Day16(8/22)

\*과제\*

**package** com.ict.bm;

**import** java.util.HashMap;

**import** java.util.Scanner;

**public** **class** HashMapExample {

**static** Scanner *scan* = **new** Scanner(System.***in***);

**static** HashMap<Integer, Object> *request* = **new** HashMap<>();

**public** HashMapExample() {

System.***out***.println(":::학생관리시스템:::");

**while** (**true**) {

System.***out***.println("1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit");

String answer = *scan*.next();

**switch** (answer) {

**case** "1":

Input input = **new** Input();

input.inputMethod();

**break**;

**case** "2":

Output output = **new** Output();

output.outputMethod();

**break**;

**case** "3":

Search search = **new** Search();

search.searchMethod();

**break**;

**case** "4":

Delete delete = **new** Delete();

delete.deleteMethod();

**break**;

**case** "5":

Modify modify = **new** Modify();

modify.modifyMethod();

**break**;

**case** "6":

Exit exit = **new** Exit();

exit.exitMethod();

**break**;

**default**:

System.***out***.println("1~5 값을 입력하세요");

}

}

}

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

**new** HashMapExample();

}

}

**package** com.ict.bm;

**import** java.util.Scanner;

**import** **static** com.ict.bm.HashMapExample.*request*;

**public** **class** Input {

**static** Scanner *scan* = **new** Scanner(System.***in***);

**static** **int** *ID* = 1; //ID값은 중복X

**static** String *name*;

**static** **int** *kor*;

**static** **int** *math*;

**static** **int** *eng*;

**static** **int** *sum*;

**static** Double *avg*;

**static** **int** *rank*;

**static** String *grade*;

**static** **public** **void** makeData() {

*sum* = *kor* + *math* + *eng*;

*avg* = (**int**) ((*kor* + *math* + *eng*) / 3.0 \* 100) / 100.0;

*rank* = 1;

*grade* = "";

**if** (*avg* >= 90)

*grade* = "A";

**else** **if** (*avg* >= 80)

*grade* = "B";

**else** **if** (*avg* >= 70)

*grade* = "C";

**else** **if** (*avg* >= 60)

*grade* = "D";

**else**

*grade* = "F";

**int** res\_avg = (**int**) (*avg* % 10);

String res = "";

**if** (*avg* >= 60 && *avg* <= 100) {

**if** (*avg* == 100)

res = "+";

**else** **if** (res\_avg >= 0 && res\_avg <= 3)

res = "-";

**else** **if** (res\_avg >= 4 && res\_avg <= 6)

res = "0";

**else** **if** (res\_avg >= 7 && res\_avg <= 9)

res = "+";

}

*grade* += res;

}

//100이하의 값만 받기

**static** **int** getScore(**int** score) {

**while** (score < 0 || score > 100) {

System.***out***.println("점수는 0~100 값만 유효합니다. 다시 입력하세요!");

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

score = *scan*.nextInt();

}

**return** score;

}

**public** **void** score() {

System.***out***.print("국어>>");

*kor* = *getScore*(*scan*.nextInt());

System.***out***.println("수학>>");

*math* = *getScore*(*scan*.nextInt());

System.***out***.println("영어>>");

*eng* = *getScore*(*scan*.nextInt());

}

**public** **void** inputMethod() {

// 입력받기

System.***out***.print("이름>>");

*name* = *scan*.next();

score();

*makeData*();

// 객체 생성, 객체에 값 넣기

Student student = **new** Student(*name*, *kor*, *math*, *eng*, *sum*, *avg*, *rank*, *grade*);

*request*.put(*ID*, student);

*ID*++;

System.***out***.println("학생 정보 저장 완료!");

}

}

**package** com.ict.bm;

**import** **static** com.ict.bm.HashMapExample.*request*;

**public** **class** Output {

**public** **void** outputMethod() {

System.***out***.println("이름|\t국어|\t수학|\t영어|\t총점|\t평균|\t학점|\t등수|\t");

