04

04-2. java.util

Java Basic API

Random

- java.util 패키지의 Random 클래스는 난수에 관한 기능을 처리하는 API입니다.

생성자	설명
Random()	난수를 발생하기 위한 Random 객체 생성
Random(long seed)	매개변수로 전달받은 값을 기초로 하는 난수를 발생하는 Random 객체 생성

제어자 및 타입	메서드	설명
boolean	nextBoolean()	true 또는 false 난수를 반환
double	nextDouble()	0.0 ~ 1.0 사이의 난수를 반환(1.0 미포함)
int	nextInt()	int 범위의 난수 반환
int	nextInt(int bound)	0 ~ bound 사이의 난수 반환(bound 미포함)
long	nextLong()	long 범위의 난수 반환
void	setSeed(long seed)	난수를 발생하기 위한 기본값 설정

Random

Test15.java

```
01: package com.ruby.java.ch09;
02:
03: import java.util.Random;
04:
05: public class Test15 {
06:   public static void main(String[] args) {
07:     Random r1 = new Random();
08:     for(int i = 0; i < 5; i++) {
09:       System.out.print(r1.nextInt()+"\t");
10:       System.out.print(r1.nextInt(10)+"\t");
11:       System.out.print(r1.nextBoolean()+"\t");
12:       System.out.print(r1.nextDouble()+"\t");
13:       System.out.println();
14:     }
15:   }
16: }
```

【실행결과】

```
-1004630131     1       true    0.19940465693659437
-219444362      7       true    0.778501040150012
-660006064      5       true    0.9145349455258247
1132303598      2       true    0.4262284833489808
-832221576      2       false   0.7232462499263005
```

Date/ Calendar

- java.util.Date 클래스
- 날짜와 시간 정보

- Calendar는 날짜와 시간 정보를 설정

표 Calendar 필드

필드 이름	의미
YEAR	연도
MONTH	월(0~11)
HOUR	시간(0~11)
HOUR_OF_DAY	24시간을 기준으로 한 시간
SECOND	초
DAY_OF_MONTH	한 달의 날짜
DAY_OF_WEEK	한 주의 요일
AM_PM	오전 또는 오후
MINUTE	분
MILLISECOND	밀리초

Date/ Calendar

Test17.java

```
01: package com.ruby.java.ch09;
02:
03: import java.util.Calendar;
04: import java.util.Date;
05:
06: public class Test17 {
07:   public static void main(String[] args) {
08:     Date d = new Date();
09:     System.out.println(d);
10:
11:     Calendar c = Calendar.getInstance();
12:
13:     System.out.println(c.get(Calendar.YEAR));
14:     System.out.println(c.get(Calendar.MONTH) + 1);
15:     System.out.println(c.get(Calendar.DAY_OF_MONTH));
16:     System.out.println(c.get(Calendar.HOUR));
17:     System.out.println(c.get(Calendar.MINUTE));
18:   }
19: }
```

【실행결과】

```
Sun Feb 25 20:33:33 KST 2018
2018
2
25
8
33
```

SimpleDataFormat

- 날짜의 표현 형식을 지원하는 객체

```
SimpleDateFormat()
SimpleDateFormat(String pattern)
SimpleDateFormat(String pattern, DateFormatSymbols formatSymbols)
SimpleDateFormat(String pattern, Locale locale)
```

패턴 문자	설명	패턴 문자	설명
y	연도	H	시(0~23)
M	월	h	시(1~12) am/pm
d	일	K	시(0~11) am/pm
D	1~365	k	시(1~24)
E	월, 화, 수, 목, 금, 토, 일	m	분
a	오전/오후	s	초
z	타임존		

Pattern

- 정규표현식
- 정규표현식(Regular Expression)은 문자열을 찾기 위한 조건을 문자열로 표현.
- 정규표현식을 만들 때는 특정한 의미가 있는 기호(메타 문자)들을 사용합니다.

메타 문자	의미
\d	숫자
\D	숫자가 아님
\w	문자 또는 숫자
\W	문자 또는 숫자가 아님
\s	공백
\S	공백이 아닌 것
a-z	a부터 z 사이에 있는 모든 소문자
A-Z	A부터 Z 사이에 있는 모든 대문자
0-9	0부터 9까지의 모든 숫자

