인턴 연구원 최종 발표

엄현서
DB 1실

2022. 08. 30

Tmax

목차

Tmax

개요

전반기

- SQL, PL/SQL 스터디
- Interpreter 코딩을 통한 쿼리 처리 과정의 이해

후반기

- DB 스터디, PL/SQL 추가 기능 실습
- 프로젝트 1 - Materalized View 구현
- 프로젝트 2 - DBMS_UTILITY.COMMA_TO_TABLE 구현
- 프로젝트 3 - UTL_SMTP 패키지 구현

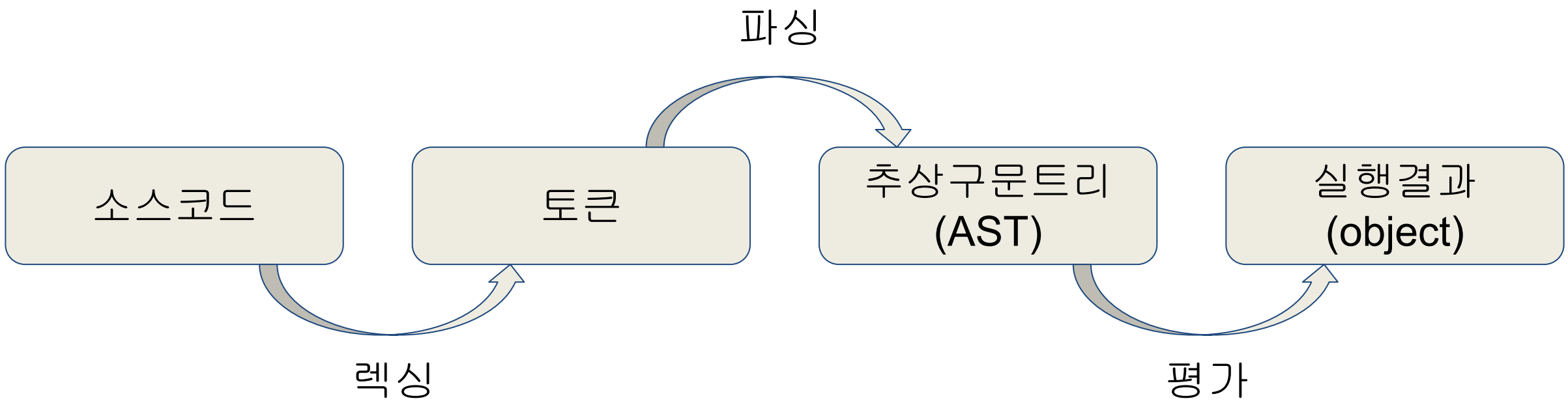
Tmax

전반기 진행 프로젝트 - Monkey 언어

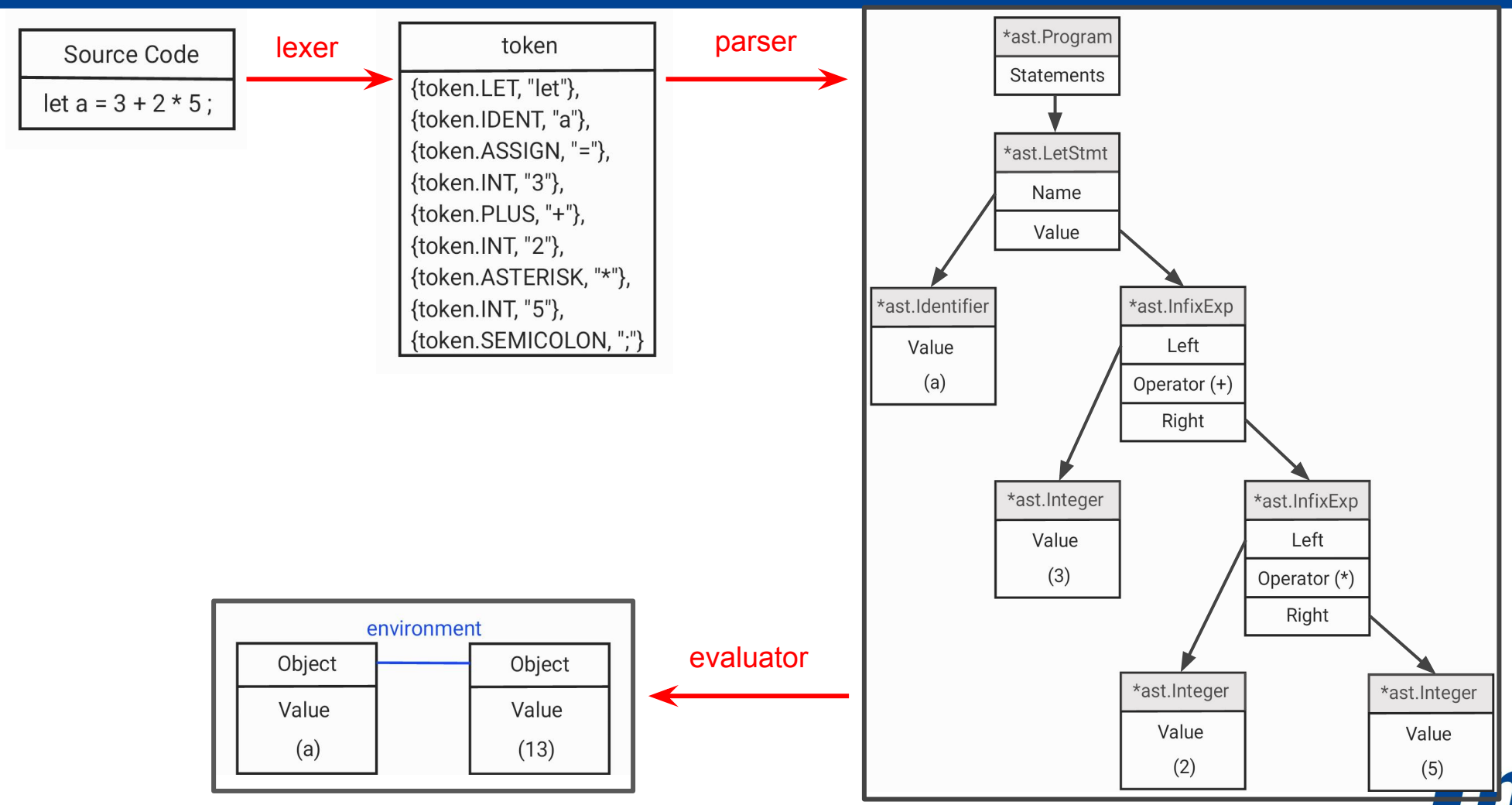
```
let age = 1;

let result = 10 * (20 / 2);

let add = fn(a, b) { return a + b; };
add(1, 2);

puts(result);
```

Tmax

전반기 진행 프로젝트 - 개요

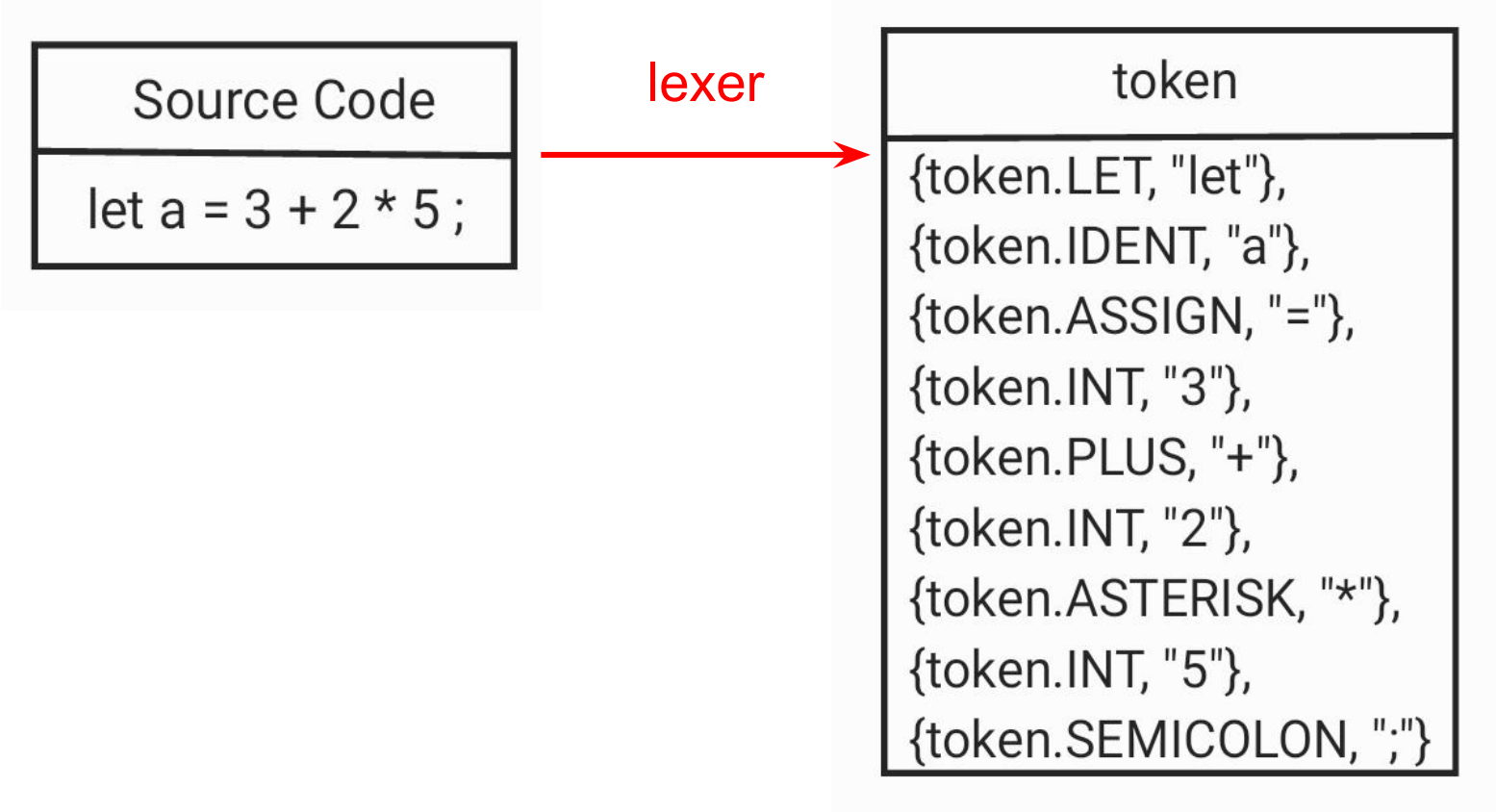


전반기 진행 프로젝트 - 개요



전반기 진행 프로젝트 - 렉싱

소스코드를 입력으로 받고, 소스코드를 표현하는 토큰 열을 결과로 출력.



