04-1. java.lang

Java Basic API

패키지(Package)

- Java 패키지(Package)
  - 서로 관련있는 클래스 또는 인터페이스들의 묶음
  - 장점
    - 클래스들을 묶음 단위로 제공하며 필요할 때만 사용 가능(import)
    - 클래스 이름의 혼란을 막아서 충돌을 방지
    - 패키지 단위의 접근 권한 지정 가능(package)
  - 사용법

【자바 소스 파일 기본 구조】

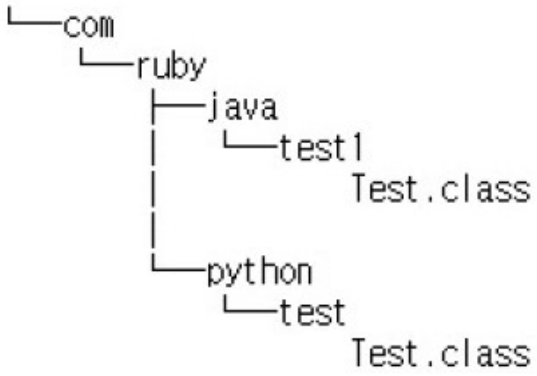
```
package 패키지명;                ← ① 패키지 선언
import 패키지명.클래스명;        ← ② 클래스 임포트
제어자 class 클래스명 {          ← ③ 클래스 선언
    필드 선언;
    생성자 선언
    메서드 선언
}
```

패키지(Package)

- 패키지란, 클래스를 관리하는 방법이며 물리적으로는 파일 시스템의 디렉터리를 의미합니다.
- 클래스를 패키지로 구성하려면 먼저 소스 파일의 첫 번째 줄에 자신이 속한 패키지를 선언해야합니다.

Test.java

```
package com.ruby.java.test1;
public class Test {
    public static void main(String[] args) {
    ...
    }
```




```
com.ruby.java.test1.Test
com.ruby.python.test.Test
```

패키지(Package)

- 사용하려는 객체가 속한 패키지 정보를 나타내주는 것이import 문입니다.

【import 문】

```
import 패키지명[.하위 패키지명].클래스명;
```

또는

```
import 패키지명[.하위 패키지명].*;
```

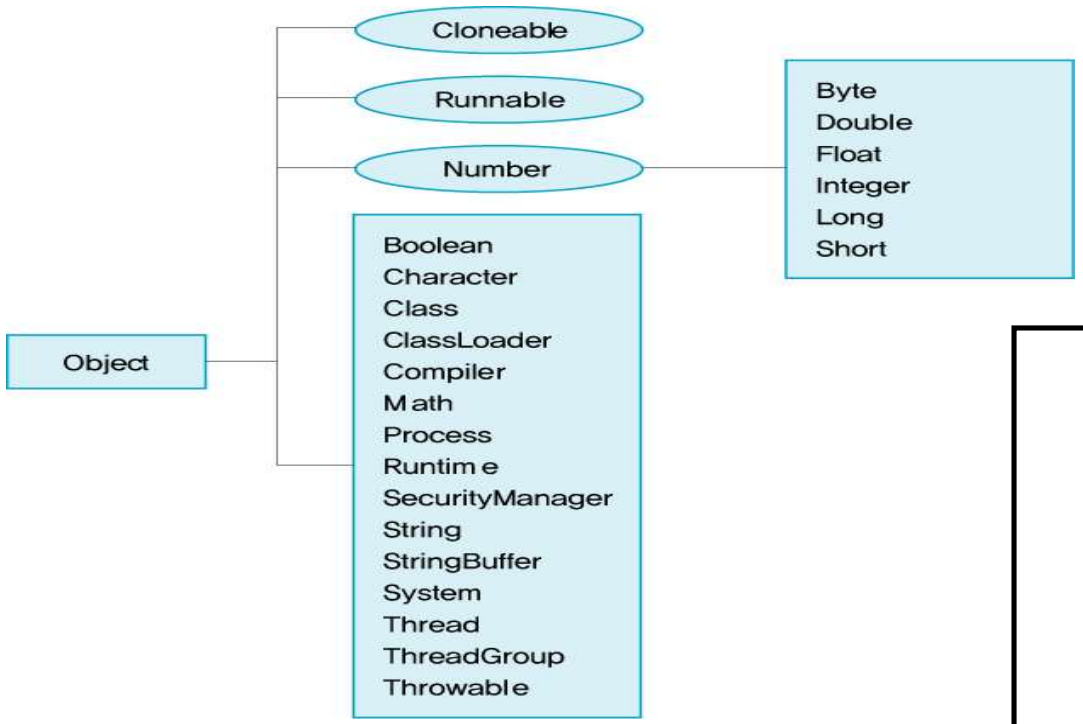
```
import com.ruby.java.test2.*;
import com.ruby.java.test2.io.*;
```

