**로고학습일지**

**kt ds University 자바 기반의 데이터 사이언티스트 양성과정**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학습일시 | 2018. 07. 31 (화) | 장소 | kt ds University B관 201호 | **시 간** | 09:00~18:00 |
| 학습범위 | UI/UX | | | | |
| 작 성 자 |  | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 |  |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| • **예외처리 :** Java program이 예외(exception)를 JVM에게 전달하지 못하도록 하기 위해서   1. IOExcetion (input,output) : 보통 파일을 핸들링 할 때, 발생하는 exception 2. NullPointerException : Null을 참조할 때 3. NumberFormatException : 문자를 숫자로 바꾸려고 할 때 발생하는 exception 4. IndexOutofBoundsException : ArrayList나 List에 잘못된 범위(인덱스)를 넣었을 경우 5. ArrayIndexOutofBoundsException : 배열에서 잘못된 범위(인덱스)를 줬을 때   현재 발생한 예외가 Exception을 직접 받은 상속이냐? RuntimeException을 직접 받은 상속이  냐?로 체크해서 해결한다!  예외 사후처리는 복구가능(RuntimeException)과 복구불가능(Exception)이 있다.  - **복구가능(RuntimeException):** 코드 수정이 가능하다! Try-Catch사용하지말기! (왜? 느림)  NullPointerException, NumberFormatException,  IndexOutofBoundsException, NumberFormatException  **Ex)** class NullPointerException extends RuntimeException  **→** "RuntimeException이네" 하고 JVM이 더 이상 위로 타고 올라가지 않음! 끝~  (위로 올라가보면 RuntimeException또한 Exception 상속을 받고있음)  - **복구불가능(Exception):** 에러를 떠넘긴다(throws) or  본인이 책임진다(본인이 예외를 처리 try-catch)  IOExcetion  **Ex)** public FileReader(File file) throws FileNotFoundException  public class FileNotFoundException extends IOException  class IOException extends Exception |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| • **RDB :** 관계형데이타베이스 ⇒ Table : 데이터가 저장될 형태  **Primary key = pk = 기본키:** 한 Low를 다른 Low와 구별해준다. 그래서 값이 중복, Null  이면 안된다.(무결성원칙) 또한 업데이트도 안된다.  특정행을 참조하기 위해서도 사용  Pk는 자동정렬기능이 있음   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **ID** | **이름** | **연락처** | **성별** | **주소** | **연령** | **비번** | | Kd | 길동 | 010-1234 | 남 | 서울 | 60 | 1234 | | Di | 둘리 | 010-5678 | 남 | 서울 | 26 | 5678 |   **Low :** 하나의 데이터(행보다는 Low라고 말하는게 좋음)  외래키(=foreign key): primary key의 참조를 받는 키  멀티키: 주키(기본키)가 여러개일때 통칭하는거  • **SQL**이란?**:** C R U D (만들고 조회하고 수정하고 지우고)  insert into, selsect, update set, delete  **1. SELECT**  - DISTINCT 컬럼명  - ORDER BY 컬럼명(될 수 있으면 사용X, 그럼 언제 사용? PK로도 안될 때)  - WHERE 컬럼명 |

|  |  |
| --- | --- |
| 특이사항 |  |

■ **NullPointerExceptionTest.java**

**package** com.ktds;

**import** java.util.ArrayList;

- **NullPointerException == Reference Type에서만 발생!**

Primitive Type에서는 발생하지않는다.

Reference Type의 값이 Null을 가질 때

발생 가능성 높음

**import** java.util.List;

**public** **class** NullPointerExceptionTest {

**private** String message; // null

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

**new** NullPointerExceptionTest().run();

**new** NullPointerExceptionTest().runList();

}

**public** **void** **run( ) {**

String name = get(); // String name = null;과 같음

System.***out***.println( name ); // 이거는 에러가 아님

**if** ( name != **null** ) {

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

}

- if문을 안 쓰고 System.***out***.println( name.length() ); 했을 시,

NullPointerException발생 **⇒** 예외가 발생한 지점부터 프로그램 종료함.