전반기 진행 프로젝트 - 렉싱

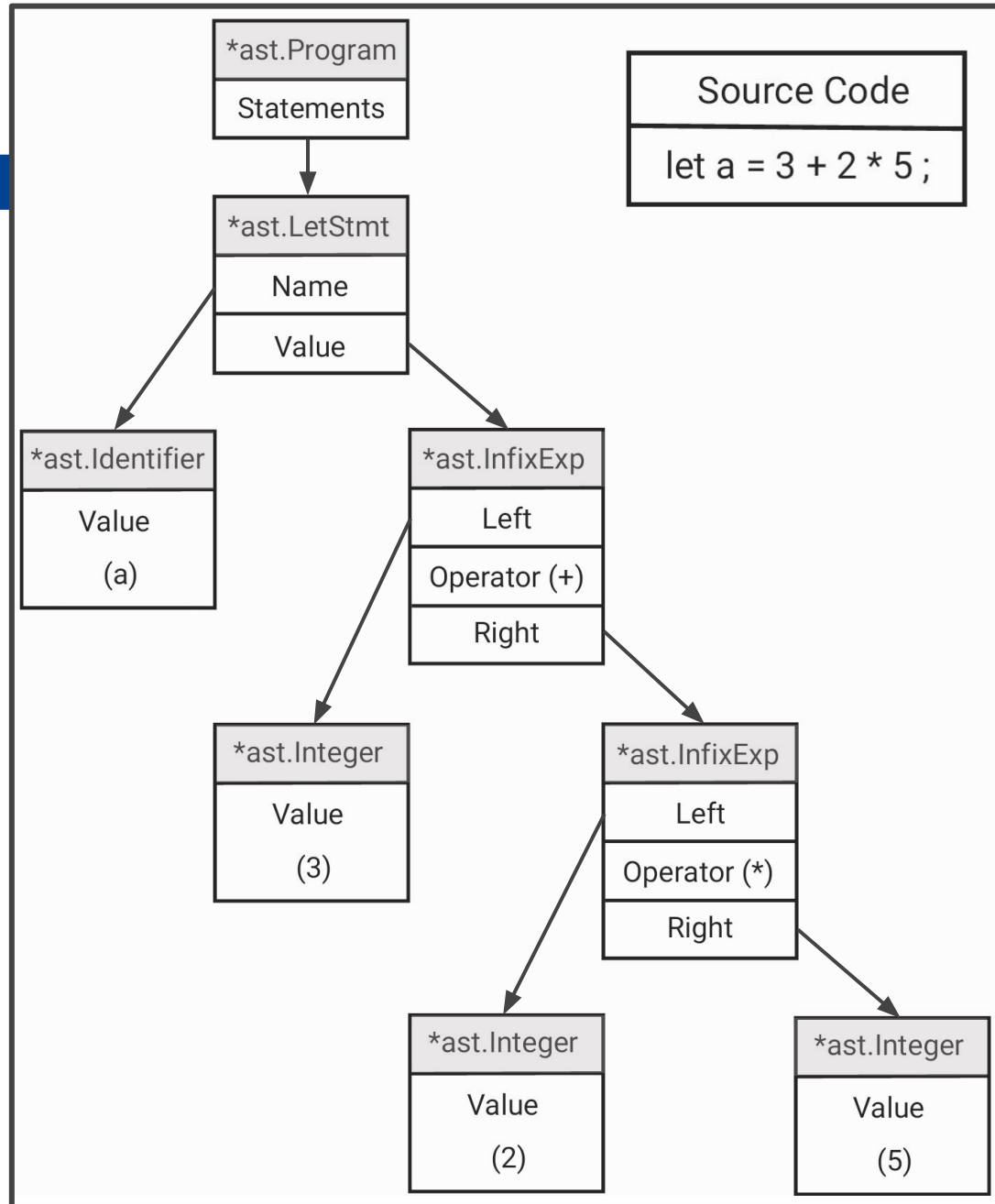
렉서가 처리할 수 있는 토큰의 종류:

- 예약어 (let, fn, if, else, return, true, false)
- 식별자
- 숫자
- 한 글자 토큰(= + - * / ! < > ( ) { })
- 두 글자 토큰(== !=)

전반기 진행 프로젝트 - 파싱

- 토큰을 추상구문트리(AST, Abstract Syntax Tree)로 변환.

- 하향식 연산자 우선순위 파서
  (top down operator precedence parser)
  = 프랫 파서(Pratt parser)



전반기 진행 프로젝트 - 파싱

AST의 두 가지 노드 타입

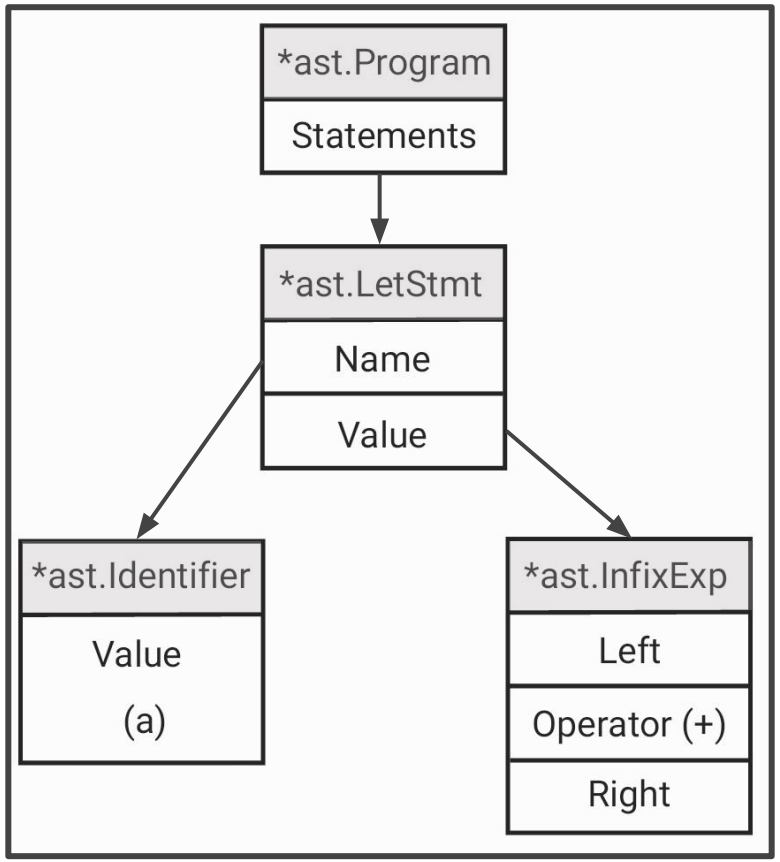
- 명령문(Statement)
- 표현식(Expression)

```
type Node interface {
    TokenLiteral() string
    String() string
}

type Statement interface {
    Node
    statementNode()
}

type Expression interface {
    Node
    expressionNode()
}
```

Tmax

전반기 진행 프로젝트 - 파싱

let <identifier> = <expression> ;



전반기 진행 프로젝트 - 파싱

표현식 파싱이 복잡한 이유

- 연산자 우선순위(operator precedence) 존재.

  5 * 5 + 10

  5 * (5 + 10)

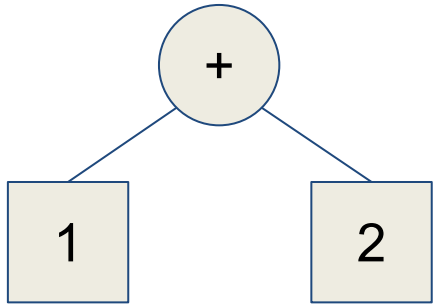
- 같은 타입의 토큰들이 여러 위치에 나타날 수 있음.

