08장 컬렉션

자바의 컬렉션

◈ 컬렉션(Collection)

다수의 데이터를 쉽게 처리할 수 있는 표준화된 방법을 제공

배열의 단점을 보완한 배열의 발전된 모델

배열은 크기를 동적으로 늘릴 수 없지만, 컬렉션은 동적으로 메모리를 확장할 수 있다.

자바의 컬렉션

◈ 컬렉션의 종류

리스트(List) : 데이터를 1차원으로 늘어놓은 형태의 자료구조

스택(Stack) : 마지막에 넣은 데이터부터 순서대로 꺼낼 수 있는 자료구조

큐(Queue) : 입력된 순서대로 데이터를 꺼낼 수 있는 자료구조

맵(Map) : 키(key)를 이용해서 데이터를 검색하는 자료구조

집합(Set) : 순서없이 데이터만을 저장, 데이터의 중복 저장을 허용하지 않는 자료구조

자바의 컬렉션

◈ 리스트(ArrayList)

인덱스로 객체를 검색, 삭제할 수 있는 기능을 제공

기능	메서드	설명
객체 추가	boolean add(E e)	주어진 객체를 맨 끝에 추가
	void add(int index, E element)	주어진 인덱스에 객체를 추가
	set(int index, E element)	주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	E get(int index)	주어진 인덱스에 저장된 객체를 리턴
	isEmpty()	컬렉션이 비어 있는지 조사
	int size()	저장되어 있는 전체 객체 수를 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	E remove(int index)	주어진 인덱스에 저장된 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

자바의 컬렉션

◈ 리스트(ArrayList) 의 선언

ArrayList<담을 타입> 리스트명 = new ArrayList<담을 타입>();

import java.util.ArrayList;

```
public static void main(String[] args) {
    // 리스트(ArrayList) 선언
    ArrayList<String> list1 = new ArrayList<String>();
    ArrayList<String> list2 = new ArrayList<>();          // new 부분에서는 제네릭<> 안에 타입을 생략해도 된다.
    ArrayList<Integer> list3 = new ArrayList<>();         // 기본 타입의 경우(숫자와 boolean 타입), Wrapper Class를 이용해야 한다.

    // 값을 담아주며 선언
    ArrayList<String> list4 = new ArrayList<>(Arrays.asList("C", "A", "B", "a"));
}
```

자바의 컬렉션

◈ 제네릭(Generic)

괄호<> 안에 해당 컬렉션에 담을 데이터의 타입 또는 클래스를 지정해준다.

잘못된 타입이 배열에 담기는 것을 컴파일 단계에서 방지할 수 있다.

타입이 지정되어 있기 때문에 컬렉션에서 데이터를 꺼내어 쓸 때 따로 타입 검사를 하지 않아도 된다.

기본 타입의(Primary Type) 경우 Wrapper 클래스를 이용한다.

자바의 컬렉션

◈ Wrapper 클래스

기본 데이터 타입의 값을 담는 객체를 생성하는 클래스

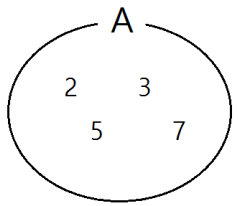
클래스 이름	해당 기본데이터 타입
Byte	byte
Short	short
Integer	int
Long	long
Character	char
Float	float
Double	double
Boolean	boolean

자바의 컬렉션

◈ 집합(HashSet)

Set 컬렉션은 저장 순서가 유지되지 않는다.

객체를 중복 저장할 수 없다. 수학의 집합과 유사하다.



기능	메서드	설명
객체 추가	boolean add(E e)	주어진 객체를 저장, 객체가 성공적으로 저장되면 true를 리턴하고 중복 객체면 false를 리턴
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	Iterator<E> iterator()	저장된 객체를 한 번씩 가져오는 반복자 리턴
	isEmpty()	컬렉션이 비어 있는지 조사
	int size()	저장되어 있는 전체 객체 수를 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

자바의 컬렉션

◈ 집합(HashSet)의 선언

HashSet<담을 타입> 집합명 = new HashSet <담을 타입>();

```
import java.util.HashSet;
```

```
public static void main(String[] args) {
    // 집합(HashSet)의 선언
    HashSet<String> set1 = new HashSet<String>();
    HashSet<String> set2 = new HashSet<>();      // new 부분에서는 제너릭 타입 생략 가능
    Set<String> set3 = new HashSet<String>();    // 변수의 타입을 Set으로 해도 괜찮다(다형성)

    // 값을 담아주며 선언
    HashSet<String> set4 = new HashSet<>(Arrays.asList("C", "A", "B", "a"));
}
```

