Week3 Review

* Java - Collection, 객체, Class 개념 종합 실습
* SQL - 데이터베이스 설계(1:1, 1:N 테이블)
* 서버

1. **Java**

실습 1. 주식 이름, 해당 주식에 대한 시세정보, 외국인비율, 기관비율, 개인비율 정보를 key로, 해당 정보들을 value로 가지고 있는 맵에서 각 종목마다 시세정보, 외국인비율, 기관비율, 개인비율을 가지는 새로운 맵을 만들어보기

{indiInfo=[7.53, 0.52, 15.75, 11.22], frgnInfo=[11.51, 20.32, 33.13, 76.91], orgInfo=[11.51, 57.22, 21.7, 45.52], stockNameInfo=[백광소재, 차이나하오란, 삼성전자, 팍스넷], siseInfo=[1000, 2050, 1330, 250]}

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.List;

**public** **class** Main {

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

DataMap map = **new** DataMap();

HashMap<String, Object> infoMap = (HashMap<String, Object>) map.getMap();

List <String> stock = (List<String>) infoMap.get("stockNameInfo");

List <String> indiInfo = (List<String>) infoMap.get("indiInfo");

List <String> frgnInfo = (List<String>) infoMap.get("frgnInfo");

List <String> orgInfo = (List<String>) infoMap.get("orgInfo");

List <String> siseInfo = (List<String>) infoMap.get("siseInfo");

HashMap <String, List> stockInfo = **new** HashMap<>();

**for** (String key : infoMap.keySet()) {

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

List list = **new** ArrayList();

list.add(indiInfo.get(i));

list.add(frgnInfo.get(i));

list.add(orgInfo.get(i));

list.add(siseInfo.get(i));

stockInfo.put(stock.get(i), list);

}

}

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

* {백광소재=[7.53, 11.51, 11.51, 1000], 삼성전자=[15.75, 33.13, 21.7, 1330], 팍스넷=[11.22, 76.91, 45.52, 250], 차이나하오란=[0.52, 20.32, 57.22, 2050]

실습 2. 가격이 1330인 종목의 정보들을 출력해보시오

**for** (String key: stockInfo.keySet()) {

**if** ((stockInfo.get(key)).contains(1330)) {

System.***out***.println(key + stockInfo.get(key));

* 삼성전자[15.75, 33.13, 21.7, 1330]

}

}

}

}

실습 3. DataMap map = new DataMap();을 해서 map.getMap()을 하면 맵이 리턴이 되는데, 이 안에는 map1 , map2 라는 맵이 들어있음. 이 각각의 맵에는 stockType이 kosdaq(코스닥) , kospi(코스피) 를 나누어져 들어가 있다. 각각의 코스닥, 코스피 종목들을 종목명, 가격, 보유 비율 등으로 재정렬하여 출력한 후, 삼성전자 종목 정보를 출력해보시오.

**public** **class** Main4\_data2 {

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

DataMap map = **new** DataMap();

HashMap<String, Object> infoMap = (HashMap<String, Object>) map.getMap();

HashMap<String, Object>map1 = (HashMap<String, Object>) infoMap.get("map1");

HashMap<String, Object>map1\_1 = (HashMap<String, Object>) map1.get("stockInfo");

List <String> stockName = (List) map1\_1.get("stockNameInfo");

HashMap<String, Object>map2 = (HashMap<String, Object>) infoMap.get("map2");

HashMap<String, Object>map2\_1 = (HashMap<String, Object>) map2.get("stockInfo");

List <String> stockName2 = (List) map2\_1.get("stockNameInfo");

HashMap<String, Object> resMap1 = **new** HashMap<>();

HashMap<String, Object> resMap2 = **new** HashMap<>();

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

List list = **new** ArrayList<>();

list.add(((List) map1\_1.get("siseInfo")).get(i));

list.add(((List) map1\_1.get("frgnInfo")).get(i));

list.add(((List) map1\_1.get("orgInfo")).get(i));

list.add(((List) map1\_1.get("indiInfo")).get(i));

resMap1.put(stockName.get(i), list);

