Chapter 20

API – java.util 패키지

API – java.util 패키지

*** java.util 패키지**

- java.util 패키지는 자바 프로그램 개발에 보조 역할을 하는 클래스들을 담고 있습니다. 주로 컬렉션 관련 클래스들을 담고 있습니다.

- java.util 패키지 주요 클래스

1. Arrays: 배열을 조작할 때 사용.
2. Date: 날짜와 시간 정보를 저장하는 클래스
3. Calendar: 운영체제의 날짜와 시간을 얻을 때 사용
4. Random: 난수를 얻을 때 사용.

Arrays클래스

* Arrays 클래스

- Arrays 클래스는 배열 조작 기능을 가지고 있습니다. 배열의 복사, 항목 정렬, 항목 검색과 같은 기능을 말합니다.

- Arrays 클래스의 모든 메서드는 정적 메서드이므로 클래스이름으로 바로 사용이 가능합니다.

* Arrays 클래스 주요 메서드

1. binarySearch(배열, 찾는값): 전체 배열 항목에서 찾는 값이 있는 인덱스를 리턴.

2. copyOf(원본배열, 복사할길이): 원본 배열의 0번 인덱스부터 복사할 길이만큼의 인덱스까지 복사한 배열을 리턴.

3. copyOfRange(원본배열, 시작인덱스, 끝인덱스): 원본 배열의 시작 인덱스에서 끝 인덱스까지 복사한 배열 리턴, 시작 인덱스는 포함되지만 끝 인덱스는 포함되지 않음.

4. sort(배열): 배열의 전체 항목을 오름차순으로 정렬.

5. toString(배열): 배열의 값들을 "[값1, 값2,...]"와 같은 문자열 형식으로 리턴.

copyOf 메서드 사용 예

```
public static void main(String[] args) {
   double[] arOrg = {1.1, 2.2, 3.3, 4.4, 5.5};

   // 배열 전체를 복사
   double[] arCpy1 = Arrays.copyOf(arOrg, arOrg.length);

   // 세번째 요소까지만 복사
   double[] arCpy2 = Arrays.copyOf(arOrg, 3);

   for(double d : arCpy1) {
      System.out.print(d + "\t");
   }
   System.out.println();


   for(double d : arCpy2) {
      System.out.print(d + "\t");
   }
   System.out.println();
}
```

배열 전체 복사
vs
길이 3 복사

sort(), binarySearch(), toString(), equals() 예

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

	int[] scores = {75, 47, 23, 56, 89};

	//배열의 데이터를 오름차순으로 정렬하는 메서드 sort()
	int[] scores2 = Arrays.copyOf(scores, scores.length);
	Arrays.sort(scores2);

	//배열내부데이터의 인덱스를 탐색하는 메서드 binarySearch()
	int index = Arrays.binarySearch(scores2, 75);
	System.out.println("75가 있는 인덱스번호: " + index);

	//배열을 문자열로 toString()
	System.out.println(Arrays.toString(scores2));

	//배열 비교 equals(배열1, 배열2)
	int[] scores3 = Arrays.copyOf(scores, scores.length);


	scores3[2] = 140;
	//확인하면 안일치해~
	if(Arrays.equals(scores, scores3)) {
		System.out.println("배열의 각 항목이 모두 일치함.");
	}else {
		System.out.println("배열의 각 항목이 일치하지 않음.");
	}

}
```

실행 결과

```
75가 있는 인덱스번호: 3
[23, 47, 56, 75, 89]
배열의 각 항목이 일치하지 않음.
```

Date클래스 – 날짜, SimpleDateFormat클래스 - 날짜포멧

* Date 클래스

- Date 클래스는 날짜를 표현하는 클래스입니다. 객체 간에 날짜 정보를 주고받을 때 주로 사용합니다.

- Date 객체를 기본생성자로 생성하면 컴퓨터의 현재 날짜를 읽어 객체로 만듭니다.

- Date 클래스의 toString() 메서드는 영문으로 된 날짜를 문자열로 리턴하는데 만약 특정 문자열 포맷으로 얻고 싶다면 java.text 패키지의 SimpleDateFormat 클래스를 이용하면 됩니다.

* SimpleDateFormat

- SimpleDateFormat 클래스는 날짜를 원하는 형식으로 표현하기 위한 클래스입니다. java.text 패키지에 구성되어 있는 API입니다.

- 패턴을 사용하여 생성자의 매개값으로 표현형식을 지정하여 객체를 생성한 후 format() 메서드를 호출하여 패턴이 적용된 문자열을 얻을 수 있습니다.

Date클래스 예제

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

	Date date = new Date();

	System.out.println(date.toString());
	//특정 문자열 포맷으로 얻고싶다면 SimpleDateFormat 클래스 사용
	SimpleDateFormat sdf = new SimpleDateFormat("yyyy년 MM월 dd일 HH시 mm분 ss초");
	String time = sdf.format(date);
	System.out.println(time);


	sdf = new SimpleDateFormat("yy-MM-dd a hh:mm:ss");
	System.out.println(sdf.format(date));

	sdf = new SimpleDateFormat("오늘은 E요일입니다. 오늘은 1년 중 D번째 날입니다.");
	System.out.println(sdf.format(date));

}
```

