우테캠프로 - 3주차 ATDD

순서

- 3주차 소개
- ATDD Intro
- 인수 테스트
- 인수 테스트 만들기

강사 소개

류성현

현 우아한형제들 우아한테크코스 코치

전 네이버 카페

전 한전 KDN 인사시스템 운영/개발

하는일 - 교육자

교육 운영

커리큘럼 개발 및 강의

교육생 멘토링

행사 기획

기타 잡일

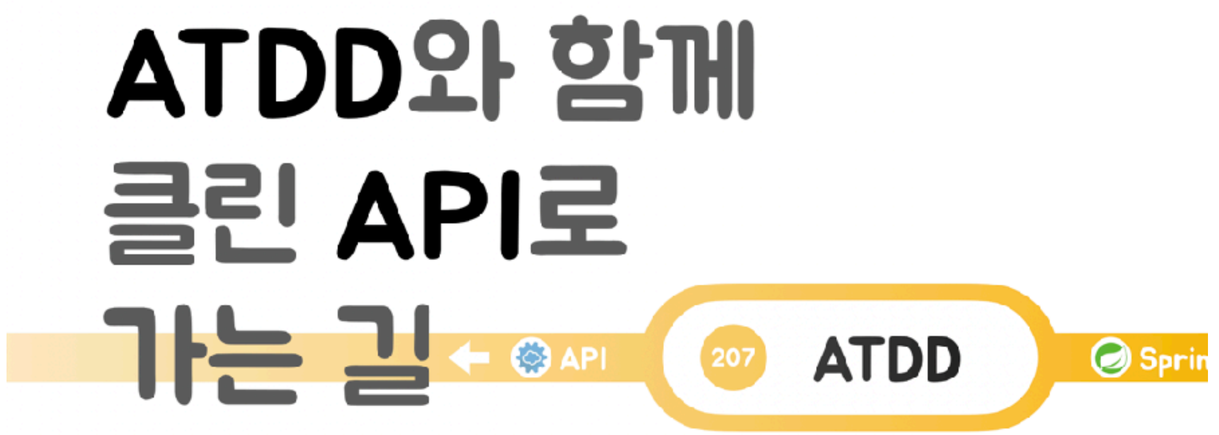
우아한테크코스

하는일 - 강사

ATDD와 함께 클린 API로 가는 길 과정 운영

학습 테스트 기반 스프링 과정 운영

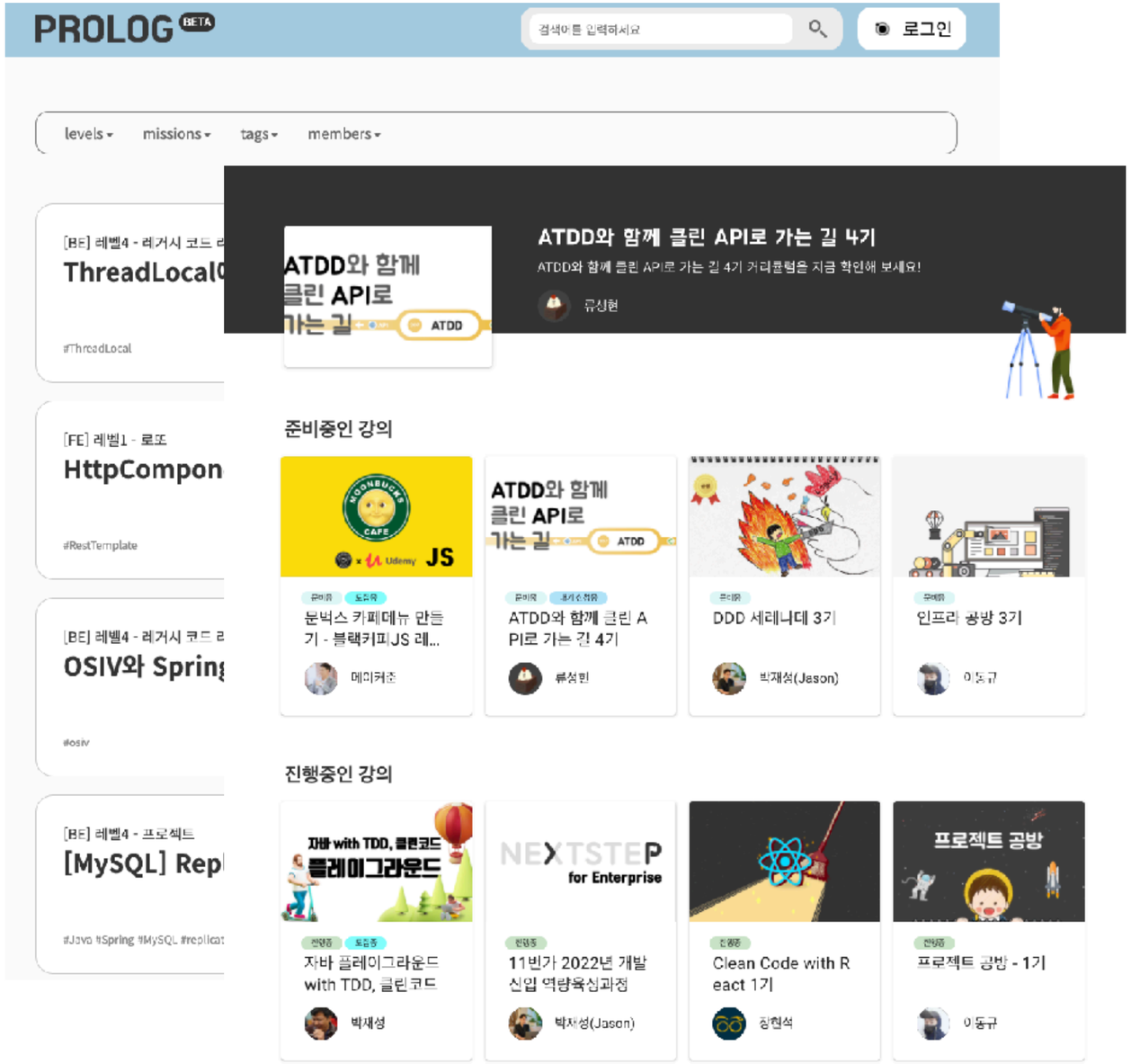
(가칭)인증/인가 with Spring Boot 준비중



하는일 - 개발자

NEXTSTEP 교육 플랫폼 개발

교육 관련 서비스 개발



3주차 소개

목적

- 웹 애플리케이션을 인수 테스트 주도 개발 방법으로 개발하는 경험하기

강의와 미션 소개

- 강의
  - ATDD가 무엇인지
  - 인수 테스트
  - 인수 테스트 도구
- 미션
  - 인수 테스트 만들기
  - 제공된 인수 조건 기반 인수 테스트 주도 개발
  - 인수 테스트 주도 개발

3주차가 끝나면 나는?

- ATDD가 무엇이고 왜 하는지 이해할 수 있다
- 인수 테스트를 작성할 수 있고 필요한 도구를 학습했다
- ATDD 개발 사이클을 이해하고 인수 테스트 주도 개발을 경험했다.

기능 구현에 목숨걸지 말기!

ATDD Intro

ATDD 이야기를 하기 전에

오늘 강의에서 이야기 하고 싶은 것

- 인수 테스트가 무엇인지
- 인수 테스트 주도 개발이 무엇이고 왜 하는지

ATDD는 모든 문제를 해결할 수 없다

필요한 상황에 맞춰 사용하는 도구

개발하다 길을 잃은적 있나요?

요구사항을 잘못 이해한적 있나요?

올바른 요구사항을 가리키는 등대가 있었다면...

ATDD를 통해
효율적인 개발 프로세스를
경험하세요

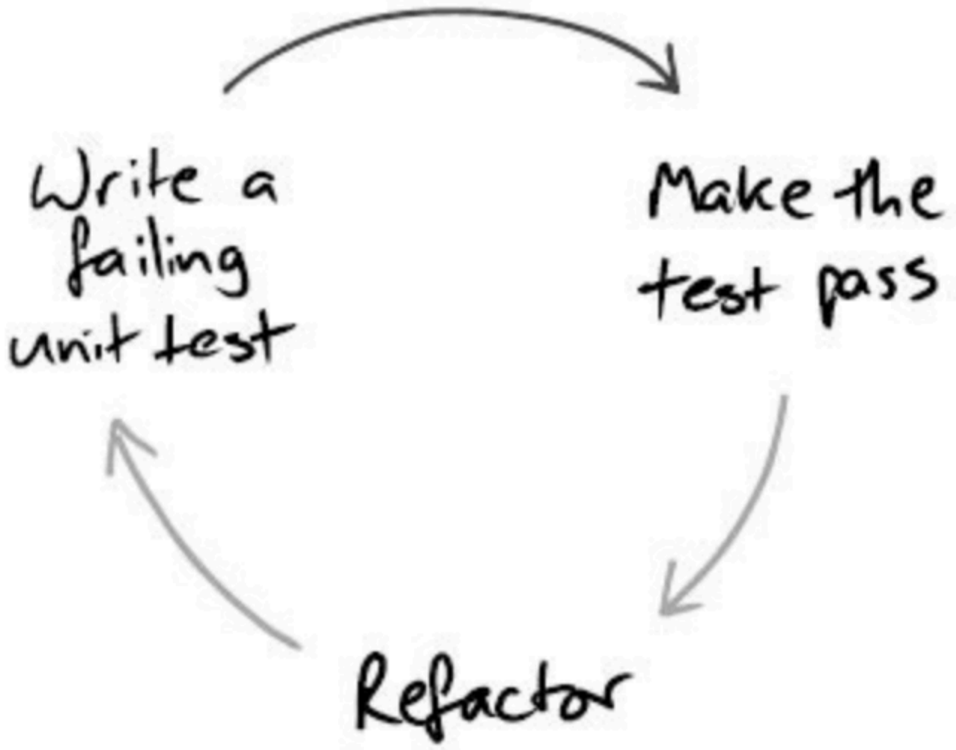
ATDD?