}

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

**if** (stockName.get(i) == "삼성전자") {

idx = i;

**break**;

}

}

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

List list = **new** ArrayList<>();

list.add(((List) map2\_1.get("siseInfo")).get(i));

list.add(((List) map2\_1.get("frgnInfo")).get(i));

list.add(((List) map2\_1.get("orgInfo")).get(i));

list.add(((List) map2\_1.get("indiInfo")).get(i));

resMap2.put(stockName.get(i), list);

}

System.***out***.println(stockName.get(idx) + resMap1.get("삼성전자"));

* 삼성전자[1330, 33.13, 21.7, 15.75]

System.***out***.println(resMap1.toString());

* {차이나하오란=[2050, 20.32, 57.22, 1.12], 샘코=[2055, 21.32, 7.22, 3.52], 유비쿼스=[10000, 80.13, 28.7, 15.21], 흥구석유=[30000, 99.91, 45.52, 11.25], 삼성전자=[1330, 33.13, 21.7, 15.75], SG세계물산=[2510, 76.91, 45.5, 15.22], 백광소재=[1000, 11.51, 11.51, 7.53], 팍스넷=[5220, 55.91, 41.55, 19.22], 한창제지=[1530, 11.51, 11.51, 7.53], 코닉글로리=[2050, 21.32, 57.22, 1.12], 이원컴포텍=[1330, 3.13, 21.7, 15.75], SG&G=[1380, 3.13, 25.72, 18.75], 우정바이오=[100050, 99.91, 45.52, 11.25], 디알텍=[1000, 11.51, 11.51, 7.53], 휴네시온=[2252, 3.13, 22.76, 11.75], 코미팜=[10000, 12.51, 52.53, 80.53], 줌인터넷=[8400, 27.51, 16.51, 11.53], 지어소프트=[5220, 27.51, 3.22, 0.48], 한창=[60030, 80.13, 28.7, 15.21], 이엠코리아=[10050, 20.32, 3.22, 0.48], 이스트소프트=[30000, 20.32, 50.22, 42.51], 현대바이오=[20, 76.91, 43.52, 11.22]}

}

}

실습4. 코스피, 코스닥별로 전체 종목을 10개씩 나눠서 출력하기

**public** **class** Main4\_data2 {

**public** **static** **void** printStocks(HashMap stockType) {

List stockNames = **new** ArrayList(stockType.keySet());

Integer idx = 0;

Integer newIdx = 0;

Integer numsPage = stockNames.size() / 10 + 1;;

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

**for** (**int** i = 0; i < numsPage;i++) {

**for** (**int** j = idx; j < idx + 10; j++) {

System.***out***.println(stockNames.get(newIdx) +": " + stockType.get(stockNames.get(newIdx)));

newIdx++;

**if** (newIdx == stockNames.size()) {

**break**;

}

}

System.***out***.println("--------------End of Page--------------");

idx = newIdx;

}

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

}

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

DataMap map = **new** DataMap();

HashMap<String, Object> infoMap = (HashMap<String, Object>) map.getMap();

HashMap<String, Object>map1 = (HashMap<String, Object>) infoMap.get("map1");

HashMap<String, Object>map1\_1 = (HashMap<String, Object>) map1.get("stockInfo");

.

.

.

System.***out***.println("코스피종목 ");

*printStocks*(resMap1);

System.***out***.println("코스닥종목 ");

*printStocks*(resMap2);

}}

* 코스피종목

------------------------------------

차이나하오란: [2050, 20.32, 57.22, 1.12]

샘코: [2055, 21.32, 7.22, 3.52]

유비쿼스: [10000, 80.13, 28.7, 15.21]

흥구석유: [30000, 99.91, 45.52, 11.25]

삼성전자: [1330, 33.13, 21.7, 15.75]

SG세계물산: [2510, 76.91, 45.5, 15.22]

백광소재: [1000, 11.51, 11.51, 7.53]

팍스넷: [5220, 55.91, 41.55, 19.22]

한창제지: [1530, 11.51, 11.51, 7.53]

코닉글로리: [2050, 21.32, 57.22, 1.12]