자바의 컬렉션

◈ 컬렉션 검색

리스트처럼 숫자형 인덱스가 존재하지 않는 컬렉션에 대해 전체 데이터를 검색하는 방법

◈ Iterator 사용

```
public static void main(String[] args) {
    Set<String> sets = new HashSet<>();
    sets.add("a");
    sets.add("b");
    sets.add("c");

    Iterator<String> iterator = sets.iterator();

    while(iterator.hasNext()) {
        String content = iterator.next();
        System.out.println(content);
    }
}
```

자바의 컬렉션

◈ 향상된 For문(Foreach) 사용

컬렉션의 항목 개수만큼 반복하며, 반복이 이루어질 때마다 배열의 항목을 순서대로 꺼내 변수에 대입해준다.

```
public static void main(String[] args) {
    Set<String> sets = new HashSet<>();
    sets.add("a");
    sets.add("b");
    sets.add("c");

    for(String content : sets) {
        System.out.println(content);
    }
}
```

자바의 컬렉션

◈ 맵(HashMap)

맵은 키(Key)와 값(Value)으로 구성된 Entry 객체를 저장하는 구조를 가진다.

동일한 키로 값을 저장하면 기존의 값은 없어지고 새로운 값으로 대체된다.

기능	메서드	설명
객체 추가	V put(K key, V value)	주어진 키와 값을 추가, 저장되면 값을 리턴
객체 검색	boolean containsKey(Object key)	주어진 키가 있는지 여부
	boolean containsValue(Object value)	주어진 값이 있는지 여부
	Set<Map.Entry<K,V>> entrySet()	키와 값의 쌍으로 구성된 모든 Map.Entry 객체를 Set에 담아서 리턴
	V get(Object key)	주어진 키의 값을 리턴
	boolean isEmpty()	컬렉션이 비어 있는지 여부
	Set<K> keySet()	모든 키를 Set 객체에 담아서 리턴
	int size()	저장된 키의 총 수를 리턴
	Collection<V> values()	저장된 모든 값을 Collection에 담아서 리턴
객체 삭제	void clear()	모든 Map.Entry(키와 값)를 제거
	V remove(Object key)	키(key)를 이용해서 데이터를 제거

자바의 컬렉션

◈ 맵(HashMap)의 선언

HashMap<담을 타입> 맵명 = new HashMap <담을 타입>();

import java.util.HashMap;

```
public static void main(String[] args) {
    // 맵(HashMap)의 선언
    HashMap<String, String> stuMap = new HashMap<String, String>();
    HashMap<String, String> stuMap2 = new HashMap<>();       // new 부분에서는 제너릭 타입 생략 가능

    Map<String, String> stuMap3 = new HashMap<String, String>();   // 변수의 타입을 Map으로 해도 괜찮다(다형성)
}
```

자바의 컬렉션

◈ 맵 검색

keySet() 과 entrySet() 을 사용

```
public static void main(String[] args) {
    Map<String, String> map=new HashMap<>();
    map.put("a", "에이");
    map.put("b", "비");
    map.put("c", "씨");

    Set<String> keySet = map.keySet();

    for(String key : keySet) {
        String content = map.get(key);
        System.out.println(content);
    }
}
```

```
public static void main(String[] args) {
    Map<String, String> map=new HashMap<>();
    map.put("a", "에이");
    map.put("b", "비");
    map.put("c", "씨");

    Set<Entry<String,String>> entrySet = map.entrySet();

    for(Entry<String,String> entry : entrySet) {
        String key = entry.getKey();
        System.out.println(key);
        String value = entry.getValue();
        System.out.println(value);
    }
}
```

자바의 컬렉션

◈ 프로퍼티(Properties) 클래스

프로퍼티는 키와 값을 String 타입으로 제한한 컬렉션이다.

어플리케이션의 옵션 정보, 데이터베이스 연결정보, 국제화(다국어) 정보 등을 저장한 *.properties 파일을 읽어올 때 주로 사용한다.

메소드		기능
String	getProperty(String key)	인자로 전달된 key를 가지는 속성값을 찾아 반환한다.
void	list(PrintWriter out)	인자로 전달된 출력 스트림을 통해 속성 목록들을 출력한다.
void	load(InputStream inStream)	인자로 전달된 입력 스트림으로부터 키와 요소가 한 쌍으로 구성된 속성 목록들을 읽어들여 현 Properties 객체에 저장한다.
Enumeration<?>	propertyName()	속성 목록에 있는 모든 속성의 key값들을 열거형 객체로 반환한다.
Object	setProperty(String key, String value)	현 Properties 객체의 속성 목록에 인자로 전달된 key 와 value를 한 쌍으로 구성되어 저장한다. 내부적으로는 Hashtable의 put()메소드가 호출된다.
void	store(OutputStream out, String comments)	모든 속성들을 load() 메소드를 사용해 Properties 테이블에 로드하고 적절한 포맷과 인자로 전달된 출력 스트림을 통해 출력한다.