ATDD?

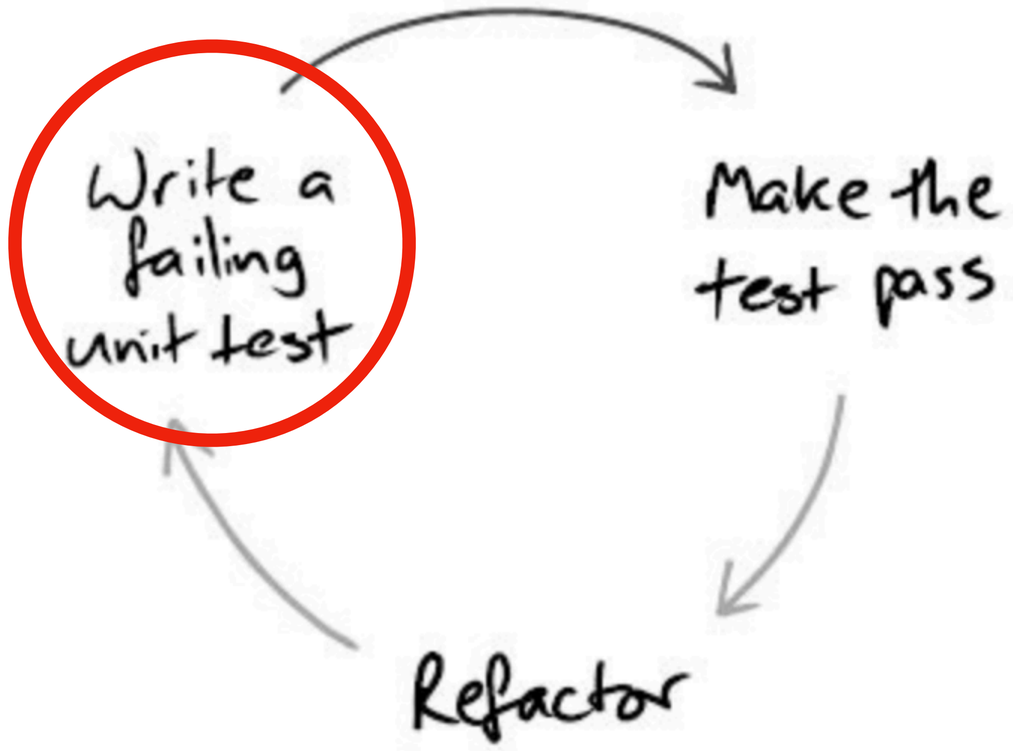
테스트가 가능한 요구사항으로

소프트웨어를 개발하는 프로세스

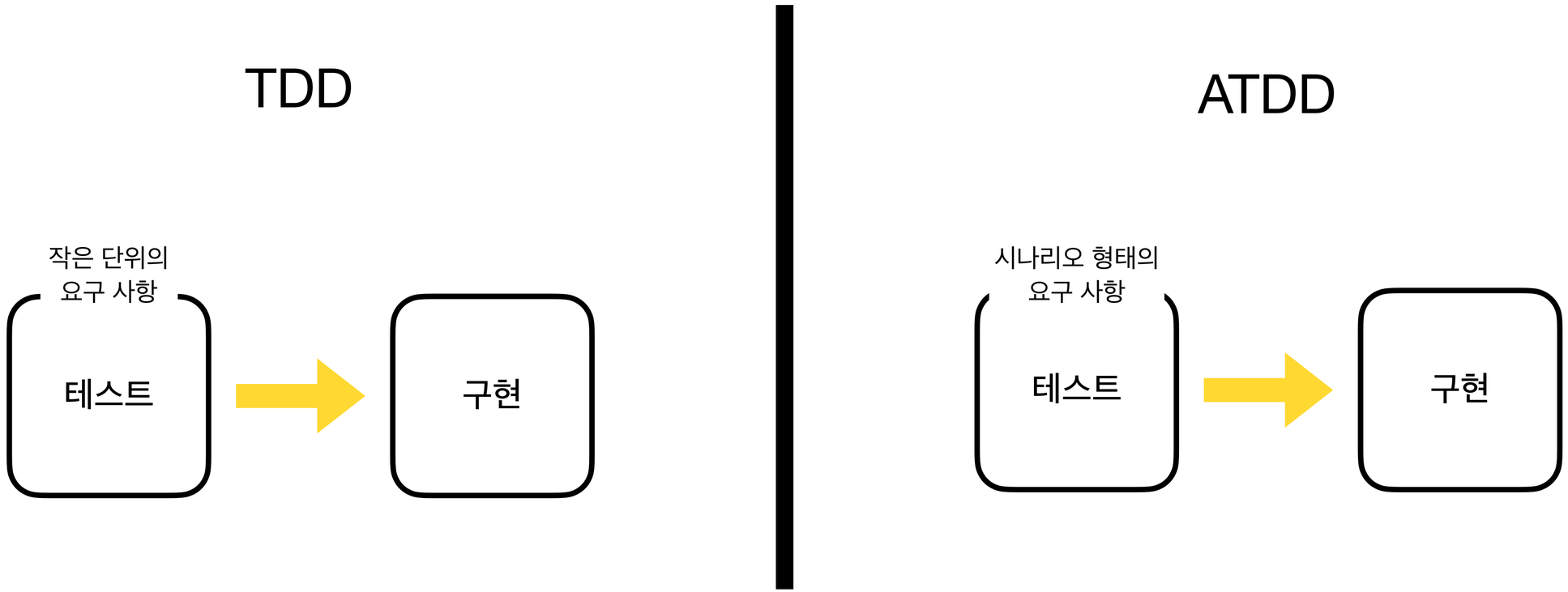
TDD



TDD



TDD vs ATDD



테스트가 가능한 요구사항으로

소프트웨어를 개발하는 프로세스

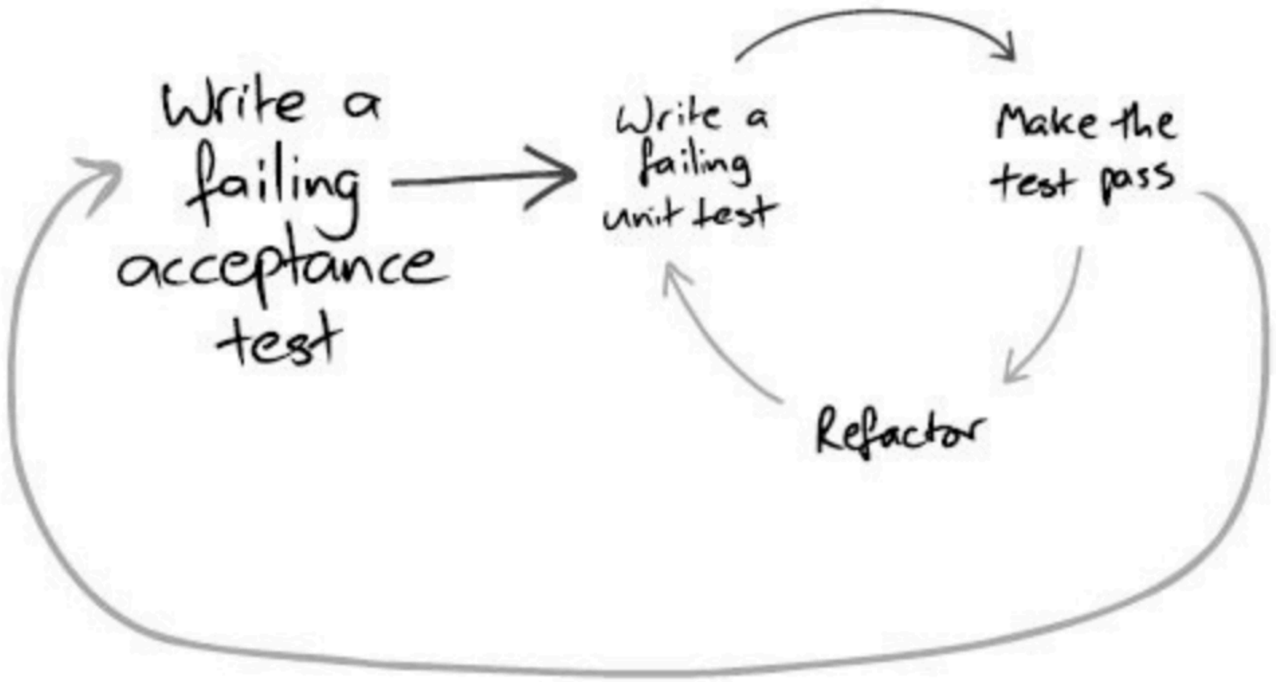
요구사항을 검증하는 테스트로

소프트웨어를 개발하는 프로세스

인수 테스트로

소프트웨어를 개발하는 프로세스

ATDD + TDD Cycle



ATDD를 하는 이유

생산성 증가

구현전에 인수 테스트를 수행하는 경우 팀의 생산성이 두 배가 되는 것을 확인했다. - 제프 서덜런드(스크럼 공동 제작자)

작업의 명확한 시작과 끝을 제시

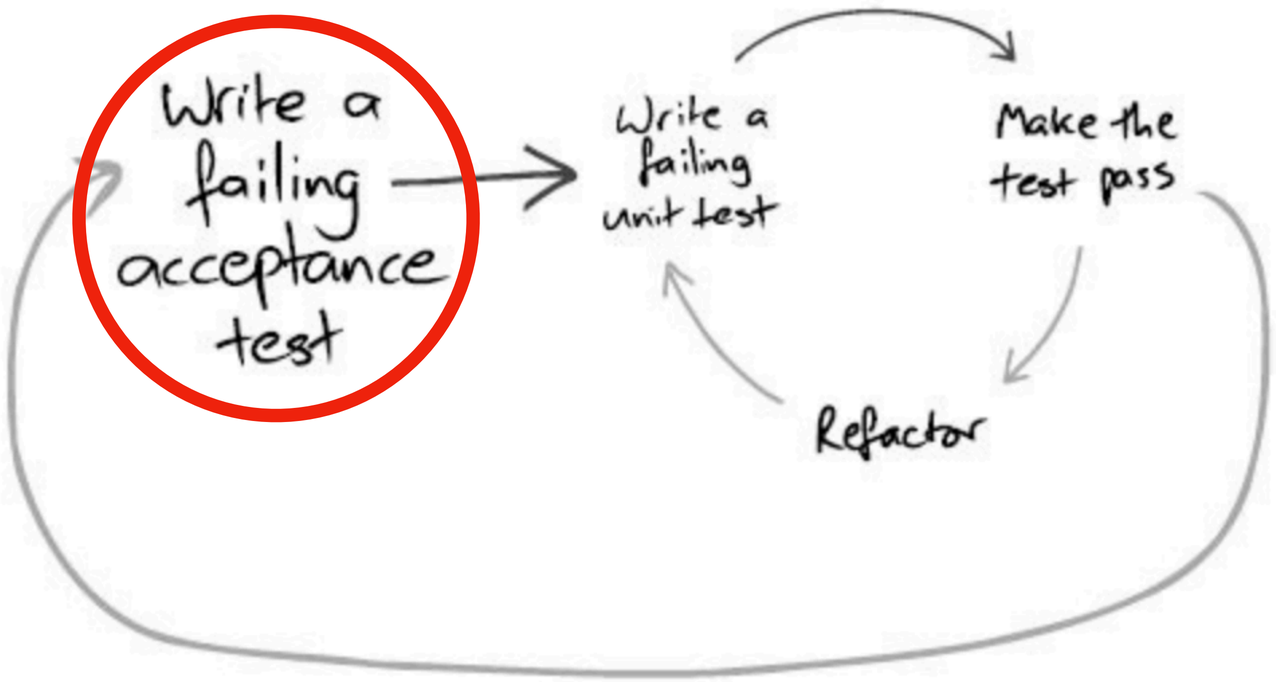
빠른 피드백

귀찮은 작업을 프로세스로 강제

인수 테스트

인수 테스트란?

인수 테스트?


Write a failing acceptance test
Write a failing unit test
Make the test pass
Refactor

Acceptance Test in extreme programming

- **사용자 스토리**를 검증하는 기능 테스트
- 사용자 스토리로 **테스트할 시나리오**를 지정

Acceptance testing in extreme programming [edit]

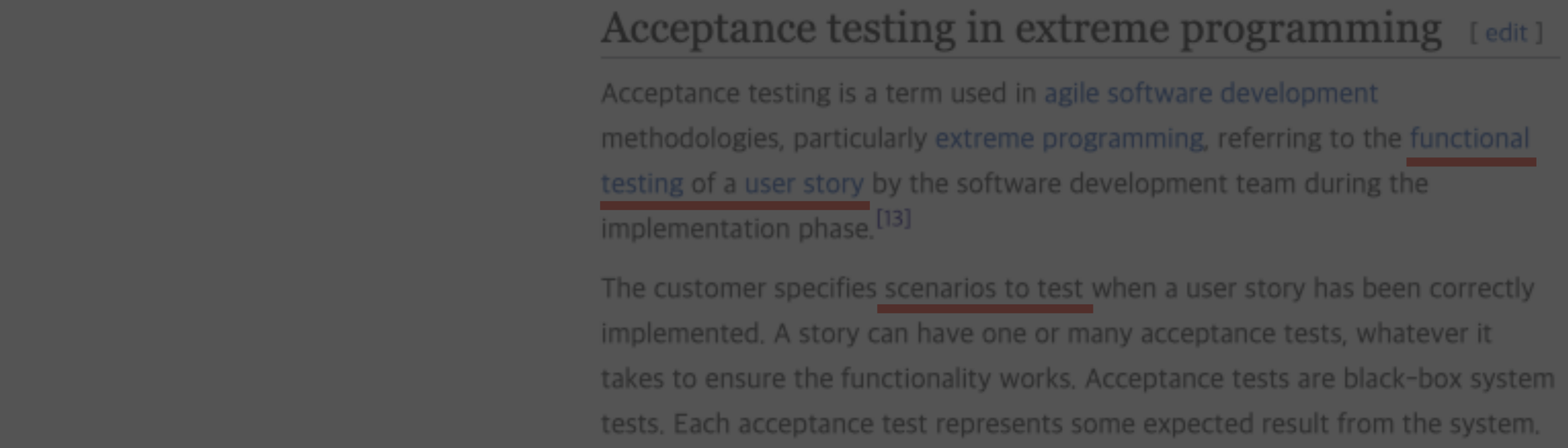
Acceptance testing is a term used in agile software development methodologies, particularly extreme programming, referring to the functional testing of a user story by the software development team during the implementation phase.[13]

The customer specifies scenarios to test when a user story has been correctly implemented. A story can have one or many acceptance tests, whatever it takes to ensure the functionality works. Acceptance tests are black-box system tests. Each acceptance test represents some expected result from the system.

Acceptance Test in extreme programming

- **사용자 스토리**를 검증하는 기능 테스트
- 사용자 스토리로 **테스트할 시나리오**를 지정

???



Acceptance testing in extreme programming [edit]

Acceptance testing is a term used in agile software development methodologies, particularly extreme programming, referring to the functional testing of a user story by the software development team during the implementation phase.[13]

The customer specifies scenarios to test when a user story has been correctly implemented. A story can have one or many acceptance tests, whatever it takes to ensure the functionality works. Acceptance tests are black-box system tests. Each acceptance test represents some expected result from the system.

