Day15(8/21)

\*compare

**package** com.ict.dm;

**import** java.util.Comparator;

**import** java.util.Objects;

**public** **class** CompareExample {

**static** **class** Student {

**int** sno;

Student(**int** sno) {

**this**.sno = sno;

}

}

//compare(대소 비교 가능)

**static** **class** StudentComparator **implements** Comparator<Student> {

@Override

**public** **int** compare(Student st1, Student st2) {

**if** (st1.sno > st2.sno) {

**return** 1;

} **else** **if** (st1.sno < st2.sno) {

**return** -1;

}

**return** 0;

}

}

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

Student s1 = **new** Student(1);

Student s2 = **new** Student(1);

Student s3 = **new** Student(2);

**int** result = Objects.*compare*(s1, s2, **new** StudentComparator());

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

result = Objects.*compare*(s1, s3, **new** StudentComparator());

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

}

}

\*컬렉션 클래스와 제네릭\*

컬렉션: 객체들을 모아서 관리하는 인터페이스들

제네릭: 컬렉션이 어떤 객체들로 이루어졌는지 표시하는 객체 타입, <객체 자료형>

API: <T> =>객체 타입, <E> =>요소, 컬렉션안에 존재하는 객체 하나를 뜻한다.

최상위 컬렉션: Collection<E> : Set<E>, List<E>, Queue<E>, Map<K,V>

\*주요 메소드:

-add(E e): boolean

-clear(): 모든 요소 삭제

-contains(Object o): 해당 객체가 현재 컬렉션에 존재하면 true

-isEmpty(): 해당 컬렉션이 비어있으면 true

-remove(Object o): 해당 객체를 삭제

-size(): 해당 컬렉션안에 존재하는 객체의 수

-toArray(): 해당 컬렉션을 배열로 만들 때 사용

-iterator(): 해당 컬렉션안에 존재하는 객체를 순서대로 접근하기 위한 Iterator 객체 반환

1.Set 인터페이스를 구현한 클래스: HashSet

특징) 중복안됨, 특정기준으로 정렬 안됨, TreeSet은 오름차순 정렬되어서 나온다.

\*HashSet

HashSet 예제1)

**package** com.ict.em;

**import** java.util.HashSet;

**import** java.util.Iterator;

**public** **class** SetExample {

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

HashSet<String> set = **null**;

set = **new** HashSet<String>();

// size(): 해당 컬렉션안에 존재하는 객체의 수

System.***out***.println("set.size() =>" + set.size()); // 0

// set 구조에 내용 추가

set.add("사과");

set.add("포도");

set.add("오렌지");

set.add("바나나");

set.add("포도"); // 중복 허용x, set에 추가 안됨

System.***out***.println("set.size() =>" + set.size());// 4

// set형태를 iterator형태로 변환

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

// Iterator에 담긴 내용을 출력

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

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

}

}

}

HashSet 예제2) 로또

**package** com.ict.em;

**import** java.util.Arrays;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.Random;

**public** **class** Lotto {

**public** **static** **final** **int** ***MAX*** = 45; // 상수

**public** **static** **final** **int** ***MIN*** = 1;

**public** **static** **final** **int** ***SIZE*** = 6;

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

// 로또 번호 생성기

Random rand = **new** Random();

HashSet<Integer> lotto = **new** HashSet<>(); // Integer값만 저장하는 HashSet

// Set인터페이스로 사용하고 있는 HashSet은 중복을 허용하지 않는다.

// 중복된 값이 입력되면 데이터가 추가되지 않는다.

**while** (lotto.size() < ***SIZE***) {

**int** num = ***MIN*** + rand.nextInt(***MAX***); // 1~45까지의 난수 발생

lotto.add(num);

}

// 출력

Iterator<Integer> elements = lotto.iterator();

System.***out***.print("lotto=> ");

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

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

}

//HashSet타입을 배열로 변경

lotto.toArray();

Integer[] lottoArr = **new** Integer[lotto.size()];

lotto.toArray(lottoArr);

//Arrays의 sort기능을 이용해서 배열을 정렬한다.

Arrays.*sort*(lottoArr);

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

System.***out***.print("lotto=> ");

//정렬된 것을 출력한다.-개선된 for문, 확장 for문

**for**(Integer number : lottoArr) {

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

}

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

//lotto Set을 비운다.

lotto.removeAll(lotto);

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

}

}

2.List

List 인터페이스: 배열과 흡사한 구조, 삽입, 삭제 자유롭다, 크기를 미리 지정하지 않아도 된다.