  -5 - 10

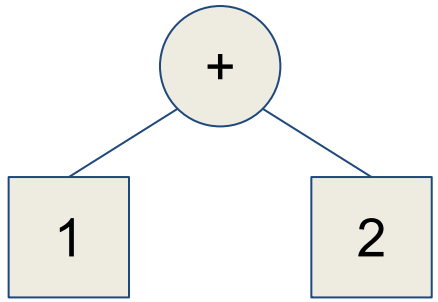
전반기 진행 프로젝트 - 파싱

예시)

1 + 2 + 3;



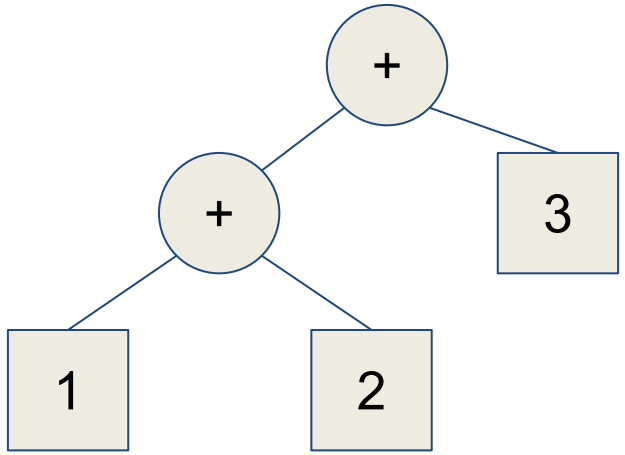
1 + 2 * 3;



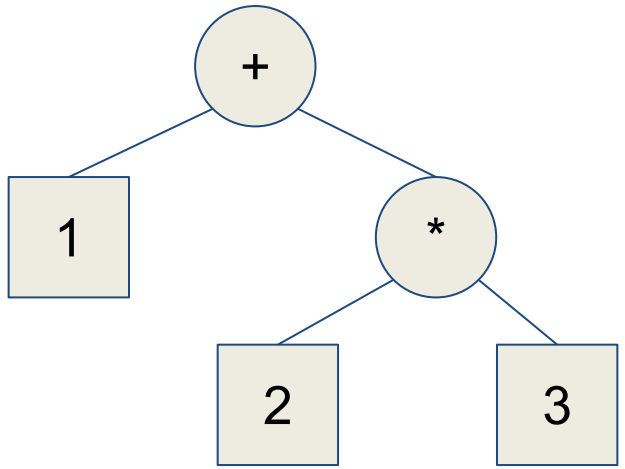
전반기 진행 프로젝트 - 파싱

예시)

1 + 2 + 3;



1 + 2 * 3;

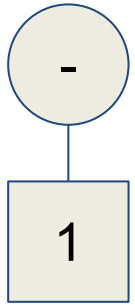


전반기 진행 프로젝트 - 파싱

예시)

-1 + 2 * -3;

1. prefix 검사

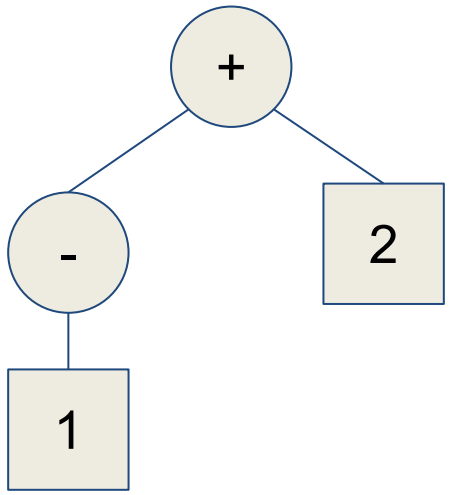


전반기 진행 프로젝트 - 파싱

예시)

-1 + 2 * -3;

1. prefix 검사
2. infix 검사

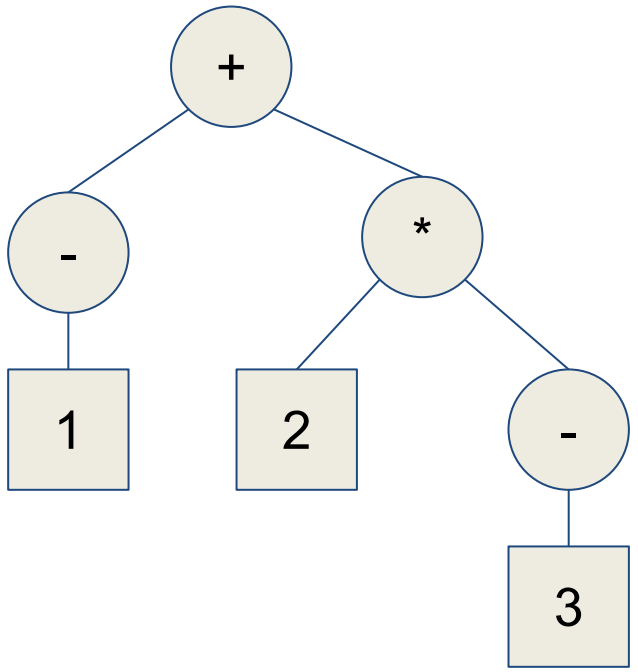


전반기 진행 프로젝트 - 파싱

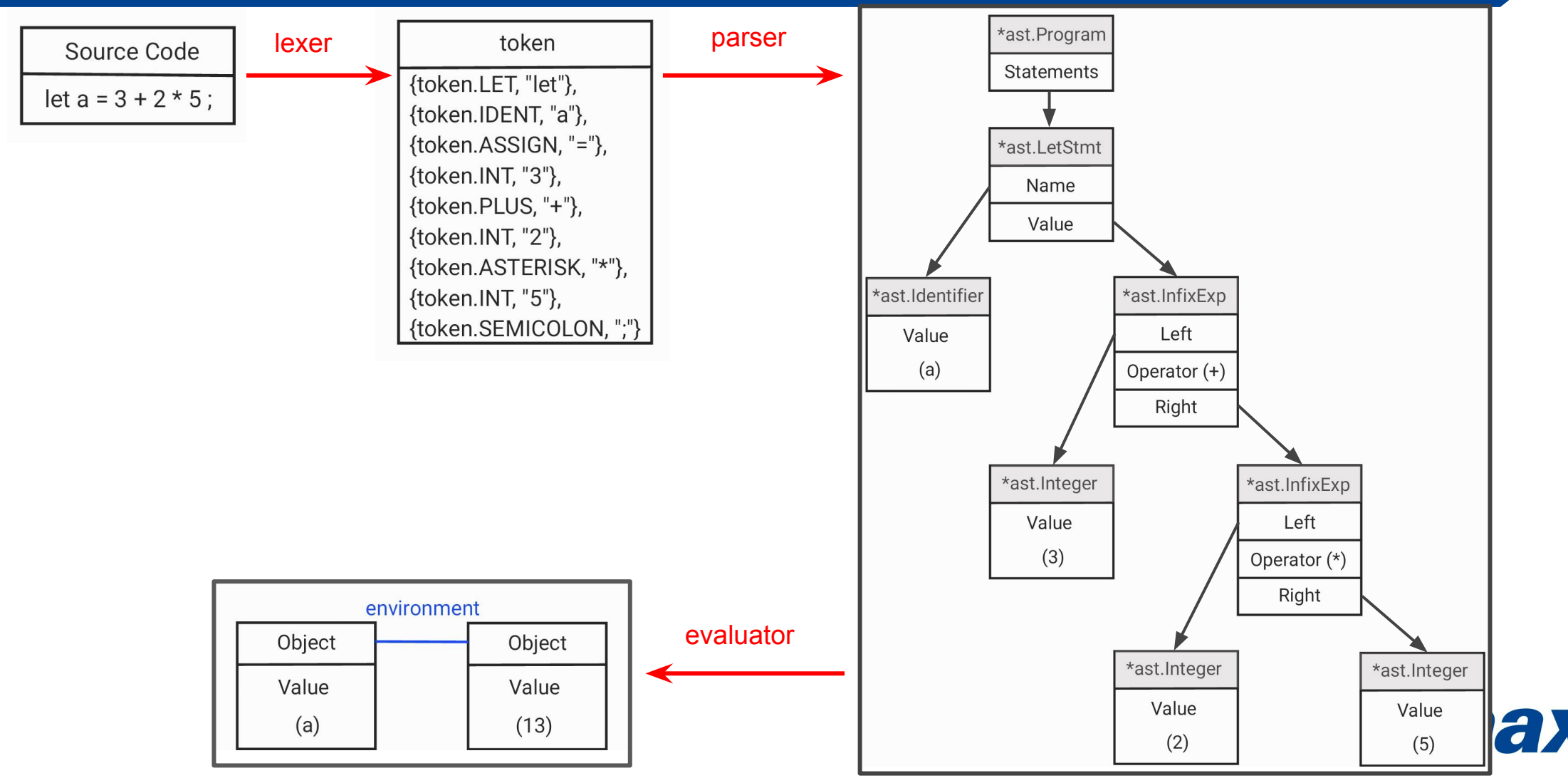
예시)

-1 + 2 * -3;