테스트 종류

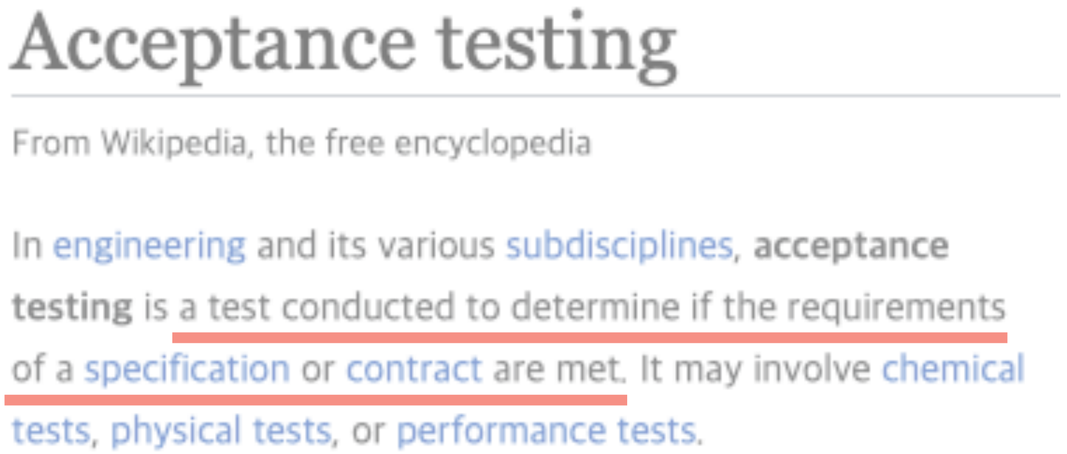
- 단위 테스트
- 통합 테스트
- E2E 테스트

인수 테스트

- **사용자 스토리**를 검증하는 기능 테스트

Acceptance Test

- 명세나 계약의 **요구 사항이 충족되는지 확인**하기 위해 수행되는 테스트



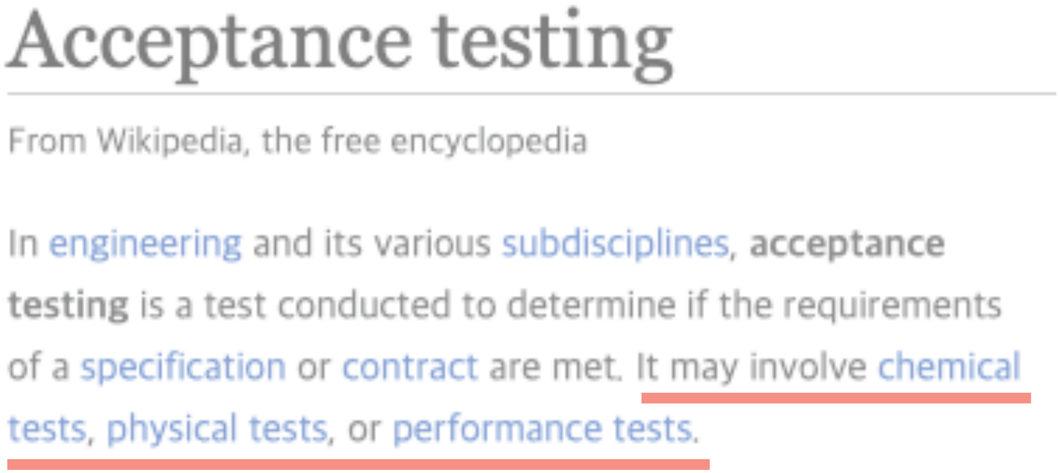
Acceptance testing

From Wikipedia, the free encyclopedia

In engineering and its various subdisciplines, **acceptance testing** is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests.

Acceptance Test

- 소프트웨어 이외 다른 분야에서도 사용되는 용어
- 보통 마지막 단계에서 수행하는 테스트를 의미



Acceptance? 인수?

- Acceptance는 수락, 받아들임이라는 뜻
- Acceptance를 인수로 해석한 이유는 인수를 수락한다는 의미로 해석

Acceptance Test (쉽게 바꾸면)

- 작업을 종료 시켜도 되는지 검증하는 테스트
- 이 테스트가 성공하면 작업 끝!

테스트 주도 개발로 배우는 객체 지향 설계와 실천

우리는 기능을 구현할 때, 만들고자 하는 기능을 수행하는 인수 테스트를 작성하는 것으로 시작한다. 인수 테스트가 실패하는 동안은 시스템이 아직 그 기능을 구현하지 않고 있다는 것을 보여준다; **인수 테스트가 통과되면, 기능 구현은 끝**이다.

콘솔 애플리케이션 개발을 위한 시나리오 기반 인수 테스트 만들기

로또 요구사항

- 콘솔 기반의 로또 애플리케이션을 만든다.
- 금액을 입력하면 금액에 맞는 갯수의 로또를 구입한다.
- 지난주 당첨 번호를 입력하면 당첨 결과와 수익률이 계산된다.

정상적인 시나리오

- 10000원을 입력한다
- 10장의 로또가 구매된다
- 구입한 10장의 로또 번호가 출력된다
- 지난주 당첨 번호로 1, 2, 3, 4, 5, 6을 입력한다
- 당첨 통계가 출력된다
- 수익률이 출력된다

비정상적인 시나리오 - 적은 금액

- 100원을 입력한다
- 로또를 구입할 수 없다.

비정상적인 시나리오 - 유효하지 않은 당첨 번호

- 10000원을 입력한다
- 10장의 로또가 구매된다
- 구입한 10장의 로또 번호가 출력된다
- 지난주 당첨 번호로 1, 1, 1, 1, 1, 1을 입력한다
- 당첨 통계와 수익률을 출력할 수 없다

로또 테스트 코드 작성 - 비정상적인 시나리오 (적은 금액)

```java
@DisplayName("로또 한장의 가격보다 적은 금액 입력")
@Test
void lessPriceThanLotto() {
    LottoApplication lottoApplication = new LottoApplication();

    assertThrows(RuntimeException.class, () -> {
        lottoApplication.insertMoney(100);
    });
}
```

로또 테스트 코드 작성 - 비정상적인 시나리오 (유효하지 않은 번호)

```java
@DisplayName("유효하지 않은 로또 당첨 번호")
@Test
void invalidWinningLottoNumber() {
    LottoApplication lottoApplication = new LottoApplication();

    Message message = lottoApplication.insertMoney(10000);
    assertThat(message.getText()).isEqualTo("10장의 로또가 구매되었습니다.");

    assertThrows(RuntimeException.class, () -> {
        lottoApplication.insertWinningLottoNumber(1, 1, 1, 1, 1, 1);
    });
}
```

로또 테스트 코드 작성 - 정상적인 시나리오

```java
@DisplayName("정상 동작 시나리오")
@Test
void common() {
    LottoApplication lottoApplication = new LottoApplication();

    Message buyMessage = lottoApplication.insertMoney(10000);
    assertThat(buyMessage.getText()).isEqualTo("10장의 로또가 구매되었습니다.");

    Message winningMessage = lottoApplication.insertWinningLottoNumber(1, 2, 3, 4,
    assertThat(winningMessage.getText()).contains("당첨 통계");
    assertThat(winningMessage.getText()).contains("당첨 수익률");
}
```

API 개발을 위한
AI나리오 기반 인수 테스트 만들기

요구사항: 지하철역 관리 기능 구현하기

- 지하철 역 생성
- 지하철 역 목록 조회
- 지하철 역 삭제

시나리오: 지하철역 관리 기능 구현하기

- 지하철 역 생성
  - 지하철 역 생성 요청을 한다.
  - 지하철 역이 생성을 되었다.
- 지하철 역 목록 조회
  - 지하철 역이 생성되어 있다.
  - 지하철 역 목록 조회 요청을 한다.
  - 지하철 역 목록이 응답되었다.

…

API 레벨의 테스트 작성

```java
@DisplayName("지하철역을 생성한다.")
@Test
void createStation() {
    // given
    Map<String, String> params = new HashMap<>();
    params.put("name", "강남역");

    // when
    ExtractableResponse<Response> response = RestAssured.given().log().all()
            .body(params)
            .contentType(MediaType.APPLICATION_JSON_VALUE)
            .when()
            .post("/stations")
            .then().log().all()
            .extract();

    ...

}
```

테스트 응답 결과를 검증

```
import static org.assertj.core.api.Assertions.assertThat;

...

@DisplayName("지하철역을 생성한다.")
@Test
void createStation() {

    ...

    ExtractableResponse<Response> response = RestAssured.given().log().all()
            .body(params)
            .contentType(MediaType.APPLICATION_JSON_VALUE)
            .when()
            .post("/stations")
            .then().log().all()
            .extract();

    // then
    assertThat(response.statusCode()).isEqualTo(HttpStatus.CREATED.value());
    assertThat(response.header("Location")).isNotBlank();
}
```

이번 과정에서의 인수 테스트

인수 테스트는 API 테스트?

인수 테스트는 E2E 테스트?

인수 테스트는 통합 테스트?

인수 테스트는

테스트의 의도에 따라

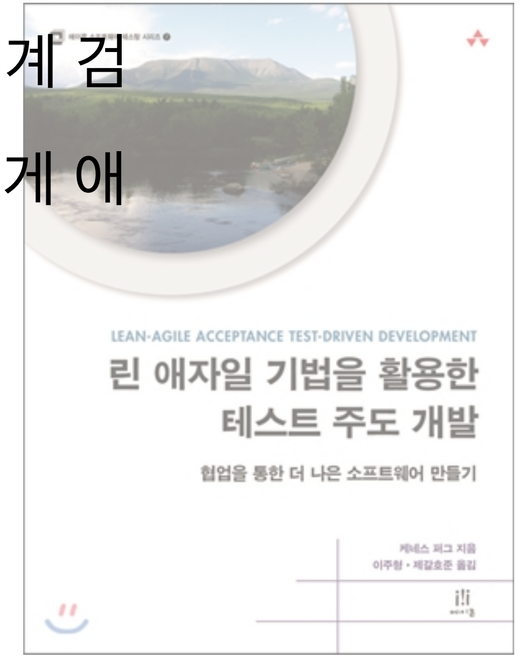
구현 방법이 달라진다.