//ID는 1부터 존재

**for**(**int** i=1; i<=*request*.size(); i++) {

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

}

}

}

**package** com.ict.bm;

**import** **static** com.ict.bm.HashMapExample.*request*;

**import** java.util.Scanner;

**public** **class** Search {

**static** Scanner *scan* = **new** Scanner(System.***in***);

**public** **void** searchData(String name) {

**for**(**int** i=1; i<=*request*.size();i++) {

Student find = (Student) *request*.get(i);

**if**(name.equals(find.getName())){

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

}**else** {

System.***out***.println("찾는 학생이 없습니다.");

}

}

}

**public** **void** searchMethod() {

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

System.***out***.println("찾을 학생의 이름 입력>>");

String name = *scan*.next();

System.***out***.println("이름|\t국어|\t수학|\t영어|\t총점|\t평균|\t학점|\t등수|\t");

searchData(name);

}

}

**package** com.ict.bm;

**import** **static** com.ict.bm.HashMapExample.*request*;

**import** java.util.Scanner;

**public** **class** Modify **extends** Input{

**static** Scanner *scan* = **new** Scanner(System.***in***);

**public** **void** modifyMethod() {

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

System.***out***.println("찾을 학생의 이름 입력>>");

String name = *scan*.next();

System.***out***.println("이름|\t국어|\t수학|\t영어|\t총점|\t평균|\t학점|\t등수|\t");

**for** (**int** i = 1; i <= *request*.size(); i++) {

Student find = (Student) *request*.get(i);

**if** (name.equals(find.getName())) {

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

score(); //점수만 입력

//나머지 정보 입력

*makeData*();

//바뀐 값들로 객체 생성

Student temp = **new** Student(name, *kor*, *math*, *eng*, *sum*, *avg*, *rank*, *grade*);

//find의 정보를 key값이 i인 Student 객체와 변경

*request*.replace(i, temp);

//수정된 정보 출력

System.***out***.println("수정 완료!");

System.***out***.println("이름|\t국어|\t수학|\t영어|\t총점|\t평균|\t학점|\t등수|\t");

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

} **else** {

System.***out***.println("찾는 학생이 없습니다.");

}

}

}

}

**package** com.ict.bm;

**import** **static** com.ict.bm.HashMapExample.*request*;

**import** java.util.Scanner;

**public** **class** Delete{

**static** Scanner *scan* = **new** Scanner(System.***in***);

**public** **void** deleteMethod() {

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

System.***out***.println("찾을 학생의 이름 입력>>");

String name = *scan*.next();

**for** (**int** i = 1; i <= *request*.size(); i++) {

Student find = (Student) *request*.get(i);

**if** (name.equals(find.getName())) {

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

System.***out***.println("정말로 삭제 하시겠습니까?(y/n)");

**while** (**true**) {

String answer = *scan*.next();

**if** (answer.equalsIgnoreCase("y")) {

*request*.remove(i);

System.***out***.println("삭제 완료!");

**break**;

} **else** **if** (answer.equalsIgnoreCase("n")) {

System.***out***.println("삭제를 취소하겠습니다.");

**return**; // 함수 종료

} **else** {

System.***out***.println("y 또는 n을 입력하세요");

**continue**;

}

}

} **else** {

System.***out***.println("찾는 학생이 없습니다.");

}

}

}

}

**package** com.ict.bm;

**public** **class** Exit {

**public** **void** exitMethod() {

System.***out***.println(":::학생관리시스템 종료:::");

System.*exit*(0);

}

}

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

\*결과창

:::학생관리시스템:::

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

1

이름>>류재영

국어>>255

점수는 0~100 값만 유효합니다. 다시 입력하세요!

>>10

수학>>24

영어>>54

학생 정보 저장 완료!

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

2

이름| 국어| 수학| 영어| 총점| 평균| 학점| 등수|

류재영 10 24 54 88 29.33 F 1

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

1

이름>>류재현

국어>>70

수학>>80

영어>>80

학생 정보 저장 완료!

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

2

이름| 국어| 수학| 영어| 총점| 평균| 학점| 등수|

류재영 10 24 54 88 29.33 F 1

류재현 70 80 80 230 76.66 C0 1

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

3

---Search---

찾을 학생의 이름 입력>>

김민경

찾는 학생이 없습니다.