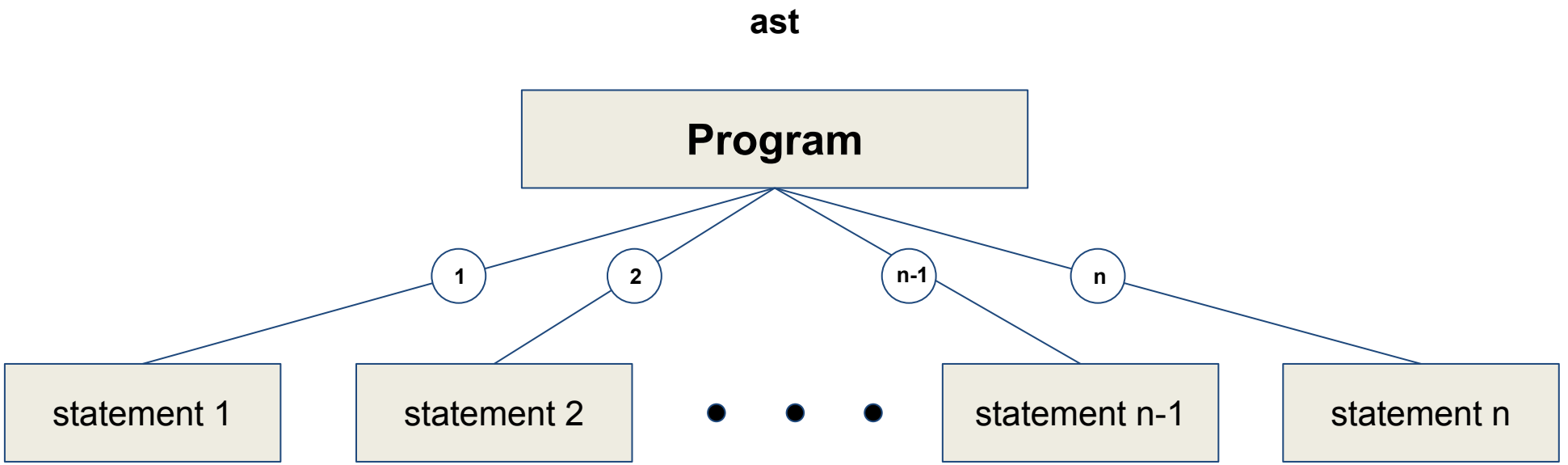
1. prefix 검사
2. infix 검사



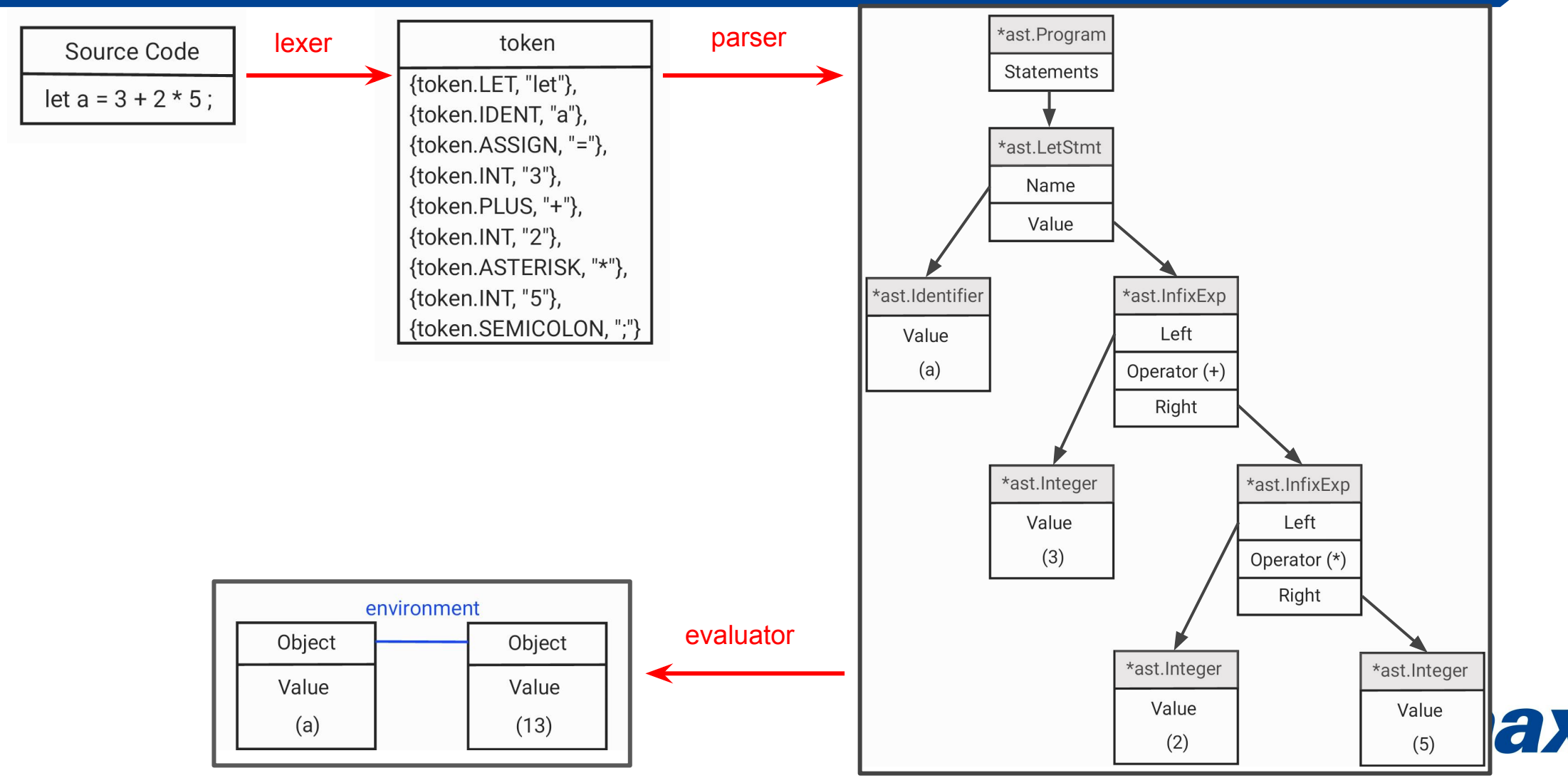
전반기 진행 프로젝트 - 평가



전반기 진행 프로젝트 - 평가

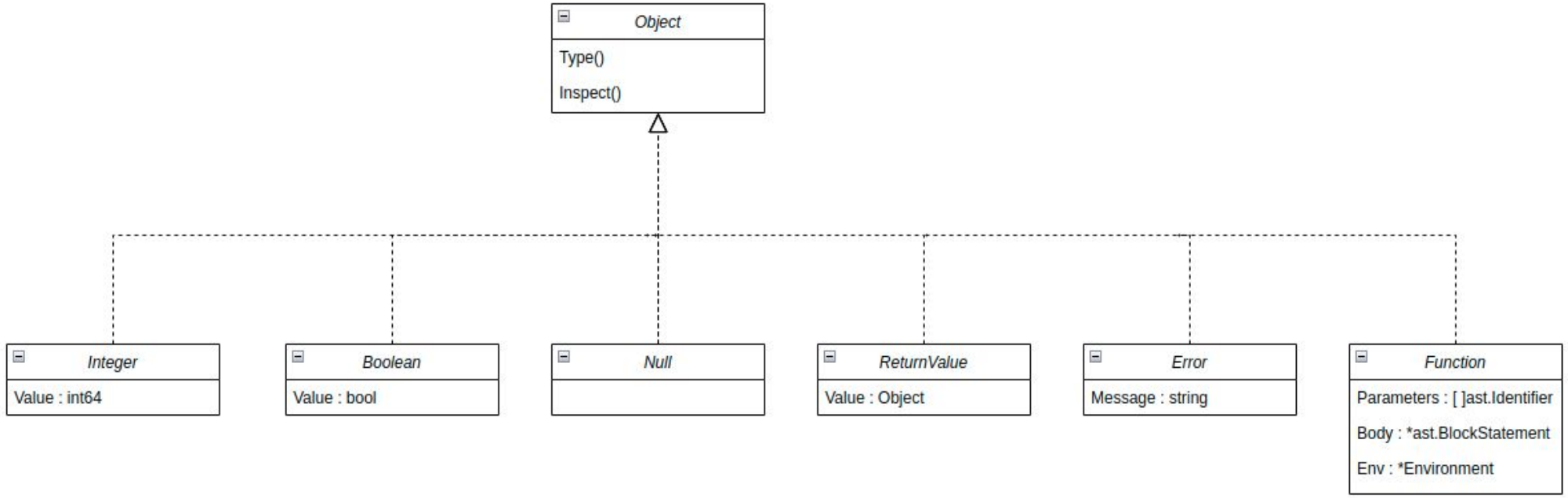


전반기 진행 프로젝트 - 평가



전반기 진행 프로젝트 - 평가

평가 결과로 반환하는 값은 정수, 불, Null 등 여러 타입이 있으므로
Object라는 인터페이스를 도입하고 이를 만족하는 구조체들을 이용해 값을 나타냄.



전반기 진행 프로젝트 - 평가

식별자의 경우 평가 과정에서 바인딩 된 값으로 계산되어야 함.

ex) let a = 5;

let b = a + a; //b = 5 + 5 = 10

string을 key로 받고 Object를 value로 반환하는 map을 통해 바인딩을 구현.

전반기 진행 프로젝트 - 평가

```
ex) let i = 5;
    let printNum = fn(i) {
        puts(i);
    };
    printNum(10);
    puts(i);
```

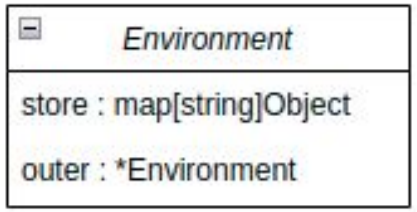
-> 단순히 map 대신 환경(Environment)을 도입함.

전반기 진행 프로젝트 - 평가

환경(Environment)은 구조체로 map과 outer라는 환경을 필드로 가짐.

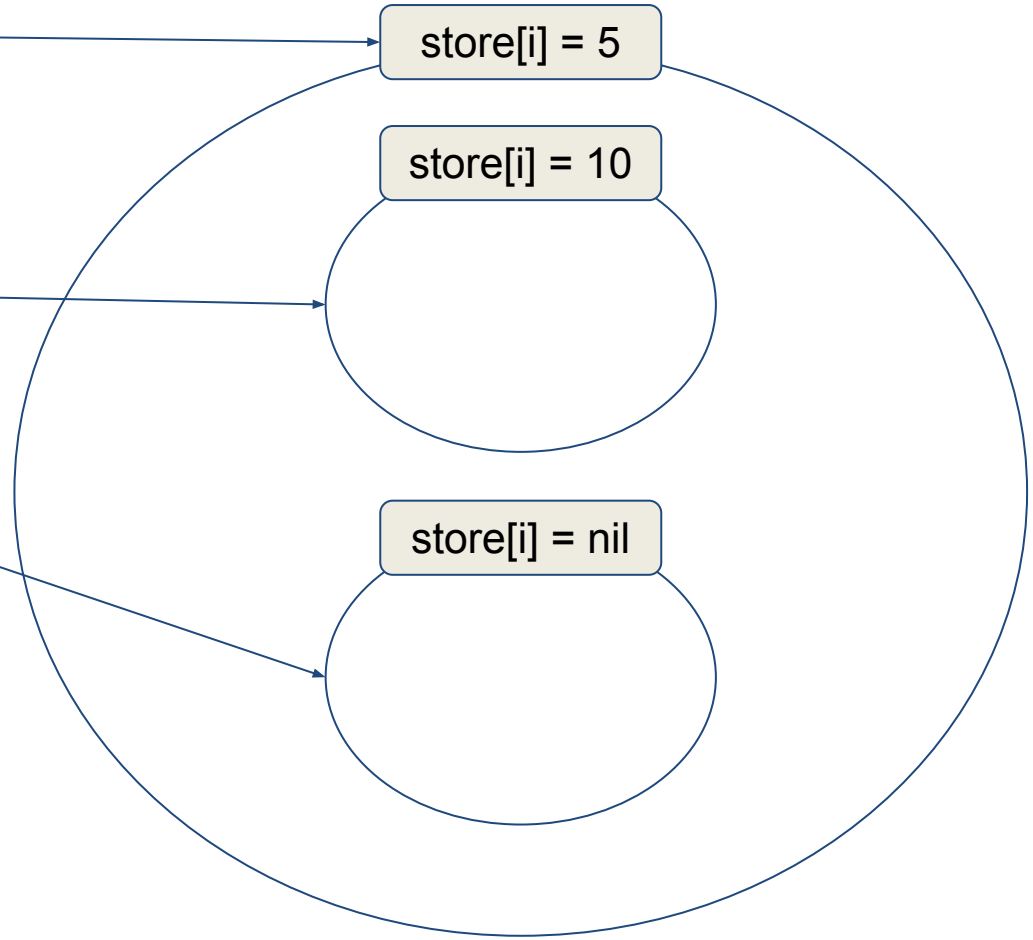
함수 호출 시 새로운 환경을 만들고 기존 환경을 outer에 대입함.