List는 Set과 달리 순차적으로 처리된다.

데이터 중복 허용, 인덱스는 중복 x

관련클래스: ArrayList, Vector, Stack;

1)ArrayList클래스

-add(int index, E) => index위치에 element로 주어진 객체를 저장한다. 해당위치의 객체는 뒤로 밀림

-set(int index, E) => index위치의 요소를 element로 주어진 객체를 대체

-remove(int index) => index위치의 객체를 지움

-int indexOf(Object o) => 전달 인자로 준 객체를 앞에서부터 찾아 해당 위치를 반환,

못 찾으면 -1반환

-int lastindexOf(Object o) => 객체를 마지막 위치부터 찾음

ArrayList예제1)

**package** com.ict.em;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**public** **class** ListEx {

//List인터페이스 - ArrayList 클래스

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

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

strList.add("김길동");

strList.add("박길동");

strList.add("이길동");

strList.add("김길자");

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

strList.add(1, "korea"); //1번지에 korea가 들어가고 그 뒷번호는 밀림

strList.set(3, "이길순"); //3번지에 있는 이길동이 사라지고 "이길순"이 대체됨

strList.remove(2); //해당 index위치의 객체를 지움

//list출력

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

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

}

//indexOf(): 존재하면 해당 index가 리턴, 없다면 -1 리턴

System.***out***.println(strList.indexOf("이길자"));

//strList의 내용을 배열에 담고 정렬해서 출력

String[] listArr = **new** String[strList.size()];

//strList를 listArr에 담음

strList.toArray(listArr);

Arrays.*sort*(listArr);

//for문이용해서 출력

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

System.***out***.print(listArr[i] +" ");

}

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

//확장된 for문 이용해서 출력

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

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

}

}

}

ArrayList와 Comparator사용해서 정렬하는 예제2)

**package** com.ict.em;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**public** **class** StudentListExample {

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

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

list.add(**new** Student("김길동", 65));

list.add(**new** Student("이길동", 55));

list.add(**new** Student("홍길동", 90));

list.add(**new** Student("박길동", 85));

list.add(**new** Student("류길동", 70));

//깊은 비교

//정렬을 위한 함수

//익명 객체로 override해서 Comparator사용

Comparator인터페이스를 상속받아서 현해서 사용해도 됨

Collections.*sort*(list, **new** Comparator<Student>() {

@Override //오름차순

**public** **int** compare(Student o1, Student o2) {

**if**(o1.getScore()>o2.getScore()) {

**return** 1;

}**else** **if**(o1.getScore()<o2.getScore()) {

**return** -1;

}

**return** 0; //같으면 return 0

}

});

**for** (Student student : list) {

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

// student.toString 자동호출 -override함

}

}

}

2)Vector클래스

생성자

-public Vector(): 디폴트 생성자로 빈 벡터 객체를 생성한다.

-public Vector(int initialCapacity): initialCapacity로 지정한 크기의 벡터 객체를 생성한다.

-public Vector(int initialCapacity, int capacityIncrement): initialCapacity로 지정한 크기의 벡터 객체를 생성하되, 새로운 요소가 추가되어 원소가 늘어나야 하면 capacityIncrement만큼 늘어난다.

메소드

-addElement(): 객체를 저장함, add()와 같은 기능

-setElement(): index로 지정한 위치의 객체를 설정한다. Set도 같은 기능 제공

-get(int index): index로 지정된 요소를 반환한다.

-void trimToSize(): Vector의 용량을 현재 크기로 줄여 준다.

-int capacity(): Vector의 용량을 반환

-Enumeration elements(): 벡터 요소들에 대한 Enumeration객체를 반환

Boolean hasMoreElements(): 요소가 있으면 true를 반환하고 없으면 false를 반환한다.

E nextElement() : 요소를 얻어낸다.

3)Stack클래스

:후입선출(LIFO, Last In First Out)

-pop(): 스택의 맨 위(top)에서 객체를 제거하고 그 객체를 반환한다.

-push(E item) : 스택의 맨 위에 객체를 추가한다.

-peek(): 스택의 맨 위에 있는 객체를 반환한다. 이때 객체를 스택에서 제거하지는 않는다.

-Boolean empty(): 현재 스택이 비어 있는지를 확인한다. isEmpty도 동일한 기능을 한다.