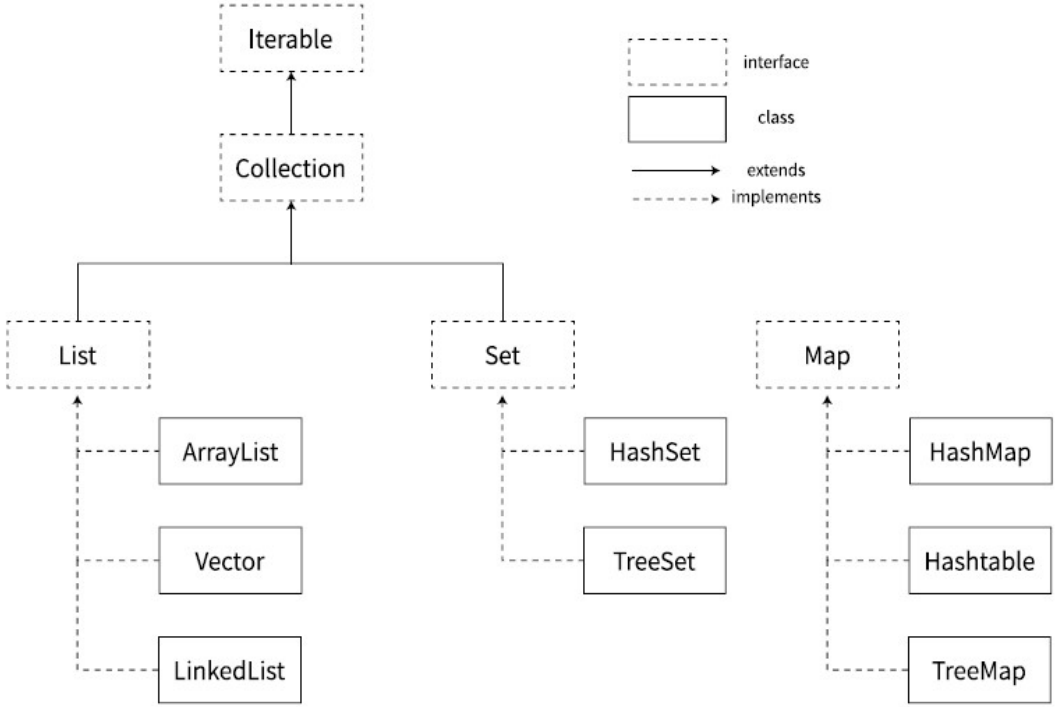
메타 문자	횟수
*	0 또는 1번 이상
+	1번 이상
?	0 또는 1번
{n}	n번
{n,}	n번 이상
{n, m}	n번 이상 m번 이하

Pattern

- 패턴검사
- 정규표현식으로 패턴을 완성한 후에는 특정 문자열들이 패턴에 맞는지 검사할 수 있습니다
- 문자열의 패턴을 검사할 때는 java.util.regex.Pattern 클래스의 matches( ) 메서드를 사용합니다.

```
static boolean matches(String regex, CharSequence input)
```

Collection type



벡터

- 배열(Array) vs. 벡터(Vector)

	배열(Array)	벡터(Vector)
장점	▪빠른 접근 (faster access) ▪기본 자료유형 사용 가능	▪가변적인 크기 ▪필요시 메모리를 할당하므로 효율적
단점	▪고정된 크기 ▪비효율적 메모리 점유 ▪최대 크기를 넘어서는 사용을 위해서는 배열을 새로 정의해야 함	▪느린 접근(slower access) ▪참조자료유형만 사용 가능
사용	▪Java에서 자체적으로 지원	▪java.util.Vector 클래스에 정의

```
import java.util.Vector;
public class VectorTest {
        public static void main(String[] args) {
                int[] a = new int[20];
                Vector v = new Vector(20);
        }
}
```

벡터

- Vector 클래스의 메쏘드(Method)
  - **생성자: `Vector() / Vector(int capacity)`**
  - **개체추가: `addElement(Object o) / insertElementAt(Object o,int index)`**
  - **개체참조: `elementAt(int index)`**
  - **역개체참조: `indexOf(Object o)`**
  - **개체변경: `setElementAt(int index)`**
  - **개체삭제: `remove(Object o)` 또는 `remove(int index)`**
  - **객체검색: `contains(Object o)`**
  - **벡터의 크기와 용량: `size() / capacity()`**
  - **벡터 복사: `clone()`**

벡터

- 자료형
  - **벡터에는 객체만 저장할 수 있음**
  - **벡터의 참조결과는 항상 Object 타입이므로 적절한 타입으로 변환후 사용**
  - **벡터에는 다른 타입의 객체를 섞어서 저장할 수 있음**
    - **자바의 모든 클래스는 Object 클래스의 서브클래스 이므로 가능**
    - **꺼낸 객체의 타입을 알고 있어야 함 또는 instanceof 연산자로 확인 후 사용**