실행결과

```
Mon Aug 05 00:51:42 KST 2019
2019년 08월 05일 00시 51분 42초
19-08-05 오전 12:51:42
오늘은 월요일입니다. 오늘은 1년 중 217번째 날입니다.
```

Random클래스

* Random 클래스

- Random 클래스는 난수를 얻어내기 위한 다양한 메서드를 제공하며 Math클래스의 random() 메서드보다 다양한 난수값을 얻을 수 있게 해줍니다.

* Random 클래스 주요 메서드

1. nextBoolean(): boolean 타입의 난수를 리턴(true or false)
2. nextDouble(): double 타입의 난수를 리턴(0.0 <= ~ < 1.0)
3. nextInt(): int타입 난수를 리턴(int의 범위)
4. nextInt(int n): int타입 난수를 리턴(0 <= ~ < n)

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

	Random r = new Random();

	//0.0이상 1.0미만의 실수 난수를 리턴.

	int i = r.nextInt(10);
	System.out.println("정수(0이상 10미만) 랜덤값: " + i);

	int i = r.nextInt(10) + 1;
	System.out.println("정수(1이상 10이하) 랜덤값: " + i);

}
```

실행결과

```
정수(0이상 10미만) 랜덤값: 3
정수(1이상 10이하) 랜덤값: 4
```

Chapter 20

API – java.util 패키지(컬렉션 프레임워크)

제네릭 이란?

* 제네릭(generic)

- 제네릭이란 클래스나 인터페이스 선언에 유형 매개변수가 들어있는 클래스를 뜻합니다.

- 제네릭 타입은 클래스 또는 인터페이스 이름 뒤에 "<>"부호가 붙고, 그 사이에 파라미터가 위치합니다.

- 자바 5 버전부터 제네릭이 도입된 이후에는 제네릭 기능으로 인해 클래스에 원하지 않는 데이터형이 들어가는 것을 방지할 수 있고, 반대로 값을 가져올 때도 형 변환을 하지 않게 되었습니다.

- 제네릭은 형 안정성(type safety)을 위해 사용합니다.

제네릭이 없는 코드가 갖는 문제

```
public class ABC {

	private Object obj;

	public void setObj(Object obj) {
		this.obj = obj;
	}

	public Object getObj() {
		return obj;
	}

}
```

ABC상자는 무엇이든 담을 수 있다.

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

	ABC abc = new ABC(); //ABC생성

	abc.setObj("홍길동");                  //이름을 저장
	String name = (String)abc.getObj(); //이름을 꺼낸다

	abc.setObj(new DEF());                 //DEF도 저장
	DEF def = (DEF)abc.getObj();           //DEF를 꺼낸다
}
```

어쩔 수 없이 해당 타입에 맞춰서 강제 형 변환이 일어나야 한다
잘못 형 변환 한다면 오류로 이어지는 결과가 된다

제네릭 등장 이후 문제 해결

제네릭 클래스의 설계 <type>

```
public class ABC<T> {

	private T t;

	public void set(T t) {
		this.t = t;
	}

	public T get() {
		return t;
	}

}
```

클래스의 저장되는 타입은 미정상태!

