Day19(8.27)

**package** com.ict.am;

**import** java.io.IOException;

//바이트 스트림: 기계-기계

//-바이트 스트림은 1byte를 입출력 할 수 있는 스트림이다.

//-일반적으로 바이트로 구성된 파일, 즉 동영상 파일, 음악 파일을 처리하기에 적합한 스트림이다.

//바이트 스트림

**public** **class** Ex01 {

**public** Ex01() {

**try** {

// test01();

// test02();

// test03();

// test04();

test05();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** test01() **throws** IOException {

// System.in을 사용해서 입력 받기

System.***out***.print("키보드 입력>>");

**int** data = System.***in***.read(); // 정수가 리턴

// \*\*하나만 읽어오기\*\*

// read() :0~255:아스키 코드로 나옴, 읽을 것이 없으면 -1

// 입력된 문자의 아스키 코드 값이 저장된다.

// 아스키 코드(숫자, 소문자, 대문자), 한글,한문,특수문자x

System.***out***.println("data => " + (**char**) data);

System.***out***.println("숫자로 변환 =>" + ((data - 48) + 100));

}

// System.in을 이용해서 2개의 값을 읽어오기

**public** **void** test02() **throws** IOException {

**int** data1 = 0;

**int** data2 = 0;

// 버퍼: [a,'/n','/r']

System.***out***.println("data1 입력>>>");

data1 = System.***in***.read(); // 한개만 읽어온다.

System.***in***.read();

**int** ch = 0;

**while** ((ch = System.***in***.read()) != 10) {

System.***out***.print(ch + " ");

}

System.***out***.println();

System.***out***.println("data2 입력>>>");

data2 = System.***in***.read();

System.***out***.println("data1: " + data1 + " data2: " + data2);

}

// byte[]을 사용하지 않고 입력받는 방법

**public** **void** test03() **throws** IOException {

System.***out***.println("숫자 입력>>>");

**int** num = getNum();

System.***out***.println("입력 받은 숫자: " + num);

System.***out***.println("num \* 100 = " + num \* 100);

}

**private** **int** getNum() **throws** IOException {

**int** num = 0;

**int** ch = 0;

// enter가 아닐때 까지 값을 받아라

// '0'의 아스키 코드 값 : 48

**while** ((ch = System.***in***.read()) != '\n') {

**if** (ch >= '0' && ch <= '9') {

num = num \* 10 + (ch - '0');

}

}

**return** num;

}

**public** **void** test04() **throws** IOException {

// byte배열을 생성해서 입력받은 문자를 byte에 저장함

**byte**[] by = **new** **byte**[255];

System.***out***.println("문자열 입력 ");

System.***in***.read(by);

// System.out.write(by);

String data = **new** String(by).trim();

System.***out***.println(data);

System.***out***.println("출력성공");

}

**public** **void** test05() **throws** IOException {

**byte**[] b = **new** **byte**[255]; // 아스키코드(0~255)를 저장하기 위함

System.***out***.println("숫자 입력>>>>");

// 입력

System.***in***.read(b); // 입력된 값을 byte[] b에 넣는다.

// 출력

String data = **new** String(b).trim(); // byte배열 b에 담긴 것을 String으로 변경하여 저장

// 문자열을 int로 변경하는 2가지 방법

// int num = new Integer(data); //String형인 data를 Integer형으로 변경한다.

**int** num = Integer.*parseInt*(data);

System.***out***.println("num=>" + num);

System.***out***.println("num\*100=>" + num \* 100);

}

**public** **static** **void** main(String[] args) {

**new** Ex01();

}

}

**package** com.ict.am;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**public** **class** Ex02\_File {

// 파일 객체가 생성된다고 해서 실제 파일이 만들어 지는 것은 아니다.

// 경로 생성 ) C:\\Mystudy\\Mystudy\\Javastudy\\day19

// C:/Mystudy/Mystudy/Javastudy/day19

// File file01 = new File("output.txt");

**static** File *file01* = **new** File("C:\\Users\\user\\Documents\\output2.txt");

//바이트 스트림(byte)

//file읽어와서 화면에 띄우는 것

**public** **static** **void** main(String[] args) **throws** IOException {

// file을 읽어옴

FileInputStream fis = **null**;

// 출력하기 위한 OutputStream

OutputStream os = System.***out***;

**try** {

fis = **new** FileInputStream(*file01*);

**int** ch = 0;

**while**((ch=fis.read())!=-1) {

os.write(ch);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}**finally** {

os.close();

fis.close();

System.***out***.println("읽기완료!");

