아이템 64.

객체는 인터페이스를 사용해 참조하라

후니

이 아이템의 핵심은 하나.

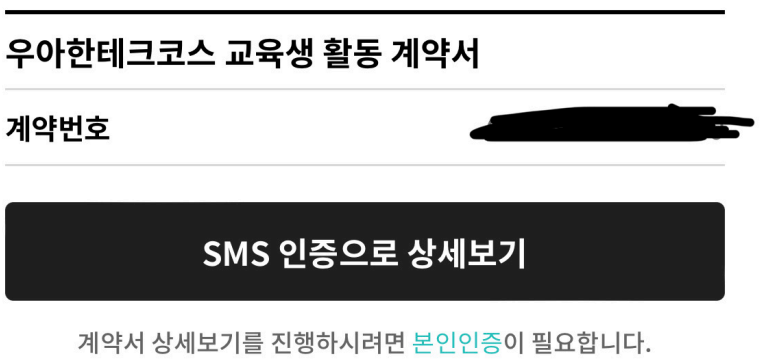
적합한 인터페이스만 있다면 전부 인터페이스 타입으로 선언하라.

But Why?

프로그램이 유연해 진다는데…

우형에 재직 중인 초보 개발자 후니는 아래와 같은 요구사항을 받았습니다.

1. 계약서를 확인할 때 SMS 인증을 보내주세요.
2. 대량 SMS 발송 회사의 api를 써주세요.



```java
public static void main(String[] args) {
    User huni = new User( name: "huni",  contractNumber: "12312412sd",  phoneNumber: "01023456789");
    SmsApi.authenticateNumber(huni);
    SmsApi.authenticate(huni,  number: "12345");
}
```

후니는 신나게 코드를 작성했습니다.

아차차! Kakao 인증도 추가해줘

우아한 형제들
계약서가 도착했습니다.

우아한테크코스 교육생 활동 계약서

계약번호 [redacted]

카카오 인증으로 상세보기

SMS 인증으로 상세보기

계약서 상세보기를 진행하시려면 본인인증이 필요합니다.

```java
public static void main(String[] args) {
    User huni = new User( name: "huni",  contractNumber: "12312412sd",  phoneNumber: "01023456789");

    // .. 인증관련 검증.....

    AuthWay authWay = AuthWay.KAKAO;
    if (authWay == AuthWay.KAKAO) {
        KakaoApi.authenticateNumber(huni);
        KakaoApi.authenticate(huni,  number: "12345");
    } else {
        SmsApi.authenticateNumber(huni);
        SmsApi.authenticate(huni,  number: "12345");
    }
}
```

후니는 점점 열이 받습니다.

인증 보내기는 똑같은데 방법이 달라진다는 이유로
프로덕션 코드가 변경됩니다.

요즘 PASS도 좋던데^^ 추가해줘~

그리고 "테스트 코드"도 짜줘^^

우아한 형제들
계약서가 도착했습니다.

우아한테크코스 교육생 활동 계약서

계약번호 [redacted]

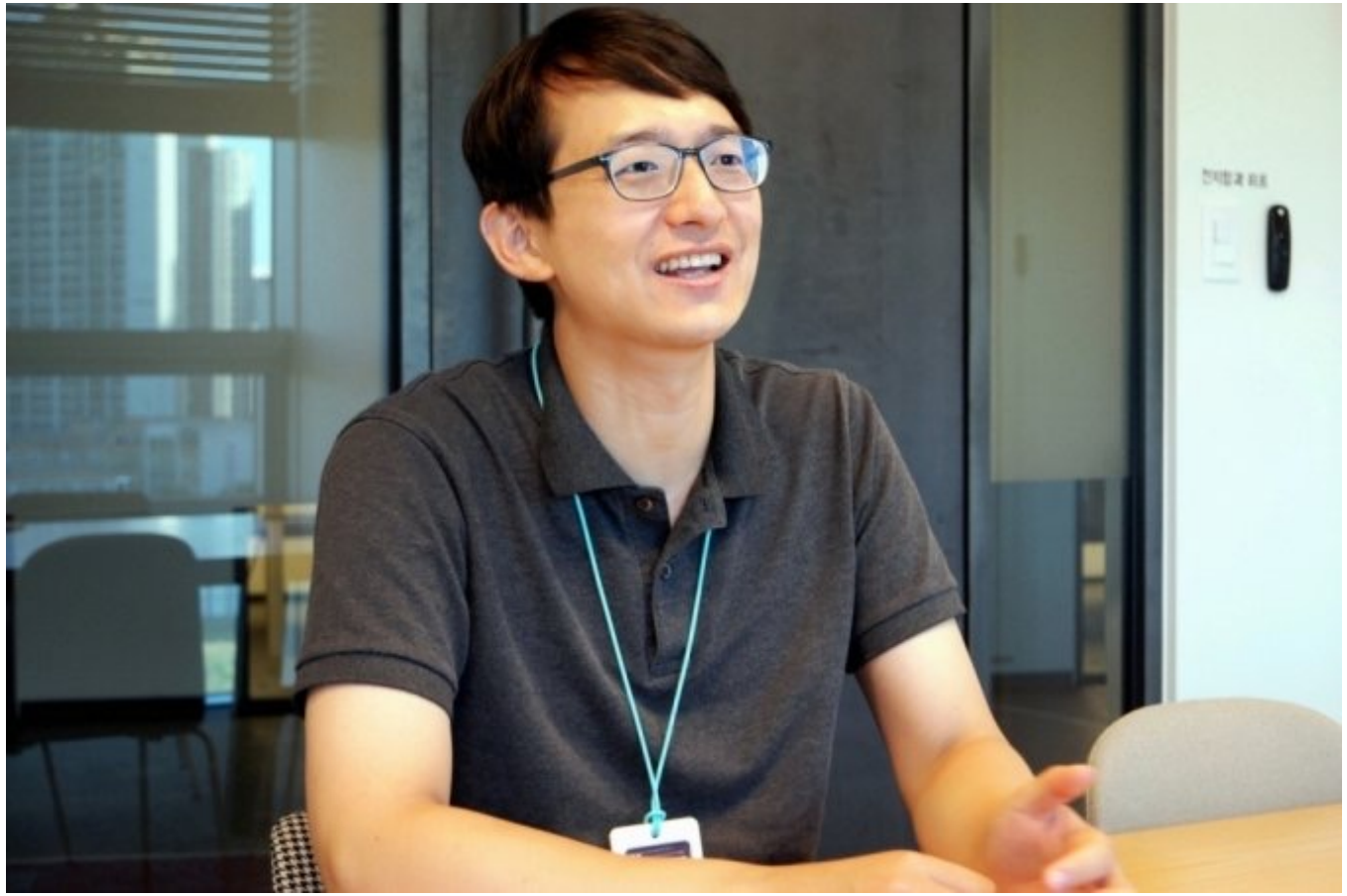
PASS 인증으로 상세보기

카카오 인증으로 상세보기

SMS 인증으로 상세보기

계약서 상세보기를 진행하시려면 본인인증이 필요합니다.

초보 개발자 후니는 망했습니다.

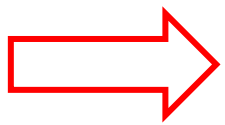


초고수 개발자 : 그럴 땐 "인터페이스"를 써라^^

분리 해보기

고수준

인증한다.



저수준

Sms로 인증 해라

Kakao로 인증 해라

Pass로 인증 해라

⋮

인증해라.

```
public interface Notifier {

    public AuthResult authenticate(final User user, final String number);

    public String sendAuthenticateNumber(final User user);
}
```

Kakao 인증해라.

```
public class KaKaoNotifier implements Notifier
```

Pass 인증해라.

```
public class PassNotifier implements Notifier
```

SMS 인증해라.

```
public class SmsNotifier implements Notifier
```

```java
public class ContractService {

    private final Notifier notifier;

    public ContractService(Notifier notifier) {
        this.notifier = notifier;
    }

    public AuthResult authenticate(final User user, final String number) {
        return notifier.authenticate(user, number);
    }

    public String sendAuthenticateNumber(final User user) {
        return notifier.sendAuthenticateNumber(user);
    }
}
```

이제 인터페이스를 참조하여 코드를 작성해보겠습니다!

1. 저수준 코드가 변경되어도 프로덕션 코드는 변경되지 않는다.

인터페이스 사용 전

```
if (authWay == AuthWay.KAKAO) {
    KakaoApi.authenticateNumber(huni);
    KakaoApi.authenticate(huni, number: "12345");

} else if (authWay == AuthWay.SMS) {
    SmsApi.authenticateNumber(huni);
    SmsApi.authenticate(huni, number: "12345");

} else {
    PassApi.authenticateNumber(huni);
    PassApi.authenticate(huni, number: "12345");
}
```

인터페이스 사용 후

```
public static void main(String[] args) {
    User huni = new User( name: "huni", contractNumber: "12312412sd", phoneNumber: "01023456789");

    // .. 인증관련 검증.....

    // 사용자가 클릭한 버튼
    AuthWay authWay = AuthWay.KAKAO;

    ContractService contractService = new ContractService(authWay.getNotifier());
    contractService.sendAuthenticateNumber(huni);
    contractService.authenticate(huni, number: "12345");

}
```

2. 원하는 구현체를 조립할 수 있다.

사용자가 SMS 클릭

```
AuthWay authWay = AuthWay.SMS; // 사용자가 클릭한 버튼

ContractService contractService = new ContractService(authWay.getNotifier());
contractService.sendAuthenticateNumber(huni);
contractService.authenticate(huni, number: "12345");
```

```
01023456789 에 인증번호 : 12345를 발송했습니다.
인증성공!
```

사용자가 Kakao 클릭

```
AuthWay authWay = AuthWay.KAKAO; // 사용자가 클릭한 버튼

ContractService contractService = new ContractService(authWay.getNotifier());
contractService.sendAuthenticateNumber(huni);
contractService.authenticate(huni, number: "12345");
```

```
01023456789 에 카카오 인증번호 : 12345를 발송했습니다.
카카오 인증성공!
```

사용자가 Pass 클릭

```
AuthWay authWay = AuthWay.PASS; // 사용자가 클릭한 버튼

ContractService contractService = new ContractService(authWay.getNotifier());
contractService.sendAuthenticateNumber(huni);
contractService.authenticate(huni, number: "12345");
```

```
01023456789 에 pass 인증번호 : 12345를 발송했습니다.
pass 인증성공!
```

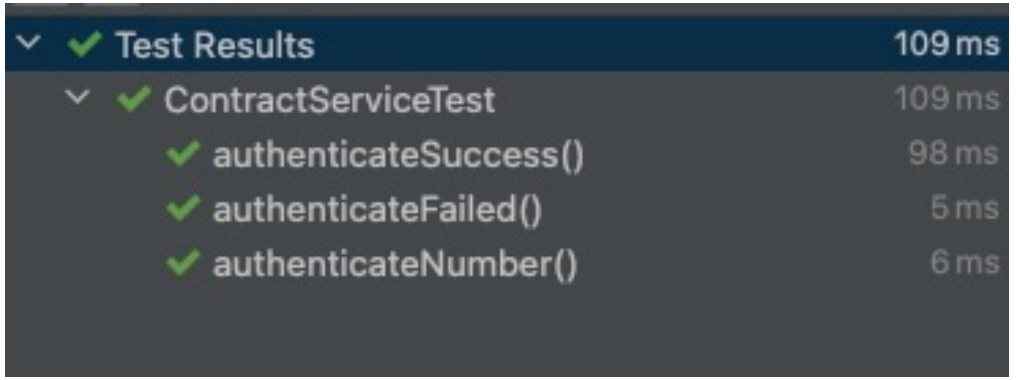
3. 테스팅이 가능하다.

```
private ContractService contractService = new ContractService(new StubNotifier());

@Test
void authenticateNumber() {
    assertThat(contractService.sendAuthenticateNumber(new User( name: "huni",
            contractNumber: "12312412sd",
            phoneNumber: "01023456789")))
            .isEqualTo("12345");
}

@Test
void authenticateSuccess() {
    assertThat(contractService.authenticate(new User( name: "huni",
            contractNumber: "12312412sd",
            phoneNumber: "01023456789"), number: "12345"))
            .isEqualTo(AuthResult.SUCCESS);
}

@Test
void authenticateFailed() {
    assertThat(contractService.authenticate(new User( name: "huni",
            contractNumber: "12312412sd",
            phoneNumber: "01023456789"), number: "123456"))
            .isEqualTo(AuthResult.FAIL);
}
```



무조건 인터페이스를 사용하라는 말은 아닙니다!

1. 실제 추상화가 필요한 상황에서 써라

* 존재하지 않는 기능에 대한 추상화 ㄴㄴ
* 실제 변경, 확장 할 때만 사용 ㅇㅇ

2. 적합한 인터페이스가 없을 때는 구현체 써라

* 값 클래스가 상속 쓰는 경우 거의 없음 (String, BigInteger)
* 인터페이스와 구현이 다를 때 (PriorityQueue)

끝