현재 환경의 map에 함수 파라미터의 이름에 대응하는 값이 있으면 사용하고, 없으면 바깥쪽 환경(outer)의 map에서 대응하는 값을 사용함.



전반기 진행 프로젝트 - 평가

```
ex) let i = 5;
    let printNum = fn(i) {
        puts(i);
    };
    printNum(10);
    puts(i);
```



Environment
store : map[string]Object
outer : *Environment

전반기 진행 프로젝트 - 결과

피보나치 함수

```
minju-kim2@minju-com:~/Desktop/study/monkey$ go run main.go
            __,__
   .--.  .-"     "-.  .--.
  / .. \/  .-. .-.  \/ .. \
 | |  '|  /   Y   \  |'  | |
 | \   \  \ 0 | 0 /  /   / |
  \ '- ,\.-"""""""-./, -' /
   ''-' /_   ^ ^   _\ '-''
       |  \._   _./  |
       \   \ '~' /   /
        '._ '-=-' _.'
           '-----'
Better Technology, Better Tomorow
Hello minju-kim2! This is the Monkey programming language!
Feel free to type in commands
>> let fibonacci = fn(x) { if (x == 0){0} else { if(x == 1){ return 1;} else {fibonacci(x - 1) + fibonacci(x - 2); } } };
>> fibonacci(1)
1
>> fibonacci(5)
5
>> fibonacci(10)
55
>> fibonacci(20)
6765
>> exit
minju-kim2@minju-com:~/Desktop/study/monkey$
```

후반기 - DB 스터디, PL/SQL 추가 기능 실습

- IF, FOR, EXCEPTION, FUNCTION, PROCEDURE
- PACKAGE, OBJECT TYPE
- PARTITION, INDEX, RECORD, COLLECTION

Tmax

후반기 - DB 스터디, PL/SQL 추가 기능 실습

- TRIGGER

  특정 DML 작업 또는 DB관련 이벤트 전 또는 후에 자동적으로 작업을 실행

- JOB

  정해진 INTERVAL마다 주기적으로 작업을 실행

후반기 - Materalized View

TABLE t

A	B	C
…	…	…
…	…	…
…	…	…
…	…	…
…	…	…
…	…	…

VIEW v

A	C
…	…
…	…
…	…
…	…
…	…
…	…

M VIEW mv

A	C
…	…
…	…
…	…
…	…
…	…
…	…

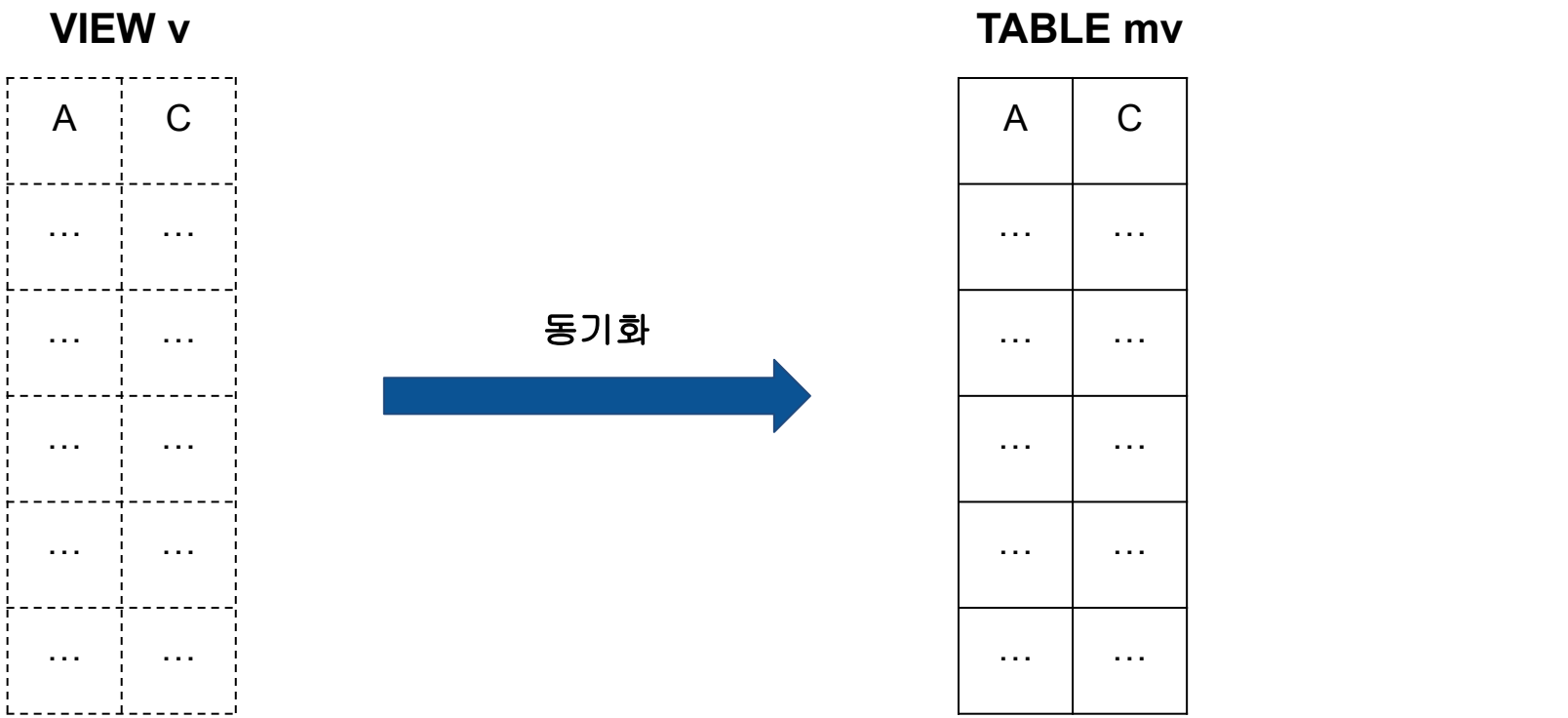
SELECT A, C FROM t

SELECT * FROM v

SELECT * FROM mv

후반기 - Materalized View

PL/SQL을 사용하여 TABLE 4개를 직접 구상하고 JOB과 TRIGGER을 통해 특정 데이터를 조작한 뒤 JOB을 통해 Materalized View처럼 행동하는 테이블을 동기화시켜주는 프로젝트



후반기 - Materalized View

```
CREATE TABLE dept (
   deptno NUMBER(2) NOT NULL
 , dname  VARCHAR2(14)
 , loc    VARCHAR2(13)
 , PRIMARY KEY (deptno)
);

CREATE TABLE emp (
   empno    NUMBER(4) NOT NULL
 , ename    VARCHAR2(10)
 , job      VARCHAR2(9)
 , hiredate DATE
 , sal      NUMBER(7,2)
 , deptno   NUMBER(2)
 , grade    NUMBER
 , PRIMARY KEY (empno)
);

CREATE TABLE salgrade (
   grade  NUMBER
 , lowsal NUMBER
 , hisal  NUMBER
);

CREATE TABLE personeval (
   ename VARCHAR2(10)
 , eval VARCHAR2(1)
);
```

연봉

1년(특정 주기)마다 연봉 상승

연봉등급

연봉등급이 달라졌는지 확인하고 변경

인사평가

인사평가에 따라 연봉 인상률이 달라짐

후반기 - Materalized View

```
CREATE TABLE dept (
   deptno NUMBER(2) NOT NULL
 , dname  VARCHAR2(14)
 , loc    VARCHAR2(13)
 , PRIMARY KEY (deptno)
);

CREATE TABLE emp (
   empno    NUMBER(4) NOT NULL
 , ename    VARCHAR2(10)
 , job      VARCHAR2(9)
 , hiredate DATE
 , sal      NUMBER(7,2)
 , deptno   NUMBER(2)
 , grade    NUMBER
 , PRIMARY KEY (empno)
);

CREATE TABLE salgrade (
   grade  NUMBER
 , lowsal NUMBER
 , hisal  NUMBER
);

CREATE TABLE personeval (
   ename VARCHAR2(10)
 , eval VARCHAR2(1)
);
```

연봉

연봉등급

인사평가

Job과 Trigger 이용

1년(특정 주기)마다 연봉 상승 : Job

연봉등급이 달라졌는지 확인하고 변경 : Trigger

인사평가에 따라 연봉 인상률이 달라짐

후반기 - Materalized View

```
CREATE OR REPLACE PROCEDURE up
IS
 rate NUMBER;
BEGIN
 FOR f_personeval IN (SELECT * FROM personeval) LOOP
  CASE f_personeval.eval
   WHEN 'A' THEN
    rate := 1.3;
   WHEN 'B' THEN
    rate := 1.2;
   WHEN 'C' THEN
    rate := 1.1;
  END CASE;
 UPDATE emp e SET e.sal = e.sal * rate WHERE e.ename = f_personeval.ename;
 END LOOP;
END;
/

DECLARE
 v_job NUMBER;
BEGIN
 DBMS_JOB.SUBMIT (
   job       => v_job
 , what      => 'up;'
 , next_date => SYSDATE
 , interval  => q'[SYSDATE + INTERVAL '1' DAY]'
 );

 COMMIT;

 DBMS_OUTPUT.PUT_LINE (v_job);
END;
```

