override fun accept(id: Long) {  
19         val trxRequest = getTrxRequest(id)  
20         trxRequest.accept()  
21         withdrawKrw(trxRequest.account, trxRequest.amount)  
22     }  
23 }  
24
```

타입을 축으로 추상화

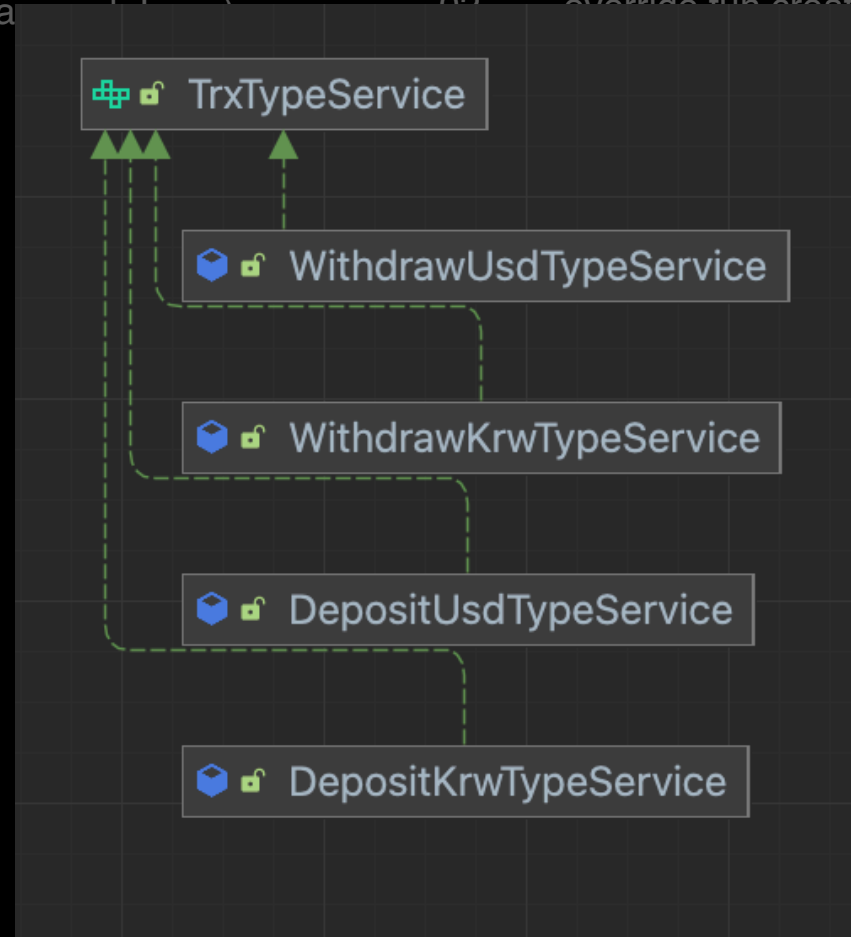
```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5 }
```

```
01 class DepositKrwTypeService: TrxTypeService {  
02     override fun create(account: String, amount: Long) {  
03         save(TrxRequest(account, amount,  
TrxRequest.Type.DEPOSIT_KRW))  
04     }  
05  
06     override fun accept(id: Long) {  
07         val trxRequest = getTrxRequest(id)  
08         trxRequest.accept()  
09         depositKrw(trxRequest.account, trxRequest.amount)  
10     }  
11 }  
12  
13 class WithdrawKrwTypeService: TrxTypeService {  
14     override fun create(account: String, amount: Long) {  
15         save(TrxRequest(account, amount,  
TrxRequest.Type.WITHDRAW_KRW))  
16     }  
17  
18     override fun accept(id: Long) {  
19         val trxRequest = getTrxRequest(id)  
20         trxRequest.accept()  
21         withdrawKrw(trxRequest.account, trxRequest.amount)  
22     }  
23 }  
24
```


타입을 축으로 추상화

```
1 interface TrxTypeService {
2     fun create(account: String, amount: Long) {
3
4     fun accept(id: Long)
5 }
```

```
01 class DepositKrwTypeService: TrxTypeService {
02     override fun create(account: String, amount: Long) {
03         // ...
04         st(account, amount,
05             IT_KRW))
06     }
07
08     fun accept(id: Long) {
09         // ...
10         = getTrxRequest(id)
11         ept()
12         request.account, trxRequest.amount)
13     }
14 }
15
16 class WithdrawKrwTypeService: TrxTypeService {
17     override fun create(account: String, amount: Long) {
18         // ...
19         st(account, amount,
20             RAW_KRW))
21     }
22
23     fun accept(id: Long) {
24         // ...
25         = getTrxRequest(id)
26         ept()
27         withdrawKrw(trxRequest.account, trxRequest.amount)
28     }
29 }
```



어떤 코드가 마음에 드시나요?

```

01 class TrxRequestService {
02     fun create(account: String, amount: Long, type: TrxRequest.Type) {
03         save(TrxRequest(account, amount, type))
04     }
05
06     fun accept(id: Long) {
07         val trxRequest = getTrxRequest(id)
08         trxRequest.accept()
09
10         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
11             depositKrw(trxRequest.account, trxRequest.amount)
12         }
13
14         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
15             withdrawKrw(trxRequest.account, trxRequest.amount)
16         }
17
18         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
19             depositUsd(trxRequest.account, trxRequest.amount)
20         }
21
22         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
23             withdrawUsd(trxRequest.account, trxRequest.amount)
24         }
25     }
26 }

```

```

01 class DepositKrwTypeService: TrxTypeService {
02     override fun create(account: String, amount: Long) {
03         save(TrxRequest(account, amount,
04             TrxRequest.Type.DEPOSIT_KRW))
05     }
06
07     override fun accept(id: Long) {
08         val trxRequest = getTrxRequest(id)
09         trxRequest.accept()
10         depositKrw(trxRequest.account, trxRequest.amount)
11     }
12
13 class WithdrawKrwTypeService: TrxTypeService {
14     override fun create(account: String, amount: Long) {
15         save(TrxRequest(account, amount,
16             TrxRequest.Type.WITHDRAW_KRW))
17     }
18
19     override fun accept(id: Long) {
20         val trxRequest = getTrxRequest(id)
21         trxRequest.accept()
22         withdrawKrw(trxRequest.account, trxRequest.amount)
23     }
24 }

```

추가 요구사항

미국계좌는 미국지사 직원이 직접 트리거 해야합니다.
달러 입출금은 승인과 처리를 분리해주세요.

IF 문 구현

```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount)
18         }
19     }
20 }
```

IF 문 구현

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount)
18         }
19     }
20 }

```

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11     }
12
13     fun process(id: Long) {
14         ...
15         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
16             depositUsd(trxRequest.account, trxRequest.amount)
17         }
18
19         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
20             withdrawUsd(trxRequest.account, trxRequest.amount)
21         }
22     }
23 }

```

IF 문 구현

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount)
18         }
19     }
20 }

```

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11     }
12
13     fun process(id: Long) {
14         ...
15         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
16             depositUsd(trxRequest.account, trxRequest.amount)
17         }
18
19         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
20             withdrawUsd(trxRequest.account, trxRequest.amount)
21         }
22     }
23 }

```

타입을 축으로 추상화

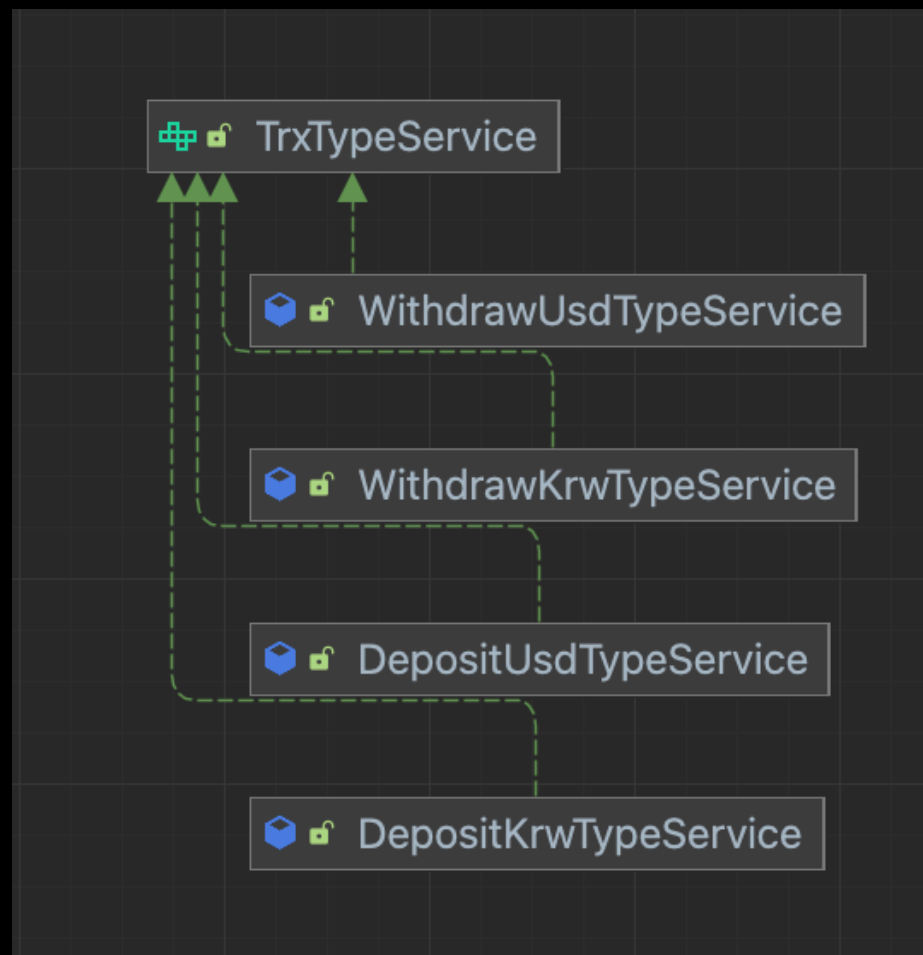
```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5 }
```

인터페이스가 수정됨