- 어느부분에서 발생해??

null.상수/ null.변수 / null.메소드와 같이 **.**뭐뭐뭐 이렇게 쓰느곳에서

발생함! **.**을 기준으로 왼쪽을 보면 됨

System.***out***.println( message ); // 이거는 에러가 아님

**if**(message != **null**) {

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

}

- 멤버변수의 해결방법은 if문을 쓰던가, 생성자를 만들어서 초기화를 하던가~

**}**

**public** **void** **runList( ) {**

List<String> nameList = **new** ArrayList<String>();

nameList.add("MCJ");

nameList.add("BKL");

nameList.add(**null**); // = nameList.add( get() );과 같음

**for** ( String name : nameList ) {

// System.out.println( name ); → 에러 안남

// System.out.println( name.indexOf("C") ); → 에러 남

**if** ( name != **null** ) {

System.***out***.println( name.indexOf("C") );

}

}

List<String> nameList = null;

for ( String name : nameList ) {

// .이 없어도 에러가남! 어쨌든 nameList.get해서 가지고 오는건데 Null이므로

}

}

**public** String get() {

**return** **null**;

}

**}**

■ **NumberFormatExceptionTest.java**

**package** com.ktds;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** NumberFormatExceptionTest {

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

**new** NumberFormatExceptionTest().run();

}

**public** **void** run() {

String numberStr = "10.0";

Pattern p = Pattern.*compile*("[0-9]+"); // [0-9]까지 하나씩 계속 반복?

Matcher m = p.matcher(numberStr); // 위에 있는 패턴과 numberStr이 같은지 비교

**int** number = 0;

**if**( m.matches() ) {

number = Integer.*parseInt*(numberStr);

}

**else** {

System.***out***.println("숫자 형식이 아닙니다.");

}

// 문자 ==> 숫자(Integer): int number = Integer.parseInt(numberStr);

// NumberFormatException발생! **.**때문에 (왜? 10.0이라고 되어있어서)

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

}

}

■ **IndexOutofBoundsExceptionTest.java**

**package** com.ktds;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** IndexOutofBoundsExceptionTest {

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

**new** IndexOutofBoundsExceptionTest().run();

}

**public** **void** run() {

List<String> nameList = **new** ArrayList<String>();

System.***out***.println(nameList.get(0)); // IndexOutofBoundsException발생!!

// 이런경우, for-each를 쓰면 에러가 절대 안나지만 for()을 쓰면 에러남!

// → i **<** nameList.size()하면 에러 안남

**for**(**int** i=0; i <**=** nameList.size(); i++) {

System.***out***.println(nameList.get(i));

}

**for** ( String name : nameList) { // 에러 안남

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

}

}

}

■ **IOExceptionTest.java**

**package** com.ktds;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** IOExceptionTest {

// 3.책임 회피하는경우: throws FileNotFoundException {

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

**new** IOExceptionTest().run();

}

**public** **void** run () { // 2.책임 회피하는경우: throws FileNotFoundException {

File file = **new** File("D:/abc.txt");

FileReader fr = **null**;

**try** {

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

}

**catch**( FileNotFoundException e ) { // 1-1.책임지는경우: 던져주는걸 잡음

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

}

**if** ( fr != **null** ) {

// 1. 책임 회피하는경우: new BufferedReader( new FileReader(file) );

BufferedReader br = **new** BufferedReader( fr );

String line = **null**;

**try** {

**while** ( (line = br.readLine()) != **null** ) {

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

}

}

**catch** ( IOException e ) { // 2-1. 책임지는 경우:

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

}

}

}

}

■ **ParseInt.java**

**package** com.ktds;

**public** **class** ParseInt {

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

**new** ParseInt().run();

}

**public** **void** run() {

String ageStr = "45.";

**int** age = 0;

**try** {

age = Integer.*parseInt*(ageStr);

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

}

**catch** ( NumberFormatException e )

{

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

}

}

}

**JAVA랑 DB연동하기**

■ **HRDao.java** (Dao란? Data access object를 말한다.)

**package** com.ktds;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**public** **class** HRDao {

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

**new** HRDao().run();

}

**public** **void** run() { //DB에 접근해 EMPLOYEES테이블의 정보를 출력한다.

// 1.DB에 접근(HR)

// 오라클에 접근하기위한 객체 Load

**try** {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

} **catch** (ClassNotFoundException e) {

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

}

// 오라클(HR)접속

Connection conn = **null**;

PreparedStatement pstmt = **null**;

ResultSet rs = **null**;

**try** {

conn = DriverManager.*getConnection*

("jdbc:oracle:thin:@localhost:1521:XE", "HR", "hr");

// 2.Query 준비 및 실행 (쿼리를 만들어서 DB에게)

pstmt = conn.prepareStatement("SELECT \* FROM EMPLOYEES");

// 3.Query 결과 (DB에서 JAVA로)

rs = pstmt.executeQuery();

// 결과출력

**while** ( rs.next() ) {

**int** employeeId = rs.getInt("EMPLOYEE\_ID");

String firstName = rs.getString("FIRST\_NAME");

String lastName = rs.getString("LAST\_NAME");

String hireDate = rs.getString("HIRE\_DATE");

**double** commissionPct = rs.getDouble("COMMISSION\_PCT");

System.***out***.printf("%d, %s, %s, %s, %f\n",

employeeId, firstName, lastName, hireDate, commissionPct);