인사평가가 A이면 1.3, B이면 1.2, C이면 1.1배 연봉이 오르는 프로시저

주기는 하루로 설정

후반기 - Materalized View

```
CREATE TABLE dept (
   deptno NUMBER(2) NOT NULL
 , dname  VARCHAR2(14)
 , loc    VARCHAR2(13)
 , PRIMARY KEY (deptno)
);

CREATE TABLE emp (
   empno    NUMBER(4) NOT NULL
 , ename    VARCHAR2(10)
 , job      VARCHAR2(9)
 , hiredate DATE
 , sal      NUMBER(7,2)
 , deptno   NUMBER(2)
 , grade    NUMBER
 , PRIMARY KEY (empno)
);

CREATE TABLE salgrade (
   grade  NUMBER
 , lowsal NUMBER
 , hisal  NUMBER
);

CREATE TABLE personeval (
   ename VARCHAR2(10)
 , eval VARCHAR2(1)
);
```

```
CREATE VIEW v AS
SELECT ename, grade FROM emp;
```

```
CREATE TABLE mv(
   ename VARCHAR2(10)
 , grade NUMBER
);
```

-> Job을 이용해 동기화

후반기 - Materalized View

테이블 정형화

After Update ON emp .. Update emp

->

After Update ON emp .. Update empgrade

```
CREATE TABLE emp (
   empno     NUMBER(4) NOT NULL
 , ename     VARCHAR2(10)
 , job       VARCHAR2(9)
 , hiredate  DATE
 , sal       NUMBER(7,2)
 , deptno    NUMBER(2)
 , grade     NUMBER
 , PRIMARY KEY (empno)
);
```

```
CREATE TABLE emp (
   empno     NUMBER(4) NOT NULL
 , ename     VARCHAR2(10)
 , job       VARCHAR2(9)
 , hiredate  DATE
 , sal       NUMBER(7,2)
 , deptno    NUMBER(2)
 , PRIMARY KEY (empno)
);

CREATE TABLE empgrade (
   ename     VARCHAR2(10)
 , grade     NUMBER
);
```

후반기 - Materalized View

테이블 정형화

```
CREATE OR REPLACE TRIGGER trg_salgrade_before
 BEFORE UPDATE ON emp
 FOR EACH ROW
DECLARE
 v_emp emp%ROWTYPE;
BEGIN
 DELETE FROM tmp;
 FOR f_salgrade IN (SELECT * FROM salgrade) LOOP
  IF :NEW.sal > f_salgrade.lowsal AND :NEW.sal < f_salgrade.hisal AND :OLD.grade != f_salgrade.grade THEN
   v_emp.empno    := :NEW.empno;
   v_emp.ename    := :NEW.ename;
   v_emp.job      := :NEW.job;
   v_emp.hiredate := :NEW.hiredate;
   v_emp.sal      := :NEW.sal;
   v_emp.deptno   := :NEW.deptno;
   v_emp.grade    := f_salgrade.grade;
   INSERT INTO tmp VALUES v_emp;
  END IF;
 END LOOP;
END;
```

```
CREATE OR REPLACE TRIGGER trg_salgrade_after
 AFTER UPDATE ON emp
BEGIN
 FOR f_tmp IN (SELECT * FROM tmp) LOOP
  UPDATE emp SET emp.grade = f_tmp.grade WHERE emp.empno = f_tmp.empno;
 END LOOP;
END;
```

```
CREATE OR REPLACE TRIGGER trg_salgrade
 AFTER UPDATE ON emp
 FOR EACH ROW
BEGIN
 FOR f_salgrade IN (SELECT * FROM salgrade) LOOP
  IF :NEW.sal > f_salgrade.lowsal AND :NEW.sal < f_salgrade.hisal AND :OLD.grade != f_salgrade.grade THEN
   UPDATE empgrade e SET e.grade = f_salgrade.grade WHERE e.ename = :NEW.ename;
  END IF;
 END LOOP;
END;
```

Tmax

후반기 - COMMA_TO_TABLE

DBMS_UTILITY.COMMA_TO_TABLE

‘A,B,C,D,E’

A	B	C	D	E

후반기 - COMMA_TO_TABLE

DBMS_UTILITY.COMMA_TO_TABLE

‘A,,C,D,E’ →

A	NULL	C	D	E

‘A,B,“,”,E’ →

A	B	“	”	E

후반기 - COMMA_TO_TABLE

INSTR과 SUBSTR 함수 이용

가장 처음에는 루프를 돌며 한 char마다 확인하여 ‘ , ’를 찾는 방법을 생각하였으나
비효율적이라고 판단

INSTR 함수를 이용해 ‘ , ’ 의 위치를 찾고,

SUBSTR 함수를 이용해 ‘ , ’ 사이에 있는 문자열을 추출 후 배열에 대입

후반기 - COMMA_TO_TABLE

INSTR과 SUBSTR 함수 이용

```
CREATE OR REPLACE PROCEDURE COMMA_TO_TABLE (
 list IN VARCHAR2,
 tablen OUT BINARY_INTEGER,
 tab OUT DBMS_UTILITY.uncl_array)
IS
 i BINARY_INTEGER := 1;
 index_start PLS_INTEGER := 1;
BEGIN
 WHILE INSTR(list, ',', 1, i) != 0 LOOP
  tab(i) := SUBSTR(list, index_start, INSTR(list, ',', 1, i) - index_start);
  index_start := INSTR(list, ',', 1, i) + 1;
  i := i + 1;
  END LOOP;
 tab(i) := SUBSTR(list, index_start);
 tablen := i;
END;
```

후반기 - COMMA_TO_TABLE

모듈화

두 가지 형태로 오버로딩, 내부 구조는 동일

프로시저로 핵심 로직을 구현해 모듈화를 함으로써 코드가 간결해짐

-> UTL_SMTP 구현에 모듈화 사용

```
DBMS_UTILITY.COMMA_TO_TABLE
(
    list            IN              VARCHAR2,
    tablen          OUT             BINARY_INTEGER,
    tab             OUT             UNCL_ARRAY
);
```

```
DBMS_UTILITY.COMMA_TO_TABLE
(
    list            IN              VARCHAR2,
    tablen          OUT             BINARY_INTEGER,
    tab             OUT             LNAME_ARRAY
);
```

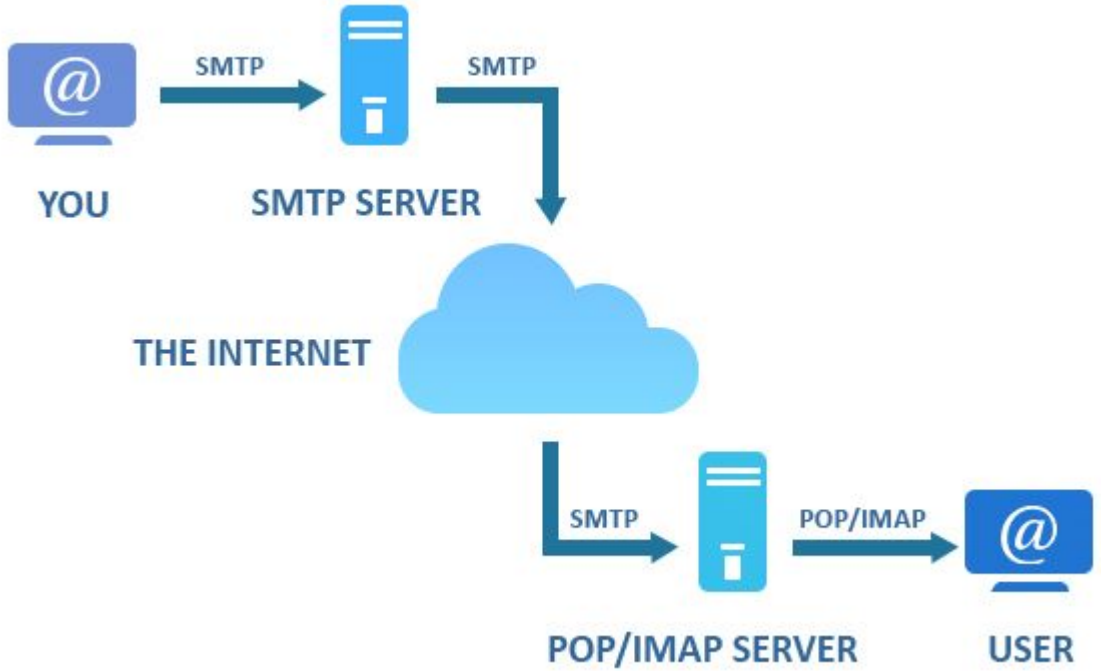
(1)

후반기 - UTL_SMTP

SMTP(Simple Mail Transfer Protocol)