3.Queue인터페이스

1)LinkedList

-offer(): 큐에 객체를 넣는다.

-E poll(): 큐에 데이터를 꺼내 온다. 만일 큐가 비어 있다면 null을 반환한다.

-E peek(): 큐의 맨 위에 있는 객체를 반환한다. 이때 객체를 큐에서 제거하지는 않는다.

만일 큐가 비어 있다면 null을 반환한다.

4.Map인터페이스

1)HashMap클래스

Map<K,V>: key와 value는 1:1로 매칭

Key와 value => not null

Key는 중복을 허용하지 않음(Set과 다름: Set은 중복된 값 저장 불가>>기존의 값이 지워지지 않음)

>>Map 기존 값을 지우고 입력됨>> 최신 정보만 가지고 있음

Key를 추가할 때 add사용 못함, push만 사용

Value를 호출할 때 get(키의 위치 값)

보통은 key만관리 한다 ->keyset() 메소드로 key만 Set 컬렉션에서 관리한다.

**package** com.ict.em;

**import** java.util.HashMap;

**import** java.util.Iterator;

//회원가입--><아이디, 비밀번호> : HashMap으로 관리? 또는 HashSet

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

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

HashMap<Integer, String> map = **new** HashMap<>();

// add x, push o

map.put(0, "한국");

map.put(1, "중국");

map.put(2, "태국");

map.put(3, "영국");

map.put(4, "미국");

map.put(3, "공화국"); // key 중복(기존의 키를 삭제하고 현재 키값 저장)

map.put(5, "공화국"); // value 중복(아무런 문제 없다)

System.***out***.println(map); // map의 key,value가 출력된다.

// 호출 방법 : 키를 호출하면 value가 나온다.

System.***out***.println(map.get(0));

System.***out***.println(map.get(1));

System.***out***.println(map.get(2));

System.***out***.println(map.get(3));

System.***out***.println(map.get(4));

System.***out***.println(map.get(5));

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

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

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

}

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

HashMap<String, String> map2 = **new** HashMap<>();

map2.put("이름", "홍길동");

map2.put("나이", "14");

map2.put("주소", "서울");

map2.put("성별", "남성");

//호출

System.***out***.println(map2.get("이름"));

System.***out***.println(map2.get("나이"));

System.***out***.println(map2.get("주소"));

System.***out***.println(map2.get("성별"));

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

//map2의 키를 Set에서 관리 =>keySet();

**for**(String k : map2.keySet()) {

System.***out***.println(map2.get(k));

}

//keySet은 Set이므로 Iterator를 이용해서 출력 가능

Iterator<String> it = map2.keySet().iterator();

**while**(it.hasNext()) {

String msg = it.next();

System.***out***.println(map2.get(msg));

}

}

}

2)Hashtable 클래스

\*hangman game

**package** com.ict.fm;

**public** **class** Customer {

**private** String id;

**private** String password;

Hangman hangman = **null**;

**public** Customer() {

}

**public** Customer(String id, String password) {

**super**();

**this**.id = id;

**this**.password = password;

}

// getter,setter

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

@Override

**public** String toString() {

**return** "Customer [id=" + id + ", password=" + password + "]";

}

}

**package** com.ict.fm;

**import** java.util.Scanner;

**public** **class** Hangman {

**static** String[] *str* = **new** String[9];

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

**private** **static** **int** *tan* = 0; // 회차에 대한 정보

**public** **int** getTan() {

**return** *tan*;

}

**public** **void** setTan(**int** tan) {

**this**.*tan* = tan;

}

**public** Hangman() {

System.***out***.println("객체가 생성되었습니다.");

}

**static** {

*str*[0] = "hello";

*str*[1] = "disco";

*str*[2] = "taste";

*str*[3] = "visit";

*str*[4] = "waste";

*str*[5] = "basic";

*str*[6] = "cisco";

*str*[7] = "money";

*str*[8] = "pushy";

// 배열 생성하고 시작

}

**void** gameStart() {

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

**int** count = 10;

String quiz = *str*[*tan*];

StringBuffer sb = **new** StringBuffer("-----");

String answer = "";

**int** end = 0;

// 답을 맞추는 코드

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

**if** (count == 0) {

System.***out***.println("기회가 모두 끝났습니다.");

**break**;

}

System.***out***.println("총 기회는" + count + "입니다.");

