**19회차(1)**

<< util package >>

- java.util.\*

- java.util.collection

- List

- Map

(1) List

- List 인터페이스를 구현한 인터페이스들

- (ex) ArrayList, LinkedList, Stack, Vector

- ArrayList

- 배열과 동일한 구조

- 배열과의 가장 큰 차이점은 : ArrayList는 가변적이다!

- (배열은 크기가 고정되어 있다) <=> but ArrayList 크기 변경 용이

- clear(), add(E e), add(int index, E element), ..

- remove(int index), contains(배열 원소), add(인덱스번호, 새로 넣을 원소)

- get(int index), size()

(2) Generics

- 지네릭스

- 배열은 하나의 타입만 원소로 가질 수 있다

- 타입에 대한 명시적인 체크를 따로 지원하지 않는다

- 자동으로 타입에 대한 체크가 가능

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

String name = arr.get(0);

(3) Map

- 배열과 유사한 형태

- Key and Value 형태의 자료형

- 배열은 이미 인덱스가 정해져 있다(0부터 순차적으로 번호가 부여된다)

- 자동으로 인덱스가 부여되지 않는다

- 지정한 키를 가지고 인덱스로 사용

- Map 인터페이스를 구현해서 여러가지 클래스들

- HashMap, TreeMap

: HashMap<String, String> map = new HashMap<String, String>();

- 원소를 추가하는 경우에는 put() 메소드를 사용

- 원소에 접근할 때는 마찬가지로 직접 접근할 수 없고 get() 메소드를 사용

- 전체 원소를 확인하려면 entrySet

- 맵의 크기는 size() 메소드를 통해서 확인

- 원소를 삭제하고 싶은 경우에는 remove() 메소드를 사용

- 키만 확인 : keySet(), 값만 확인 : values()

**19회차(2)**

package util;

// ArrayList를 사용하기 위해서 패키지를 임포트

import java.util.\*;

public class test01 {

public static void main(String[] args) {

// ArrayList 타입의 객체를 생성

// -> 비어있는 배열 하나가 만들어지는 것과 동일

ArrayList arr = new ArrayList();

// 배열과 다른 점? : 배열은 arr.length를 이용한다 <-> size() 메소드 이용

// 그래서 크기를 확인해보면 -> 0 출력

System.out.println(arr.size());

// ArrayList에서 원소를 추가할 때는 add() 메소드를 이용한다

arr.add(10);

arr.add(20);

arr.add(30);

//=> {10, 20, 30}

// add() 메소드는 맨 마지막 원소 뒤에 추가

System.out.println(arr.size());

// ArrayList 내의 원소를 확인하고 싶은 경우에는 get() 메소드를 사용할 수 있다.

// String처럼 직접적인 접근은 허용하지 않는다.

System.out.println(arr.get(0));

// ArrayList 내의 모든 원소를 확인하고 싶다면? -> 반복문을 통해서 순회해야 한다.

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

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

}

// 원소 사이에 새 원소를 추가하고 싶다면? (삽입)

// {10, 20, 30}에서 10과 20 사이에 15를 넣어보자

// arr.add(인덱스번호, 새로 넣을 원소)

arr.add(1, 15);

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

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

}

// 배열 내에서 원소를 검색하는 경우에는 contains() 메소드를 사용

// boolean 값 출력 <- 있다(true) / 없다(false)

System.out.println(arr.contains(15));

System.out.println(arr.contains(10));

// 배열 내의 원소를 삭제하는 경우에는 remove() 메소드를 사용

// 만약 숫자 20을 삭제하기를 원한다면?

// arr.remove(20); -> error

// 20번째 인덱스는 존재하지 않는다 -> 오류

// 원소를 찾아서 삭제하길 원한다면 참조형만 가능

// 숫자인 경우에는 인덱스로 인식한다!(숫자 20을 삭제하는 것은 불가능)

// => 인덱스번호로!

arr.remove(2);

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

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

}

}

}

**19회차(3)**

package util;

import java.util.\*;

public class test02 {

public static void main(String[] args) {

// [ 도서 등록 프로그램 구현 ]

// 사용자에게서 계속해서 도서명을 입력 받아 저장하는 저장소를 구현

// 이미 등록된 책이면 등록이 거절되도록 한다

// 예상 결과

// 도서명 : 어린 왕자

// 도서 등록이 완료되었습니다..현재 도서 1개

// 도서명 : 키다리 아저씨

// 도서 등록이 완료되었습니다..현재 도서 2개

// 도서명 : 어린 왕자

// 이미 등록된 도서입니다.

// 도서명 : 종료

// 프로그램을 종료 합니다.

// 도서관 arr 배열 만들기

ArrayList arr = new ArrayList();

System.out.println("도서 등록 프로그램 구현");

Scanner sc = new Scanner(System.in);

while (true) {

System.out.print("도서명 : ");

// book 입력받기

String book = sc.nextLine();

// 기존 배열에 동일한 도서가 있는지 검사

if (arr.contains(book)) {

System.out.println("프로그램을 종료합니다.");

}

// 입력된 문자열이 "종료"라면 프로그램 종료

else if (book.equals("종료")) {

System.out.println("프로그램을 종료합니다");

break;

}

// 그렇지 않으면 arr 배열에 도서를 추가해준다

else {

arr.add(book);

System.out.printf("도서 목록이 완료되었습니다...현재 도서 %d개\n", arr.size());

}

}

}

}

**19회차(4)**

package util;

import java.util.\*;

public class Test03 {

public static void main(String[] args) {

// [ 도서 등록 프로그램 구현 ]

// 사용자에게서 계속해서 도서명을 입력 받아 저장하는 저장소를 구현

// 이미 등록된 책이면 등록이 거절되도록 한다

// 예상 결과

// 도서명 : 어린 왕자

// 도서 등록이 완료되었습니다..현재 도서 1개

// 도서명 : 키다리 아저씨

// 도서 등록이 완료되었습니다..현재 도서 2개

// 도서명 : 어린 완자

// 이미 등록된 도서입니다.