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

	ABC<String> abc = new ABC<String>();

	abc.set("홍길동");		//저장
	String name = abc.get();	//꺼냄


	ABC<DEF> abc2 = new ABC<>();

	abc2.set(new DEF());		//저장
	DEF def = abc2.get();		//꺼냄

}
```

값을 꺼내는데 형 변환 하지 않는다
new로 생성할 때 클래스를 다양한 형태로 만들어 쓸 수 있다

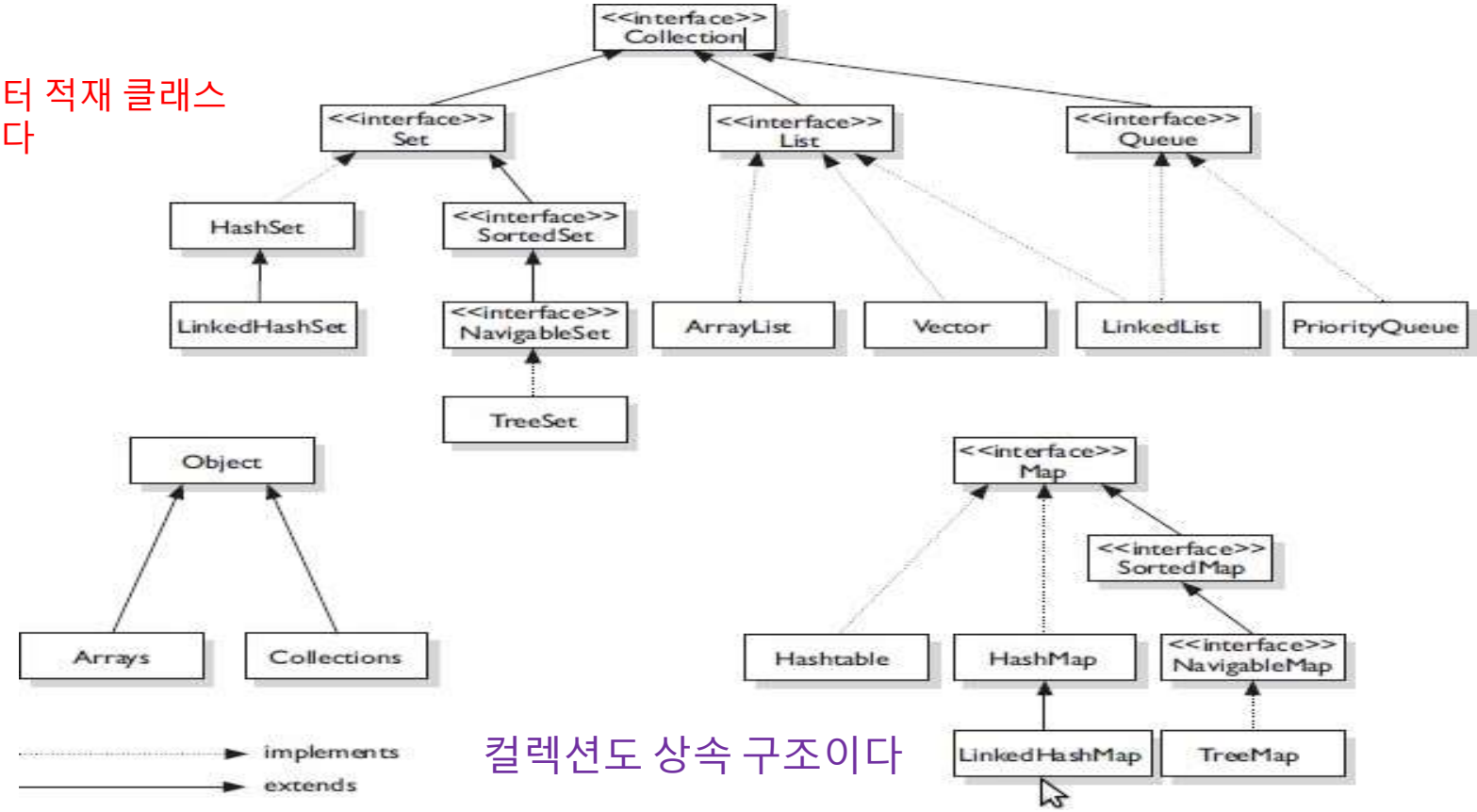
Collection Framework

* Collection

- 컬렉션은 배열과 유사하지만 데이터를 저장/조회/수정/삭제하는 작업을 쉽게 처리할 수 있으며, 동적인 크기를 갖는다는 장점이 있습니다.

- 컬렉션 계열은 Set/List/Map 등의 인터페이스가 있으며 이를 구현한 클래스를 이용하면 객체들을 모음저장할 수 있습니다.

컬렉션은 데이터 적재 클래스
자료구조 입니다



컬렉션도 상속 구조이다

List, Set, Map 계열

- Collection 인터페이스의 하위 클래스들
  - Set 계열
    - Set 인터페이스를 구현한 클래스들
    - 순서 X, 중복 허용 X
    - HashSet, TreeSet
  - List 계열
    - List 인터페이스를 구현한 클래스들
    - 순서 O, 중복 허용 O
    - ArrayList, LinkedList, Queue, Stack(Vector)
  - Map 계열
    - Map 인터페이스를 구현한 클래스들
    - HashMap, TreeMap

- 객체를 저장할 수 있는 자료구조들을 제공합니다.
- 컬렉션에 저장된 객체(or 변수)들을 엘리먼트라 합니다.
- 배열과 달리 동적인 공간을 갖습니다.
- 컬렉션마다 관리할수있는 메서드가 존재합니다.
- 기존의 배열에 비해 높은 성능을 보장합니다.

인터페이스	순서	중복	구현된 클래스
Collection	X	O	
Set	X	X	HashSet TreeSet
List	O	O	ArrayList LinkedList

List계열 (순서 o, 중복 o)

*** List 컬렉션**

- List 컬렉션은 객체를 인덱스로 관리하기 때문에 객체를 저장하면 자동으로 인덱스 번호가 부여되고 인덱스를 통해 객체를 검색, 삭제할 수 있는 기능을 제공합니다.

- List 는 객체를 순서대로 저장하며 동일한 객체를 중복 저장할 수 있습니다.

*** List 계열 주요 메서드**

- 객체 추가 기능
1. add(E e): 주어진 객체를 List의 맨 끝부분에 추가.
2. add(int index, E e): 주어진 인덱스에 객체를 추가.
3. set(int index, E e): 주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈.

- 객체 검색 기능
1. contains(Object o): 주어진 객체가 저장되어있는지의 여부를 판단.
2. get(int index): 주어진 인덱스에 저장되어 있는 객체를 리턴.
3. isEmpty(): 컬렉션이 비어있는지의 여부를 판단.
4. size(): 저장되어 있는 전체 객체 수를 리턴.

- 객체 삭제 기능
1. clear(): 저장된 모든 객체를 삭제.
2. remove(int index): 주어진 인덱스에 저장된 객체를 삭제.
3. remove(Object o): 주어진 객체를 삭제.

ArrayList예제

- 배열과 흡사 하지만, 자동 사이즈를 조절하는 ArrayList
- ArrayList에 특정 인덱스의 객체를 제거하면 자동으로 바로 뒤 인덱스부터 마지막 인덱스까지
- 모두 앞으로 1칸씩 당겨집니다.

