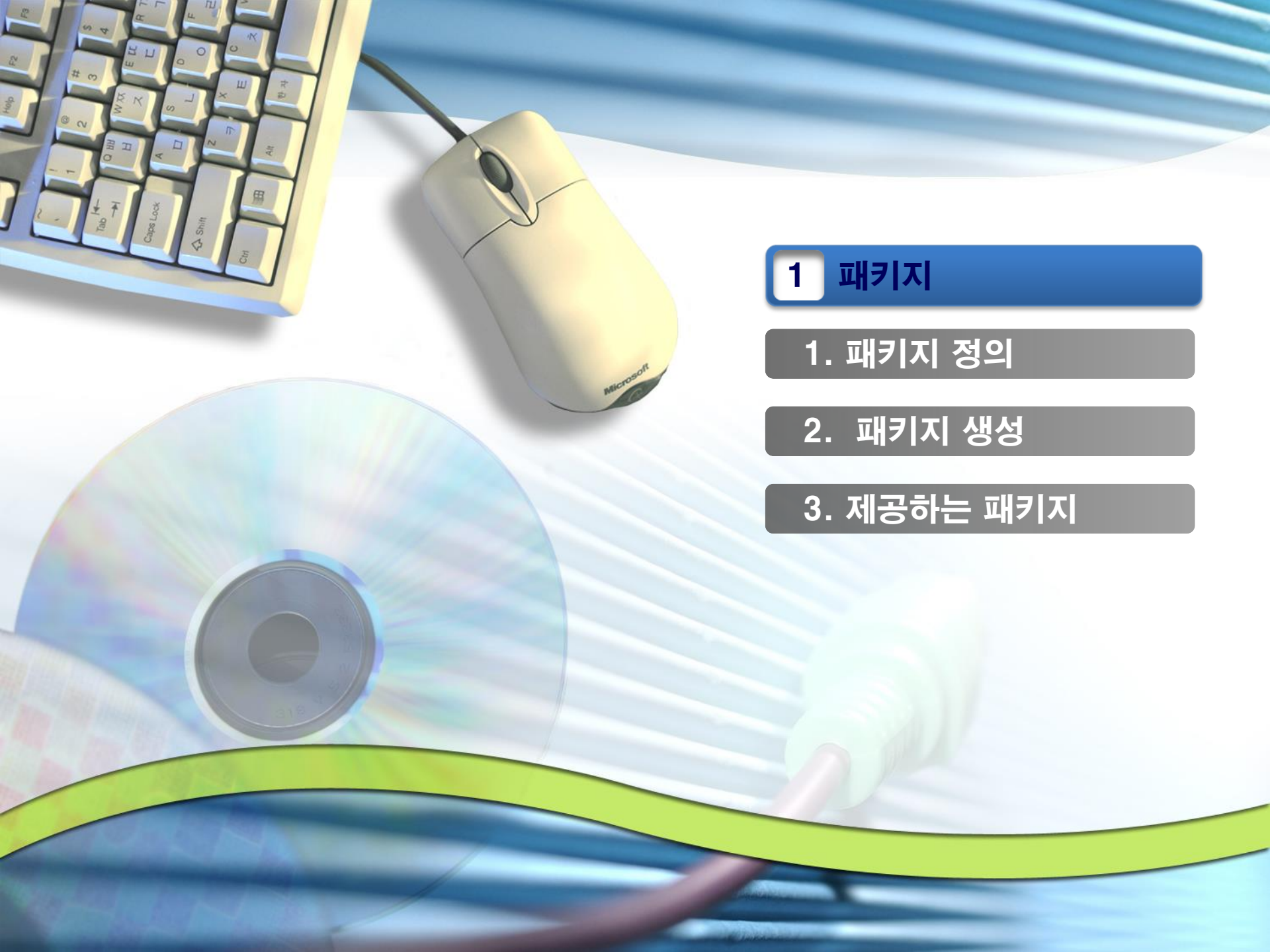

JAVA
패키지



1 패키지

1. 패키지 정의

2. 패키지 생성

3. 제공하는 패키지

패키지 정의

1) 패키지란?

패키지(package)는 서로 관련있는 클래스나 인터페이스들을 하나로 묶은 것이다. 클래스나 인터페이스의 개수가 많아지면 관리하기가 상당히 힘들어진다. 또한 각 클래스마다 서로 다른 이름을 사용하여야 한다. 자바에서는 클래스나 인터페이스를 패키지(package)라는 단위로 묶을 수 있다.