인터넷 상에서 메일을 주고받기 위한 규약

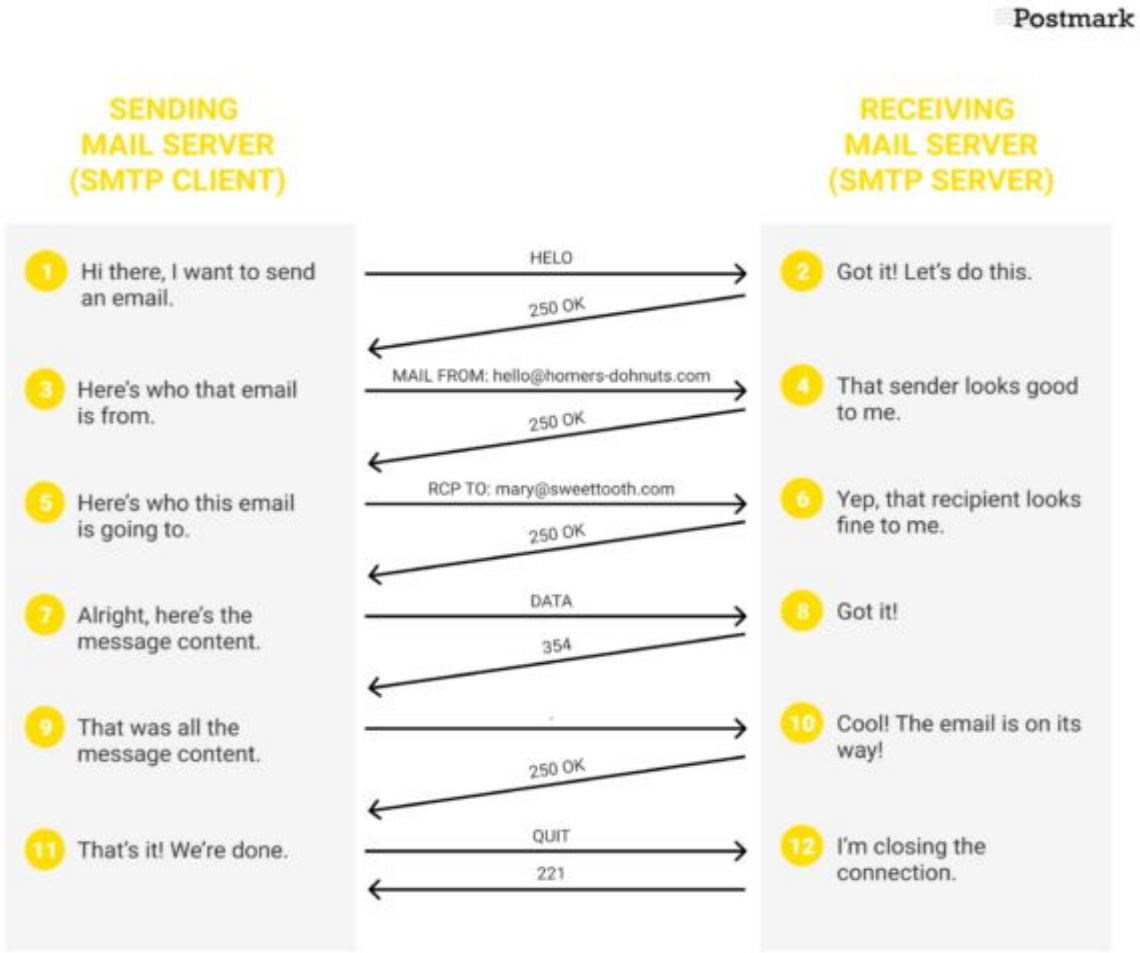
Protocol
- **TCP : 신뢰성 ↑ 속도 ↓**
- **UDP : 신뢰성 ↓ 속도 ↑**



(2)

UTL_TCP 패키지를 이용해 구현

후반기 - UTL_SMTP



핵심 COMMAND

HELO

MAIL FROM

RCPT TO

DATA

(mail body)

.

QUIT

후반기 - UTL_SMTP

UTL_SMTP

DB 상에서 SMTP를 이용해 메일을 보낼 수 있게 하는 기능을 가진 패키지

타입

```
TYPE connection IS RECORD (
    host              VARCHAR2(255),
    port              PLS_INTEGER,
    tx_timeout        PLS_INTEGER,
    private_tcp_con   utl_tcp.connection,
    private_state     PLS_INTEGER);
```

```
TYPE reply IS RECORD (
  code    PLS_INTEGER,
  text    VARCHAR2(508));

TYPE replies IS TABLE OF reply INDEX BY BINARY_INTEGER;
```

제72장 UTL_SMTP

내용 목차

Tmax

후반기 - UTL_SMTP

UTL_SMTP

DB 상에서 SMTP를 이용해 메일을 보낼 수 있게 하는 기능을 가진 패키지

필수 서브프로그램

OPEN_CONNECTION
HELO
MAIL
RCPT
OPEN_DATA
WRITE_DATA
CLOSE_DATA
QUIT

제72장 UTL_SMTP

내용 목차

Tmax

후반기 - UTL_SMTP

UTL_SMTP	SMTP
OPEN_CONNECTION	
HELO	HELO
MAIL	MAIL FROM
RCPT	RCPT TO
OPEN_DATA	DATA
WRITE_DATA	(mail body)
CLOSE_DATA	.
QUIT	QUIT

후반기 - UTL_SMTP

공통점

(대부분) SMTP 서버로 특정 COMMAND + (DATA or NULL) 형태의 한 줄을 전송

(모두)　SMTP 서버로 전송 후 서버에서 REPLY 혹은 REPLIES를 받음

WRITE_COMMAND_LINE 와 **GET_REPLY** 서브프로그램을 만들어 **모듈화**

-> 피드백을 받은 후 코드 수정이나 GET_REPLIES, GET_LAST_REPLY

서브프로그램을 만들 때 효율적으로 진행함

Tmax

후반기 - UTL_SMTP

WRITE_COMMAND_LINE

```
PROCEDURE WRITE_COMMAND_LINE(C IN OUT NOCOPY CONNECTION,
                             COMMAND IN VARCHAR2,
                             DATA IN VARCHAR2)
IS
 rc PLS_INTEGER;
 msg VARCHAR2(100);
BEGIN
 IF DATA IS NULL THEN
  msg := COMMAND || DATA;
 ELSE
  msg := COMMAND || ' ' || DATA;
 END IF;
 rc := UTL_TCP.WRITE_LINE(C.private_tcp_con, msg);
END;
```

후반기 - UTL_SMTP

GET_REPLY

```
FUNCTION GET_REPLY(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
 msg VARCHAR2(508);
 codeChar VARCHAR2(3);
 code PLS_INTEGER;
 rep reply;
BEGIN
 msg := UTL_TCP.GET_LINE(C.private_tcp_con);
 codeChar := SUBSTR(msg, 1, 3);
 code := TO_NUMBER(codeChar);
 msg := SUBSTR(msg, 5);
 rep.code := code;
 rep.text := msg;
 RETURN rep;
EXCEPTION
 WHEN UTL_TCP.END_OF_INPUT THEN
  RAISE_APPLICATION_ERROR(-20001, 'ERROR: REPLY DOES NOT EXIST');
 WHEN UTL_TCP.NETWORK_ERROR THEN
  RAISE_APPLICATION_ERROR(-20002, 'ERROR: NETWORK ERROR');
END;
```

```
PROCEDURE GET_REPLY(C IN OUT NOCOPY CONNECTION)
IS
 msg VARCHAR2(508);
BEGIN
 msg := UTL_TCP.GET_LINE(C.private_tcp_con);
EXCEPTION
 WHEN UTL_TCP.END_OF_INPUT THEN
  RAISE_APPLICATION_ERROR(-20001, 'ERROR: REPLY DOES NOT EXIST');
 WHEN UTL_TCP.NETWORK_ERROR THEN
  RAISE_APPLICATION_ERROR(-20002, 'ERROR: NETWORK ERROR');
END;
```

후반기 - UTL_SMTP

GET_REPLIES & GET_LAST_REPLY

```
PROCEDURE GET_REPLIES(C IN OUT NOCOPY CONNECTION, reps OUT REPLIES, repslen OUT BINARY_INTEGER)
IS
 i BINARY_INTEGER := 1;
 tmp VARCHAR2(508);
BEGIN
 LOOP
  tmp := UTL_TCP.GET_LINE(C.private_tcp_con, FALSE, TRUE);
  reps(i) := GET_REPLY(C);
  i := i + 1;
 END LOOP;
EXCEPTION
 WHEN OTHERS THEN
  repslen := i - 1;
END;
```

```
FUNCTION GET_LAST_REPLY(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
 reps replies;
 repslen BINARY_INTEGER;
BEGIN
 GET_REPLIES(C, reps, repslen);
 RETURN reps(repslen);
END;
```

후반기 - UTL_SMTP

OPEN_CONNECTION

SMTP 서버와 연결하는 함수

```
FUNCTION "OPEN_CONNECTION"(HOST       IN  VARCHAR2,
                           PORT       IN  PLS_INTEGER DEFAULT 25,
                           C          OUT CONNECTION,
                           TX_TIMEOUT IN  PLS_INTEGER DEFAULT NULL)
                           RETURN REPLY
IS
 c_smtp connection;
 c_tcp UTL_TCP.connection;
BEGIN
 c_tcp := UTL_TCP.OPEN_CONNECTION(HOST, PORT, TX_TIMEOUT);
 c_smtp.host := HOST;
 c_smtp.port := PORT;
 c_smtp.tx_timeout := TX_TIMEOUT;
 c_smtp.private_tcp_con := c_tcp;
 c_smtp.private_state := NULL;
 C := c_smtp;
 RETURN GET_REPLY(C);
END;
```

```
FUNCTION "OPEN_CONNECTION"(HOST       IN  VARCHAR2,
                           PORT       IN  PLS_INTEGER DEFAULT 25,
                           TX_TIMEOUT IN  PLS_INTEGER DEFAULT NULL)
                           RETURN CONNECTION
IS
 c_smtp connection;
 c_tcp UTL_TCP.connection;
BEGIN
 c_tcp := UTL_TCP.OPEN_CONNECTION(HOST, PORT, TX_TIMEOUT);
 c_smtp.host := HOST;
 c_smtp.port := PORT;
 c_smtp.tx_timeout := TX_TIMEOUT;
 c_smtp.private_tcp_con := c_tcp;
 c_smtp.private_state := NULL;
 GET_REPLY(c_smtp);
 RETURN c_smtp;
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

Tmax

후반기 - UTL_SMTP

HELO

SMTP 서버에 ‘HELO (DOMAIN)’ 전송