```
1:  import java.util.LinkedList;
2:
3:  public class LinkedListExample {
4:      public static void main(String[] args) {
5:          LinkedList<String> list = new LinkedList<String>();
6:
7:          list.add("hello");
8:          list.add("java");
9:          list.add("banana");
10:         list.addFirst("apple");
11:         list.addLast("zoo");
12:
13:         System.out.println("list data : " + list);
14:
15:         list.remove();      //head 엘리먼트 삭제
16:         System.out.println("list data after remove() : " + list);
17:
18:         list.remove(2);     //2번 인덱스 엘리먼트 삭제
19:         System.out.println("list data after remove(2) : " + list);
20:
21:         list.set(1, "new element"); //1번째 엘리먼트 변경
22:         System.out.println("list data after set() : " + list);
23:
24:         String str1 = list.peek();   //엘리먼트 조회
25:         System.out.println("str1 : " + str1);
26:         System.out.println("list data after peek() : " + list);
27:
28:         String str2 = list.poll();   //엘리먼트 조회 후 삭제
29:         System.out.println("str2 : " + str2);
30:         System.out.println("list data after poll() : " + list);
31:     }
32: }
```

```
Console
<terminated> LinkedListExample [Java Application] C:₩Program Files₩Java₩jre1.8.0_91₩
list data : [apple, hello, java, banana, zoo]
list data after remove() : [hello, java, banana, zoo]
list data after remove(2) : [hello, java, zoo]
list data after set() : [hello, new element, zoo]
str1 : hello
list data after peek() : [hello, new element, zoo]
str2 : hello
list data after poll() : [new element, zoo]
```

LinkedList 예제

- ArrayList는 내부 배열에 객체를 저장하여 인덱스로 관리하지만 LinkedList는 인접 참조를 링크하여 체인처럼 관리합니다.
- LinkedList는 특정 인덱스의 객체를 제거하면 앞 뒤 링크만 변경되고 나머지 링크는 변경되지 않아 빈번한 객체의 삭제와 삽입은 ArrayList보다 좋은 성능을 발휘합니다.