// 도서명 : 종료

// 프로그램을 종료 합니다.

// 1. 도서 등록 2. 도서 삭제 3. 도서 검색

ArrayList arr = new ArrayList();

System.out.println("도서 등록 프로그램 구현");

Scanner sc = new Scanner(System.in);

while (true) {

// 메뉴 선택

System.out.println("1.도서 등록 2.도서 삭제 3.도서 검색 ");

int select = sc.nextInt();

sc.nextLine();

// 1. 도서 등록

if (select == 1) {

System.out.print("도서명 : ");

String book = sc.nextLine();

if (arr.contains(book)) {

System.out.println("프로그램을 종료합니다.");

} else if (book.equals("종료")) {

System.out.println("프로그램을 종료합니다");

break;

} else {

arr.add(book);

System.out.printf("도서 목록이 완료되었습니다...현재 도서 %d개\n", arr.size());

}

}

else if (select == 2) {

System.out.println("삭제할 도서 이름 : ");

String name = sc.nextLine();

// 입력한 도서명이 존재하면 삭제 진행

if (arr.contains(name)) {

arr.remove(name);

System.out.println("정상적으로 삭제되었습니다.");

} else {

System.out.println("삭제하려는 도서가 존재하지 않습니다.");

}

}

else if (select == 3) {

System.out.println("검색할 도서 이름 : ");

String look = sc.nextLine();

if (arr.contains(look)) {

// 그냥 있다라고만 출력하기 보다는 배열 내의 인덱스를 출력

// 배열 내의 해당 원소의 인덱스는 indexOf() 메소드를 사용

System.out.printf("%s 도서는 %d번째에 있습니다.", look, arr.indexOf(look) + 1);

} else {

System.out.println("검색한 도서가 존재하지 않습니다.");

}

}

else {

System.out.println("잘못된 번호를 입력하였습니다.");

}

}

}

}

**19회차(5)**

package util;

import java.util.\*;

public class Test04 {

public static void main(String[] args) {

// 지네릭을 사용하지 않고 배열을 사용했을 때

// ArrayList arr = new ArrayList();

// 지네릭스는 객체를 생성할 때 타입을 명시해줄 수 있다!

ArrayList<String> arr = new ArrayList();

// 입력되는 자료의 타입을 보고 자동으로 타입이 결정

arr.add("Java");

arr.add("Python");

arr.add("C++");

// 배열 내의 원소를 꺼내오는 경우

// ArrayList 내의 타입을 알 수 없기 때문에 문제 발생

// String name = arr.get(0); -> error

// String name = (String)arr.get(0);

// => 별도의 타입을 체크하지 않아도 된다

String name = arr.get(0);

System.out.println(name);

// 배열에 다른 타입의 원소를 추가하는 경우

// => 별도의 타입 체크를 필요로 하지 않게 된다.

}

}

**19회차(6)**

package util;

import java.util.\*;

public class Test05 {

public static void main(String[] args) {

// HashMap<key type, value type>

// HashMap에 들어갈 키와 원소의 타입이 모두 문자열임을 의미

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

// 원소를 추가하는 경우에는 put() 메소드를 사용

// 원소를 추가할 때에는 타입에 맞춰서 Key와 Value를 같이 입력

map.put("first", "장동건");

map.put("second", "현빈");

map.put("third", "진");

// 원소에 접근할 때는 마찬가지로 직접접 근할 수 없고 get() 메소드를 사용

// 이때, 숫자 인덱스는 사용할 수 없고 입력된 Key를 통해서 원소에 접근

System.out.println(map.get("first"));

// 전체 원소를 확인하려면

System.out.println(map.entrySet());

// entrySet() 메소드를 이용해서 반복적으로 접근하는 것도 가능

// for each

// for each는 자바에서 제공하는 또다른 반복문

// 반복가능한 객체(배열, ArrayList, ..)의 원소의 개수만큼 반복하는 반복문

// 원소의 개수만큼 반복하면서 원소를 꺼내온다!

int[] arr = { 10, 20, 30 };

for (int value : arr) {

System.out.println(value);

}

// 이 방법을 응용하면 entrySet() 메소드를 이용해서 map 전체를 순회할 수 있다

for (Map.Entry<String, String> item : map.entrySet()) {

System.out.printf("key : %s, value : %s\n", item.getKey(), item.getValue());

}

// 키만 확인

System.out.println(map.keySet());

// 값만 확인

System.out.println(map.values());

// 원소를 삭제하고 싶은 경우에는 remove() 메소드를 사용

// 삭제는 키를 통해서 가능

map.remove("first");

System.out.println(map.entrySet());

// 맵의 크기는 size() 메소드를 통해서 확인

System.out.println(map.size());

}

}