```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5  
6     fun process(id: Long)  
7 }
```


구현체 전체가 수정되어야 함

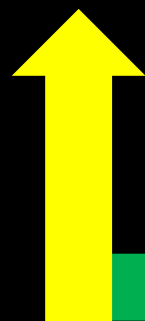
```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5  
6     fun process(id: Long)  
7 }
```



인터페이스 자체를 변경하는 것은 쉽지 않다

어느 방향으로 발전할 것인가?

타입 (원화 입출금, 달러 입출금)

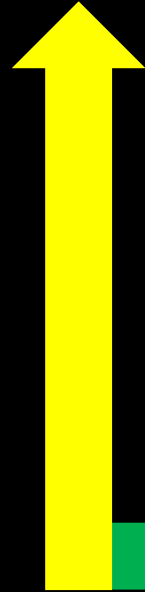


행위 (생성, 승인)



우리가 진행했던 추상화 방향

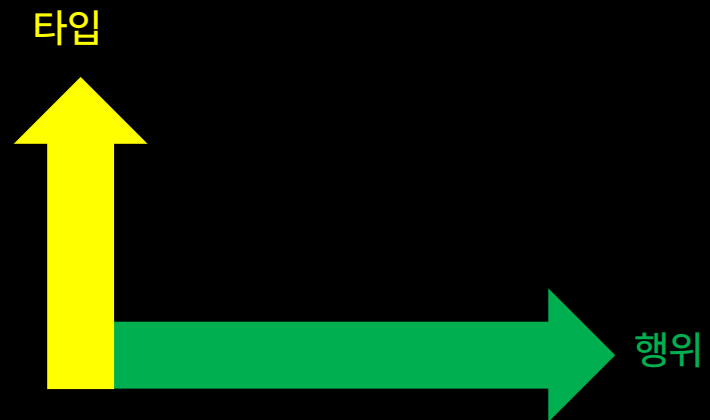
타입



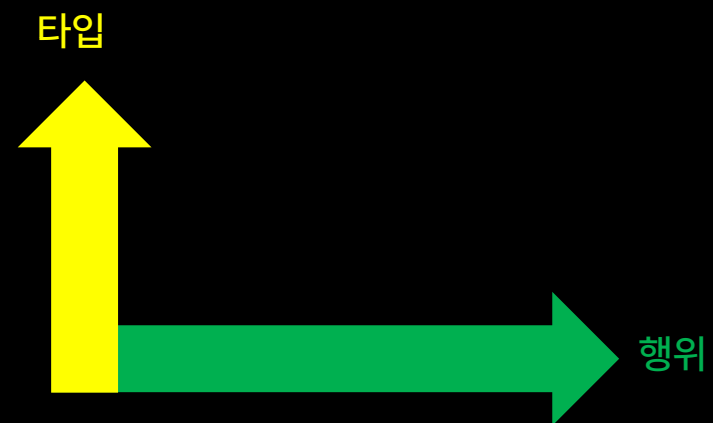
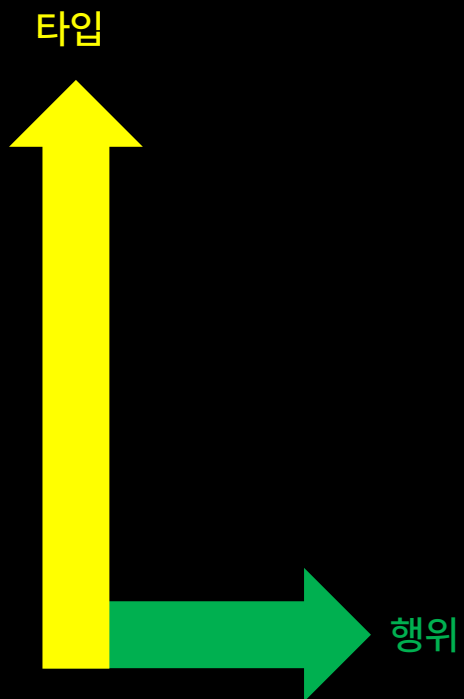
행위



소프트웨어가 발전한 방향



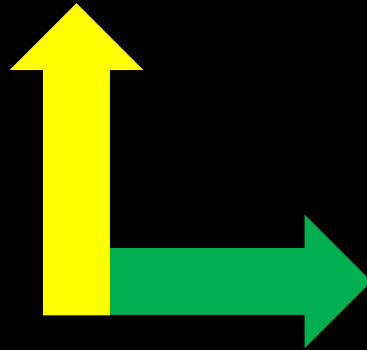
어느 방향으로 발전할지 개발초기에는 알기 쉽지 않다



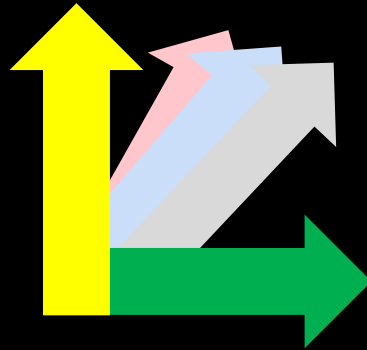
소프트웨어 발전방향과 일치하지 않는 추상화

소프트웨어 발전방향과 일치하지 않는 추상화
오히려 더 유지보수하기 힘든 코드가 탄생

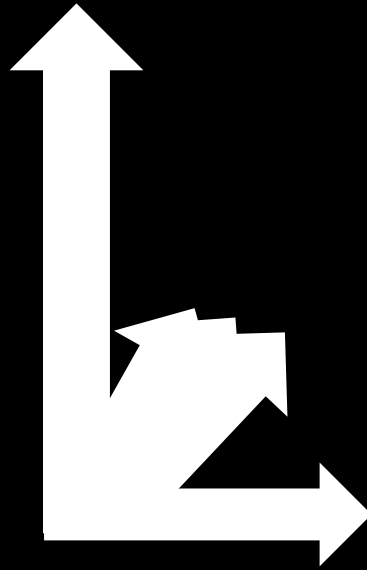
현실은 수 많은 축 들이 존재



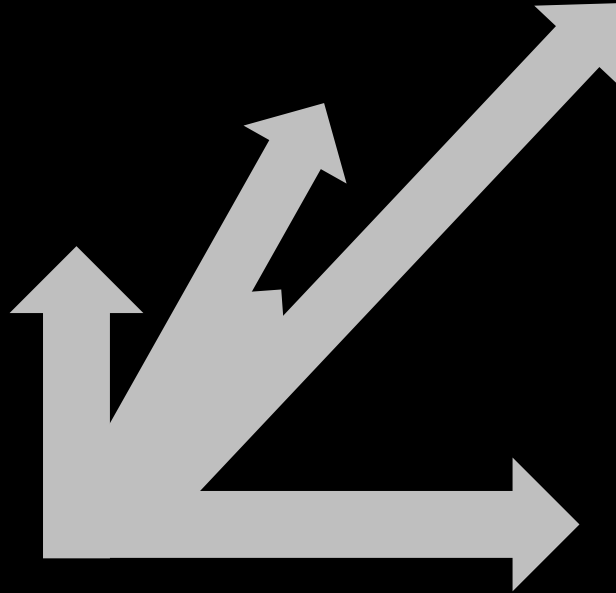
현실은 수 많은 축 들이 존재



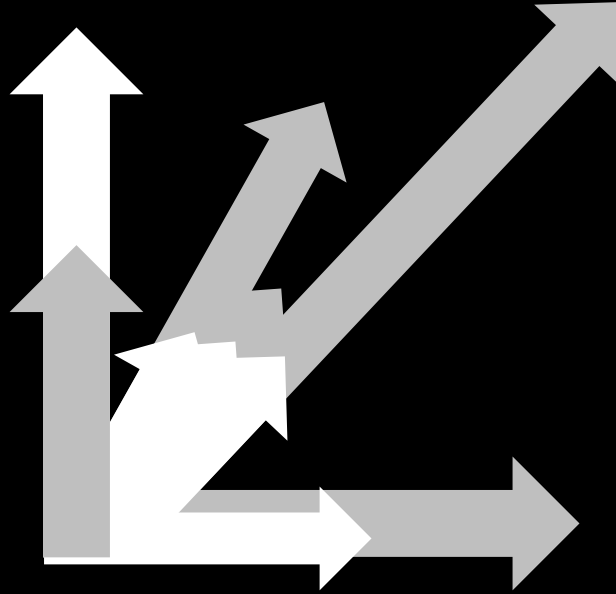
소프트웨어 발전방향을 예측



현실은...



구현과 소프트웨어 사이의 괴리가 발생



방향을 예측하기란 쉽지 않다. 그렇다면

2. 추상화 범위

추상화의 범위를 조금 좁혔다면

```
01 class DepositKrwTypeService: TrxTypeService {
02     override fun create(account: String, amount: Long) {
03         save(TrxRequest(account, amount,
04             TrxRequest.Type.DEPOSIT_KRW))
05     }
06     override fun accept(id: Long) {
07         val trxRequest = getTrxRequest(id)
08         trxRequest.accept()
09         depositKrw(trxRequest.account, trxRequest.amount)
10     }
11 }
12
```

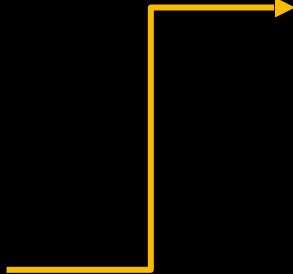

코어로직만 추상화 계층으로