```
FUNCTION "HELO"(C       IN OUT NOCOPY CONNECTION,
                DOMAIN  IN            VARCHAR2) RETURN REPLY
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'HELO', DOMAIN);
 RETURN GET_LAST_REPLY(C);
END;


PROCEDURE "HELO"(C       IN OUT NOCOPY CONNECTION,
                 DOMAIN  IN            VARCHAR2)
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'HELO', DOMAIN);
 GET_REPLY(C);
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

Tmax

후반기 - UTL_SMTP

MAIL

SMTP 서버에 ‘MAIL FROM: SENDER’ 전송

```
FUNCTION "MAIL"(C          IN OUT NOCOPY CONNECTION,
                SENDER     IN            VARCHAR2,
                PARAMETERS IN            VARCHAR2 DEFAULT NULL) RETURN REPLY
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'MAIL FROM:', SENDER);
 RETURN GET_LAST_REPLY(C);
END;


PROCEDURE "MAIL"(C          IN OUT NOCOPY CONNECTION,
                 SENDER     IN            VARCHAR2,
                 PARAMETERS IN            VARCHAR2 DEFAULT NULL)
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'MAIL FROM:', SENDER);
 GET_REPLY(C);
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

후반기 - UTL_SMTP

RCPT

SMTP 서버에 ‘RCPT TO: RECIPIENT’ 전송

```
FUNCTION "RCPT"(C          IN OUT NOCOPY CONNECTION,
                RECIPIENT  IN           VARCHAR2,
                PARAMETERS IN           VARCHAR2 DEFAULT NULL) RETURN REPLY
IS
BEGIN
  WRITE_COMMAND_LINE(C, 'RCPT TO:', RECIPIENT);
 RETURN GET_LAST_REPLY(C);
END;


PROCEDURE "RCPT"(C          IN OUT NOCOPY CONNECTION,
                 RECIPIENT  IN           VARCHAR2,
                 PARAMETERS IN           VARCHAR2 DEFAULT NULL)
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'RCPT TO:', RECIPIENT);
 GET_REPLY(C);
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

Tmax

후반기 - UTL_SMTP

OPEN_DATA

SMTP 서버에 ‘DATA’ 전송, private_state 1로 변경

```
FUNCTION "OPEN_DATA"(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'DATA', NULL);
 C.private_state := 1;
 RETURN GET_LAST_REPLY(C);
END;


PROCEDURE "OPEN_DATA"(C IN OUT NOCOPY CONNECTION)
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'DATA', NULL);
 C.private_state := 1;
 GET_REPLY(C);
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

Tmax

후반기 - UTL_SMTP

WRITE_DATA

SMTP 서버에 메일 본문 DATA 전송
private_state가 1이 아니면 에러

```
PROCEDURE "WRITE_DATA"(C     IN OUT NOCOPY CONNECTION,
                       DATA  IN            VARCHAR2 )
IS
 rc PLS_INTEGER;
BEGIN
 IF C.private_state = 1 THEN
  rc := UTL_TCP.WRITE_TEXT(C.private_tcp_con, DATA);
 ELSE
  RAISE_APPLICATION_ERROR(-20000, 'ERROR: DATA NOT OPENED');
 END IF;
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

Tmax

후반기 - UTL_SMTP

CLOSE_DATA

SMPT 서버에 '.' 전송
private_state가 1이 아니면 에러

```
FUNCTION "CLOSE_DATA"(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
BEGIN
 IF C.private_state = 1 THEN
  WRITE_COMMAND_LINE(C, '.' ||  UTL_TCP.CRLF, NULL);
  C.private_state := NULL;
  RETURN GET_LAST_REPLY(C);
 ELSE
  RAISE_APPLICATION_ERROR(-20000, 'ERROR: DATA NOT OPENED');
  RETURN NULL;
 END IF;
END;
```

```
PROCEDURE "CLOSE_DATA"(C IN OUT NOCOPY CONNECTION)
IS
BEGIN
 IF C.private_state = 1 THEN
  WRITE_COMMAND_LINE(C, '.' ||  UTL_TCP.CRLF, NULL);
  C.private_state := NULL;
  GET_REPLY(C);
 ELSE
  RAISE_APPLICATION_ERROR(-20000, 'ERROR: DATA NOT OPENED');
 END IF;
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

후반기 - UTL_SMTP

QUIT

SMPT 서버에 'QUIT' 전송, REPLY 에러가 없으면 서버와 연결 종료

```
FUNCTION "QUIT"(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
 rep reply;
BEGIN
 WRITE_COMMAND_LINE(C, 'QUIT', NULL);
 rep := GET_LAST_REPLY(C);
 UTL_TCP.CLOSE_CONNECTION(C.private_tcp_con);
 RETURN rep;
END;


PROCEDURE "QUIT"(C IN OUT NOCOPY CONNECTION)
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'QUIT', NULL);
 GET_REPLY(C);
 UTL_TCP.CLOSE_CONNECTION(C.private_tcp_con);
END;
```

UTL_SMTP

OPEN_CONNECTION

HELO

MAIL

RCPT

OPEN_DATA

WRITE_DATA

CLOSE_DATA

QUIT

Tmax

후반기 - UTL_SMTP

추가 서브프로그램 1

테스트 환경과 맞지 않았던 COMMAND, COMMAND_REPLIES, HELP를 제외하고 모두 구현

```
FUNCTION "DATA"(C     IN OUT NOCOPY CONNECTION,
                "BODY"  IN           VARCHAR2 )
                RETURN REPLY
IS
 rc PLS_INTEGER;
 rep reply;
BEGIN
 OPEN_DATA(C);
 rc := UTL_TCP.WRITE_TEXT(C.private_tcp_con, "BODY" || UTL_TCP.CRLF);
 CLOSE_DATA(C);
 rep := GET_LAST_REPLY(C);
 RETURN rep;
END;
```

```
FUNCTION "EHLO"(C       IN OUT NOCOPY CONNECTION,
                DOMAIN  IN           VARCHAR2) RETURN REPLIES
IS
 reps replies;
 repslen BINARY_INTEGER;
BEGIN
 WRITE_COMMAND_LINE(C, 'EHLO', DOMAIN);
 GET_REPLIES(C, reps, repslen);
 RETURN reps;
END;
```

```
FUNCTION "NOOP"(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'NOOP', NULL);
 RETURN GET_LAST_REPLY(C);
END;
```

제72장 UTL_SMTP

내용 목차

Tmax

후반기 - UTL_SMTP

추가 서브프로그램 2

```
FUNCTION "RSET"(C IN OUT NOCOPY CONNECTION) RETURN REPLY
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'RSET', NULL);
 RETURN GET_LAST_REPLY(C);
END;
```

```
FUNCTION "VRFY"(C          IN OUT NOCOPY CONNECTION,
                RECIPIENT  IN           VARCHAR2) RETURN REPLY
IS
BEGIN
 WRITE_COMMAND_LINE(C, 'VRFY', RECIPIENT);
 RETURN GET_LAST_REPLY(C);
END;
```

```
PROCEDURE "WRITE_RAW_DATA"(C     IN OUT NOCOPY CONNECTION,
                           DATA  IN           RAW)
IS
  rc PLS_INTEGER;
BEGIN
 IF C.private_state = 1 THEN
  rc := UTL_TCP.WRITE_RAW(C.private_tcp_con, DATA);
 ELSE
  RAISE_APPLICATION_ERROR(-20000, 'ERROR: DATA NOT OPENED');
 END IF;
END;
```

제72장 UTL_SMTP

내용 목차

Tmax

후반기 - UTL_SMTP

실행 사진

```
DECLARE
    t_host    VARCHAR2(30) := 'localhost';
    t_port    NUMBER := 25;
    t_domain  VARCHAR2(30) := 'tmax.co.kr';

    t_from    VARCHAR2(50) := 'tibero@tmax.co.kr';
    t_to      VARCHAR2(50) := 'ducco705@snu.ac.kr';

    c CLONE_UTL_SMTP.connection;
BEGIN
    c := CLONE_UTL_SMTP.OPEN_CONNECTION(
        HOST => t_host,
        PORT => t_port,
        TX_TIMEOUT => 2);

    CLONE_UTL_SMTP.HELO(c, t_domain);

    CLONE_UTL_SMTP.MAIL(c, t_from);
    CLONE_UTL_SMTP.RCPT(c, t_to);

    CLONE_UTL_SMTP.OPEN_DATA(c);

    CLONE_UTL_SMTP.WRITE_DATA(c,'From: ' || '"tibero" <tibero@tmax.co.kr>' || UTL_TCP.CRLF );
    CLONE_UTL_SMTP.WRITE_DATA(c,'To: ' || '"minju" <ducco705@snu.ac.kr>' || UTL_TCP.CRLF );
    CLONE_UTL_SMTP.WRITE_DATA(c,'Subject: Test' || UTL_TCP.CRLF );
    CLONE_UTL_SMTP.WRITE_DATA(c, UTL_TCP.CRLF );
    CLONE_UTL_SMTP.WRITE_DATA(c,'THIS IS SMTP_TEST2 ' || UTL_TCP.CRLF );

    CLONE_UTL_SMTP.CLOSE_DATA(c);

    CLONE_UTL_SMTP.QUIT(c);
END;
```

소감

- 협업의 중요성을 느낌
- 유지보수 등에 있어 코드를 효율적으로 짜는 것이 큰 도움이 된다는 것을 느낌
- **TDD(Test Driven Development)**방식을 이용하고 직접 에러를 발생시키며 에러 처리의 중요성을 느낌

Tmax

사진 출처

1. Tibero7 Online Manual
2. https://www.business2community.com/email-marketing/what-is-smtp-and-how-does-it-work-part-1-02413330
3. https://postmarkapp.com/guides/everything-you-need-to-know-about-smtp
4. Tibero7 Online Manual

감사합니다