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

3

---Search---

찾을 학생의 이름 입력>>

류재현

이름| 국어| 수학| 영어| 총점| 평균| 학점| 등수|

류재현 70 80 80 230 76.66 C0 1

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

5

---Search---

찾을 학생의 이름 입력>>

류재현

이름| 국어| 수학| 영어| 총점| 평균| 학점| 등수|

류재현 70 80 80 230 76.66 C0 1

국어>>100

수학>>100

영어>>100

수정 완료!

이름| 국어| 수학| 영어| 총점| 평균| 학점| 등수|

류재현 100 100 100 300 100.0 A+ 1

1.Input 2.Output 3.Search 4.Delete 5.Modify 6.Exit

6

:::학생관리시스템 종료:::

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

\*Map 인터페이스 – HashMap, HashTable 클래스

HashMap 예제1)

**package** com.ict.am;

**import** java.util.Enumeration;

**import** java.util.Hashtable;

**public** **class** Ex01MapExample {

// 자바 중요도

// 상속과 다형성, 인터페이스,

// \*\*java.util package, java.lang package\*\*

// Collections(set, map, list)

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

// Map구조 - Key와 Value를 쌍으로 값을 가진다.

// key를 이용해서 value를 검색한다.

// key의 중복이 있으면 안된다.

// Map 인터페이스 상속 1)HashTable 클래스, 2)HashMap 클래스

// Map은 인터페이스이므로 직접 인스턴스 생성이 불가능

// Set구조와 Map구조는 순서가 없다.

// 정렬을 하기위해서는 배열로 변환해야 한다.

Hashtable<String, Object> map = **new** Hashtable<>();

map.put("성명", "홍길동");

map.put("주소", "서울시 강서구 염창동");

map.put("전화번호", "010-5603-5771");

map.put("직업", "프로그래머");

map.put("친구들", **new** String[] { "일지매, 루팡, 임꺽정" });

Enumeration<String> keys = map.keys(); // 키값

**while** (keys.hasMoreElements()) {

String key = keys.nextElement();

//instanceof: 상속관계에 놓여져 있는 클래스끼리 형변환이 가능하면 true, 가능하지 않다면 false

**if** (map.get(key) **instanceof** String) {

String value = (String) map.get(key);

System.***out***.println(key + ": " + value);

}

**if** (map.get(key) **instanceof** String[]) {

System.***out***.print("친구들: ");

String[] arr = (String[]) map.get(key);

**for** (String friend : arr) {

System.***out***.println(friend+"\t");

}

}

}

}

}

HashMap 예제2)

**package** com.ict.am;

**import** java.util.ArrayList;

**import** java.util.Enumeration;

**import** java.util.Hashtable;

**import** java.util.Iterator;

**public** **class** Ex02MapExample {

**static** Hashtable<String, Object> *map* = **new** Hashtable<>();

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

// Map구조에 ArrayList구조를 담아보자

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

// ArrayList: 배열과 비슷, 순차적으로 처리, 중복 허용, 인덱스 중복x

// add(): 객체추가

// set(): 객체대체

// remove(index): 인덱스위치의 객체를 지움

// ArrayList출력

// ArrayList는 Iterator를 이용해서 출력 또는 for문

// Map은 Enumeration를 이용해서 출력한다.

list.add("오징어");

list.add("꼴두기");

list.add("대구");

list.add("명태");

list.add("거북이");

*map*.put("list", list);

*map*.put("String", "안뇽");

// 출력 메소드

*mapPrint*();

}

//key를 모를 때 출력하는 것

**private** **static** **void** mapPrint() {

Enumeration<String> keys = *map*.keys();

**while** (keys.hasMoreElements()) {

String key = keys.nextElement();

System.***out***.print(key + ": ");

**if** (*map*.get(key) **instanceof** ArrayList) {

ArrayList<String> value = (ArrayList<String>) *map*.get(key);

Iterator<String> elements = value.iterator();