```
01 class DepositKrwTypeService: TrxTypeService {  
02     override fun create(account: String, amount: Long) {  
03         save(TrxRequest(account, amount,  
TrxRequest.Type.DEPOSIT_KRW))  
04     }  
05  
06     override fun accept(id: Long) {  
07         val trxRequest = getTrxRequest(id)  
08         trxRequest.accept()  
09         depositKrw(trxRequest.account, trxRequest.amount)  
10     }  
11 }  
12
```

코어로직만 추상화 계층으로

```
01 class DepositKrwTypeService: TrxTypeService {  
02     override fun create(account: String, amount: Long) {  
03         save(TrxRequest(account, amount,  
TrxRequest.Type.DEPOSIT_KRW))  
04     }  
05  
06     override fun accept(id: Long) {  
07         val trxRequest = getTrxRequest(id)  
08         trxRequest.accept()  
09         depositKrw(trxRequest.account, trxRequest.amount)  
10     }  
11 }  
12
```

```
1 class DepositKrwProcessor: TrxRequestProcessor {  
2     override fun process(account: String, amount: BigDecimal) {  
3         ....  
4     }  
5 }
```



코어로직만 추상화 계층으로

```
1 interface TrxTypeService {  
2     fun create(account: String, amount: Long)  
3  
4     fun accept(id: Long)  
5 }
```

```
1 interface TrxRequestProcessor {  
2     fun process(account: String, amount: BigDecimal)  
3 }
```

최초 요구사항 – 원화/달러 모두 승인 후 처리

```
1 class TrxRequestService {  
2     fun accept(id: Long, processor: TrxRequestProcessor) {  
3         val trxRequest = getTrxRequest(id)  
4         trxRequest.accept()  
5         processor.process(trxRequest.account, trxRequest.amount)  
6     }  
7 }
```

인터페이스의 변경없이 대응

```
1 class TrxRequestService {  
2     fun accept(id: Long, processor: TrxRequestProcessor) {  
3         val trxRequest = getTrxRequest(id)  
4         trxRequest.accept()  
5         processor.process(trxRequest.account, trxRequest.amount)  
6     }  
7 }
```

```
01 class TrxRequestService {  
02     fun accept(id: Long, processor: TrxRequestProcessor) {  
03         val trxRequest = getTrxRequest(id)  
04         trxRequest.accept()  
05  
06         if (trxRequest.type.isUsd()) {  
07             return  
08         }  
09  
10         processor.process(trxRequest.account, trxRequest.amount)  
11     }  
12  
13     fun process(id: Long, processor: TrxRequestProcessor) {  
14         ...  
15         if (trxRequest.type.isUsd()) {  
16             processor.process(trxRequest.account, trxRequest.amount)  
17         }  
18     }  
19 }
```

너무 넓은 범위를 추상화 할 경우
인터페이스가 변경되는 위험이 커짐

인터페이스 추상화는 필요한 만큼만

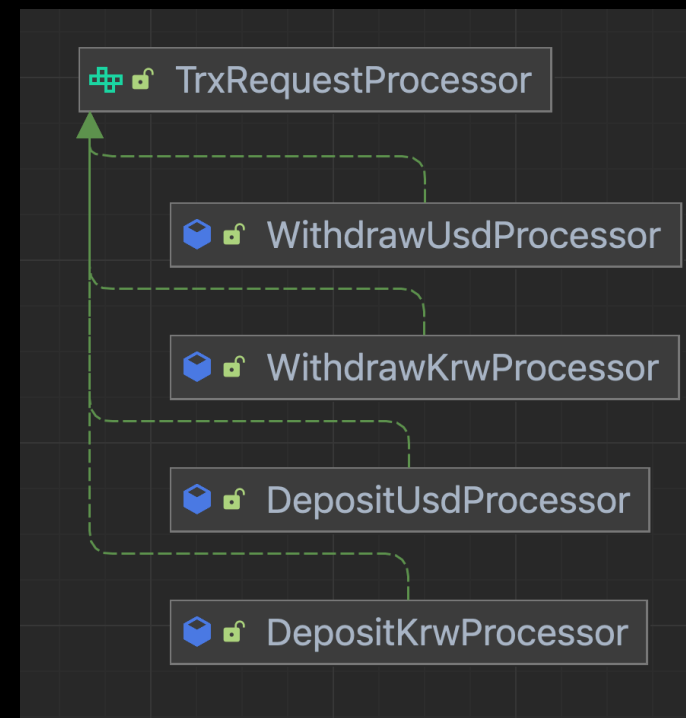
3. 추상화 시기

끊임없는 요구사항

인프콘 포인트 계좌가 추가되었어요.
인프콘 포인트도 입출금이 가능하도록 해주세요.
인프콘 포인트는 승인 즉시 수행되어야 합니다.

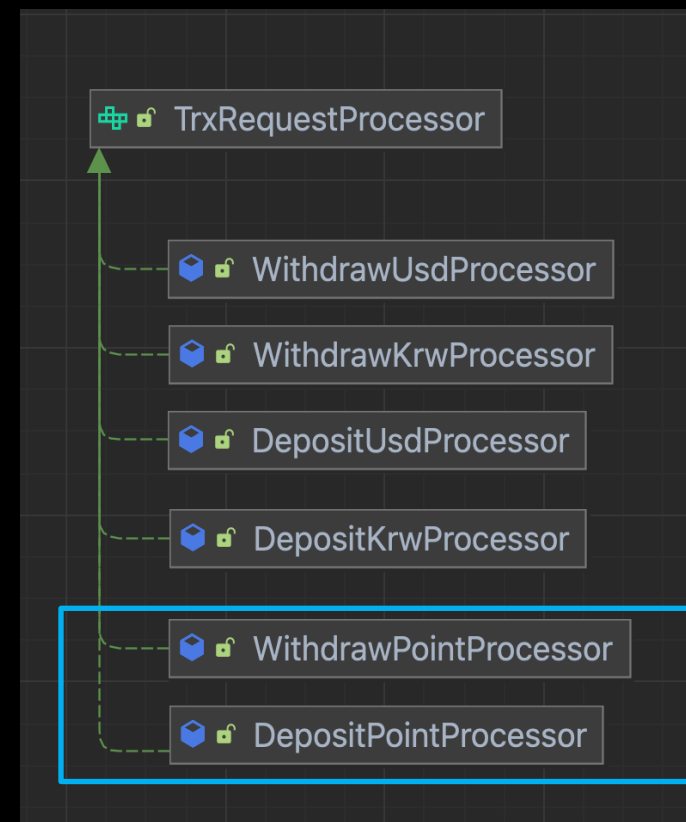
어떻게 요구사항을 대응 할 것인가?

```
01 class TrxRequestService {
02     fun accept(id: Long, processor: TrxRequestProcessor) {
03         val trxRequest = getTrxRequest(id)
04         trxRequest.accept()
05
06         if (trxRequest.type.isUsd()) {
07             return
08         }
09
10         processor.process(trxRequest.account, trxRequest.amount)
11     }
12
13     fun process(id: Long, processor: TrxRequestProcessor) {
14         ...
15         if (trxRequest.type.isUsd()) {
16             processor.process(trxRequest.account, trxRequest.amount)
17         }
18     }
19 }
```



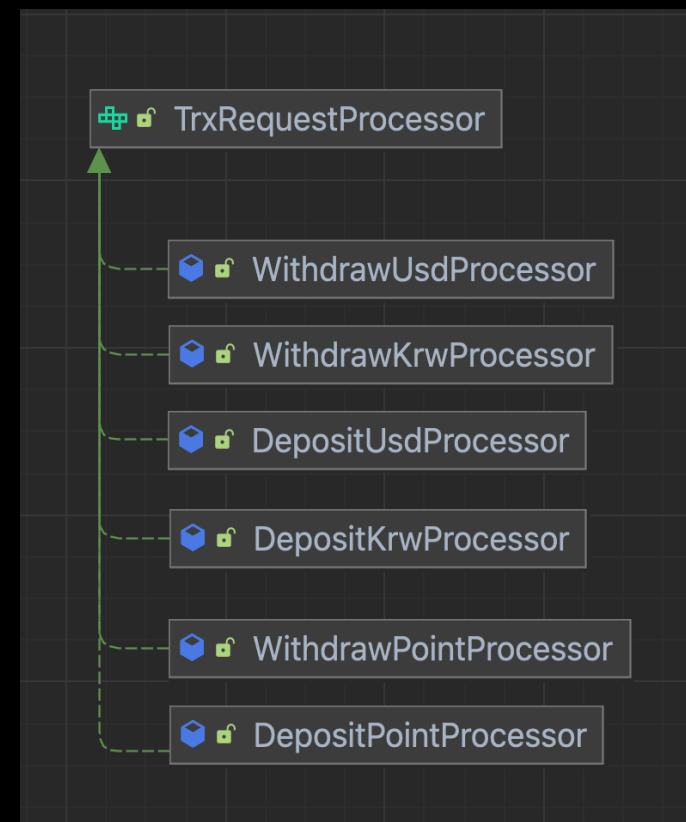
구현체 추가