테스트가 검증하는 대상

- 단위 테스트: 구현한 부분, 단위를 검증
- 통합 테스트: 각 단위들이 유기적으로 잘 동작하는지 검증
- 인수 테스트: 요구사항을 만족하는지를 검증

린 애자일 기법을 활용한 (인수) 테스트 주도 개발

- 인수 테스트의 개념은 테스트 의도에 따라 정해지는 것이지 테스트를 어떻게 구현하는지에 따라 정해지는 것이 아니다. 유닛 레벨이나 통합 레벨, 사용자 인터페이스 레벨에서 인수 테스트를 적용할 수 있다. … 더 나아가, 인수 테스트를 유닛이나 컴포넌트가 의도한 동작을 하는지 확인하는 설계 검증 테스트로 사용할 수 도 있다. 어떤 경우든 인수 테스트는 사용자에게 애플리케이션이 인도될 수 있는 지를 확인한다.



이번 과정에서의 인수 테스트

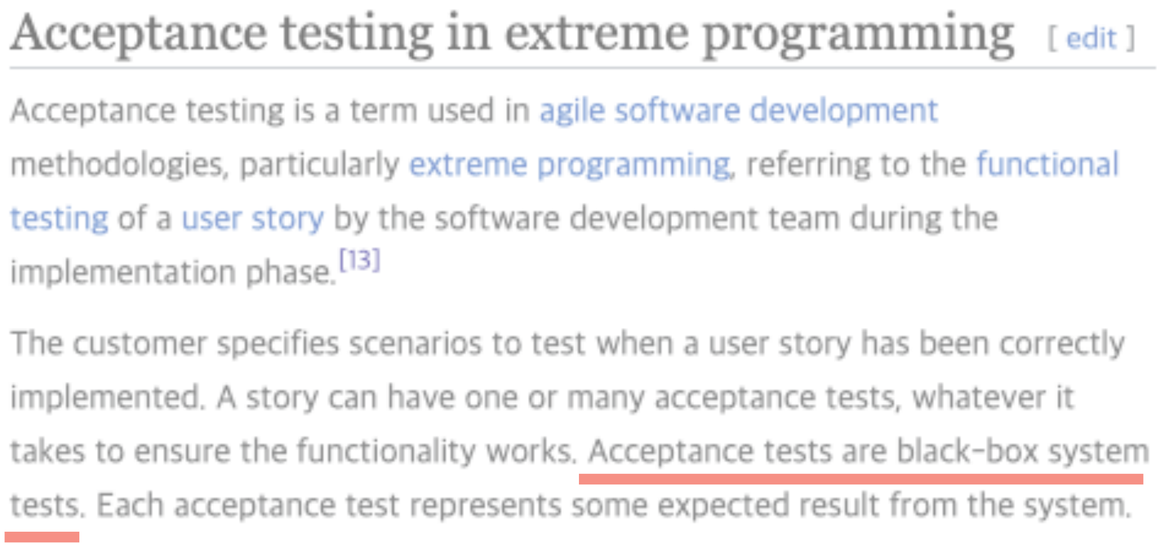
- 백엔드 개발자가 단독적으로 적용해서 실천해볼 수 있는 범위
- 고객은 프론트엔드 개발자 혹은 API 활용하는 사람 대상
- API 접점에서 검증하는 **E2E 테스트**
- API의 Request와 Response 정보 이외 내부 정보는 최대한 가리는 **블랙박스 형식의 테스트**

인수 테스트 만들기

인수 테스트 만들기 전 알아야 할 것

블랙 박스 테스트

- 인수 테스트는 블랙 박스 테스트의 성격을 가지고 있음



Acceptance testing in extreme programming [edit]

Acceptance testing is a term used in agile software development methodologies, particularly extreme programming, referring to the functional testing of a user story by the software development team during the implementation phase.[13]

The customer specifies scenarios to test when a user story has been correctly implemented. A story can have one or many acceptance tests, whatever it takes to ensure the functionality works. Acceptance tests are black-box system tests. Each acceptance test represents some expected result from the system.

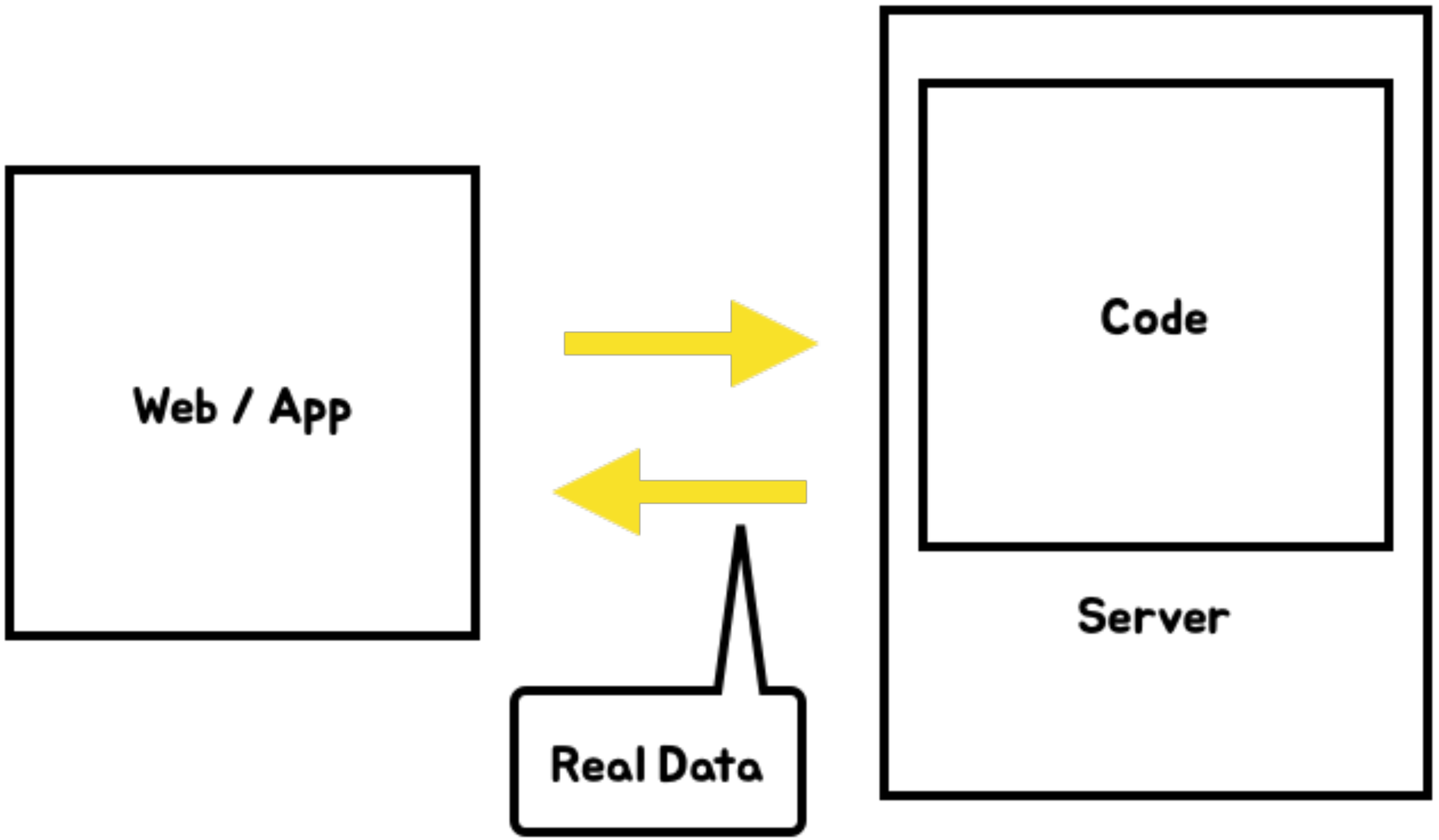
블랙박스?