**while** (elements.hasNext()) {

System.***out***.print(elements.next() + ", ");

}

}

**if**(*map*.get(key) **instanceof** String) {

String value = (String)*map*.get(key);

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

}

}

}

}

HashMap 예제3)

**package** com.ict.am;

**import** java.util.Hashtable;

**public** **class** Ex03MapExample {

**public** **static** Hashtable<String, Object> *request* = **new** Hashtable<>();

**public** Ex03MapExample() {

System.***out***.println("기본 생성자");

//입력 객체 생성

Input input = **new** Input();

input.show();

// show에 생성한 student객체가 request에 담겨져 있다.

//printMap();

Output output = **new** Output();

output.show();

}

**private** **void** printMap() {

// request(map)에서 key가 "student"인 요소를 가져와서 사용한다.

Student student = (Student) *request*.get("student");

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

}

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

**new** Ex03MapExample();

}

}

**package** com.ict.am;

**import** java.util.Scanner;

**import** **static** com.ict.am.Ex03MapExample.*request*;

**public** **class** Input {

**static** Scanner *scan* = **new** Scanner(System.***in***);

**public** **void** show() {

//사용자 정보를 입력 받아서 Map에 저장

//사용자 정보를 입력 받아서 map에 저장

//성명, 점수를 Student 객체에 담는다.

Student student = **new** Student();

System.***out***.print("성명>>");

student.setName(*scan*.next());

System.***out***.print("점수>>");

student.setScore(*scan*.nextInt());

//Ex03MapExample.request - import static을 해서 request객체를 바로 사용 가능

*request*.put("student", student);

}

}

**package** com.ict.am;

**import** **static** com.ict.am.Ex03MapExample.*request*;

**public** **class** Output {

**public** **void** show() {

**if**((*request*.get("student"))!=**null**){

Student student = (Student) *request*.get("student");

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

}

}

}

**package** com.ict.am;

**public** **class** Student {

**private** String name;

**private** **int** score;

**public** Student() {

}

**public** Student(String name, **int** score) {

**super**();

**this**.name = name;

**this**.score = score;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getScore() {

**return** score;

}

**public** **void** setScore(**int** score) {

**this**.score = score;

}

@Override

**public** String toString() {

**return** "Student [name=" + name + ", score=" + score + "]";

}

}

\*예외(Exception)

-자바를 컴파일하여 에러가 발생하지 않았다고 원하는 결과를 얻은 것은 아니다.

-프로그램이 실행되는 동안에 발생하는 예기치 않은 에러를 예외라고 한다.

-즉, 예외는 일반적이지 않은 상화에서 발생하는 에러이다.

-try-catch

try{

예외가 발생할 만한 코드

}catch(해당 Exception e){

예외 처리를 위한 루틴

}

-예외를 출력하는 다양한 방법

e.toString();

e.getMessage();

e.printStackTrace();

\*finally

-try-catch구문에서 예외가 발생하거나 발생하지 않더라도 반드시 실행해야 하는 문장들이 있을 때,

바로 finally블록 내에 기술한다.

-finally 블록은 try/catch 구문의 제어가 끝난 후에 반드시 처리해야 할 마무리 작업들을 처리하는 구문이다.

\*finally는 예외가 발생여부와 상관없이 반드시 수행한다.

**package** com.ict.cm;

**public** **class** Ex02Throw {

**public** Ex02Throw() {

throwEx();

}

**private** **void** throwEx() {

**try** {

System.***out***.println("예외 처리 try문");

**int** res = 10 / 0; // java.lang.ArithmeticException: / by zero

System.***out***.println("예외가 발생하면 실행 안됨");

} **catch** (ArithmeticException e) {

System.***err***.println("0으로 나눌 수 없습니다.");

} **catch** (Exception e) {

System.***err***.println("예외발생!");

e.printStackTrace();

} **finally** {

System.***out***.println("예외 유무와 관계없이 실행");

}

System.***out***.println("정상종료!");

}

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

**new** Ex02Throw();

}

}

\*예외의 인위적인 발생

-throw 예외객체;