```
01 class TrxRequestService {
02     fun accept(id: Long, processor: TrxRequestProcessor) {
03         val trxRequest = getTrxRequest(id)
04         trxRequest.accept()
05
06         if (trxRequest.type.isUsd()) {
07             return
08         }
09
10         processor.process(trxRequest.account, trxRequest.amount)
11     }
12
13     fun process(id: Long, processor: TrxRequestProcessor) {
14         ...
15         if (trxRequest.type.isUsd()) {
16             processor.process(trxRequest.account, trxRequest.amount)
17         }
18     }
19 }
```



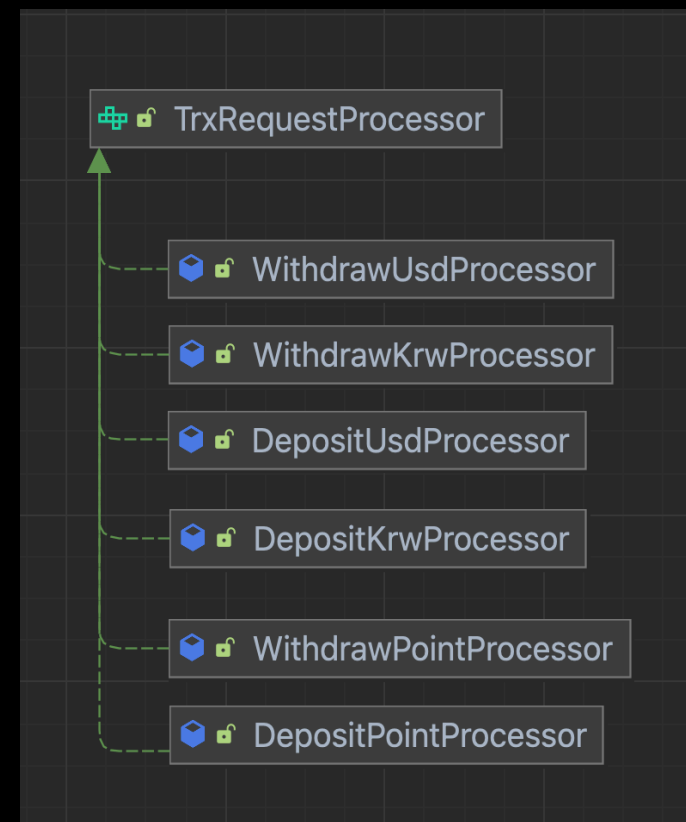
구현체 추가

```
01 class TrxRequestService {
02     fun accept(id: Long, processor: TrxRequestProcessor) {
03         val trxRequest = getTrxRequest(id)
04         trxRequest.accept()
05
06         if (trxRequest.type.isUsd()) {
07             return
08         }
09         인프콘 포인트는 승인 즉시 수행되어야 합니다.
10         processor.process(trxRequest.account, trxRequest.amount)
11     }
12
13     fun process(id: Long, processor: TrxRequestProcessor) {
14         ...
15         if (trxRequest.type.isUsd()) {
16             processor.process(trxRequest.account, trxRequest.amount)
17         }
18     }
19 }
```



운이 좋았다.

```
01 class TrxRequestService {
02     fun accept(id: Long, processor: TrxRequestProcessor) {
03         val trxRequest = getTrxRequest(id)
04         trxRequest.accept()
05
06         if (trxRequest.type.isUsd()) {
07             return
08         }
09
10         processor.process(trxRequest.account, trxRequest.amount)
11     }
12
13     fun process(id: Long, processor: TrxRequestProcessor) {
14         ...
15         if (trxRequest.type.isUsd()) {
16             processor.process(trxRequest.account, trxRequest.amount)
17         }
18     }
19 }
```



IF문 분기 구현에서 인프콘 포인트 요구사항을 마주했다면

```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
04             depositKrw(trxRequest.account, trxRequest.amount)
05         }
06
07         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
08             withdrawKrw(trxRequest.account, trxRequest.amount)
09         }
10
11         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
12             return
13         }
14
15         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
16             return
17         }
18     }
19 }
```

```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
04             depositKrw(trxRequest.account, trxRequest.amount)
05         }
06
07         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
08             withdrawKrw(trxRequest.account, trxRequest.amount)
09         }
10
11         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
12             return
13         }
14
15         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
16             return
17         }
18
19         if (trxRequest.type == TrxRequest.Type.DEPOSIT_POINT) {
20             depositPoint(trxRequest.account, trxRequest.amount)
21         }
22
23         if (trxRequest.type == TrxRequest.Type.WITHDRAW_POINT) {
24             withdrawPoint(trxRequest.account, trxRequest.amount)
25         }
26     }
27 }
```



```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
04             depositKrw(trxRequest.account, trxRequest.amount)
05         }
06
07         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
08             withdrawKrw(trxRequest.account, trxRequest.amount)
09         }
10
11         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
12             return
13         }
14
15         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
16             return
17         }
18
19         if (trxRequest.type == TrxRequest.Type.DEPOSIT_POINT) {
20             depositPoint(trxRequest.account, trxRequest.amount)
21         }
22
23         if (trxRequest.type == TrxRequest.Type.WITHDRAW_POINT) {
24             withdrawPoint(trxRequest.account, trxRequest.amount)
25         }
26     }
27 }
```

함수로 분리되어 있어 전체 흐름이 명확히 보임

```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
04             depositKrw(trxRequest.account, trxRequest.amount)
05         }
06
07         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
08             withdrawKrw(trxRequest.account, trxRequest.amount)
09         }
10
11         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
12             return
13         }
14
15         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
16             return
17         }
18
19         if (trxRequest.type == TrxRequest.Type.DEPOSIT_POINT) {
20             depositPoint(trxRequest.account, trxRequest.amount)
21         }
22
23         if (trxRequest.type == TrxRequest.Type.WITHDRAW_POINT) {
24             withdrawPoint(trxRequest.account, trxRequest.amount)
25         }
26     }
27 }
```

계좌를 기준으로 추상화하라는 시그널



```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
04             depositKrw(trxRequest.account, trxRequest.amount)
05         }
06
07         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
08             withdrawKrw(trxRequest.account, trxRequest.amount)
09         }
10
11         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
12             return
13         }
14
15         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
16             return
17         }
18
19         if (trxRequest.type == TrxRequest.Type.DEPOSIT_POINT) {
20             depositPoint(trxRequest.account, trxRequest.amount)
21         }
22
23         if (trxRequest.type == TrxRequest.Type.WITHDRAW_POINT) {
24             withdrawPoint(trxRequest.account, trxRequest.amount)
25         }
26     }
27 }
```

KrwAccountAdapter

UsdAccountAdapter

PointAccountAdapter

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
04             depositKrw(trxRequest.account, trxRequest.amount)
05         }
06
07         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
08             withdrawKrw(trxRequest.account, trxRequest.amount)
09         }
10
11         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
12             return
13         }
14
15         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
16             return
17         }
18
19         if (trxRequest.type == TrxRequest.Type.DEPOSIT_POINT) {
20             depositPoint(trxRequest.account, trxRequest.amount)
21         }
22
23         if (trxRequest.type == TrxRequest.Type.WITHDRAW_POINT) {
24             withdrawPoint(trxRequest.account, trxRequest.amount)
25         }
26     }
27 }

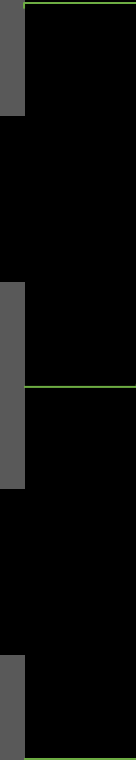
```

KrwAccountAdapter

UsdAccountAdapter

PointAccountAdapter

AccountAdapter

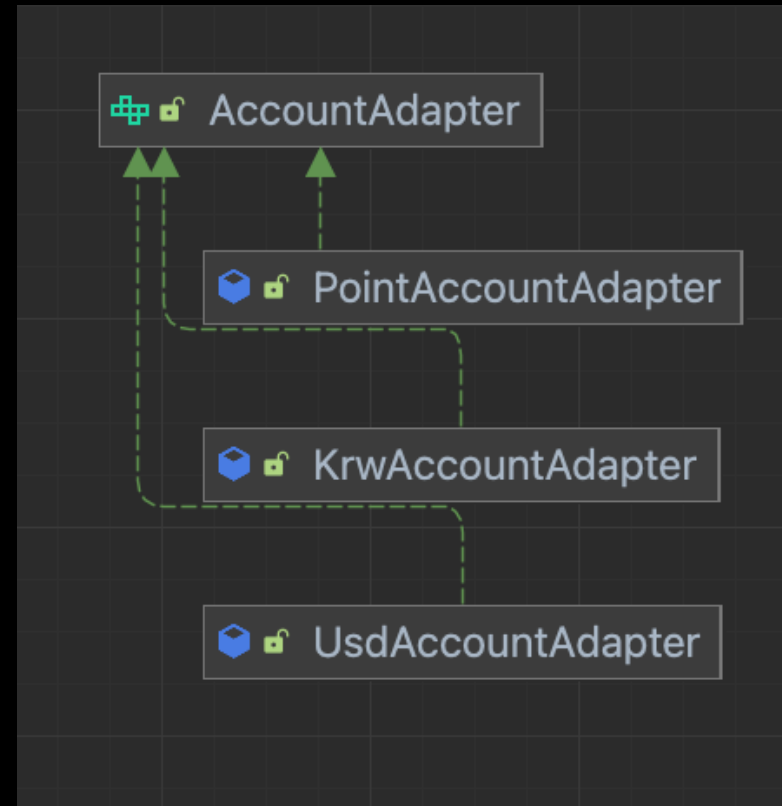


계좌기준으로 인터페이스 재정의

```
1 interface AccountAdapter {  
2     fun canProcessInAccepting(): Boolean  
3  
4     fun deposit(account: String, amount: BigDecimal)  
5  
6     fun withdraw(account: String, amount: BigDecimal)  
7 }
```