}

}

// 키보드로 입력 받은 값 --> 파일에 쓰기

**public** **static** **void** test01(String[] args) **throws** IOException {

// 입력받기

InputStream is = System.***in***;

// 출력하기

FileOutputStream fos = **new** FileOutputStream(*file01*);

**int** b = 0;

**while** ((b = is.read()) != -1) {

fos.write(b);

}

fos.flush();

System.***out***.println("파일쓰기완료");

}

// FileWriter-->BufferedWriter-->입력

**public** **static** **void** test03(String[] args) **throws** IOException {

String[] strArr = { "대한민국", "서울", "마포구", "동교동", "ICT인재 개발원" };

FileWriter fw = **new** FileWriter(*file01*);

BufferedWriter bw = **new** BufferedWriter(fw);

**for** (**int** i = 0; i < strArr.length; i++) {

// System.out.println(strArr[i]);

bw.write(strArr[i]);

bw.write("|"); // 실행할때마다 파일이 덮어 씌워진다.

}

bw.flush();

fw.flush();

System.***out***.println("파일 쓰기 완료!");

}

// FileReader, FileWriter 를 이용해서 file 입/출력

**public** **static** **void** test02(String[] args) **throws** IOException {

// 파일 내용이 존재한다면 파일 내용을 읽어오자

**if** (*file01*.exists()) {

System.***out***.println("파일이 존재한다.");

String path = *file01*.getAbsolutePath();

System.***out***.println("path" + path);

// FileReader를 이용해서 file 읽어오기

FileReader fr = **new** FileReader(*file01*);

**char**[] cb = **new** **char**[255];

fr.read(cb);

fr.close();

String data = **new** String(cb).trim();

System.***out***.println("data=>" + data);

}

}

**public** **static** **void** test01() **throws** IOException {

// File 클래스 : 특정 위치에 존재하는 파일이나 디렉토리를 처리하는 폴더

// 주요 생성자: File(String 경로), File(String 상위 경로, 하위 경로)

// File(File 상위경로, String 하위경로)

// 주요 메소드: createNewFile() : 파일 생성, mkdir(), mkdirs() : 디렉토리 생성

// delete(): 삭제, exists(): 존재유무, isDirectory(): 디렉토리인가?, isFile():파일이냐?

// getName(): 이름 반환, length():파일의 크기(Byte)표시; list(): 특정내용을 배열로 저장

// getAbsolutePath(): 절대주소, getCanonicalPath(): 정규화 주소, getPath(): 상대주소

// 파일이 존재하지 않으면 파일을 생성 하고

// 파일이 존재한다면 data를 파일 출력한다.

**if** (*file01*.exists()) {

System.***out***.println("파일이 존재합니다.");

System.***out***.println(*file01*.getPath()); // 상대주소

System.***out***.println(*file01*.getAbsolutePath()); // 절대주소

String data = "우리나라 대한민국 만세";

FileWriter fw = **new** FileWriter(*file01*); // file01에 쓰기

fw.write(data);

fw.close(); // 파일 닫아줘야함 .

} **else** {

System.***out***.println("파일이 존재하지 않습니다.");

**boolean** result = *file01*.createNewFile();

**if** (result) {

System.***out***.println("파일이 생성 되었습니다.");

} **else** {

System.***out***.println("파일이 생성되지 않았습니다.");

}

}

}

}

MySql

#mysql에 접속하는 명령어

mysql -uroot -p;

12345;

#데이터베이스 생성 명령어

create database test;

#데이터베이스 제거 명령어

drop database test;

#데이터 베이스 생성 명령어

use test;

#tabel 생성 명령어

create table member(

id varchar(20),

name varchar(20),

age int

);

#member table 구조를 확인하는 병령어

describe member;

#table 제거 명령어

drop table member;

#생성된 테이블에 새로운 row를 삽입하는 명령어

insert into member (id,name,age) values('Hong' , 'gildong', 20);

inser into member values('Ryu', 'jae', 25);

#table 내용을 확인하는 명령어

select \* from member;

select id, name, age

from member;

#table에서 튜플을 삭제하는 명령어

delete from member where id = 'Ryu';

#삽입된 데이터를 수정하는 명령어

update member

set id = '마앙', name = 'ryujaeyoung'

where id = '2013112111';

#personal tabale 생성

create table personal(

pno int not null primary key,

pname varchar(10) not null,

job varchar(15) not null,

manager int,

startdate date,

pay int,

bonus int,

dno int not null

);