- 클라이언트는 결과물의 내부 구현이나 사용된 기술을 기반으로 검증하기 보다는 표면적으로 확인할 수 있는 요소를 바탕으로 검증하려 함
- 실제 사용하는 상황의 시나리오를 바탕으로 요구사항을 작성
- 내부 구현이나 기술에 의존적이지 않는 블랙 박스 테스트

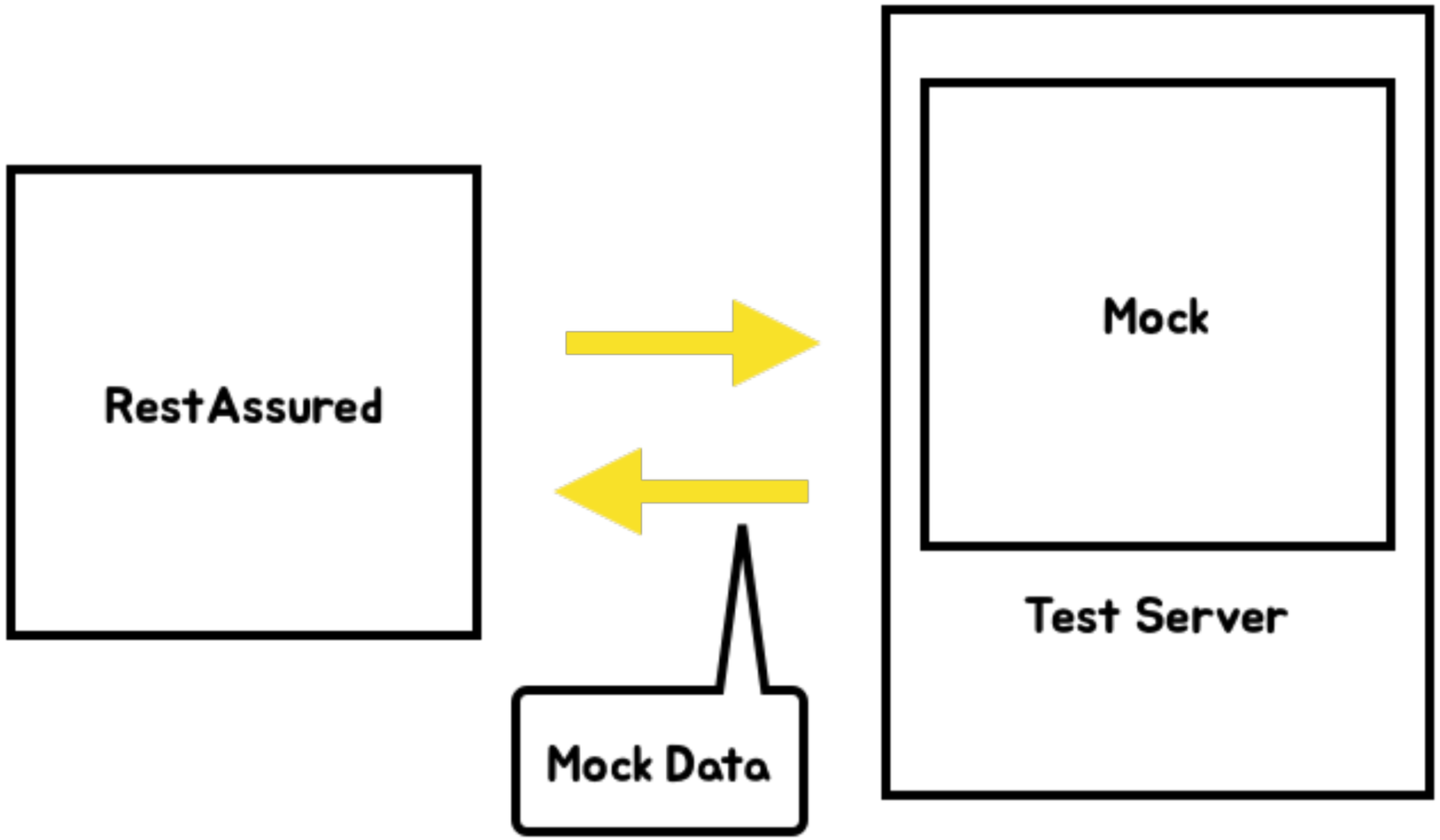
E2E 테스트

- API레벨의 블랙박스 테스트 이므로 요청과 응답 기준으로 API레벨의 E2E 테스트로 검증

클라이언트와 서버의 요청과 응답



테스트 환경에서 클라이언트와 서버의 요청과 응답



인수 테스트 도구 설명

Spring Boot Test

- 테스트에 사용할 ApplicationContext를 쉽게 지정하게 도와줌
- 기존 @ContextConfiguration의 발전된 기능
- SpringApplication에서 사용하는 ApplicationContext를 생성해서 작동

https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-testing-spring-boot-applications

예시) 인수 테스트 클래스

```
@DisplayName("지하철 역 관련 기능")
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class StationAcceptanceTest {
    ...
}
```

RestAssured

REST-assured는 REST API의 테스트 및 검증을 단순화하도록 설계

HTTP 작업에 대한 검증을 위한 풍부한 API를 활용 가능

```
@Test public void
lotto_resource_returns_200_with_expected_id_and_winners() {

    when().
        get("/lotto/{id}", 5).
    then().
        statusCode(200).
        body("lotto.lottoId", equalTo(5),
                "lotto.winners.winnerId", hasItems(23, 54));

}
```

예시) RestAssured 객체 설정

```java
@DisplayName("지하철 역 관련 기능")
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class StationAcceptanceTest {
    @LocalServerPort
    int port;

    @BeforeEach
    public void setUp() {
        RestAssured.port = port;
    }
    ...
}
```

예시) RestAssured 객체로 요청 보내기

```java
@DisplayName("지하철역을 생성한다.")
@Test
void createStation() {
    // given
    Map<String, String> params = new HashMap<>();
    params.put("name", "강남역");

    // when
    ExtractableResponse<Response> response = RestAssured.given().log().all()
            .body(params)
            .contentType(MediaType.APPLICATION_JSON_VALUE)
            .when()
            .post("/stations")
            .then().log().all()
            .extract();

    ...

}
```

JsonPath

Jayway JsonPath

A Java DSL for reading JSON documents.

build passing | maven central 2.5.0 | javadoc 2.5.0

Jayway JsonPath is a Java port of Stefan Goessner JsonPath implementation.

```
response.jsonPath().getList(".", StationResponse.class)
```

JsonPath

```
{
    "store": {
        "book": [
            {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
            },
            {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
            }
        ],
        "bicycle": {
            "color": "red",
            "price": 19.95
        }
    },
    "expensive": 10
}
```

JsonPath (click link to try)	Result
$.store.book[*].author	The authors of all books
$..author	All authors
$.store.*	All things, both books and bicycles
$.store..price	The price of everything
$..book[1]	The second book

인수 테스트 실습

- 지하철역 생성 인수 테스트 만들기
- 지하철역 조회 인수 테스트 만들기

인수 테스트 만들기 피드백

@SpringBootTest

- 테스트에 사용할 ApplicationContext를 쉽게 지정하게 도와줌
- 기존 @ContextConfiguration의 발전된 기능
- SpringApplication에서 사용하는 ApplicationContext를 생성해서 작동

https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-testing-spring-boot-applications

webEnvironment

@SpringBootTest의 webEnvironment 속성을 사용하여 테스트 서버의 실행 방법을 설정

- MOCK: Mocking된 웹 환경을 제공, MockMvc를 사용한 테스트를 진행할 수 있음
- RANDOM_PORT: 실제 웹 환경을 구성
- DEFINED_PORT: 실제 웹 환경을 구성, 지정한 포트를 listen
- NONE: 아무런 웹 환경을 구성하지 않음

RestAssured

REST-assured는 REST API의 테스트 및 검증을 단순화하도록 설계

HTTP 작업에 대한 검증을 위한 풍부한 API를 활용 가능

```
@Test public void
lotto_resource_returns_200_with_expected_id_and_winners() {

    when().
        get("/lotto/{id}", 5).
    then().
        statusCode(200).
        body("lotto.lottoId", equalTo(5),
                "lotto.winners.winnerId", hasItems(23, 54));

}
```

MockMvc vs WebTestClient vs RestAssured

- MockMvc
  - @SpringBootTest의 webEnvironment.MOCK과 함께 사용 가능하며 **mocking 된 web environment**(ex tomcat) 환경에서 테스트
- WebTestClient
  - @SpringBootTest의 webEnvironment.RANDOM_PORT 나 DEFINED_PORT와 함께 사용, Netty를 기본으로 사용
- RestAssured
  - **실제 web environment(Apache Tomcat)**을 사용하여 테스트

테스트 격리?

- 테스트를 각각 실행하면 성공하는데 한번에 실행하면?
- 테스트의 결과가 다른 테스트에 영향을 주는 이유는?
  - 힌트: Context Caching

인수 테스트 주도 개발

이게 맞나?