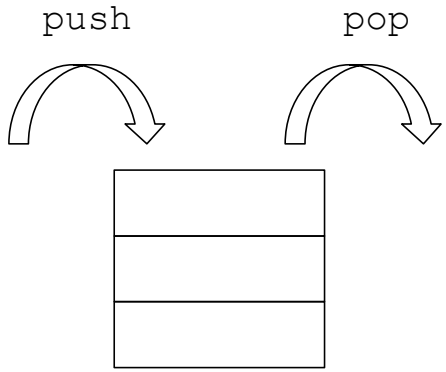
```
Vector v = new Vector();
v.addElement(1); //불가 1은 int 기본자료형
v.addElement(new Integer(1)); //int에 대응되는 Wrapper 클래스인 Integer 사용
v.addElement(new Double(3.14)); //어떤 타입의 객체도 저장 가능

Integer i = v.elementAt(0); //elementAt 메쏘드의 반환타입은 Object 이므로
Integer i = (Integer)v.elementAt(0);//명시적 내림변환 필요(downcasting)
Double d = (Double)v.elementAt(1);
```

ArrayList

- Vector 와 사용 목적 동일
- Vector가 동기화 된다면 ArrayList는 동기화가 되지않은 상태
- Vector는 동기화 되어있기 때문에 한번에 하나의 스레드만 접근할 수 있기때문에 스레드 안전
- ArrayList는 동기화되지 않았기 때문에 명시적으로 동기화 할 필요
- ArrayList는 동기화 되지않았기 때문에 동기화 된 벡터보다 더 빠름.
- Vector와 ArrayList 모두 동적 배열 클래스로 최대 인덱스를 초과할 때 추가
- Vector는 현재 배열의 크기의 100%가 증가하며, ArrayList의 경우 현재 배열의 크기의 50%가 증가.

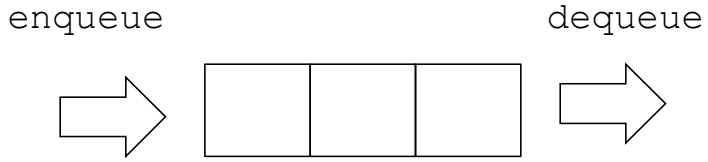
스택(Stack)

- 스택(Stack)
  - 여러 개의 데이터 항목들이 일정한 순서로 나열된 자료구조
  - 한쪽 끝에서만 새로운 항목을 삽입하거나 기존의 항목을 제거할 수 있음
    - Last In First Out (LIFO)
    - 쌓여있는 접시: 새로운 접시를 위에 쌓거나 가장 위의 접시부터 사용 가능
  - java.util.Stack 클래스로 제공되며 Vector 클래스를 상속받아 구현됨
  - 스택에서의 동작들(메쏘드)
    - push()
    - pop()
    - peek()
    - empty()



큐(Queue)

- 큐(Queue)
  - 리스트와 유사한 자료구조
  - 항목들의 리스트에서 가장 마지막에 새로운 항목이 추가됨
  - 기존 항목의 제거는 리스트의 처음에서 일어남
  - First In First Out (FIFO)
  - java.util.Queue 클래스로 제공됨
  - 큐에서의 동작들(메쏘드)
    - enqueue
    - dequeue
    - empty



해시테이블(Hashtable)

- 해시테이블(Hash Table)
  - 벡터나 리스트가 순차적인 자료의 나열을 다루는 자료구조임에 반해 해시테이블은 맵(Map) 인터페이스를 다루는 자료구조임
  - 맵에서는 자료들의 순서가 아닌 키(key)와 값(value)의 쌍을 저장
  - 키와 값은 모든 임의의 객체가 허용됨
  - java.util.Hashtable 클래스로 제공
  - 해시테이블에서 사용 가능한 동작(메쏘드)
    - put
    - get
    - isEmpty
    - containsKey

key	value
"Car"	new Car()
"Man"	new Man()
"Cat"	new Cat()

이뉴머레이션(Enumeration)과 이터레이터(Iterator)

- Enumeration & Iterator
  - 벡터와 해시테이블에서 존재하는 모든 요소에 대한 접근 방식을 제공해주는 인터페이스
  - 자바 콜렉션 프레임웍으로 확장되면서 이터레이터(Iterator) 도입(List, Set 등)
  - 이뉴머레이션 (Enumeration)
    - hasMoreElements()와 nextElement() 두 개의 메쏘드 제공
    - Vector::elements()
    - Hashtable::keys()
    - Hashtable::values()

이뉴머레이션(Enumeration)과 이터레이터(Iterator)

- Enumeration & Iterator
  - 이터레이터 (Iterator)
    - hasNext(), next(), remove() 메쏘드 제공
    - Set::iterator()
    - List::iterator()

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare