

자바 강의 2주차(4)

Set을 배울것이야.

Set은 java.util 패키지에 존재하는 인터페이스입니다.

특정 코드에서 중복된 값의 허용이 있어서는 안될 때 사용한다.

즉, Set을 사용하면 복잡한 코드구성 없이 중복된 요소들을 쉽게 제거할 수 있다는 장점이 있다.

여기서는 가장 많이 사용되는 HashSet과 TreeSet을 알아보도록 하겠습니다.

SetEx 클래스 정의

```
public class SetEx {
```

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

```
        // Set은 값의 중복을 허용하지 않는다.
```

```
        // - HashSet : 정렬이 안됨
```

```
        // - TreeSet : 오름차순
```

```
        HashSet<Integer> hs = new HashSet<Integer>();
```

```
        while(true){
```

```
            // 난수 발생하기
```

```
int v  
= (int)(Math.random()*45+1);
```

```
int v  
=
```

```
new Random().nextInt( (큰 수- 작은 수) + 1 ) + 작은 수
```

```
int v  
= new Random().nextInt(45) + 1;
```

```
hs.add(v);
```

```
if(hs.size() >= 6)
```

```
break;
```

```
}
```

```
System.out.println(hs);
```

```
// 여
```

```
기까지 HashSet예제
```

```
TreeSet<Integer> ts = new TreeSet<Integer>();
```

```
while(true){
```

```

        = (int)(Math.random()*45+1);
        int v

        =
        int v

        new Random().nextInt( ( 큰 수- 작은 수) + 1 ) + 작은 수

        = new Random().nextInt(45) + 1;
        int v

        ts.add(v);

        if(ts.size() >= 6)

        break;

        }

        System.out.println(ts);

        //여
        기는 TreeSet예제

        //위의 결과에서 10보단 크고 30보단 작은 수를 가려낸 것

```

```
System.out.println(ts.subSet(10, 30));  
}  
}
```

HashSet을 이용하여 5 * 5의 랜덤 빙고판 만들기

```
class Bingo  
{  
  
    public static void main(String[] args)  
    {
```

```
        HashSet<Integer> set = new HashSet<>();
```

```
        int[][] board = new int[5][5];
```

```
        for(int i=0; set.size() < 25; i++) {
```

```
            set.add(new Random().nextInt(50) + 1);  
        }
```

```
        // Set구조는 arrayList와 같이 get()메서드를 이용하여 특정 인덱스로 접근
```

```
        // 할수 없기 때문에, 내용을 순차적으로 얻어오기 위해서는
```

```
        // iterator라고 하는 반복자를 이용해야 한다.
```

```
Iterator<Integer> it = set.iterator();
```

```
for(int i=0; i < board.length; i++) {
```

```
for(int j=0; j < board[i].length; j++) {
```

```
board[i][j] = (Integer)it.next();
```

```
System.out.printf("%02d ", board[i][j]);
```

```
}
```

```
System.out.println();
```

```
}
```

```
}
```

```
}
```

Map

Map은 키(key)와 값(value)을 묶어서 하나의 데이터로 저장한다는 특징을 갖습니다.

키를 통해 벨류를 검색하므로, 많은양의 데이터를 검색하는데 있어서 뛰어난 성능을 발휘합니다.

가장 많이 쓰이는 HashMap을 살펴봅니다.

MapEx1클래스 정의

```
public class MapEx1 {
```

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

/ /

Map구조는 Key값과 Value가 하나의 쌍을 이루어 저장된다.

```
        HashMap<String, Boolean> map = new HashMap<String, Boolean>();
```

```
        map.put("k1", true);
```

```
        map.put("k2", true);
```

```
        map.put("k2", false);
```

/ /

key값이 중복될 경우 추가가 아님. k2의 값을 false로 변경한다.

```
        System.out.println("map의 사이즈:"+map.size());//map은 .length가 아닌
```

```
// .size()를 사용
```

```
        boolean b1 = map.get("k1");
```

```
        System.out.println("b1 : "+ b1);
```

```
boolean b2 = map.get("k2");
```

```
System.out.println("b2 : " + b2);//true가 아닌 false다.
```

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

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

```
map2.put("s1", "나의 key는 s1");
```

```
map2.put("s2", "나는 key는 s2");
```

```
map2.put("s3", "나는 key는 s3");
```

```
String str1 = map2.get("s1");
```

```
System.out.println("str1 : " + str1);
```

```
String str2 = map2.get("s2");
```

```
System.out.println("str2 : " + str2);
```

```
String str3 = map2.get("s3");
```

```
System.out.println("str3 : " + str3);
```

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

Map구조는 Key값으로 value를 찾기 때문에,

//

배열처럼 메모리공간에 순차적으로 자리를 잡고있지 않다.

//

때문에 for문 등을 이용하여 값을 얻을 수 없다.

//

Map구조의 내용을 순차적으로 얻어오기 위해서는

//

iterator라고 하는 반복자를 이용해야 한다.

//

저 Map구조의 내용을 출력하기 위해 Key값들을 모두 가져온다.

// 먼

```
Set<String> keys = map2.keySet();//map2의 모든 key값을 keys에 저장
```

```
//map2의 키는 String이기 때문에 keys의 제네릭 타입 또한 String으로
```

한다.

키값들을 메모리에서 가져와 순차적으로 나열화 시킨다.

//

Java.util패키지의 Iterator인터페이스를 사용

//

```
Iterator<String> it = keys.iterator();
```

```
while(it.hasNext()){
```

// 위

의 while문의 it.hasNext()는

// 자

신의 현재 위치로부터 순차적으로 다음 위치로 이동할수 있는

지 를

판단

// 가

능하다면 true, 불가능하다면 false를 반환

//hasNext()를 통해 다음 위치로 이동할수 있다는 것이 알려지면

//next()를 통해 실제로 다음 위치로 이동한다.

```
String k = it.next(); // 키값 하나 가져오기
```

```
키값으로 value를 얻어낸다.
```

```
String v = map2.get(k);
```

```
System.out.println(k+":"+v);
```

```
}
```

```
-----예제 1
```

```
MapEx2클래스 정의
```

```
public class MapEx2 {
```

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

```
Map구조는 Key값과 Value가 하나의 쌍을 이루어 저장된다.
```

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

```
map.put("kim", 1111);
```

```
map.put("lee", 2222);
```

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

```
while(true){
```

```
System.out.print("id를 입력하세요 : ");
```

```
String id = scan.next();
```

```
System.out.print("password를 입력하세요 : ");
```

```
int  
pwd = scan.nextInt();
```

```
if(!map.containsKey(id)){//입력받은 값이 map객체의 Key값으로
```

```
//저  
장되어 있지 않은경우
```

```
System.out.println("아이디가 존재하지 않습니다.");
```

```
}else{//아이디가 일치할 경우
```

```
if(map.get(id) != pwd){
```

```
System.out.println("비밀번호가 일치하지 않습니다.");
```

```
}else{
```

```
System.out.println("로그인 성공!!");
```

```
break;
```

```

}

}

}

}

}

```

-----예제2

set과 map은 여기까지!!

ArrayList 를 배워보자~

바로 예제로 고고싱

ArrayEx1클래스 정의

```
public class ArrayEx {
```

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

```
/ /
```

배열과 같지만 배열은 크기가 정해져야만 한다.

```
/ /
```

```
int[] ar = new int[10]; 이런식으로.
```

```
/ /
```

하지만 List구조는 size가 늘었다가 줄었다가 유동적이다.

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

```
System.out.println("list.size():"+list.size());
```

```
list.add(100);
```

```
list.add(20);
```

```
System.out.println("list.size():"+list.size());//2
```

```
for(int i=0; i<list.size(); i++)
```

```
System.out.println(list.get(i));
```

// 인

덱스로 접근하는 구조는 깊이가 있으면 있을 수록 속도가 느리다.

/ /

그것을 해결하기 위해 나온 개념이 바로 Iterator다.

// 1)

List구조의 자원을 반복자로 변환하다.

```
Iterator<Integer> it = list.iterator();
```

// 2)

반복자를 반복문으로 수행한다.

```
while(it.hasNext()){ // 반복자 안에는 커서가 존재하며,
```

// 현

재 커서의 위치에서 다음 요소에 객체가 있다면...

int v

```
= it.next(); //unBoxing 객체를 기본자료형에 대입.
```

```
System.out.println(v);
```

```
    }  
    }  
}
```

-----예제1

ArrayEx2클래스 정의

```
public class ArrayEx2 {
```

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

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

```
        list.add("홍길동");
```

```
        list.add("이순신");
```

```
        list.add("강감찬");
```

```
        list.add("을지문덕");
```

```
        list.add("연개소문");
```

```
System.out.println(list);
```

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

```
System.out.println("list[0] : " + list.get(0));
```

```
System.out.println("list[2] : " + list.get(2));
```

```
}
```

```
}
```

```
-----예제2
```

ArrayList문제

아래의 결과를 도출하시오.

아이디 생성 : abc

abc

아이디 생성 : abc2

abc

abc2

아이디 생성 : abc3

abc

abc2

abc3

아이디 생성 :

```
ArrayList<String> arr = new ArrayList<String>();
```

```
while(true){
```

```
System.out.print("아이디 생성 : ");
```

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



```
String id = scan.next();
```

```
arr.add(id);
```

```
for(int i = 0; i < arr.size(); i++){
```

```
System.out.println(arr.get(i));
```

```
}
```

```
}
```

ArrayList문제

바로 위의 코드에서 중복된 아이디를 검사하는 로직을 추가하기.

아이디 생성 : abc

abc

아이디 생성 : abc

중복된 아이디

아이디 생성 : abc2

abc

abc2

아이디 생성 :

```
ArrayList<String> arr = new ArrayList<String>();
```

```
outer : while(true){
```

```
System.out.print("아이디 생성 : ");
```

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

```
String id = scan.next();
```

```
for(int i = 0; i < arr.size(); i++){
```

```
if(id.equals(arr.get(i))){
```

```
System.out.println("중복된 아이디");
```

```
continue outer;
```

}

}

/*

i f (

```
arr.contains( id ) {
```

```
System.out.println("중복된 아이디");
```

```
continue outer;
```

}

* /

```
arr.add(id);
```

```
for(int i = 0; i < arr.size(); i++){
```

```
System.out.println(arr.get(i));
```

}

```
//while()
```

ArrayList문제

유저의 아이디와 패스워드를 가지는 UserInfo클래스를 하나 만들고 Main클래스에서 유저의 정보를 어레이리스트에 추가하여 출력해보자

---결과---

아이디 생성 : aaa

패스워드 입력 : 1234

aaa

1234

아이디 생성 : bbb

패스워드 입력 : 5678

aaa

1234

bbb

5678

아이디 생성 :

UserInfo클래스 정의

```
public class UserInfo {
```

```
    private String id;//아이디
```

```
    private int pwd;//패스워드
```

```
    public String getId() {
```

```
        return id;
```

```
    }
```

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

```
        this.id = id;
```

```
    }
```

```
    public int getPwd() {
```

```
        return pwd;
```

```

    }

    public void setPwd(int pwd) {

        this.pwd = pwd;

    }

}

```

UserMain클래스 정의

```

public class UserMain {

    public static void main(String[] args) {

        ArrayList<UserInfo> arr = new ArrayList<UserInfo>();

        //ArrayList<UserInfo> arr = new ArrayList<>();

        // 요
        령게 써도 된다. 근데 난 습관이 돼서 생성할때도 제네릭 타입을

        // 넣
        는게 좋더라

        while(true){

            System.out.print("아이디 생성 : ");

```

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

```
//아
```

```
이디를 입력할때마다 새로운 UserInfo객체 생성
```

```
UserInfo ui = new UserInfo();
```

```
ui.setId(scan.next());
```

```
System.out.print("패스워드 입력 : ");
```

```
Scanner scan2 = new Scanner(System.in);
```

```
ui.setPwd(scan2.nextInt());
```

```
arr.add(ui);
```

```
for(int i = 0; i < arr.size(); i++){
```

```
System.out.println(arr.get(i).getId());
```

```
System.out.println(arr.get(i).getPwd());
```

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

```
    }  
    }  
    }  
}
```

-----예제3

문제 :

바로 직전에 만든 UserInfo클래스와 UserMain클래스를 활용하여
아이디가 생성될 때 기존에 사용중인 같은 아이디가 있을 경우
이를 판단하여, “아이디가 중복됩니다”라는 메시지와 함께
while()문의 처음으로 돌아가는 로직을 구현하기.

문제 정답풀이

UserInfo클래스 정의

```
public class UserInfo {
```

```
String id;
```

```
int
```

```
pwd;
```

```
public String getId() {
```

```

return id;
}

public void setId(String id) {

this.id = id;
}

public int getPwd() {

return pwd;
}

public void setPwd(int pwd) {

this.pwd = pwd;
}

}

```

UserMain클래스 정의

```

public class UserMain {

public static void main(String[] args) {

ArrayList<UserInfo> arr = new ArrayList<UserInfo>();

outer : while(true){

System.out.print("아이디 생성 : ");

```

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

```
UserInfo ui = new UserInfo();
```

```
ui.setId(scan.next());
```

```
for(int i = 0; i < arr.size(); i++){
```

```
if(ui.getId().equals(arr.get(i).getId())){
```

```
System.out.println("아이디가 중복됩니다. 다른
```

```
아이디를 생성하세요");
```



```
continue outer;
```

```
}
```

```
}
```

```
System.out.print("패스워드 입력 : ");
```

```
Scanner scan2 = new Scanner(System.in);
```

```
ui.setPwd(scan2.nextInt());
```

```
arr.add(ui);
```

```
for(int i = 0; i < arr.size(); i++){
```

```
System.out.println(arr.get(i).getId());
```

```
System.out.println(arr.get(i).getPwd());
```

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

```
}
```

```
}
```

```
}
```

```
}
```

-----문제(예제)4

고객의 인적사항을 추가하고, 삭제하고, 확인하기 위한 문제출제.