#personal table에 data 삽입

insert into personal values (1111, 'smith', 'manager', 1001, '1990-12-17', 1000, NULL, 10);

insert into personal values (1112, 'ally', 'salesman', 1116, '1991-02-20', 1600, 500, 30);

insert into personal values (1113, 'word', 'salesman', 1116, '1992-02-24', 1450, 300, 30);

insert into personal values (1114, 'james', 'manager', 1001, '1990-04-12', 3975, NULL, 20);

insert into personal values (1001, 'bill', 'president', NULL, '1989-01-10', 7000, NULL, 10);

insert into personal values (1116, 'johnson', 'manager', 1001, '1991-05-01', 3550, NULL, 30);

insert into personal values (1118, 'martin', 'analyst', 1111, '1991-09-09', 3450, NULL, 10);

insert into personal values (1121, 'kim', 'clerk', 1114, '1990-12-08', 4000, NULL, 20);

insert into personal values (1123, 'lee', 'salesman', 1116, '1991-09-23', 12000, 0, 30);

insert into personal values (1226, 'park', 'analyst', 111, '1990-01-03', 2500, NULL, 10);

#출력하기

1) personal table의 전체 데이터를 출력하는 명령어

select \*

from personal table;

2) personal table에서 pno, pname, pay만 출력하는 명령어

select pno, pname, pay

from personal;

3) personal table에서 dno가 10을 만족하는 튜플만 출력하는 명령어

select \*

from personal

where dno=10;

#비교연산

4)dno가 20 초과인 튜플의 속성만 출력하는 명령어

select \*

from personal

where dno>20;

#논리연산(and(&&), or(||)

5)dno가 20, pno>1111 의 조건을 만족하는 속성만 출력하는 명령어

select \*

from personal

where dno=20 and pno>1111;

#between

6)dno가 20과 30사이의 데이터만 출력하는 명령어

select \* from personal where dno between 20 and 30;

#in

7)select \* from personal where dno in(10,20);

where dno = 10 or dno = 20;

where dno = 10 || dno = 20;

#부분검색

8)like

select \* from personal where job like 's%'; //job이 s로 시작하는 튜플

select \* from personal where job like '%r'; //job이 r로 끝나는 튜플

select \* from personal where dno like '3\_'; //dno가 3로 시작하고 2자릿 수

#부정 : not like, not between

\*테이블의 검색 결과 출력 관련

#order by

1) bonus를 기준으로 오름차순으로 정렬해서 personal table의 튜플을 출력하는 명령어

select \*

from personal

order by bonus asc;

#distinct

2)select dno from personal; //중복을 모두 포함하여 데이터를 가져옴

3)select distinct dno from personal; //중복된 데이터는 생략하고 가져옴

#As :명명 (생략가능)

4) select pname, pay, pay+bonus as income

from personal;

//pay+bonus의 값을 income으로 명명하여 만들어서 출력하라

5) select pname, pay, pay+bonus as 'month income'

from personal

order by 'month income';

6) select pname, pay, startdate

from personal

order by 2;

// pname, pay, startdate를 두번째 속성(pay)로 오름차순 시켜서 출력하라

#표현식을 이용한 값   
ex) select pname, pay, pay+bonus from personal;  
참고) null+값 = null

ex) select pname, pay, pay + ifnull(bonus,0) from personal;

#컬럼 함수(sum, avg, min, max, cont)

\*\*null데이터는 제외됨

\*\*count(where 조건을 충족하는 테이블의 건수)는 null을 포함해서 count한다.

7)select sum(pay), avg(pay), min(pay), max(pay), count(\*) from personal;

#group by

having 그룹 조건

order by 필드명 (asc 또는 desc)

8)select dno, sum(pay) from personal;

9)select dno, sum(pay) from personal

group by dno;

//dno로 그룹 지어서 pay의 합을 출력하는 명령어

10) select dno, sum(pay) from personal

where pay>1500

group by dno;

//pay가 1500 초과인 사람들만 dno로 그룹지어서 그 합을 구하는 명령어

11) select dno, sum(pay) from personal

-> where pay>1500

-> group by dno

-> having sum(pay)>15000;

//pay가 1500초과인 사람만 dno로 그룹화하여 합을 구하고

//그 합 중 sum(pay)가 15000초과인 그룹만 출력하는 명령어

#scalar 함수: 단일 값 추출

[1] select substr(job, 1,3)

from personal;

// (속성 명, 시작 점, 끝 점)

//job에서 시작점과 끝점을 추출해서 출력하는 명령어

[2] 문자열 결합하기

select concat(pname, ‘job is’ , job) from personal;