패키지(Package) – Java API Package

- Java API
  - JDK에 정의된 자바 표준 패키지
  - 프로그래머가 새로운 클래스를 정의하여 사용하는 것 보다 자바API에서 제공하는 많은 클래스들을 재사용하여 쉽게 프로그램 작성 가능

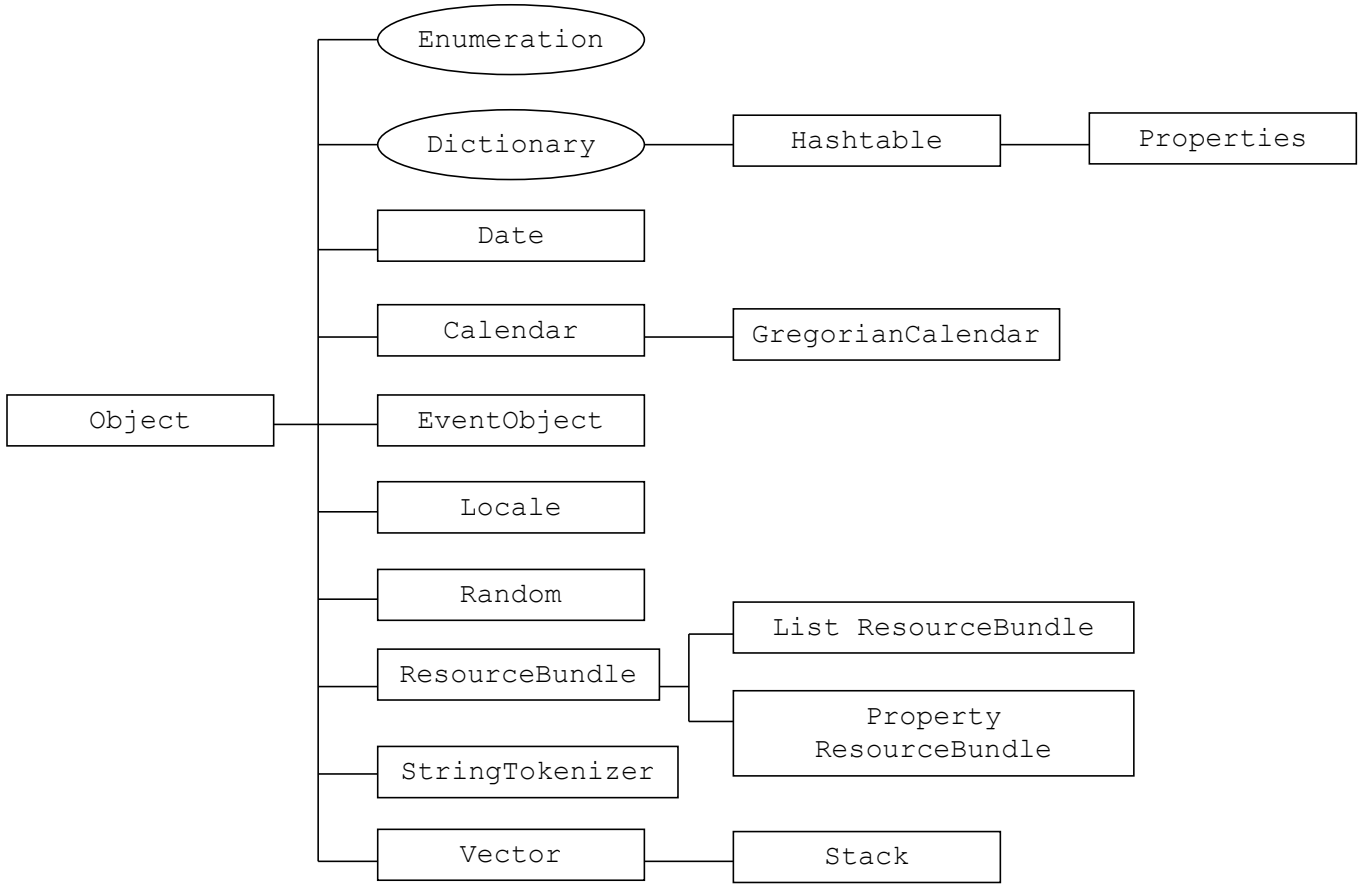
자바 API	제공하는 기능
java.lang.*	명시적으로 지정하지 않아도 자동으로 **import** 되는 패키지. 자바 프로그램이 기본적으로 필요로 하는 클래스와 인터페이스 포함
java.io.*	데이터를 입력받고 출력할 수 있도록 하는 클래스 포함
java.util.*	날짜/시간 조작, 난수 발생 등 각종 유틸리티 클래스와 인터페이스 포함

패키지(Package) – Java API Package



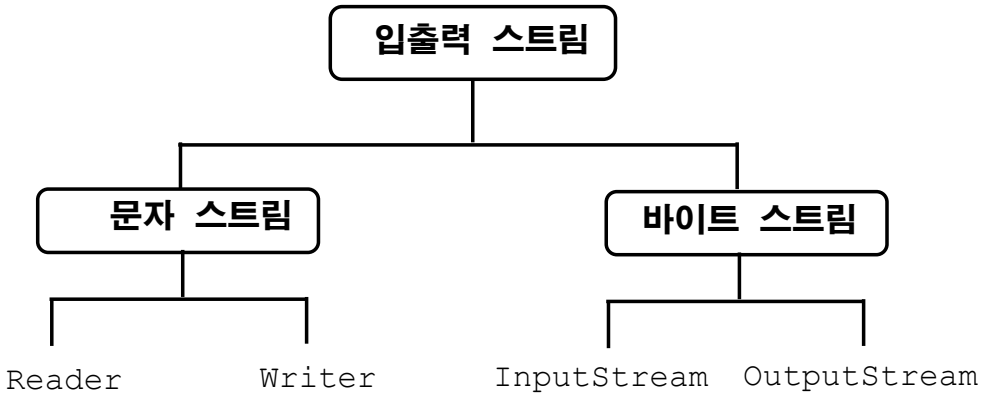
- **Math class**
  - **x = Math.abs(y)**
- **Integer class**
  - **int x = Integer.parseInt("1984"); // 문자열을 int 로 변환**
- **Double class**
  - **double d = Double.parseDouble("3.14");**

패키지(Package) – java.util 패키지



패키지(Package) – java.io 패키지

- 문자 스트림
  - 16 Bit로 코드화 된 문자나 문자열을 읽거나 쓰는 데 이용(unicode)
- 바이트 스트림
  - 숫자나 binary 자료를 읽거나 쓰는 데 이용



Object 클래스

- 자바 최상위 클래스
- hashCode( )는 메모리에 생성된 인스턴스의 주솟값을 가지고 일련번호를 만들어 반환하는 메서드입니다.
- toString()

Test01.java

```
package com.ruby.java.ch09;

public class Test01 {
  public static void main(String[] args) {
    …
    System.out.println(obj1.toString());
    System.out.println(obj2.toString());
    System.out.println(obj3.toString());
  }
}
```

Test01.java

```
package com.ruby.java.ch09;

public class Test01 {
  public static void main(String[] args) {
    …
    System.out.println(obj1);
    System.out.println(obj2);
    System.out.println(obj3);
  }
}
```

【실행결과】

```
…
java.lang.Object@15db9742
java.lang.Object@6d06d69c
java.lang.Object@7852e922
```

Object 클래스

- equals() : 인자로 전달된 객체와 현재 객체가 같은지를 판단합니다. 이때 판단 기준은 해시 코드값 입니다.

Test01.java

```
01: package com.ruby.java.ch09;
02:
03: public class Test01 {
04:   public static void main(String[] args) {
05:     …
06:     MyObject obj4 = new MyObject();
07:     MyObject obj5 = new MyObject();
08:     if(obj4.equals(obj5)) {
09:       System.out.println("동일 객체이다.");
10:     } else {
11:       System.out.println("다른 객체이다.");
12:     }
13:
14:     if(obj4 == obj5) {
15:       System.out.println("동일 객체이다.");
16:     } else {
17:       System.out.println("다른 객체이다.");
18:     }
19:   }
20: }
```

【실행결과】

```
…
다른 객체이다.
다른 객체이다.
```

String 클래스

- 문자(Character):
  - 단순 자료형(char)
  - 문자 선언 **`char letter;`**
  - 문자 할당 **`letter = 'A';`**
  - 문자 사용

```
System.out.println(letter);
System.out.println((int)letter);
```

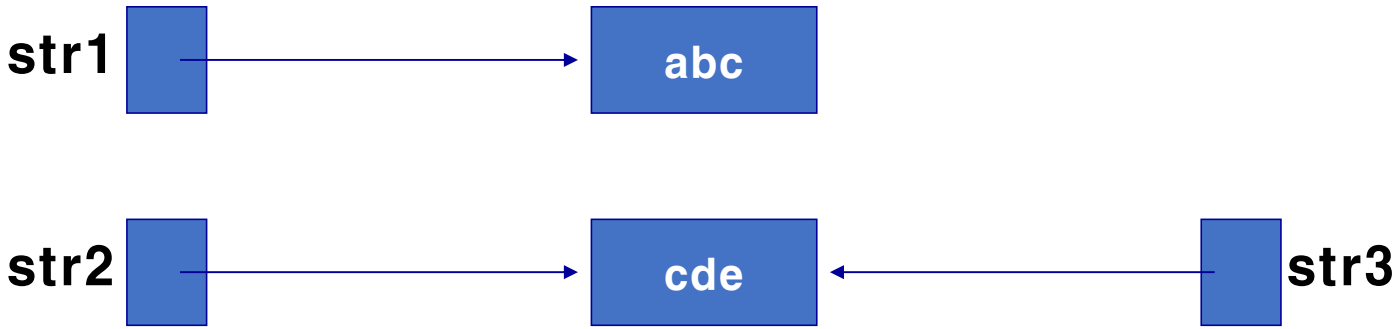
String 클래스

- 문자열(String) 클래스
  - 연속된 문자(character)들을 저장하고 다루기 위한 클래스
  - 문자열 상수: " "로 둘러싸인 문자열 → **한번 만들어진 String 객체는 변경 불가함(immutable)**
  - 문자열 선언 **`String greeting;`**
  - 문자열 할당 **`greeting = "Hello JAVA!";`**
  - 문자열 사용 **`System.out.println(greeting);`**
  - 특수 문자의 표현(Escape characters)
    - ₩' , ₩" , ₩₩, ₩n, ₩t

```
System.out.print("Hello "JAVA!"");

System.out.print("Hello \"JAVA!\"");

System.out.print("Hello \"JAVA!\"\n");
```

String 클래스

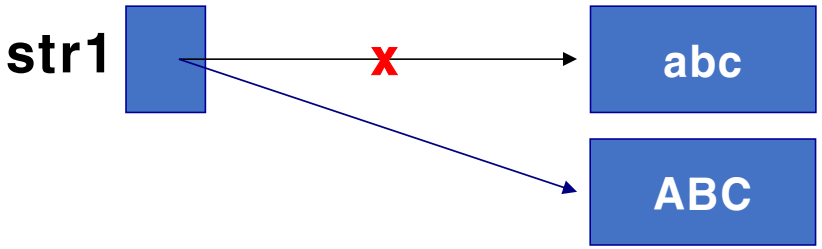
```
String str1;

String str2, str3;  // String 클래스변수 str1, str2, str3 선언

str1 = "abc"; // str1은 생성된 String 클래스의 객체(Object)를 가리킴

str2 = “cde"; // str2은 생성된 String 클래스의 객체(Object)를 가리킴

str3 = str2; // str3에 str2의 값 할당
```



String 클래스

※Java의 String 객체는 한번 생성되고 나면 변경할 수 없고 완전히 새로운 String 클래스 객체가 생성됨

```
str1 = str1.toUppercase();
```



String 클래스

- "+" 연산자

```
String greet = “Hello”;
String name = “JAVA”;
System.out.println(greet+name+”!”);
System.out.println(greet+” “+name+”!”);
```

String 클래스

- charAt()
  - String 객체 내의 문자 인덱싱은 배열과 같이 0부터 시작됨
  - charAt(position) : 해당 위치의 문자를 반환
  - substring(start, end) : start부터 end까지의 문자들을 새로운 문자열로 반환

```
String greeting = "Hello JAVA!";
greeting.charAt(0); //'H'
greeting.charAt(10);
greeting.substring(1,3); //"ell"
```

Index	0	1	2	3	4	5	6	7	8	9	10
char	H	e	l	l	o		J	A	V	A	!

String 클래스

- equals()
- String 인스턴스의 문자열을 비교하여 같으면 true, 다르면 false를 반환합니다

- `toString()`
- toString( ) 메서드는 참조변수를 출력할 때 자동으로 호출되는 메서드로서 문자열을 반환하도록 재정의하였습니다.

Test02.java

```
package com.ruby.java.ch09;

public class Test02 {

  public static void main(String[] args) {
    ...
    System.out.println(s1);
    System.out.println(s2);
    System.out.println(s3);
    System.out.println(s4);
  }
}
```

【실행결과】

```
...
java
java
java
java
```

String 클래스

- indexOf( ), lastIndexOf( )는인자로 지정된 문자 또는 문자열이 시작되는 인덱스를 구할 때 사용합니다.
- indexOf( )는 문자열의 처음부터 검색하고, lastIndexOf( ) 메서드는 문자열의 끝에서부터 검색합니다.
- 검색할 문자열이 없으면 -1을 반환합니다.

- startsWith( )와 endsWith( )는 인자로 전달받은 문자열이 대상 문자열의 시작 부분 또는 끝 부분에 포함되었는지를 판단할 때 사용합니다.
- startsWith( ) 메서드는 인자로 전달받은 문자열로 시작하는지를 판단합니다.
- endsWith( ) 메서드는 인자로 전달받은 문자열로 끝나는지를 판단합니다

【프로토타입】

```
boolean startsWith(String prefix)
boolean startsWith(String prefix, int toffset)
boolean endsWith(String suffix)
```

String 클래스

- length( ) 메서드는 문자열의 전체 길이를 반환합니다.
- trim( )는 문자열의 양끝에 있는 공백을 제거합니다.
- isEmpty( )는 문자열의 길이가 0인지를 판단합니다.

Test04.java

```
01: package com.ruby.java.ch09;
02:
03: public class Test04 {
04:   public static void main(String[] args) {
05:     String s = "    Amy    "; // Amy 앞뒤로 공백 4개가 있습니다.
06:     int len1 = s.length();
07:
08:     String s2 = s.trim();
09:     int len2 = s2.length();
10:
11:     System.out.println(len1);  // 11
12:     System.out.println(len2);  // 3
13:
14:     s = "";
15:     System.out.println(s.length());  // 0
16:     System.out.println(s.isEmpty()); // true
17:   }
18: }
```

【실행결과】

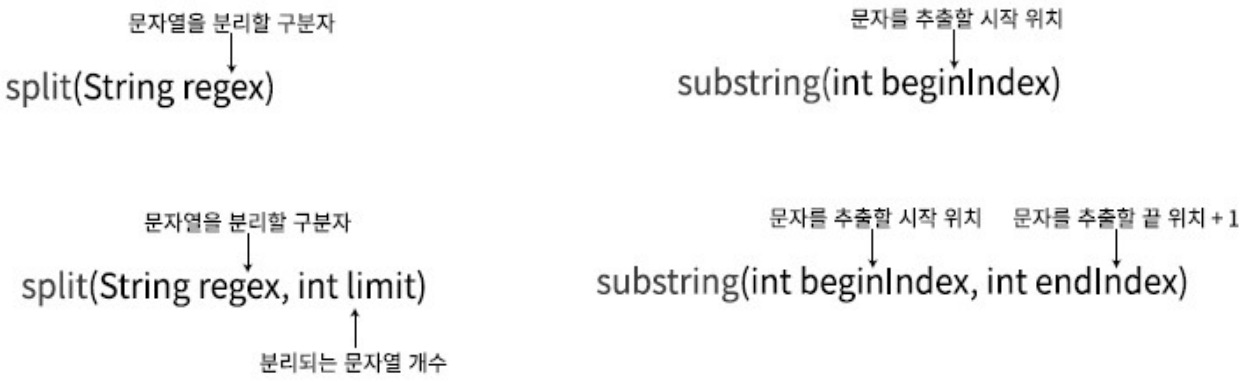
```
11
3
0
true
```

String 클래스

- slipt() – 문자열 자르기
- substring() – 문자열 추출

【프로토타입】

```
String[] split(String regex)
String[] split(String regex, int limit)
String substring(int beginIndex)
String substring(int beginIndex, int endIndex)
```



StringBuffer

- 한번 만들어진 String 객체는 변경 불가함(immutable)
- 동적으로 변경되는 문자열을 다루기에 String 클래스는 부적절함
- StringBuffer 클래스는 문자열이 변경될 때 새로운 객체를 생성하는 대신 메모리상에서 문자열을 조작
- 동적으로 문자열의 내용을 바꾸거나 문자의 위치를 변경할 때 용이
  - append
  - insert
  - replace
  - reverse
  - delete

StringBuffer - 예제

```
public class StringBufferTest {

  public static void main(String[] args) {

        StringBuffer s1=new StringBuffer();

        s1.append("Hello ");

        s1.append("JAVA!");

        System.out.println(s1.toString());


        s1.reverse();

        s1.delete(0,3);

        System.out.println(s1.toString());


        System.out.println(s1.length());

        System.out.println(s1.capacity());
```

Wrapper 클래스

- Wrapper 클래스는 Wrapper라는 이름의 클래스가 존재하는 것이 아니라 java.lang 패키지에 있는 클래스 중 자바의 기본 데이터 타입과 매핑되는 클래스들을 의미합니다.

표 Wrapper 클래스

Wrapper 클래스	기본 데이터 타입
Boolean	boolean
Byte	byte
Character	char
Double	double
Float	float
Integer	int
Long	long
Short	short

Math 클래스

- Math는 수학적인 계산에 관한 처리를 하는 클래스입니다.
- Math 클래스가 가지는 필드, 메서드는 모두static으로 선언되었으므로 'Math.변수' 또는 'Math.메서드( )' 형태로 사용합니다.
- Math 클래스는 생성자가 private 접근 제한자로 선언되었으므로 인스턴스를 생성할 수 없습니다.

표 Math 클래스의 메서드(* 표시는 오버로딩 메서드)

제어자 및 타입	메서드	설명	예
static *	abs(*)	절댓값을 구함	Math.abs(-12) → 12
static double	ceil(double a)	매개변수 값을 올림함	Math.ceil(12.5) → 13.0
static double	floor(double a)	매개변수 값을 내림함	Math.floor(12.5) → 12.0
static *	max(*)	최댓값을 구함	Math.max(5, 8) → 8
static *	min(*)	최솟값을 구함	Math.min(5, 8) → 5
static double	pow(double a, double b)	a의 b승을 구함	Math.pow(2, 3) → 8
static double	random()	0.0 이상 1.0 미만의 난수를 구함	Math.random() → 실행할 때마다 값이 달라짐
static *	round(*)	소수점 첫째 자리에서 반올림함	Math.round(12.5) → 13.0
static double	sqrt(double a)	a의 제곱근을 구함	Math.sqrt(4) → 2.0

감사합니다

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

윌리엄 셰익스피어
William Shakespeare