--------------End of Page--------------

이원컴포텍: [1330, 3.13, 21.7, 15.75]

SG&G: [1380, 3.13, 25.72, 18.75]

우정바이오: [100050, 99.91, 45.52, 11.25]

디알텍: [1000, 11.51, 11.51, 7.53]

휴네시온: [2252, 3.13, 22.76, 11.75]

코미팜: [10000, 12.51, 52.53, 80.53]

줌인터넷: [8400, 27.51, 16.51, 11.53]

지어소프트: [5220, 27.51, 3.22, 0.48]

한창: [60030, 80.13, 28.7, 15.21]

이엠코리아: [10050, 20.32, 3.22, 0.48]

--------------End of Page--------------

이스트소프트: [30000, 20.32, 50.22, 42.51]

현대바이오: [20, 76.91, 43.52, 11.22]

--------------End of Page--------------

실습5. searchStockVo, searchStockVO2 클래스 객체로 만들어진 dataMap3에서 객체를 꺼내 각 종목별로 정보를 추출하고, 전일대비값이 가장 큰 종목과 전일대비비율을 출력해보시오.

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**public** **class** Main {

**public** **static** **double** findMax(Map <String, Double> a) {

**double** max = 0;

**for** (String b : a.keySet()) {

**if** (a.get(b) > max) {

max = a.get(b);

}

}

**return** (**double**) max;

}

**public** **static** **double** findMin(Map <String, Double> a) {

**double** min = 0;

**for** (String b : a.keySet()) {

**if** (a.get(b) < min) {

min = a.get(b);

}

}

**return** (**double**) min;

}

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

DataMap3 dm = **new** DataMap3();

HashMap<String, Object>map = dm.getMap();

List <HashMap> siseList = (List<HashMap>) map.get("siseList");

List <searchStockVO2> vo2 = **new** ArrayList<>();

Map <String, String> nameCode = **new** HashMap<>();

List <searchStockVO> vo = **new** ArrayList<>();

**for** (HashMap s: siseList) {

List <String> keyset = **new** ArrayList(((HashMap<String, Object>) s).keySet());

**for** (String k : keyset) {

String values = (s.get(k)).toString();

**if**(values.substring(0,14).equals("searchStockVO2")){

vo2.add((searchStockVO2)(s.get(k)));

} **else** {

vo.add((searchStockVO)(s.get(k)));

}

}

}

**for** (Object s : vo2) { // 종목명, 종목코드

s = (searchStockVO2) s;

nameCode.put(((searchStockVO2) s).getStockName(), ((searchStockVO2) s).getStockCode());

}

Map <String, Double> dayPriceGaps = **new** HashMap<>();

**for** (Object s : vo) {

// symbol, 종목코드, 현재가, 전일가, 상한가, 하한가, 외국인비율, 기관비율, 개인비율

s = (searchStockVO) s;

**for** (String i : nameCode.keySet()) {

**if** (((searchStockVO) s).getStockCode() == (nameCode.get(i))) {

**int** closePrice = ((searchStockVO) s).getClosePrice();

**int** ydayPrice = ((searchStockVO) s).getYdayPrice();

**int** dayPriceGap = closePrice - ydayPrice;

**double** dayPriceDifPercent = Math.*round*(((**double**) dayPriceGap / closePrice) \* 100 \* 100)/100.0;

((Map <String, Double>) dayPriceGaps).put(i, (**double**) dayPriceDifPercent);

System.***out***.println(i + ((searchStockVO) s).toString());

}

}

}

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

**for** (String s: dayPriceGaps.keySet()) {

**if** (dayPriceGaps.get(s) == *findMax*(dayPriceGaps)) {

System.***out***.println("상승폭이 가장 큰 종목: " + s + " " +*findMax*(dayPriceGaps) + "%" );

} **else** **if** (dayPriceGaps.get(s) == *findMin*(dayPriceGaps)) {

System.***out***.println("하락폭이 가장 큰 종목: " + s + " " +*findMin*(dayPriceGaps) + "%" );