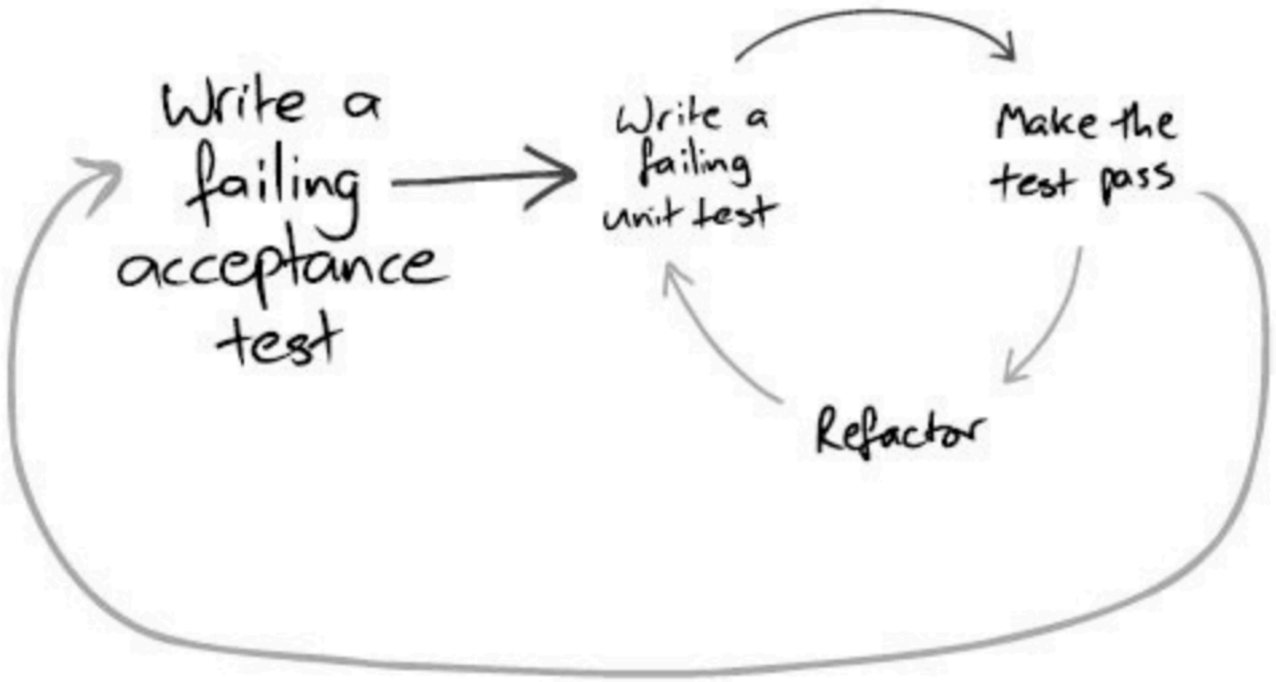
ATDD란 in wikipedia

인수 테스트는 고객과 개발자, 그리고 테스터간의 커뮤니케이션을 기반으로 한 개발 방법론이다. … 이러한 프로세스는 개발자와 테스터가 구현 전에 고객의 요구를 이해하는 데 도움이되며 고객이 자신의 도메인 언어로 대화 할 수 있도록한다.

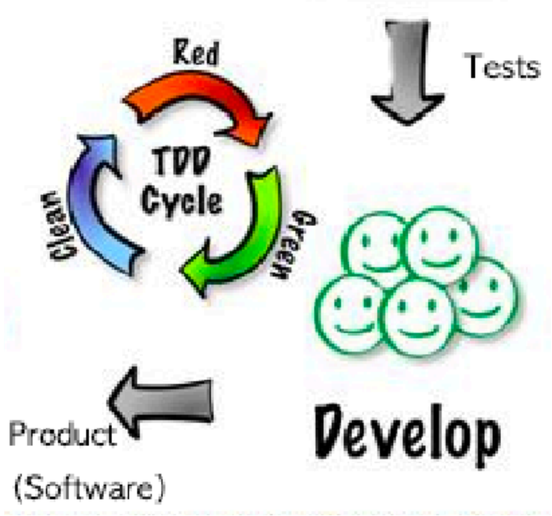
출처:

https://en.wikipedia.org/wiki/Acceptance_test%E2%80%93driven_development

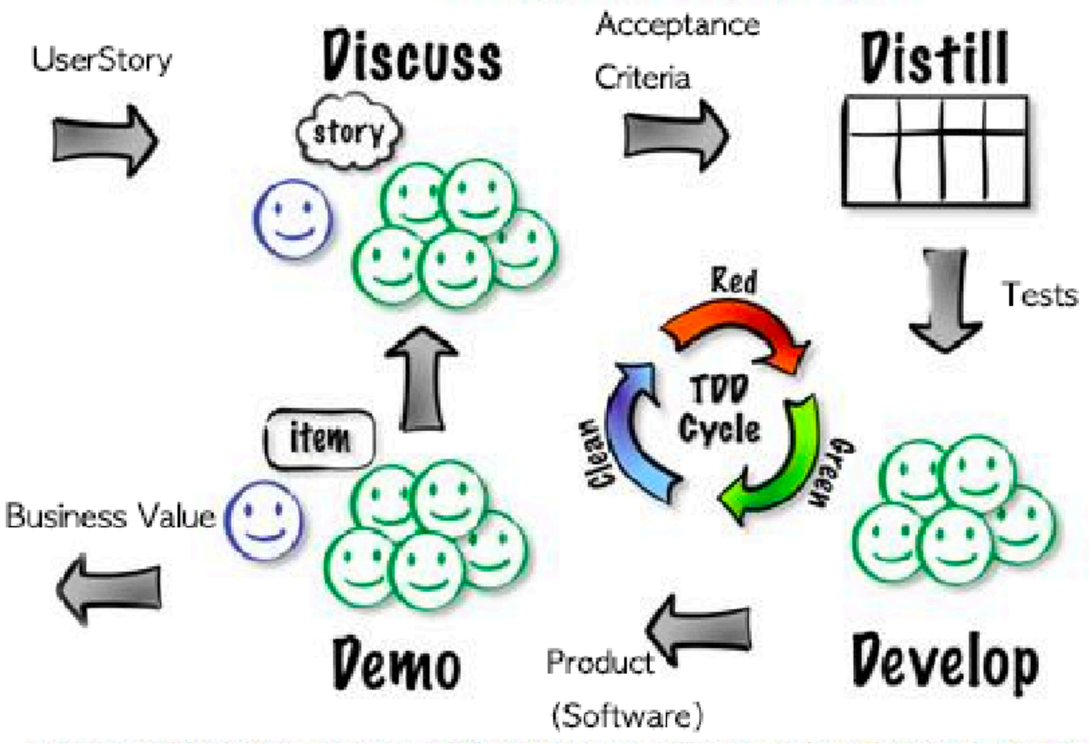
ATDD + TDD Cycle



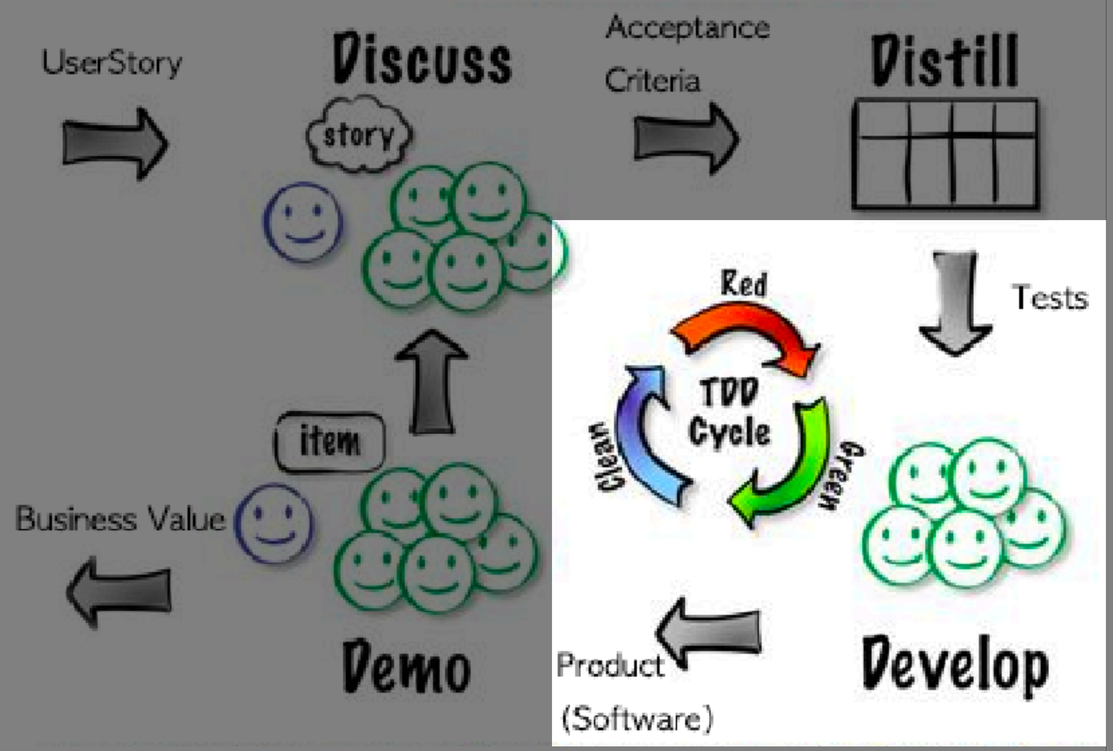
ATDD Cycle - 개발 관련 부분



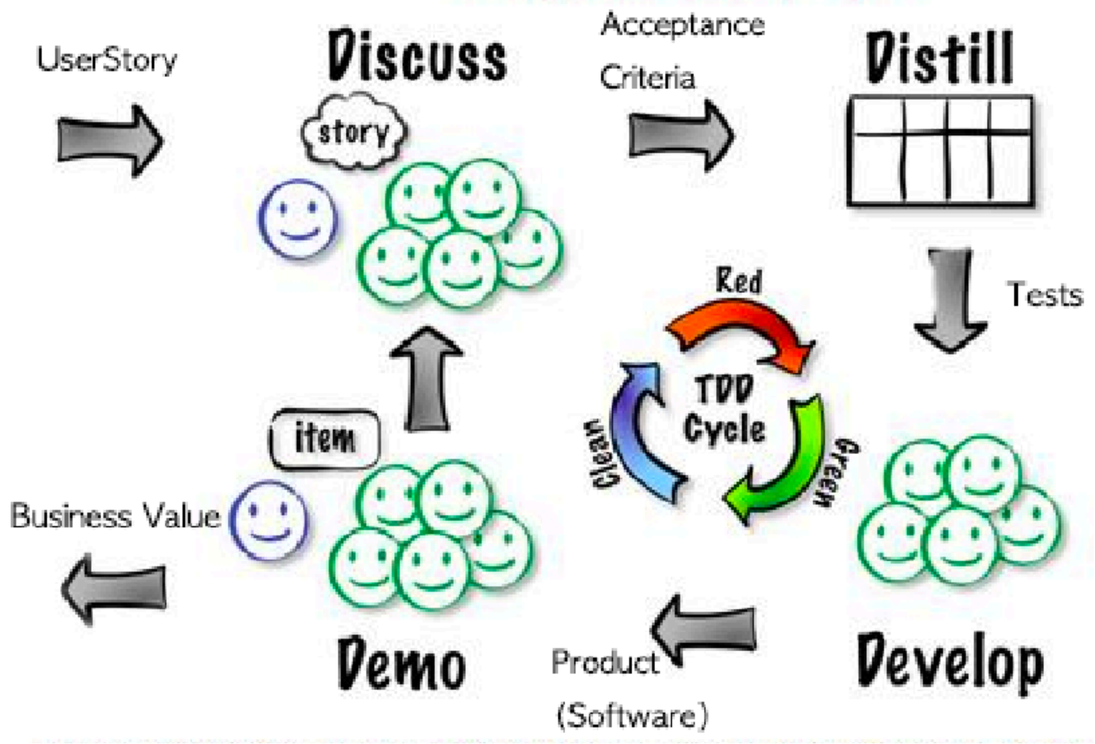
구체적인 ATDD Cycle



ATDD Cycle

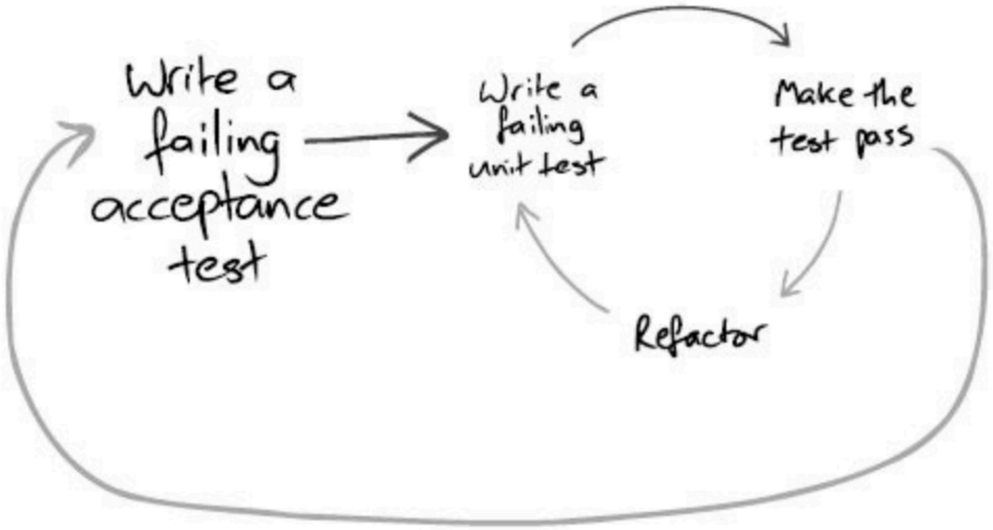

UserStory
Discuss
story
Acceptance
Criteria
Distill
Red
Tests
TDD
Cycle
Clean
Green
item
Business Value
Demo
Product
(Software)
Develop

ATDD Cycle


UserStory
Discuss
story
Acceptance
Criteria
Distill
Tests
Red
TDD
Cycle
Green
Clean
item
Business Value
Demo
Product
(Software)
Develop

인수 테스트 주도로 개발 하기

인수 테스트 주도로 개발은 어떻게 하나요?



인수 테스트 주도로 개발 하기

인수 조건 정의

인수 테스트 작성

기능 구현

인수 조건

- 인수 테스트가 충족해야하는 조건
- 이번 과정에서는 시나리오 형태로 표현

인수 조건 형식

- scenario-oriented (the Given/When/Then template)
- rule-oriented (the checklist template)
- custom formats.

출처: https://www.altexsoft.com/blog/business/acceptance-criteria-purposes-formats-and-best-practices/

인수 조건 예시

```
Feature: 최단 경로 구하기

  Scenario: 지하철 최단 경로 조회
    Given 지하철역들이 등록되어 있다.
    And 지하철노선이 등록되어 있다.
    And 지하철노선에 지하철역들이 등록되어 있다.
    When 사용자는 출발역과 도착역의 최단 경로 조회를 요청한다.
    Then 사용자는 최단 경로의 역 정보를 응답받는다.
```

인수 테스트

- 인수 조건을 검증하는 테스트
- 실제 요청/응답하는 환경과 유사하게 테스트 환경을 구성

인수 테스트 클래스

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class LineAcceptanceTest {

    @DisplayName("최단 경로 구하기")
    @Test
    void findPath() {
    ...
}
```

인수 테스트 예시

```java
@DisplayName("지하철 노선을 조회한다.")
@Test
void getLine() {
    // given
    // 지하철_노선_등록되어_있음
    Map<String, String> params = new HashMap<>();
    params.put("name", "신분당선");
    params.put("color", "bg-red-600");
    params.put("startTime", LocalTime.of( hour: 05,  minute: 30).format(DateTimeFormatter.ISO_TIME));
    params.put("endTime", LocalTime.of( hour: 23,  minute: 30).format(DateTimeFormatter.ISO_TIME));
    params.put("intervalTime", "5");

    ExtractableResponse<Response> createResponse = RestAssured.given().log().all().
            contentType(MediaType.APPLICATION_JSON_VALUE).
            body(params).
            when().
            post( path: "/lines").
            then().
            log().all().
            extract();

    // when
    // 지하철_노선_조회_요청
    String uri = createResponse.header( name: "Location");

    ExtractableResponse<Response> response = RestAssured.given().log().all().
            accept(MediaType.APPLICATION_JSON_VALUE).
            when().
            get(uri).
            then().
            log().all().
            extract();

    // then
    // 지하철_노선_응답됨
    assertThat(response.statusCode()).isEqualTo(HttpStatus.OK.value());
    assertThat(response.as(LineResponse.class)).isNotNull();
}
```

인수 테스트 예시2

```
@DisplayName("지하철 노선에 등록된 지하철역을 제외한다.")
@Test
void removeLineSection1() {
    // given
    지하철_노선에_지하철역_등록_요청(신분당선, 강남역, 양재역, 2);
    지하철_노선에_지하철역_등록_요청(신분당선, 양재역, 정자역, 2);

    // when
    ExtractableResponse<Response> removeResponse
                = 지하철_노선에_지하철역_제외_요청(신분당선, 양재역);

    // then
    지하철_노선에_지하철역_제외됨(removeResponse);
    ExtractableResponse<Response> response
                = LineAcceptanceTest.지하철_노선_조회_요청(신분당선);
    지하철_노선에_지하철역_순서_정렬됨(response, Arrays.asList(강남역, 정자역, 광교역));
}
```

기능 구현

- 인수테스트 기반으로 하나씩 기능 개발을 진행

미션 소개 - 단계별 요구사항