//출력 결과: pname job is job 으로 출력된다

+--------------------------------+

| concat(pname, ' job is ', job) |

+--------------------------------+

| bill job is president |

| smith job is manager |

| ally job is salesman |

| word job is salesman |

| james job is manager |

| johnson job is manager |

| martin job is analyst |

| kim job is clerk |

| lee job is salesman |

| park job is analyst |

+--------------------------------+

비교) select pno, ' job is ', job from persona;

+------+----------+-----------+

| pno | job is | job |

+------+----------+-----------+

| 1001 | job is | president |

| 1111 | job is | manager |

| 1112 | job is | salesman |

| 1113 | job is | salesman |

| 1114 | job is | manager |

| 1116 | job is | manager |

| 1118 | job is | analyst |

| 1121 | job is | clerk |

| 1123 | job is | salesman |

| 1226 | job is | analyst |

+------+----------+-----------+

[3] round(3.141592, 3)

truncate(3.141592, 3)

round(3.141592, 3)

truncate(3.141592, -3)

[4] 치환 replace

select replace(job, 'manager','man') from personal;

// job이 manager 🡪 man으로 대체하는 명령어

[5]날짜

//요일

select dayname(startdate) from personal;

//현재 날짜 – 일하기 시작한 날짜 == 일한 일수

Select to\_days(now())-to\_days(startdate) from personal;

#null값 추출

select pname from personal where bonus is null;

#Limit

select pname from personal limit 5;

//limit : 5명

\*\*연습 문제

1.업무가 clerk가 아니고 manager이 아닌 사원

select pname

from personal

where job != 'manager' and job !='clerk';

+--------+

| pname |

+--------+

| bill |

| ally |

| word |

| martin |

| lee |

| park |

+--------+

2.입사일이 1991년 이후 사원 중 급여가 2000이하인 사원

select \*

from personal

where startdate>’1991-01-01’ and pay<=2000

+------+-------+----------+---------+------------+------+-------+-----+

| pno | pname | job | manager | startdate | pay | bonus | dno |

+------+-------+----------+---------+------------+------+-------+-----+

| 1112 | ally | salesman | 1116 | 1991-02-20 | 1600 | 500 | 30 |

| 1113 | word | salesman | 1116 | 1992-02-24 | 1450 | 300 | 30 |

+------+-------+----------+---------+------------+------+-------+-----+

3.업무가 prisident이거나 saleman인 사원

select \*

from personal

where job ='president' or job = 'salesman';

+------+-------+-----------+---------+------------+-------+-------+-----+

| pno | pname | job | manager | startdate | pay | bonus | dno |

+------+-------+-----------+---------+------------+-------+-------+-----+

| 1001 | bill | president | NULL | 1989-01-10 | 7000 | NULL | 10 |

| 1112 | ally | salesman | 1116 | 1991-02-20 | 1600 | 500 | 30 |

| 1113 | word | salesman | 1116 | 1992-02-24 | 1450 | 300 | 30 |

| 1123 | lee | salesman | 1116 | 1991-09-23 | 12000 | 0 | 30 |

+------+-------+-----------+---------+------------+-------+-------+-----+

4.사원이름 첫 글자가 a부터 k까지인 사원을 이름순으로 정렬

select \* from personal where substr(pname,1,1)>='a' && substr(pname,1,1)<='k';

+------+---------+-----------+---------+------------+------+-------+-----+

| pno | pname | job | manager | startdate | pay | bonus | dno |

+------+---------+-----------+---------+------------+------+-------+-----+

| 1001 | bill | president | NULL | 1989-01-10 | 7000 | NULL | 10 |

| 1112 | ally | salesman | 1116 | 1991-02-20 | 1600 | 500 | 30 |

| 1114 | james | manager | 1001 | 1990-04-12 | 3975 | NULL | 20 |

| 1116 | johnson | manager | 1001 | 1991-05-01 | 3550 | NULL | 30 |

| 1121 | kim | clerk | 1114 | 1990-12-08 | 4000 | NULL | 20 |

+------+---------+-----------+---------+------------+------+-------+-----+

#논리함수

[1] if(논리식, 논리식이 참일 때 값, 논리식이 거짓일 때 값)

select pno, pname, pay, if(pay>=1500, 'good','poor') as result from personal;

[2]ifnull(값1, 값2) :해당 필드가 null일 경우 대체해라

select pno, pname, pay, ifnull(bonus, 100) from personal;

//bonus가 null이면 100을 넣어라

\*\*select 문을 이용해서 값을 변경하더라도 실제 database의 값은 변경되지 않는다.