}

}

}

}

* 결과

차이나하오란 [symbol=HK2000120302, stockCode=900090, closePrice=5000, ydayPrice=4750, diffPrice=0, upLimit=6500, downLimit=3500, diffRate=0.0, foreignRate=0.1, orgRate=21.35, indiRate=30.22]

sk하이닉스 [symbol=HK1004210421, stockCode=000660, closePrice=51000, ydayPrice=40500, diffPrice=0, upLimit=50000, downLimit=20000, diffRate=0.0, foreignRate=1.99, orgRate=50.77, indiRate=40.21]

동성제약 [symbol=HK0004242132, stockCode=002210, closePrice=2200, ydayPrice=4200, diffPrice=0, upLimit=5500, downLimit=1300, diffRate=0.0, foreignRate=1.0, orgRate=90.0, indiRate=33.01]

지어소프트 [symbol=HK0004242001, stockCode=051160, closePrice=8950, ydayPrice=8090, diffPrice=0, upLimit=10100, downLimit=5670, diffRate=0.0, foreignRate=5.01, orgRate=1.1, indiRate=21.11]

에이치엔티 [symbol=HK0004242123, stockCode=176440, closePrice=1111, ydayPrice=2142, diffPrice=0, upLimit=3010, downLimit=500, diffRate=0.0, foreignRate=51.01, orgRate=50.1, indiRate=1.19]

이스트소프트 [symbol=HK1004242124, stockCode=047560, closePrice=11150, ydayPrice=8610, diffPrice=0, upLimit=11150, downLimit=6030, diffRate=0.0, foreignRate=21.71, orgRate=30.12, indiRate=11.19]

하락폭이 가장 큰 종목: 에이치엔티 -92.8%

상승폭이 가장 큰 종목: 이스트소프트 22.78%

1. **SQL**

실습1. 1:1 관계인 테이블 설계 후 생성 및 데이터 추가

create table economicsStudent (

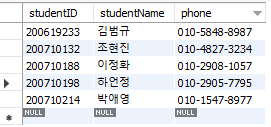
studentID integer(9) not null,

studentName varchar(10),

phone varchar(15),

primary key(studentID)

);



create table studentScore (

studentID integer(9) not null,

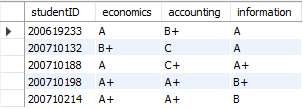
economics varchar(5),

accounting varchar(5),

information varchar(5),

primary key (studentID)

);



실습 1-1. 1 : 1 관계인 테이블끼리 조인해서 데이터 출력

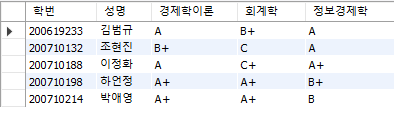
select

s.studentid "학번", s.studentname "성명", sc.economics "경제학이론",

sc.accounting "회계학", sc.information "정보경제학"

from economicsStudent s, studentscore sc

where (s.studentID = sc.studentID);



실습2. 1:N 관계인 테이블 설계 후 생성 및 데이터 추가

create table mandatorySubject (

subjectCode varchar(10) not null,

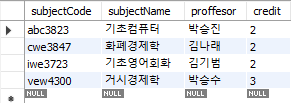
subjectName varchar(10),

proffesor varchar(10),

credit integer(5),

primary key(subjectCode)

);



create table classapplication (

subjectCode varchar(10) not null,

subjectName varchar(10),

proffesor varchar(10),

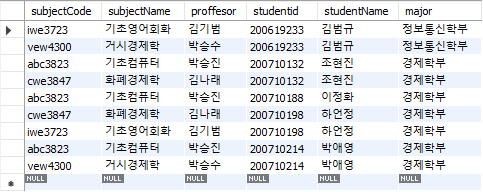
studentid integer(9),

studentName varchar(10),

major varchar(10),

primary key(studentid, subjectcode)

);



실습 2-1. 1: N 관계인 테이블끼리 조인해서 데이터 출력

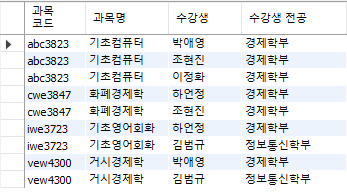
select

ms.subjectcode "과목코드", ms.subjectname "과목명", c.studentname "수강생", c.major "수강생 전공"

from mandatorySubject ms, classapplication c

where (ms.subjectcode=c.subjectcode)

order by ms.subjectcode;



1. **서버**
2. Port, SERVER, WAS의 개념

* 포트:

장치에서 데이터를 전송하거나 수신하는 하드웨어의 위치. HTTP는 80번 포트, HTTPS는 443번 포트를 사용.

