# Java 프로그래밍

<지난 시간 개념 복습>

```
package com.day09;
class Point{
	//private int x;
	//private int y;
	protected int x, y;
	public Point(int x, int y) {
		this.x = x;
		this.y = y;
	}
	public void showPoint() {
		System.out.println("("+x+", "+y+")");
	}

}
class  ColorPoint extends Point{
	private String color;
	public ColorPoint(int x, int y, String color) {
		super(x,y);
		this.color =color;
	}
	@Override
	public void showPoint() {
		//System.out.println("("+super.x+", "+super.y+")"+color);
		super.showPoint();
		System.out.println(color);
	}

	@Override
	public String toString() {
		return color +"색의 ("+ super.x +"," + super.y+")";
	}
}
public class ColorPointEx {

	public static void main(String[] args) {
		 Point p  = new Point(3,4);
		p.showPoint() ; //(3,4)
		ColorPoint cp =  new ColorPoint(2,5,"red");
		cp.showPoint() ; //(2,5) red

		Point p1 = new ColorPoint(7,9,"yellow");
		p1.showPoint() ; //(7,9) yellow
		System.out.println(p1);  //yellow 색의 (7,9)

	}

}
```

<Wrapper Class>

```
package com.day09;
class Point{
	//private int x;
	//private int y;
	protected int x, y;
	public Point(int x, int y) {
		this.x = x;
		this.y = y;
	}
	public void showPoint() {
		System.out.println("("+x+", "+y+")");
	}

}
class  ColorPoint extends Point{
```

Wrapper 클래스란 기본 데이터 타입(primitive type)을 객체로 감싸는 클래스를 말합니다. 기본 데이터 타입은 int, double, char, boolean 등과 같은 데이터를 표현하는 타입이며, Wrapper 클래스는 이러한 기본 데이터 타입을 객체로 다룰 수 있게 해줍니다.

```
package com.day09;

// boolean - Boolean
// int - Integer
// char - Character
// short - Short
// long - Long
// double - Double
// float - Float
// byte - Byte

public class WrapperTest {
	int a;

	public void setValue(Integer i) {
		this.a = i;
	}

	public Integer returnValue() {
		return 5; // Integer(리턴형) <- 5(int)
	}

	public static void main(String[] args) { // i(Integer) = 20(int);
		int value = 7; // int != Integer
		Integer val = new Integer(value);
```

```
		WrapperTest test = new WrapperTest();
		test.setValue(20);
	}
}

package com.day09.collection;

public class MemberArrayListTest {


	public static void main(String[] args) {
	MemberArrayList memberArrayList = new MemberArrayList();

	Member memberLee = new Member(1001, "이지원");
	Member memberSon = new Member(1002, "손민국");
	Member memberPark = new Member(1003, "박서훤");
	Member memberHong = new Member(1004, "홍길동");

	memberArrayList.addMember(memberLee);
	memberArrayList.addMember(memberSon);
	memberArrayList.addMember(memberPark);
	memberArrayList.addMember(memberHong);

	memberArrayList.showAllMember();
	memberArrayList.removeMember(memberHong.getMemberId());
	boolean flag = memberArrayList.removeMember(memberHong.getMemberId());;
	if(flag == true) {
		System.out.println("삭제 성공");
	}
	else {
		System.out.println(memberHong.getMemberId()+"삭제 실패");
	}
	memberArrayList.showAllMember();
}
}

package com.day09.collection;

import java.util.ArrayList;

public class MemberArrayList {
	private ArrayList<Member> arrayList;

	public MemberArrayList() {
		arrayList = new ArrayList<Member>();
	}

	public void addMember(Member member) {
		arrayList.add(member);
	}
```

```
	// 삭제
	public boolean removeMember(int memberId) {
		for (Member m : arrayList) {
			if (m.getMemberId() == memberId) {
				arrayList.remove(m);
				return true;
			}
		}
		return false;
	}

	public void showAllMember() {
		for (Member m : arrayList) {
			System.out.println(m);
		}

		System.out.println();
	}

}


package com.day09.collection;

public class Member {
private int memberId;
private String memberName;

	public Member(int memberId, String memberName) {
		this.memberId=memberId;
		this.memberName=memberName;
	}

	@Override
		public String toString() {
			// TODO Auto-generated method stub
			return memberName+" 회원님의 아이디는 " + memberId + "입니다.";
		}

	//getter
	public int getMemberId() {
		return memberId;
	}

	public String getMemberName() {
		return memberName;
	}
```

}

<Map>
Map은 자바에서 제공하는 인터페이스 중 하나로, Key-Value 쌍으로 데이터를 저장하는 자료구조를 구현하는 데 사용됩니다. Map은 데이터를 검색하고 저장하는 데에 있어서 매우 효율적인 구조입니다.