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

System.***out***.println("입력>>>");

answer = *scan*.next();

**boolean** result = **false**;

// - --> 입력한 문자로 바꾸기

**char** a = answer.charAt(0);

**for** (**int** i = 0; i < 5; i++) {

**if** (quiz.charAt(i) == a) {

sb.setCharAt(i, a);

end++;

result = **true**;

}

}

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

**if** (result) {

System.***out***.println("입력한 문자" + a + "존재합니다.");

} **else** {

System.***out***.println("입력한 문자" + a + "존재하지않습니다.");

}

// 10회 이내에 답을 맞출경우

**if** (end == 5) {

System.***out***.println("정답입니다");

System.***out***.println("현재" + (*tan* + 1) + "탄 입니다");

**int** nextan = *tan* + 1;

setTan(nextan);

**break**;// while문 빠져나감

}

count--;

}

// 정답을 맞춤

System.***out***.println("계속하시겠습니까?");

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

String answer2 = *scan*.next();

**if** (answer2.equalsIgnoreCase("y")) {

**if** (*tan* <= 8) {

System.***out***.println(*tan* + 1 + "탄 시작합니다.");

gameStart();

} **else** {

System.***out***.println("모두 깨셨습니다. 게임을 종료합니다");

**break**;

}

} **else** **if** (answer2.equalsIgnoreCase("n")) {

**break**;

} **else** {

System.***out***.println("y/n중 입력하세요");

**continue**;

}

}

}

}

package com.ict.fm;

import java.util.HashMap;

import java.util.Scanner;

import java.util.Set;

import javax.swing.plaf.synth.SynthSeparatorUI;

public class Register\_HashMap {

// 회원가입 만들기 - HashMap이용

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

static Customer[] cm = new Customer[100];

static HashMap<String, String> customer = new HashMap<String, String>(); // 회원 관리

static Set<String> keys = customer.keySet(); // HashSet의 key를 관리하기 위함

static int top = 0;

// 초기값

static {

makeCustomer("fbwodud", "1234");

makeCustomer("maxok65", "2222");

}

// 로그인

public static void assign() {

System.out.println("::::로그인::::");

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

String id = scan.next();

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

String password = scan.next();

if (keys.contains(id)) {

if (password.equals(customer.get(id))) {

System.out.println("로그인 성공!");

gameStart(id); //id값 넘겨서 게임시작

return;// 함수 종료

} else {

System.out.println("비밀번호가 틀렸습니다");

}

} else {

System.out.println("아이디가 존재하지 않습니다");

}

}

//id - 현재 로그인한 아이디값 넘어옴

public static void gameStart(String id) {

//id값과 일치하는 객체 찾아서 게임정보 불러오기

int act = 0; //게임정보 불러오기

for(int i=0; i<top; i++) {

if(id.equals(cm[i].getId())) {

act = i;

break;

}

}

cm[act].hangman.gameStart(); //hangman객체의 게임 시작

}

public static void makeCustomer(String id, String password) {

// 객체 생성

cm[top] = new Customer(id, password);

cm[top].hangman = new Hangman(); //아이디 만들때 hangman 객체 생성

//HashMap 추가

customer.put(id, password);

top++;

}

// 회원가입

public static void register() {

System.out.println("::::회원가입::::");

String id = "";

while (true) {

System.out.println("사용할 ID를 입력 하세요>>>");

id = scan.next();

// 중복검사

if (keys.contains(id)) {

System.out.println("사용할 수 없는 아이디 입니다.");

continue;

} else {

System.out.println("사용가능합니다.");

break;

}

}

System.out.println("사용할 비밀 번호를 입력하세요");

String password = scan.next();

//객체생성 함수

makeCustomer(id, password);

}

// 전체 목록 보기

public static void view() {

System.out.println("::::고객정보::::");

for(int i =0; i<top; i++) {

System.out.println(cm[i]);

}

}

// 종료

public static void close() {

System.out.println("::::종료::::");

System.out.println("수고하셨습니다");

System.exit(0);

}

public static void main(String[] args) {

while (true) {

System.out.println("1.회원가입 2.로그인 3.회원목록 4.종료");

String answer = scan.next();

switch (answer) {

case "1":

register();

break;

case "2":

assign();

break;

case "3":

view();

break;

case "4":

close();

break;

default:

System.out.println("잘못누르셨습니다.");

}

}

}

}