* 웹서버:

사용자에게 웹을 제공하기 위한 서버. 웹 브라우저와 같은 클라이언트의 요청이 있는 경우 이를 받아들여 요청한 페이지를 응답. 주로 HTML, CSS, JS, 이미지파일 등 정적 컨텐츠를 리턴. Apache, Nginx

* WAS ( Web Application Server):

웹 애플리케이션과 서버 환경을 만들어 동작시키는 기능을 제공하는 소프트웨어 프레임워크. 인터넷 상에서 HTTP를 통해 사용자 컴퓨터나 장치에 애플리케이션을 수행해 주는 미들웨어(소프트웨어 엔진). 비즈니스 로직, 동적 콘텐츠를 처리하고 리턴. Tomcat, Weblogic, Jeus

* 사용자가 웹 브라우저를 통해 페이지에 접속 –> L7(url, id차단, 요청 분배) -> 웹서버(정적 파일 처리) -> WAS(톰캣. 동적 컨텐츠, 비즈니스 로직 등 처리하여 리턴)

1. 톰캣

: 웹서버에서 서블릿(servlet), JSP와 같은 자바 코드를 이해하기 위해서는 톰캣 같은 자바를 이해하는 엔진이 있어야 함(jsp를 실행하는 웹컨테이너, 서블릿컨테이너).

: WAS, 웹서버의 기능을 동시에 수행. 하나의 서버장비에 여러 개의 톰캣을 구성할 수 있으며 하나의 톰캣에 여러 개의 애플리케이션을 구성할 수 있음

1. 톰캣 다운로드, 설치, 실행, 디렉토리 구조

3-1) 설치

구글 – Tomcat 7.0.94 다운로드 – 압축풀기 – c드라이브내 JSP 폴더 내에 저장 - 환경변수 설정: 내컴퓨터 - 속성 - 고급시스템설정 – 환경변수 – 새 시스템 변수 – 변수 이름: JRE\_HOME 변수 값: C:\Program Files\Java\jre7

3-2) 실행

Cmd – cd C:\JSP\apache-tomcat-7.0.94\bin>STARTUP

웹브라우저 – localhost:8080 / <http://127.0.0.1:8080/> 접속

C:\JSP\apache-tomcat-7.0.94\bin>SHUTDOWN 페이지 접속 불가

3-3) 테스트 폴더 만들기

C:\JSP\apache-tomcat-7.0.94\webapps 내에 test폴더 만들기 – test1.jsp 파일 만든 후 저장.

웹브라우저 <http://127.0.0.1:8080/test/test1.jsp> -> 생성한 파일 확인 가능

3-4) 톰캣 접속 포트번호 변경하기

1. 기본 포트 확인: C:\JSP\apache-tomcat-7.0.94\conf 내 **server.xml** 설정파일 – 문서 중간쯤 <Connector port="**8080**" redirectPort="**8443**" connectionTimeout="**20000**" protocol="**HTTP/1.1**"/>
2. “80”으로 변경하기

Server.xml 파일 우클릭 – 편집 – Connector port="80" – 저장 – 웹 브라우저 [http://localhost:80](http://localhost/) / 127.0.0.1:80

<Connector port="80" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

3-5) 톰캣 웹 루트 디렉토리 찾기

웹 루트 디렉토리: <http://localhost:8080/> 접속 시 찾아가는 디렉토리

경로: C:\JSP\apache-tomcat-7.0.94\webapps\ROOT

기본 설정 내용:

Server.xml에서 - <Engine name="**Catalina**" defaultHost="**localhost**">

<Host name="**localhost**" autoDeploy="**true**" unpackWARs="**true**" appBase="**webapps**">