인터페이스 = 추상 클래스

```
package com.day09;

import java.util.HashMap;
import java.util.Set;

public class HashMapTest {

	public static void main(String[] args) {
		HashMap<String, String> hm = new HashMap<>();
		hm.put("one", "첫번째"); // map 추가
		hm.put("two", "두번째");
		hm.put("three", "세번째");
		hm.put("four", "네번째");
		System.out.println(hm); // 순서 없음, 중복 허용 안 함(key에 대한 중복을 허용하지 않음.)
		System.out.println("=======================================");
		hm.put("one", "첫번째첫번째");
		System.out.println(hm);
		System.out.println("=======================================");
		hm.put("oneone", "첫번째첫번째");
		System.out.println(hm);
		System.out.println("크기 : " + hm.size()); // map의 크기
		System.out.println(hm.get("two")); // value값을 알려준다.
		hm.remove("oneone");
		System.out.println(hm); // map 제거, 지운 것을 알 수 있음.
		String value = hm.remove("four");
		System.out.println("제거 : " + value);
		System.out.println(hm); // 제거 확인
		System.out.println("=======================================");
		HashMap<String, String> h = new HashMap<>();
		h.put("1", "딸기1");
		h.put("2", "딸기2");
		h.put("3", "딸기3");
		h.put("4", "딸기4");
		// h에 있는 value 값 출력
//		System.out.println(h.get("1"));
//		System.out.println(h.get("2"));
//		System.out.println(h.get("3"));
//		System.out.println(h.get("4"));
//		System.out.println(h.get(i));
		for (int i = 1; i < 5; i++) {
```

```
//		System.out.println(h.get(i+""));
		System.out.println(h.get(String.valueOf(i)));
	}
	System.out.println("=======================================");
	// h 해쉬맵에서 키값이 "1"이 있는가?
	System.out.println(h.containsKey("1"));
	// h라는 해쉬맵에서 value값에서 "딸기"가 있나요?
	System.out.println(h.containsValue("딸기")); // 값이 없기 때문에 false로 출력됨
	System.out.println("=======================================");
	// h라는 해쉬맵에서 키값만 출력
	System.out.println(h.keySet()); // key만 나오게 한다.
	Set<String> set = h.keySet();
	System.out.println("set : " + set);
	// h라는 해쉬맵에서 value 값만 출력
	System.out.println(h.values());
	System.out.println("=======================================");

	HashMap<String, String> hash = new HashMap<>();
	hash.put("one", "바나나1");
	hash.put("two", "바나나2");
	hash.put("three", "바나나3");
	hash.put("four", "바나나4");
	// value 값만 출력
	System.out.println(hash.values());
	// get을 이용해서 value 값을 출력
	for (String key : hash.keySet()) {
		System.out.println("value = " + hash.get(key));
	}
	for (String v : hash.values()) {
		System.out.println("v = " + v);
	}

	HashMap<Integer, String> h1 = new HashMap<>(); // 자료형 선언에는 class형
써줘야함.
	h1.put(1, "사과");
	h1.put(2, "사과2");
	h1.put(3, "사과3");
	h1.put(4, "사과4");
	for(int i=1; i<=h1.size(); i++) {
		System.out.println(h1.get(i));
	}

	}
}
```

<ArrayList와 Map의 차이점>
ArrayList 순서가 있고 중복 허용, Map은 순서가 없으며 중복 허용하지 않음.

```
package com.day09;
```