이름과 나이, 번호를 갖는 Person클래스를 만든 후, ArrayList를 사용하여 아래의 결과처럼 Person객체의 정보추가와 전체정보 보기를 할 수 있도록 만들어보자
정보삭제부분은 안배웠으니 나랑같이 고고싱

결과 :

1. 정보추가
2. 정보삭제
3. 전체정보
4. 종료

항목선택 : 1 <- 정보추가 항목

-----정보추가-----

이름 : 1

나이 : 1

전화 : 1

정보가 저장되었습니다.

1. 정보추가
2. 정보삭제
3. 전체정보
4. 종료

항목선택 : 3 <- 정보보기 항목

-----전체정보-----

등록인원 1명

이름 : 1

나이 : 1

전화 : 1

풀이 ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Person클래스 정의

```
public class Person {
```

```
// //
```

한 사람의 정보를 담당하는 Person클래스를 정의

```
private String name;
```

```
private int age;
```

```
private String tel;
```

```
public void setAge(int age) {
```

```
    this.age = age;
```

```
}
```

```
public int getAge() {
```

```
    return age;
```

```
}
```

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

```
    this.name = name;
```

```
}
```

```
public String getName() {
```

```
    return name;
```

```
}
```

```
public void setTel(String tel) {
```

```
this.tel = tel;
}
```

```
public String getTel() {

return tel;
}
```

PersonManager클래스 정의

```
public class PersonManager {
```

```
public void personMgr(){

select;

Person p;
```

```
//Person객체만 저장할 수 있는 ArrayList객체 생성
```

```
ArrayList<Person> personArr = new ArrayList<>();
```

```
while(true){
```

```
System.out.println("1. 정보추가");
```

```
System.out.println("2. 정보삭제");
```

```
System.out.println("3. 전체정보");
```

```
System.out.println("4. 종료");
```

```
System.out.print("항목선택 : ");
```

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

```
select = scan.nextInt();
```

```
switch (select) {
```

```
1: //정보추가
```

case

p =

new Person();//정보를 추가할때마다 Person객체를

새로 생성한다.

System.out.println("-----정보추가-----");

System.out.print("이름 : ");

p.setName(scan.next());

System.out.print("나이 : ");

```
p.setAge(scan.nextInt());
```

```
System.out.print("전화 : ");
```

```
p.setTel(scan.next());
```

```
personArr.add(p);
```

```
System.out.println("정보가 저장되었습니다.");
```

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

```
break;
```

case

2: //정보삭제

System.out.println("-----정보삭제-----");

System.out.print("삭제할 이름 : ");

String name = scan.next();

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


```
if((personArr.get(i).getName()).equals(name)){
```

```
personArr.remove(i); //personArr의 i
```

```
정보를 삭제한다.
```

번 째

```
System.out.println(name + "의 정보를
```

```
삭제했습니다.");
```

```
break;
```

```
}else{
```

```
if(i +
```

```
1 == personArr.size())
```

```
System.out.println(name + "
```

```
이 존재하지 않습니다.");
```

```
}
```

}

break;

case

3: //전체정보

System.out.println("-----전체정보-----");

System.out.println("등록인원 " + personArr.size() + "명");

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

```
System.out.println("이름 : " +
```

```
personArr.get(i).getName());
```

```
System.out.println("나이 : " +
```

```
personArr.get(i).getAge());
```

```
System.out.println("번호 : " +
```

```
personArr.get(i).getTel());
```

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

```
}
```

```
//for문 대신 Iterator를 이용한 while문 사용 가능
```

```
/*Iterator<Person> it = personArr.iterator();
```

```
while(it.hasNext()){
```

```
it.next();
```

```
p =
```

```
System.out.println("이름 : " + p.getName());
```

```
System.out.println("나이 : " + p.getAge());
```

```
System.out.println("번호 : " + p.getTel());
```

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

```
}*/
```

break;

default:

System.out.println("프로그램 종료");

return;