```
1:  import java.util.LinkedList;
2:
3:  public class LinkedListExample {
4:      public static void main(String[] args) {
5:          LinkedList<String> list = new LinkedList<String>();
6:
7:          list.add("hello");
8:          list.add("java");
9:          list.add("banana");
10:         list.addFirst("apple");
11:         list.addLast("zoo");
12:
13:         System.out.println("list data : " + list);
14:
15:         list.remove();      //head 엘리먼트 삭제
16:         System.out.println("list data after remove() : " + list);
17:
18:         list.remove(2);     //2번 인덱스 엘리먼트 삭제
19:         System.out.println("list data after remove(2) : " + list);
20:
21:         list.set(1, "new element"); //1번째 엘리먼트 변경
22:         System.out.println("list data after set() : " + list);
23:
24:         String str1 = list.peek();   //엘리먼트 조회
25:         System.out.println("str1 : " + str1);
26:         System.out.println("list data after peek() : " + list);
27:
28:         String str2 = list.poll();   //엘리먼트 조회 후 삭제
29:         System.out.println("str2 : " + str2);
30:         System.out.println("list data after poll() : " + list);
31:     }
32: }
```

Console

<terminated> LinkedListExample [Java Application] C:₩Program Files₩Java₩jre1.8.0_91₩

```
list data : [apple, hello, java, banana, zoo]
list data after remove() : [hello, java, banana, zoo]
list data after remove(2) : [hello, java, zoo]
list data after set() : [hello, new element, zoo]
str1 : hello
list data after peek() : [hello, new element, zoo]
str2 : hello
list data after poll() : [new element, zoo]
```

Set계열 (순서 o, 중복 x)

*** Set 계열 컬렉션**

- Set 컬렉션은 저장 순서를 보장하지 않으며 객체의 중복 저장을 허용하지 않습니다.
- Set 컬렉션은 인덱스로 관리하지 않으며 들어갈 때의 순서와 나올 때의 순서가 다를 수도 있습니다.
- Set 컬렉션은 인덱스로 객체를 검색하는 기능이 없고 전체 객체를 대상으로 한번씩 반복해서 객체의 값을 가져오는 반복자(Iterator)를 제공합니다.

- Iterator 인터페이스의 주요 메서드

1. hasNext(): 가져올 객체가 있으면 true를 리턴하고 없으면 false를 리턴.
2. next(): 컬렉션에서 하나의 객체를 가져옴.
3. remove(): Set 컬렉션에서 객체를 제거함.

*** Set 계열 컬렉션 주요 메서드**

- 객체 추가 기능
1. add(E e): 주어진 객체를 저장, 성공적으로 저장되면 true를 리턴, 중복 객체를 저장하면 false를 리턴.

- 객체 검색 기능
1. contains(Object o): 주어진 객체가 저장되어 있는지의 여부를 판단.
2. isEmpty(): 컬렉션이 비어있는지를 조사.
3. iterator(): 저장된 객체를 한번씩 가져오는 반복자 객체를 리턴.
4. size(): 저장되어 있는 전체 객체 수를 리턴.

- 객체 삭제 기능
1. clear(): 저장된 모든 객체를 삭제.
2. remove(Object o): 주어진 객체를 삭제.

HashSet 예제

-HashSet 클래스는 Set 인터페이스를 구현한 컬렉션이므로 저장된 객체의 순서를 보장하지 않고 중복을 허용하지 않습니다.

-순차적으로 데이터를 관리하는 것에 비하여 속도가 향상됩니다.

```
1:  import java.util.*;
2:
3:  public class HashSetExample {
4:
5:      public static void main(String args[]) {
6:
7:          Set set = new HashSet();
8:
9:          set.add("three");
10:         set.add("one");
11:         set.add("two");
12:         set.add("four");
13:         set.add("five");
14:         set.add(new Integer(4));
15:         boolean isAdded = set.add("five");
16:
17:         System.out.println(set);
18:         System.out.println(isAdded);
19:
20:         System.out.println(set.size());
21:
22:         set.remove("two");
23:         System.out.println(set);
24:
25:         set.clear();
26:         System.out.println(set);
27:
28:         if (set.isEmpty()) {
29:             System.out.println("set is Empty");
30:         }
31:     }
32: }
```

Console

<terminated> HashSetExample [Java Application] C:₩Pr