-throw new 예외 클래스

**package** com.ict.cm;

**public** **class** Ex01Throw {

**public** **static** **void** throwException() {

**try** {

System.***out***.println(">>throwException 함수 호출<<");

**throw** **new** Exception("사용자가 예외를 발생시킴!");

} **catch** (Exception e) {

System.***out***.println(">>catch구문: 예외를 처리한다.<<");

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

}

}

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

System.***out***.println(">>main 함수에서 예외를 발생시키는 함수 호출<<");

*throwException*();

System.***out***.println("정상 종료");

}

}

\*예외처리 미루기

**package** com.ict.cm;

**public** **class** Ex03Throws {

**public** Ex03Throws() **throws** Exception {

throwsEx();

}

**public** **void** throwsEx() **throws** Exception {

//throw new Exception();

//try-catch가 먼저 처리--> throws

**try** {

**throw** **new** Exception();

} **catch** (Exception e) {

System.***err***.println("강제로 예외발생!");

e.printStackTrace();

}

}

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

**new** Ex03Throws();

}

}

Exception은 파일 I/O에 사용된다.

**package** com.ict.cm;

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**public** **class** Ex04IOStreamExample {

**public** Ex04IOStreamExample() **throws** Exception{

inputEx();

}

**public** **void** inputEx() **throws** Exception{

// 입력 스트림을 이용한 정보입력

InputStreamReader isr = **new** InputStreamReader(System.***in***);

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

String name = **null**;

**int** age = 0;

System.***out***.println("성명입력>>>");

name = br.readLine();

System.***out***.println("나이입력>>>");

age = **new** Integer(br.readLine());

/\* try {

System.out.println("성명입력>>>");

name = br.readLine();

System.out.println("나이입력>>>");

age = new Integer(br.readLine());

} catch (IOException e) {

e.printStackTrace();

}\*/

}

// Main-프로그램의 시작점->객체의 생성자에서 부터 코딩하는 것이 자바의 정석

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

**new** Ex04IOStreamExample();

}

}

Multitasking :

-프로세스란 운영체제에서 실행중인 하나의 프로그램을 말한다.

-멀티 프로세스란 두 개 이상의 프로세스가 실행되는 것을 말한다.

-멀티 테스킹이란 두개 이상의 프로세스를 실행하여 일을 처리하는 것을 말한다.

\*멀티 스레드

-자바는 멀티스레드 가능

-스레드란 프로세스 내에서 실행되는 세부 작업 단위이다.

-멀티 스레드란, 하나의 프로세스에서 여러 개의 스레드가 병행적으로 처리되는 것을 말한다.

스레드의 생명주기

-new: new Thread() -> start()

-Runnable: 준비

-Running: 실행(CPU를 할당 받은 상태) -> run()메소드 수행

>>Runnable상태에 있는 스레드들은 scheduler에 의해서 running 상태가 된다.

>>Thread가 CPU를 차지하고 있는 상태에서 스케줄러에 의해 다른 Thread가 CPU를 선점하면

>>run중이던 Thread는 blocking되어 Blocked 상태가 된다.

>>unblocked상태가 되면 다시 Runnable상태가 된다.

-Dead

Thread의 생성 방법

-Thread() – 기본생성자

-Thread(Runnable target)

-Thread(Runnable target, String name)

-Thread(String name)

Thread 클래스의 주요 메소드

-sleep(long millis): millis에 지정된 시간만큼 대기한다

-getName(): 스레드의 이름을 반환한다

-setName(String name): 스레드의 이름 지정

-start(): 스레드를 시작시킨다

-join(): 현재 스레드는 join() 메소드를 호출한 스레드가 종료할 때까지 기다리게 된다.

-yield(): 수행중인 스레드 중 우선순위가 같은 다른 스레드에게 제어권을 넘긴다.

\*자바에서 스레드 생성법

1. Thread 클래스 상속 받기

2. Runnable 인터페이스 상속

\*start();

스레드 시작, 무조건 run()을 찾아서 run()내용을 실행한다.

\*\*java.util package 공부하기