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Random;

public class HashMapTest01 {
/*
 * 1. HashMap hm 생성
 * 2. Math.random 함수를 이용하여 1~20까지의 난수를 발생
 * 3. 난수 10개를 hm에 추가 value값은 중복허용하지 않음.
 */

	public static void main(String[] args) {
	HashMap<Integer, Integer> hm = new HashMap<>();
	for(int i=1; i<11; i++) {
		while(true) {
			int v = (int)(Math.random()*20)+1;
			if(!hm.containsValue(v)) {
			hm.put(i, v);
			break;
		} //if
	} // while

}
	System.out.println(hm);

	// value 값만 출력
	System.out.println(hm.values());
	// get이용해서 value 출력
	for(Integer num : hm.keySet()) {
		System.out.println("value = "+hm.get(num));
	}
	// value를 오름차순으로 정렬
	List<Integer> list = new ArrayList<>(hm.values()); //List형 생성
	// ArrayList<Integer> list = new ArrayList<>(hm.values()); - 똑같은 표현.
	Collections.sort(list); // List형으로 해야한다.
	System.out.println();
	System.out.println("=== 정렬 후 ==="); // ascending
	for(int i : list)
		System.out.print(i+"\t");
	System.out.println();
	System.out.println("\n=== reverse ==="); // descending
	Collections.reverse(list);
	for(int i : list)
		System.out.print(i+"\t");
	System.out.println();
	System.out.println("\n=== max & min ===");
	System.out.println("최대값 : "+Collections.max(list));
	System.out.println("최소값 : "+Collections.min(list));
```

```
	}
}
```

```
package com.day09.collection;

import java.util.HashMap;
import java.util.Scanner;

public class HashTest02 {
	/* 1. HashMap map을 생성
	 * 2. 단어를 입력받아서 단어와 입력 횟수 저장
	 * 3. 대소문자 구분없이 X를 입력시 종료
	 * 4. map 출력
	 * ex)java test java test java test test test abc x
	 */


	public static void main(String[] args) {
	HashMap<String, Integer> map = new HashMap<>();
	Scanner sc = new Scanner(System.in);
	System.out.println("단어를 입력하세요.");


	while(true) {
		String word=sc.next();
		int value;
		if(word.equalsIgnoreCase("x")) break;
	/*if(map.containsKey(word)) {
		// map.put(word, map.get(word)+1);
		value = map.get(word)+1;
		}else {
			// map.put(word, 1);
			value = 1;
		} */


		map.put(word, map.containsKey(word)? map.get(word)+1 : 1); // 3항 연산자로
1줄 요약해서 표현


		// map.put(word, value);
}


	System.out.println(map);

	}
}
```

<반복자 - Iterator>
객체(Object)를 반복문 같이 돌려주는 것
Iterator(반복자)를 사용하게 됨으로 객체를 순회하면서 요소에 접근하기 위한 인터페이스입니다.
Iterator를 사용하여 컬렉션의 요소를 순차적으로 접근하고 수정할 수 있습니다.

```
package com.day09.collection;

import java.util.Iterator;
import java.util.Vector;

public class IteratorEx {

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

```
		Vector<Integer> vc = new Vector<Integer>();
		vc.add(5);
		vc.add(-1);
		vc.add(new Integer(4));
		for (Integer i : vc) {
			System.out.println(i);
		}
		vc.add(2, 100);
		System.out.println("===");
		for (Integer i : vc) {
			System.out.println(i);
		}
		System.out.println("===");
		Iterator<Integer> it = vc.iterator(); // for문으로도 사용이 가능하지만 객체 반복 반환은 Iterator를 사용.
		while (it.hasNext()) {
		System.out.println(it.next());
		}


	}

}

package com.day09.collection;

import java.util.ArrayList;

// p.419

class MyStack{
	private ArrayList<String> arrayStack = new ArrayList<String>();


	public void push1(String string) {
	arrayStack.add(string);
	}

	public String pop1() {
		int len = arrayStack.size();
		if(len == 0) {
			System.out.println("스택이 비었습니다.");
			return null;
		}
		return(arrayStack.remove(len-1));
	}

}

public class StackTest {
	public static void main(String[] args) {
		MyStack stack = new MyStack();
		stack.push1("A");
		stack.push1("B");
		stack.push1("C");
		System.out.println(stack.pop1());
		System.out.println(stack.pop1());
		System.out.println(stack.pop1());
}
}
```