계좌기준으로 인터페이스 재정의

```
1 interface AccountAdapter {  
2     fun canProcessInAccepting(): Boolean  
3  
4     fun deposit(account: String, amount: BigDecimal)  
5  
6     fun withdraw(account: String, amount: BigDecimal)  
7 }
```

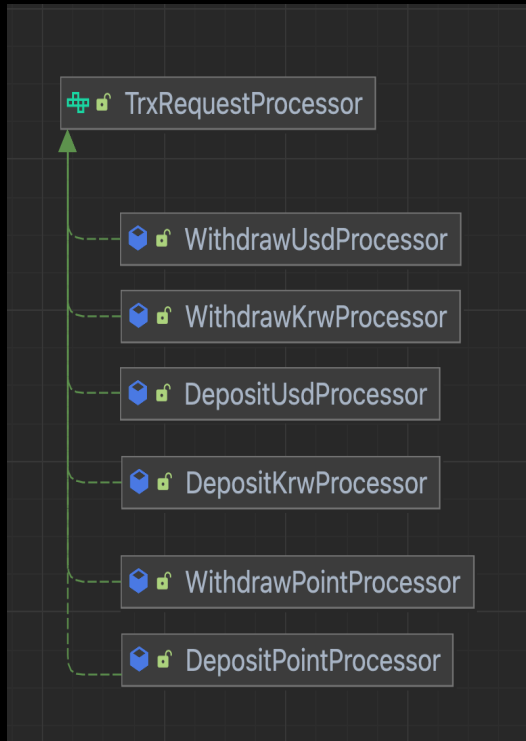


단일 함수 추상화와 계좌 추상화 인터페이스

```

1 interface TrxRequestProcessor {
2     fun process(account: String, amount: BigDecimal)
3 }

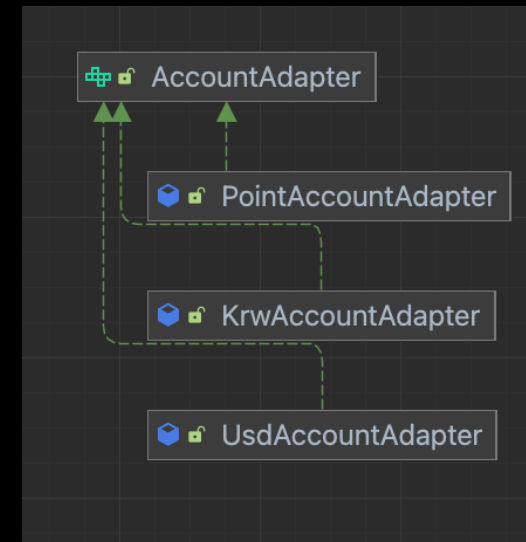
```



```

1 interface AccountAdapter {
2     fun canProcessInAccepting(): Boolean
3
4     fun deposit(account: String, amount: BigDecimal)
5
6     fun withdraw(account: String, amount: BigDecimal)
7 }

```



단일 함수 추상화와 계좌 추상화 구현

```
01 class TrxRequestService(  
02 ) {  
03  
04     fun accept(id: Long, processor: TrxRequestProcessor) {  
05         val trxRequest = getTrxRequest(id)  
06         trxRequest.accept()  
07  
08         if (trxRequest.type.isUsd()) {  
09             return  
10         }  
11  
12         processor.process(trxRequest.account, trxRequest.amount)  
13     }
```

```
01 class TrxRequestService(  
02 ) {  
03  
04     fun accept(id: Long, accountAdapter: AccountAdapter) {  
05         val trxRequest = getTrxRequest(id)  
06         trxRequest.accept()  
07  
08         if (!accountAdapter.canProcessInAccepting()) {  
09             return  
10         }  
11  
12         if (trxRequest.type.isDeposit()) {  
13             accountAdapter.deposit(trxRequest.account,  
14                                     trxRequest.amount)  
15         }  
16  
17         if (trxRequest.type.isWithdraw()) {  
18             accountAdapter.withdraw(trxRequest.account,  
19                                     trxRequest.amount)  
20         }  
21     }
```

더 함축된 이름과 더 구체적인 이름

```
01 class TrxRequestService(  
02 ) {  
03  
04     fun accept(id: Long, processor: TrxRequestProcessor) {  
05         val trxRequest = getTrxRequest(id)  
06         trxRequest.accept()  
07  
08         if (trxRequest.type.isUsd()) {  
09             return  
10         }  
11  
12         processor.process(trxRequest.account, trxRequest.amount)  
13     }
```

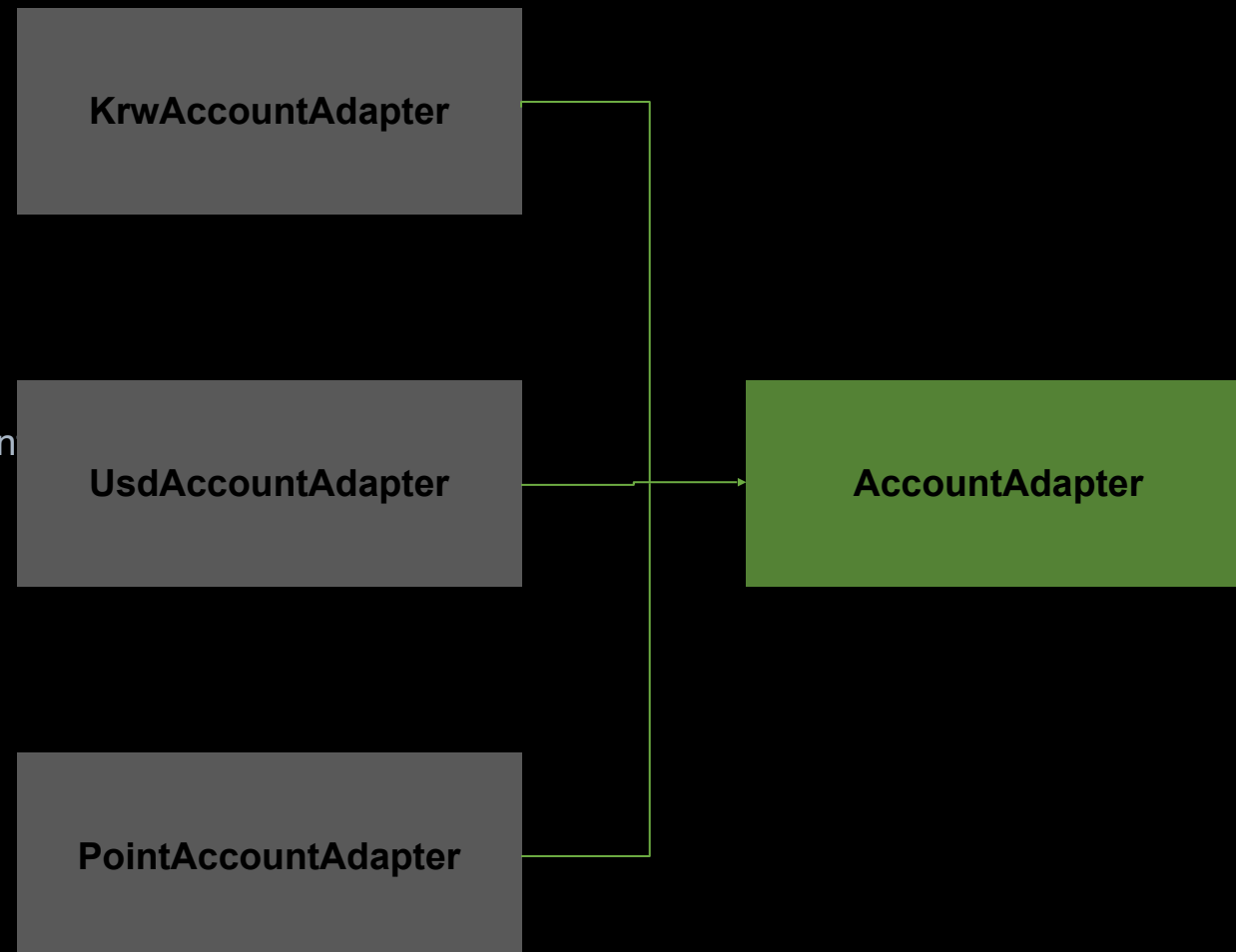
```
01 class TrxRequestService(  
02 ) {  
03  
04     fun accept(id: Long, accountAdapter: AccountAdapter) {  
05         val trxRequest = getTrxRequest(id)  
06         trxRequest.accept()  
07  
08         if (!accountAdapter.canProcessInAccepting()) {  
09             return  
10         }  
11  
12         if (trxRequest.type.isDeposit()) {  
13             accountAdapter.deposit(trxRequest.account,  
14                                     trxRequest.amount)  
15         }  
16  
17         if (trxRequest.type.isWithdraw()) {  
18             accountAdapter.withdraw(trxRequest.account,  
19                                     trxRequest.amount)  
20         }  
21     }
```

지금까지의 추상화들

	TrxTypeService	TrxRequestProcessor	AccountAdapter
범위	요청 생성, 승인, 처리	처리	처리
구현책임	create, accept, process	process	deposit, withdraw
구현체	6개	6개	3개
정보	?	요청 처리	입출금, 계좌

하지만 TrxRequestProcessor에서 AccountAdapter를 떠올리긴 쉽지 않을 수 있다

```
01 class TrxRequestService {  
02     fun accept(id: Long, processor: TrxRequestProcessor) {  
03         val trxRequest = getTrxRequest(id)  
04         trxRequest.accept()  
05  
06         if (trxRequest.type.isUsd()) {  
07             return  
08         }  
09  
10         processor.process(trxRequest.account, trxRequest.amount)  
11     }
```



과거의 코드는 그 위치에 있다는 사실만으로도 상당한 영향력을 가진다

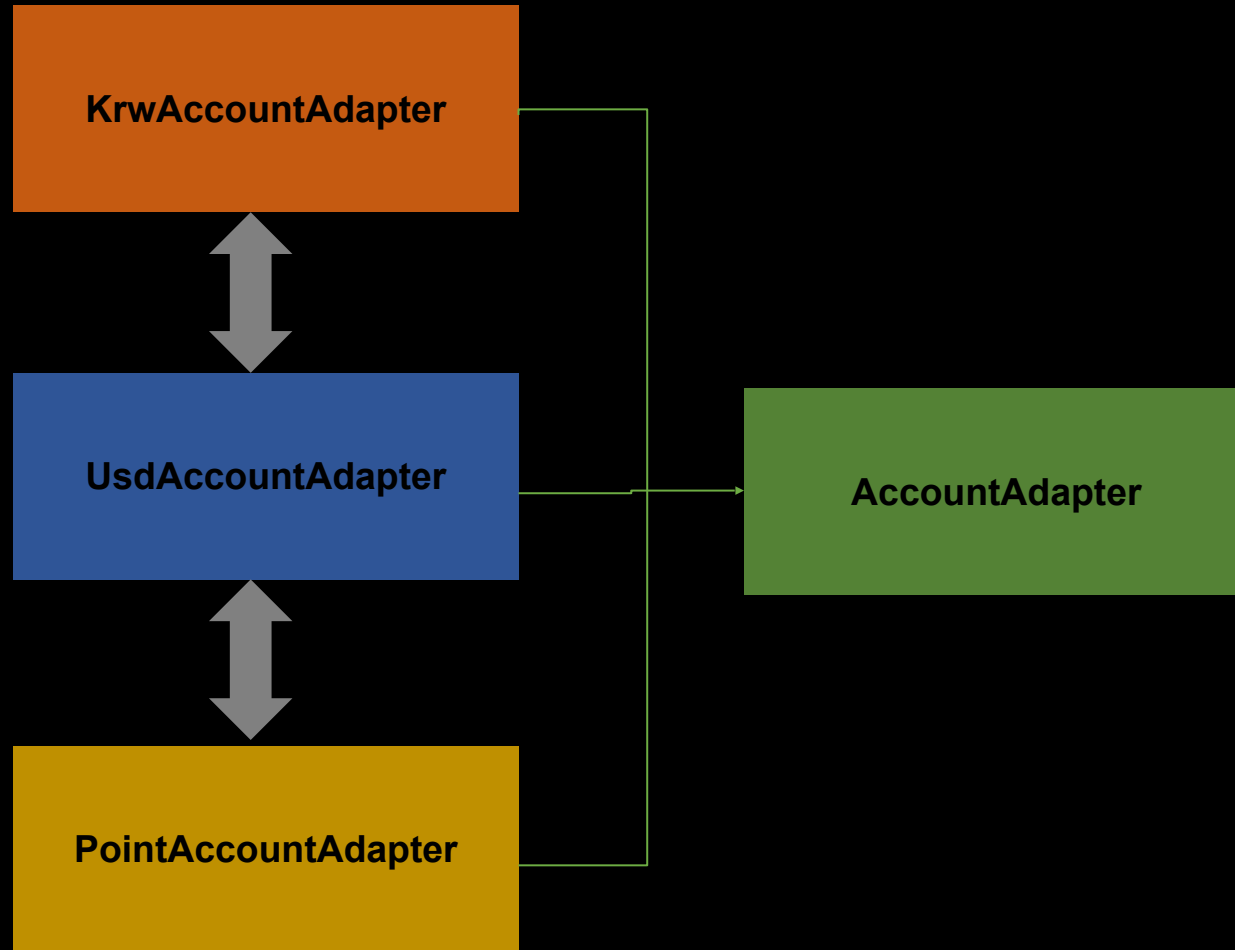
과거의 코드는 그 위치에 있다는 사실만으로도 상당한 영향력을 가진다
때론 잘 구어진 스테이크를 김치찌개에 넣는 용기가 필요하다.

때로는 날 것의 코드가 좋은 인사이트를 주기도 한다.

좋은 추상화는 더 많은 맥락 속에서 이루어짐

기능 개발을 하고 있는 개발자 vs 리팩토링을 하고 있는 개발자

인터페이스 추출시, 구현체들은 보이지 않은 의존성을 가짐



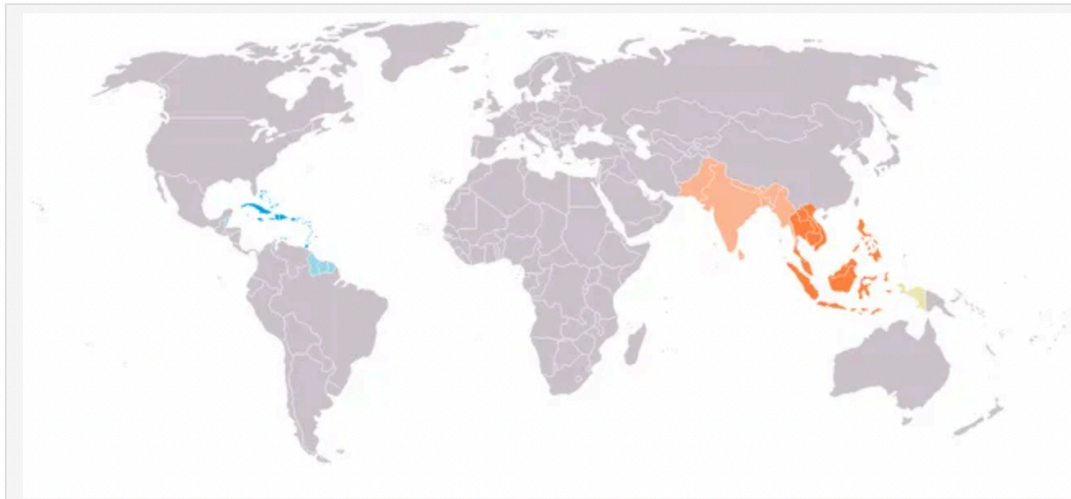
클래스 추출에서 멈추는 것도 방법

KrwAccountAdapter

UsdAccountAdapter

PointAccountAdapter

서인도 연방



주황: 동인도, 연한 주황: 인도(지명), 살구색: 서뉴기니
파랑: 서인도, 하늘색: 때때로 서인도에 포함되는 지역

- 영어: (the) West Indies, West India

사실 이것은 크리스토퍼 콜럼버스가 자신이 탐험한 아메리카 대륙 동부(카리브)를 인도라고 착각한 데에서 비롯된 명칭이다. 해당 지역이 인도가 아니라는 사실이 밝혀진 뒤에도 유럽인들은 관습적으로 카리브 일대를 서인도라고 불렀다.^[1]

따라서 서인도와 동인도 같은 용어는 제국주의적 용어이므로 현재는 역사적 사실을 거론할 때 빼놓고는 잘 사용하지 않는다. 그래도 카리브해 섬나라들을 묶어 분류할만한 용어가 이것 말고는 좀 애매하다보니 서인도란 말은 여전히 종종 쓰기도 하는데 동인도는 그냥 인도, 인도네시아라고 실제 지명으로 대체되어 거의 도태된 표현이라고 봐도 무방하다.

한국에서는 그냥 '서인도'라고 부르기보다 '서인도 제도(西印度諸島)'라고 부르는 경우가 많다.

설부른 확신은 서인도 연방과 같이 오해하기 쉬운 이름을 만들어낸다.

처음부터 인터페이스를 정의하기보다는

처음부터 인터페이스를 정의하기보다는
함수 분리부터 시작

처음부터 인터페이스를 정의하기보다는
함수 분리부터 시작
필요하다면 클래스를 통해 응집성 확보

적절한 시기를 찾는 것은 균형의 예술과도 같다.

적절한 시기를 찾는 것은 균형의 예술과도 같다.
하지만 많은 경우 개발 기능을 구현하는 초기 단계에서

적절한 시기를 찾는 것은 균형의 예술과도 같다.
하지만 많은 경우 개발 기능을 구현하는 초기 단계에서
좋은 추상화를 만들어 내는 것은 쉽지 않다.

4. 변화무쌍한 요구사항을 유연하게

끊임없는 요구사항

현실은 제한된 시간에서 이보다 훨씬 변화무쌍

끊임없는 요구사항

환차손익 계산을 위해 달러 입금시에는 환율정보를 기록해야 합니다.

환차손익: 환율변동으로 인한 손실 또는 이익

IF 문 구현

```
01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount)
18         }
19     }
20 }
```


IF 문 구현

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount)
18         }
19     }
20 }

```

```

01 class TrxRequestService {
02     fun accept(id: Long, exchangeRate: BigDecimal) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount, exchangeRate)
18         }
19     }
20 }

```

IF 문 구현

```

01 class TrxRequestService {
02     fun accept(id: Long) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount)
18         }
19     }
20 }

```

```

01 class TrxRequestService {
02     fun accept(id: Long, exchangeRate: BigDecimal) {
03         ...
04         if (trxRequest.type == TrxRequest.Type.DEPOSIT_KRW) {
05             depositKrw(trxRequest.account, trxRequest.amount)
06         }
07
08         if (trxRequest.type == TrxRequest.Type.WITHDRAW_KRW) {
09             withdrawKrw(trxRequest.account, trxRequest.amount)
10         }
11
12         if (trxRequest.type == TrxRequest.Type.DEPOSIT_USD) {
13             depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
14         }
15
16         if (trxRequest.type == TrxRequest.Type.WITHDRAW_USD) {
17             withdrawUsd(trxRequest.account, trxRequest.amount, exchangeRate)
18         }
19     }
20 }

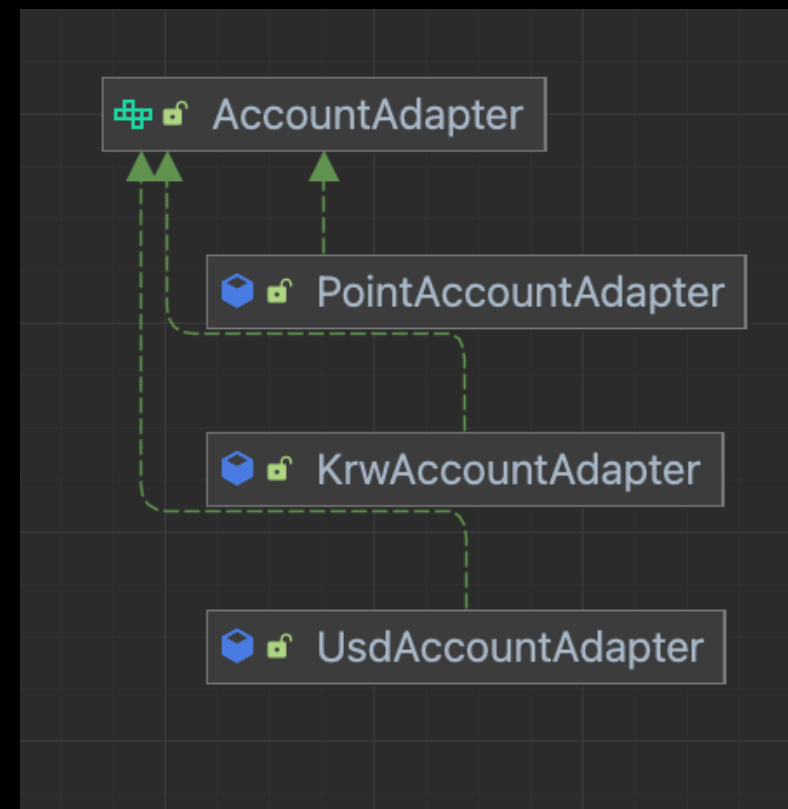
```

AccountAdapter

```
1 interface AccountAdapter {  
2     fun canProcessInAccepting(): Boolean  
3  
4     fun deposit(account: String, amount: BigDecimal, exchangeRate:  
BigDecimal?)  
5  
6     fun withdraw(account: String, amount: BigDecimal)  
7 }
```

달러입금을 위해 모든 구현체가 Nullable한 환율 파라미터가 추가 됨

```
1 interface AccountAdapter {  
2     fun canProcessInAccepting(): Boolean  
3  
4     fun deposit(account: String, amount: BigDecimal, exchangeRate:  
5         BigDecimal?)  
6     fun withdraw(account: String, amount: BigDecimal)  
7 }
```



인터페이스의 변경은 전체 구현체에 영향을 미친다.

파라미터도 추상화 할 것 인가?

함수가 일급객체인 언어에서는 함수를 통해 전체 구현체의 변경 없이
구현체 별 시그니처를 독립적으로 가져갈 수 있음

함수를 파라미터로

```
01 fun accept(id: Long, process: (trxRequest: TrxRequest) -> Unit) {  
02     val trxRequest = getTrxRequest(id)  
03     trxRequest.accept()  
04     process(trxRequest)  
05 }  
06  
07 fun process(id: Long, process: (trxRequest: TrxRequest) -> Unit) {  
08     val trxRequest = getTrxRequest(id)  
09     process(trxRequest)  
10 }
```


호출하는 쪽 코드는 이런 모습

```
01 fun acceptOuter(id: Long) {
02     accept(id) { trxRequest ->
03         when (trxRequest.type) {
04             TrxRequest.Type.DEPOSIT_KRW -> depositKrw(trxRequest.account, trxRequest.amount)
05             TrxRequest.Type.WITHDRAW_KRW -> withdrawKrw(trxRequest.account, trxRequest.amount)
06             TrxRequest.Type.DEPOSIT_USD -> {}
07             TrxRequest.Type.WITHDRAW_USD -> {}
08             TrxRequest.Type.DEPOSIT_POINT -> depositPoint(trxRequest.account, trxRequest.amount)
09             TrxRequest.Type.WITHDRAW_POINT -> withdrawPoint(trxRequest.account, trxRequest.amount)
10         }
11     }
12 }
13
14 fun processOuter(id: Long) {
15     val exchangeRate = getExchangeRate()
16     process(id) { trxRequest ->
17         when (trxRequest.type) {
18             TrxRequest.Type.DEPOSIT_KRW -> {}
19             TrxRequest.Type.WITHDRAW_KRW -> {}
20             TrxRequest.Type.DEPOSIT_USD -> depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
21             TrxRequest.Type.WITHDRAW_USD -> withdrawUsd(trxRequest.account, trxRequest.amount)
22             TrxRequest.Type.DEPOSIT_POINT -> {}
23             TrxRequest.Type.WITHDRAW_POINT -> {}
24         }
25     }
26 }
```

호출하는 쪽에서 분기

```
01 fun acceptOuter(id: Long) {
02     accept(id) { trxRequest ->
03         when (trxRequest.type) {
04             TrxRequest.Type.DEPOSIT_KRW -> depositKrw(trxRequest.account, trxRequest.amount)
05             TrxRequest.Type.WITHDRAW_KRW -> withdrawKrw(trxRequest.account, trxRequest.amount)
06             TrxRequest.Type.DEPOSIT_USD -> { }
07             TrxRequest.Type.WITHDRAW_USD -> { }
08             TrxRequest.Type.DEPOSIT_POINT -> depositPoint(trxRequest.account, trxRequest.amount)
09             TrxRequest.Type.WITHDRAW_POINT -> withdrawPoint(trxRequest.account, trxRequest.amount)
10         }
11     }
12 }
13
14 fun processOuter(id: Long) {
15     val exchangeRate = getExchangeRate()
16     process(id) { trxRequest ->
17         when (trxRequest.type) {
18             TrxRequest.Type.DEPOSIT_KRW -> { }
19             TrxRequest.Type.WITHDRAW_KRW -> { }
20             TrxRequest.Type.DEPOSIT_USD -> depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
21             TrxRequest.Type.WITHDRAW_USD -> withdrawUsd(trxRequest.account, trxRequest.amount)
22             TrxRequest.Type.DEPOSIT_POINT -> { }
23             TrxRequest.Type.WITHDRAW_POINT -> { }
24         }
25     }
26 }
```

호출하는 쪽에서 분기

```
01 fun acceptOuter(id: Long) {
02     accept(id) { trxRequest ->
03         when (trxRequest.type) {
04             TrxRequest.Type.DEPOSIT_KRW -> depositKrw(trxRequest.account, trxRequest.amount)
05             TrxRequest.Type.WITHDRAW_KRW -> withdrawKrw(trxRequest.account, trxRequest.amount)
06             TrxRequest.Type.DEPOSIT_USD -> { }
07             TrxRequest.Type.WITHDRAW_USD -> { }
08             TrxRequest.Type.DEPOSIT_POINT -> depositPoint(trxRequest.account, trxRequest.amount)
09             TrxRequest.Type.WITHDRAW_POINT -> withdrawPoint(trxRequest.account, trxRequest.amount)
10         }
11     }
12 }
13
14 fun processOuter(id: Long) {
15     val exchangeRate = getExchangeRate()
16     process(id) { trxRequest ->
17         when (trxRequest.type) {
18             TrxRequest.Type.DEPOSIT_KRW -> { }
19             TrxRequest.Type.WITHDRAW_KRW -> { }
20             TrxRequest.Type.DEPOSIT_USD -> depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
21             TrxRequest.Type.WITHDRAW_USD -> withdrawUsd(trxRequest.account, trxRequest.amount)
22             TrxRequest.Type.DEPOSIT_POINT -> { }
23             TrxRequest.Type.WITHDRAW_POINT -> { }
24         }
25     }
26 }
```

Caller에서 더 많은 정보를 유연하게 넘길 수 있음

```
01 fun acceptOuter(id: Long) {
02     accept(id) { trxRequest ->
03         when (trxRequest.type) {
04             TrxRequest.Type.DEPOSIT_KRW -> depositKrw(trxRequest.account, trxRequest.amount)
05             TrxRequest.Type.WITHDRAW_KRW -> withdrawKrw(trxRequest.account, trxRequest.amount)
06             TrxRequest.Type.DEPOSIT_USD -> { }
07             TrxRequest.Type.WITHDRAW_USD -> { }
08             TrxRequest.Type.DEPOSIT_POINT -> depositPoint(trxRequest.account, trxRequest.amount)
09             TrxRequest.Type.WITHDRAW_POINT -> withdrawPoint(trxRequest.account, trxRequest.amount)
10         }
11     }
12 }
13
14 fun processOuter(id: Long) {
15     val exchangeRate = getExchangeRate()
16     process(id) { trxRequest ->
17         when (trxRequest.type) {
18             TrxRequest.Type.DEPOSIT_KRW -> { }
19             TrxRequest.Type.WITHDRAW_KRW -> { }
20             TrxRequest.Type.DEPOSIT_USD -> depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
21             TrxRequest.Type.WITHDRAW_USD -> withdrawUsd(trxRequest.account, trxRequest.amount)
22             TrxRequest.Type.DEPOSIT_POINT -> { }
23             TrxRequest.Type.WITHDRAW_POINT -> { }
24         }
25     }
26 }
```

분기에 해당되는 구현체들의 시그니처는 통일되지 않아도 됨

```
01 fun acceptOuter(id: Long) {
02     accept(id) { trxRequest ->
03         when (trxRequest.type) {
04             TrxRequest.Type.DEPOSIT_KRW -> depositKrw(trxRequest.account, trxRequest.amount)
05             TrxRequest.Type.WITHDRAW_KRW -> withdrawKrw(trxRequest.account, trxRequest.amount)
06             TrxRequest.Type.DEPOSIT_USD -> { }
07             TrxRequest.Type.WITHDRAW_USD -> { }
08             TrxRequest.Type.DEPOSIT_POINT -> depositPoint(trxRequest.account, trxRequest.amount)
09             TrxRequest.Type.WITHDRAW_POINT -> withdrawPoint(trxRequest.account, trxRequest.amount)
10         }
11     }
12 }
13
14 fun processOuter(id: Long) {
15     val exchangeRate = getExchangeRate()
16     process(id) { trxRequest ->
17         when (trxRequest.type) {
18             TrxRequest.Type.DEPOSIT_KRW -> { }
19             TrxRequest.Type.WITHDRAW_KRW -> { }
20             TrxRequest.Type.DEPOSIT_USD -> depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
21             TrxRequest.Type.WITHDRAW_USD -> withdrawUsd(trxRequest.account, trxRequest.amount)
22             TrxRequest.Type.DEPOSIT_POINT -> { }
23             TrxRequest.Type.WITHDRAW_POINT -> { }
24         }
25     }
26 }
```

분기를 선언적으로 표현

```
01 fun acceptOuter(id: Long) {
02     accept(id) { trxRequest ->
03         when (trxRequest.type) {
04             TrxRequest.Type.DEPOSIT_KRW -> depositKrw(trxRequest.account, trxRequest.amount)
05             TrxRequest.Type.WITHDRAW_KRW -> withdrawKrw(trxRequest.account, trxRequest.amount)
06             TrxRequest.Type.DEPOSIT_USD -> {}
07             TrxRequest.Type.WITHDRAW_USD -> {}
08             TrxRequest.Type.DEPOSIT_POINT -> depositPoint(trxRequest.account, trxRequest.amount)
09             TrxRequest.Type.WITHDRAW_POINT -> withdrawPoint(trxRequest.account, trxRequest.amount)
10         }
11     }
12 }
13
14 fun processOuter(id: Long) {
15     val exchangeRate = getExchangeRate()
16     process(id) { trxRequest ->
17         when (trxRequest.type) {
18             TrxRequest.Type.DEPOSIT_KRW -> {}
19             TrxRequest.Type.WITHDRAW_KRW -> {}
20             TrxRequest.Type.DEPOSIT_USD -> depositUsd(trxRequest.account, trxRequest.amount, exchangeRate)
21             TrxRequest.Type.WITHDRAW_USD -> withdrawUsd(trxRequest.account, trxRequest.amount)
22             TrxRequest.Type.DEPOSIT_POINT -> {}
23             TrxRequest.Type.WITHDRAW_POINT -> {}
24         }
25     }
26 }
```

AccountAdapter에서의 분기


```
01 class TrxRequestService(  
02 ) {  
  
03  
04     fun accept(id: Long, accountAdapter: AccountAdapter) {  
05         val trxRequest = getTrxRequest(id)  
06         trxRequest.accept()  
07  
08         if (!accountAdapter.canProcessInAccepting()) {  
09             return  
10         }  
11  
12         if (trxRequest.type.isDeposit()) {  
13             accountAdapter.deposit(trxRequest.account,  
trxRequest.amount)  
14         }  
15  
16         if (trxRequest.type.isWithdraw()) {  
17             accountAdapter.withdraw(trxRequest.account,  
trxRequest.amount)  
18         }  
19     }  
20 }
```

canProcessInAccepting 분기가 사라짐

```
01 class TrxRequestService(  
02 ) {  
  
03  
04     fun accept(id: Long, accountAdapter: AccountAdapter) {  
05         val trxRequest = getTrxRequest(id)  
06         trxRequest.accept()  
07  
08         if (!accountAdapter.canProcessInAccepting()) {  
09             return  
10         }  
11  
12         if (trxRequest.type.isDeposit()) {  
13             accountAdapter.deposit(trxRequest.account,  
trxRequest.amount)  
14         }  
15  
16         if (trxRequest.type.isWithdraw()) {  
17             accountAdapter.withdraw(trxRequest.account,  
trxRequest.amount)  
18         }  
19     }  
20 }
```

```
01 fun acceptOuter(id: Long) {  
02     accept(id) { trxRequest ->  
03         when (trxRequest.type) {  
06             TrxRequest.Type.DEPOSIT_USD -> {}  
07             TrxRequest.Type.WITHDRAW_USD -> {}  
11         }  
12     }
```


하지만 이것 역시 완벽하지는 않음

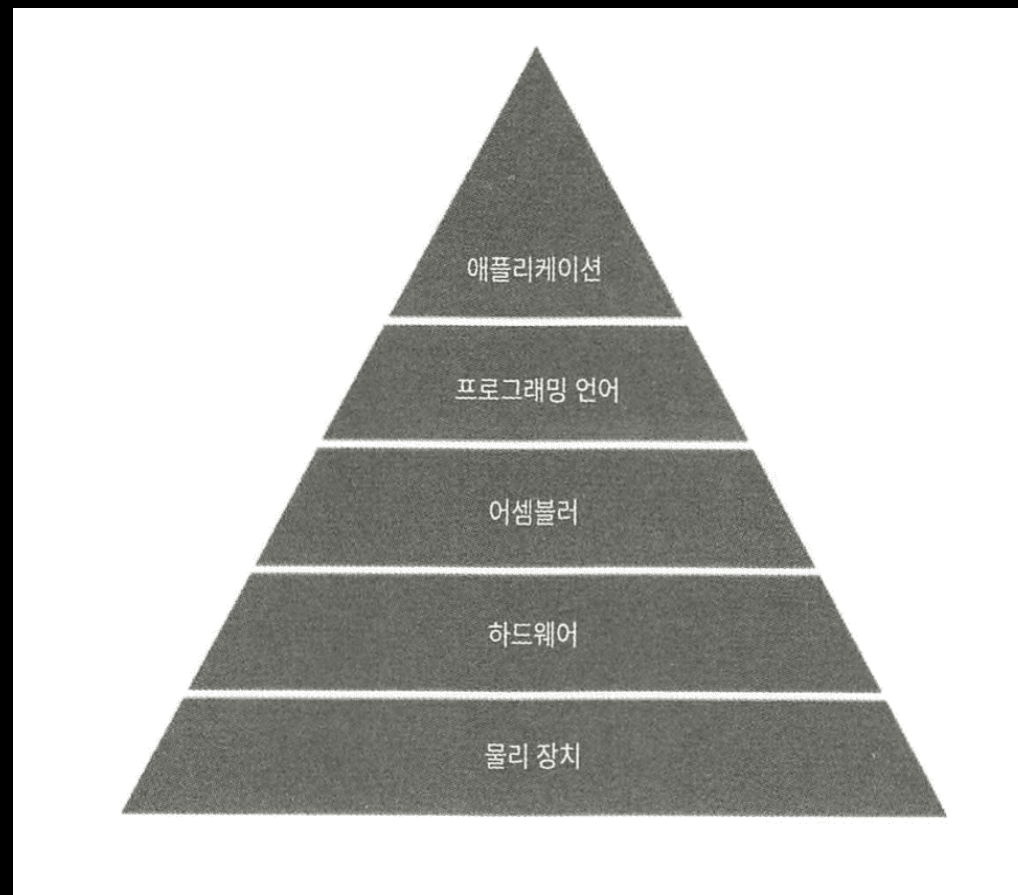


```
1 function hell(win) {
2   // for listener purpose
3   return function() {
4     loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
5       loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
6         loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
7           loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
8             loadLink(win, REMOTE_SRC+'/lib/underscore.min.js', function() {
9               loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
10                loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
11                  loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
12                    loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
13                      async.eachSeries(SCRIPTS, function(src, callback) {
14                        loadScript(win, BASE_URL+src, callback);
15                      });
16                    });
17                  });
18                });
19              });
20            });
21          });
22        });
23      });
24    });
25  };
26 }
```

그럼에도 추상화가 주는 혜택은 상당하다.

좋은 추상화는 더 복잡한 문제를 풀 수 있게 해준다.

그 덕에 우리는 아주 높은 추상화 계층에 살고 있다



하지만 추상화에는 많은 고민들이 필요함

오늘 이야기 3줄 요약

1. 추상화는 비용이 있고 잘못된 추상화는 혜택없이 비용만 감당해야하기도 함
2. 추상화는 적절한 방향, 적절한 범위, 적절한 시기를 고민해야 함.
3. 함수를 추출하는 것부터 시작해서 점진적으로 추상화

토스증권에서 사람 구하고 있어요

Server Developer

토스

토스페이먼츠

토스증권

토스뱅크

토스플레이스

Platform

Product

차세대