```
[4, four, one, three, two, five]
false
6
[4, four, one, three, five]
[]
set is Empty
```

TreeSet 예제

- Tree구조 기반으로 생성된 클래스로 Set기능에 +자동정렬 기능을 갖습니다

```
1:  import java.util.TreeSet;
2:
3:  public class TreeSetExample {
4:
5:      public static void main(String[] args) {
6:          TreeSet<String> ts = new TreeSet<String>();
7:
8:          ts.add("hello");
9:          ts.add("java");
10:         ts.add("aaa");
11:         ts.add("computer");
12:
13:         for(String str : ts) {
14:             System.out.print(str + "\t");
15:         }
16:     }
17: }
```

TreeSet 예제

1. TreeSet을 생성 하세요
2. 무한루프에서 45까지의 난수를 발생시키세요
3. 난수를 추가합니다..
4. 크기가 6이 되면 빠져나오세요

```
TreeSet<Integer> lotto = new TreeSet<>();
Random r = new Random();

while(true) {
    int rn = r.nextInt(45) + 1;

    lotto.add(rn);

    if(lotto.size() == 6) {
        break;
    }
}
```

Map 계열 (key-value)

*** Map 계열 컬렉션**

- Map 컬렉션은 키(key)와 값(value)으로 구성된 Entry객체를 저장하는 구조를 가지고 있습니다.
- 키는 중복저장 될 수 없지만 값은 중복저장 될 수 있습니다.

*** Map 계열 주요 메서드**

- 객체 추가 기능
1. put(K key, V value): 주어진 키와 값을 추가, 정상적으로 저장되면 그 값(value)를 리턴.

- 객체 검색 기능
1. containsKey(Object Key): 주어진 키가 있는지의 여부를 확인.
2. containsValue(Object value): 주어진 값이 있는지의 여부를 확인.
3. get(Object key): 주어진 키에 들어있는 값을 리턴.
4. isEmpty(): 컬렉션이 비어있는지의 여부를 확인.
5. size(): 저장된 키의 총 수를 리턴.
6. values(): 저장된 모든 값을 컬렉션에 담아서 리턴.
7. keySet(): 저장된 모든 키를 Set객체에 담아서 리턴.
8. entrySet(): 키와 값의 쌍으로 구성된 모든 Entry객체를 Set에 담아서 리턴.

- 객체 삭제 기능
1. clear(): 모든 Entry를 삭제
2. remove(Object key): 주어진 키와 일치하는 Entry객체를 삭제.

HashMap 예제

```
8:      Map maps = new HashMap();
9:
10:     String s1 = new String("홍길동");
11:     maps.put("name", s1);
12:     maps.put("hiredate", new Date());
13:     maps.put("salary", 20000);
14:
15:     System.out.println(maps);
16:
17:     System.out.println();
18:     System.out.println(maps.get("hiredate"));
19:     System.out.println(maps.get("salary"));
20:     System.out.println(maps.get("name"));
21:
22:     System.out.println();
23:     //map안의 엘리먼트를 entrySet() 메서드를 이용하여 조회
24:     Set<Map.Entry<String, Object>> s = maps.entrySet();
25:     for(Map.Entry<String, Object> me : s) {
26:        System.out.println(me.getKey() + " : " + me.getValue());
27:     }
28:
29:     System.out.println();
30:     //keySet() 메서드로 map 키를 리턴받고 get(key) 메서드로 조회
31:     Set<String> ss = maps.keySet();
32:     for(String key : ss) {
33:        System.out.println(key + " :: " + maps.get(key));
34:     }
```

Console

<terminated> HashMapExample [Java Application] C:₩Program Files₩Java₩jre1.8.0_91₩bin₩javaw.exe (201

```
{name=홍길동, salary=20000, hiredate=Sat Jul 16 17:45:19 KST 2016}

Sat Jul 16 17:45:19 KST 2016
20000
홍길동

name : 홍길동
salary : 20000
hiredate : Sat Jul 16 17:45:19 KST 2016

name :: 홍길동
salary :: 20000
hiredate :: Sat Jul 16 17:45:19 KST 2016
```

List, Map은 자바프로그램에서 사용이 빈번하다
반드시 복습해서 기본 사용방법을 알아두도록 하자!

Chapter 20

수고하셨습니다