}

}

**catch** (SQLException e) {

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

}

**finally** { // 아래에 있는 3개는 try-catch문을 합치면 안된다!

**if** ( rs != **null** ) {

**try** {

rs.close();

} **catch** (SQLException e) {}

}

**if** ( pstmt != **null** ) {

**try** {

pstmt.close();

} **catch** (SQLException e) {}

}

**if** ( conn != **null** ) { // 커넥션이 연결을 하고 있다면

**try** {

conn.close();

} **catch** (SQLException e) {}

}

}

}

**- finally를 쓴 이유는?**

일단, finally의 역할은 블록에는 예외가 발생하거나 안 하거나 반드시 실행되어

야 할 로직이들어간다. 메모리를 잡아먹지 않기위해서는 연결했던걸 닫아줘야한

다. 만약 finally를 사용하지 않았다고 가정하면, pstmt에 쿼리를 DB에 보낸 반환값이

들어있어야하는데 오류가 나서 catch문이 실행되어버렸다. 그러면 나머지 아래

에 있는 코드는 실행이 안되고 사용했던걸 닫아줘야하는데 못 닫게 된다. (메모리차지)

}

**DataBase**

**SELECT** FIRST\_NAME

**,** LAST\_NAME

**FROM** EMPLOYEES

**;**

SELECT **\***

FROM EMPLOYEES

;

SELECT CITY

FROM LOCATIONS

;

SELECT DEPARTMENT\_NAME

FROM DEPARTMENTS

;

SELECT COUNTRY\_ID

, COUNTRY\_NAME

FROM COUNTRIES

;

SELECT **DISTINCT** MANAGER\_ID

**DISTINCT:** MANAGER\_ID와 FIRST\_NAME이 똑같은 값을 가지고 있으면 그 LOW 중복제거

**,** FIRST\_NAME

FROM EMPLOYEES

**ORDER BY** MANAGER\_ID **DESC**

**ORDER BY:** 데이터정렬해줌. 기본값은 오름차순(ASC)

내림차순은 DESC

;

/\* EMPLOYEES 테이블의 모든 컬럼을 조회한다.

EMPLOYEE\_ID 기준 오름차순으로 정렬한다.\*/

SELECT \*

FROM EMPLOYEES

ORDER BY EMPLOYEE\_ID

;

/\* EMPLOYEES테이블에서 모든 컬럼을 조회한다.

단, SALARY가 가장 높은 ROW부터 출력한다.\*/

SELECT \*

FROM EMPLOYEES

ORDER BY SALARY DESC

;

/\* 입사일: HIRE\_DATE

먼저 입사한 순서부터 모든 사원들의 모든 정보를 출력한다.\*/

SELECT \*

FROM EMPLOYEES

ORDER BY HIRE\_DATE

;

SELECT \*

FROM EMPLOYEES

ORDER BY HIRE\_DATE

, SALARY DESC

;

SELECT FIRST\_NAME

FROM EMPLOYEES

ORDER BY FIRST\_NAME

;

SELECT \*

FROM EMPLOYEES

**WHERE** SALARY = 24000

**AND** EMPLOYEE\_ID = 100

;

SELECT \*

FROM EMPLOYEES

WHERE FIRST\_NAME = **'**Steven**'**

AND SALARY = 2200

;

**오라클**

오라클사이트에 접속해서[Database 11g Express Edition](https://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html)을 다운받는다. **압출만 풀기!!!!**

**Toad For Oracle**

구글에서 Toad For Oracle 검색 후, 제일 첫번째에 있는 사이트클릭 후,

메뉴에서 다운로드 누른 후 3개중에서 맨 왼쪽 select를 누른다.

**톰캣**

[https://tomcat.apache.org/download-80.cgi에서 8.0.53](https://tomcat.apache.org/download-80.cgi에서%208.0.53)을 다운받는다. **압출만 풀기!!!!**

**설치하기**

1. 아파치 톰캣 실행

D:\apache-tomcat-8.0.49\apache-tomcat-8.0.49\bin 에서 startup클릭 후, **실행 창 끄지 말기!**

1. 오라클 설치

D:\OracleXE112\_Win64\DISK1에서 setup클릭

TNS Port 오라클로 접속하기 위한 포트 (수정안함)

MTS Port ??? (수정안함)

HTTP Port <- 이거를 바꾸기 위해서 톰캣창 안 닫음. **9999로 바꾸기**

1. Toad For Oracle 설치하기
2. 톰캣 창 닫기~
3. Toad For Oracle실행 SYSTEM, 비번 입력!!

Schema browser 에서 두번째 선택창에서Users선택 후

HR 우측마우스클릭후 언클락

다시 우측마우스클릭후 Alter 선택한 후

https://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html