}

}

}

}

PersonMain클래스 정의

public class PersonMain {

public static void main(String[] args) {

PersonManager pMgr = new PersonManager();

```
pMgr.personMgr();  
}  
}
```

-----문제(예제)5

리스트의 정렬기능.(sort)

User클래스 생성 및 내용 추가.

```
public class User {  
  
    private String name;//이름  
    private int no;//일련번호  
  
    public String getName() {  
  
        return name;  
    }  
  
    public void setName(String name) {  
  
        this.name = name;  
    }  
  
    public int getNo() {  
  
        return no;  
    }  
  
    public void setNo(int no) {  
  
        this.no = no;  
    }  
}
```

CompareTest클래스 생성 및 내용 추가.

```
public class CompareTest {
```



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

```
//제
```

```
네릭 타입을 User형으로 갖는 ArrayList생성.
```

```
ArrayList<User> users = new ArrayList<User>();
```

```
User user = new User();
```

```
user.setName("고철수");
```

```
user.setNo(1);
```

```
users.add(user);
```

```
user = new User();
```

```
user.setName("박영희");
```

```
user.setNo(2);
```

```
users.add(user);
```

```
user = new User();
```

```
user.setName("감수왕");
```

```
user.setNo(3);
```

```
users.add(user);
```

```
user = new User();
```

```
user.setName("이사람");
```

```
user.setNo(4);
```

```
users.add(user);
```

```
System.out.println("==== 정렬 하기전 =====");
```

```
f o r
```

```
(User temp : users) {
```

```
System.out.print(temp.getNo() + " : ");
```

```
System.out.println(temp.getName());
```

```
}
```

```
//Comparator<T>를 구현하는 클래스를 한 개씩 생성하며 확인
```

```
System.out.printf("\n\n====문자 오름 차순 정렬 ==== \n");
```

```
Collections.sort(users, new NameAscCompare());
```

```
f o r
```

```
(User temp : users) {
```

```
System.out.print(temp.getNo() + " : ");
```

```
System.out.println(temp.getName());
```

```
}
```

```
System.out.printf("\n\n==== 문자 내림 차순 정렬 ==== \n");
```

```
Collections.sort(users, new NameDescCompare());
```

```
f o r
```

```
(User temp : users) {
```

```
System.out.print(temp.getNo() + " : ");
```

```
System.out.println(temp.getName());
```

```
}
```

```
Collections.sort(users, new NoAscCompare());
```

```
System.out.printf("\n\n==== 숫자 오름 차순 정렬 ==== \n");
```

```
f o r
```

```
(User temp : users) {
```

```
System.out.print(temp.getNo() + " : ");
```

```
System.out.println(temp.getName());
```

```
}
```

```
Collections.sort(users, new NoDescCompare());
```

```
System.out.printf("\n\n=== 숫자 내림 차순 정렬 ===\n");
```

```
f o r
```

```
(User temp : users) {
```

```
System.out.print(temp.getNo() + " : ");
```

```
System.out.println(temp.getName());
```

```
}
```

```
}//main
```

```
static class NameAscCompare implements Comparator<User> {
```

```
//문
```

```
자 오름차순(ASC)
```

```
@Override
```

```
public int compare(User o1, User o2) {
```

```
//o1은 홀수번째, o2는 짝수번째 위치의 객체를 나타내는데,
```

```
//이
```

```
건 굳이 신경 쓸 필요는 없고 공식대로 코딩해주면 된다.
```

```
return o1.getName().compareTo(o2.getName());
```

```
}
```

```
}
```

```
static class NameDescCompare implements Comparator<User> {
```

```
//문
```

```
자 내림차순(DESC)
```

```
@Override
```

```
public int compare(User o1, User o2) {
```

```
return o2.getName().compareTo(o1.getName());
```

```
}  
}
```

```
static class NoAscCompare implements Comparator<User> {
```

```
//숫
```

```
자 오름차순(ASC)
```

```
@Override
```

```
public int compare(User o1, User o2) {
```

```
return o1.getNo() < o2.getNo() ?
```

```
-1 :
```

```
o1.getNo() > o2.getNo() ? 1:0;
```

```
}  
}
```

```
static class NoDescCompare implements Comparator<User> {
```

```
//숫
```

```
자 내림차순(DESC)
```

```
@Override
```

```
public int compare(User o1, User o2) {
```

```
    return o1.getNo() > o2.getNo() ?
```

```
        -1 :
```

```
        o1.getNo() < o2.getNo() ? 1:0;
```

```
    }
```

```
}
```

```
}//class end
```

자바 강의 2주차(4) 문제(스레드와 ArrayList를 활용한 영타연습 게임)
출제!!!!