자바가 제공하는 라이브러리도 패키지로 구성되어 있다

패키지 장점

※ 패키지의 장점

① 관련된 클래스들을 쉽게 파악할 수 있다.

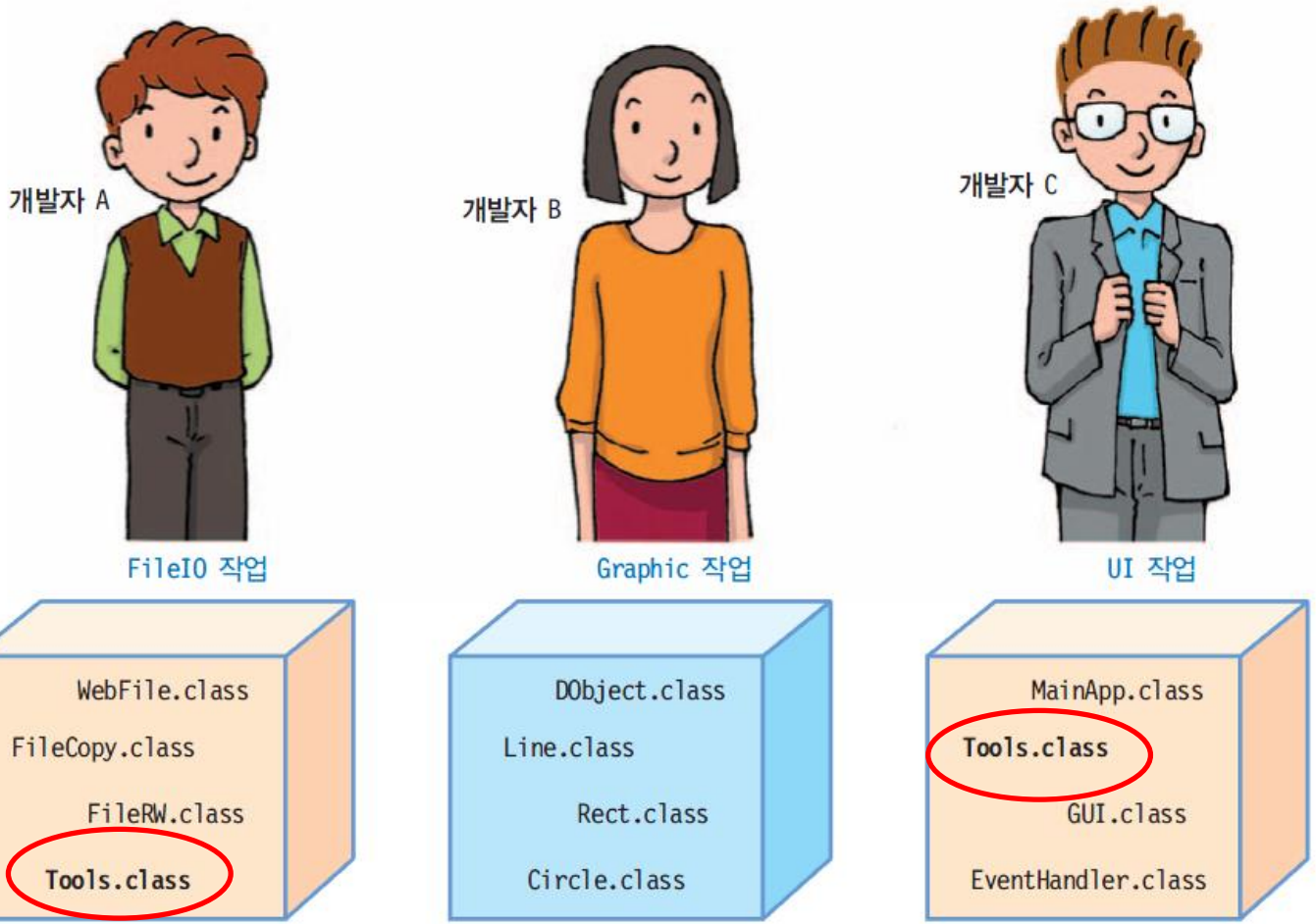
② 원하는 클래스들을 쉽게 찾을 수 있다.

③ 패키지마다 이름 공간을 따로 갖기 때문에 같은 클래스 이름을 여러 패키지가 사용할 수 있다.

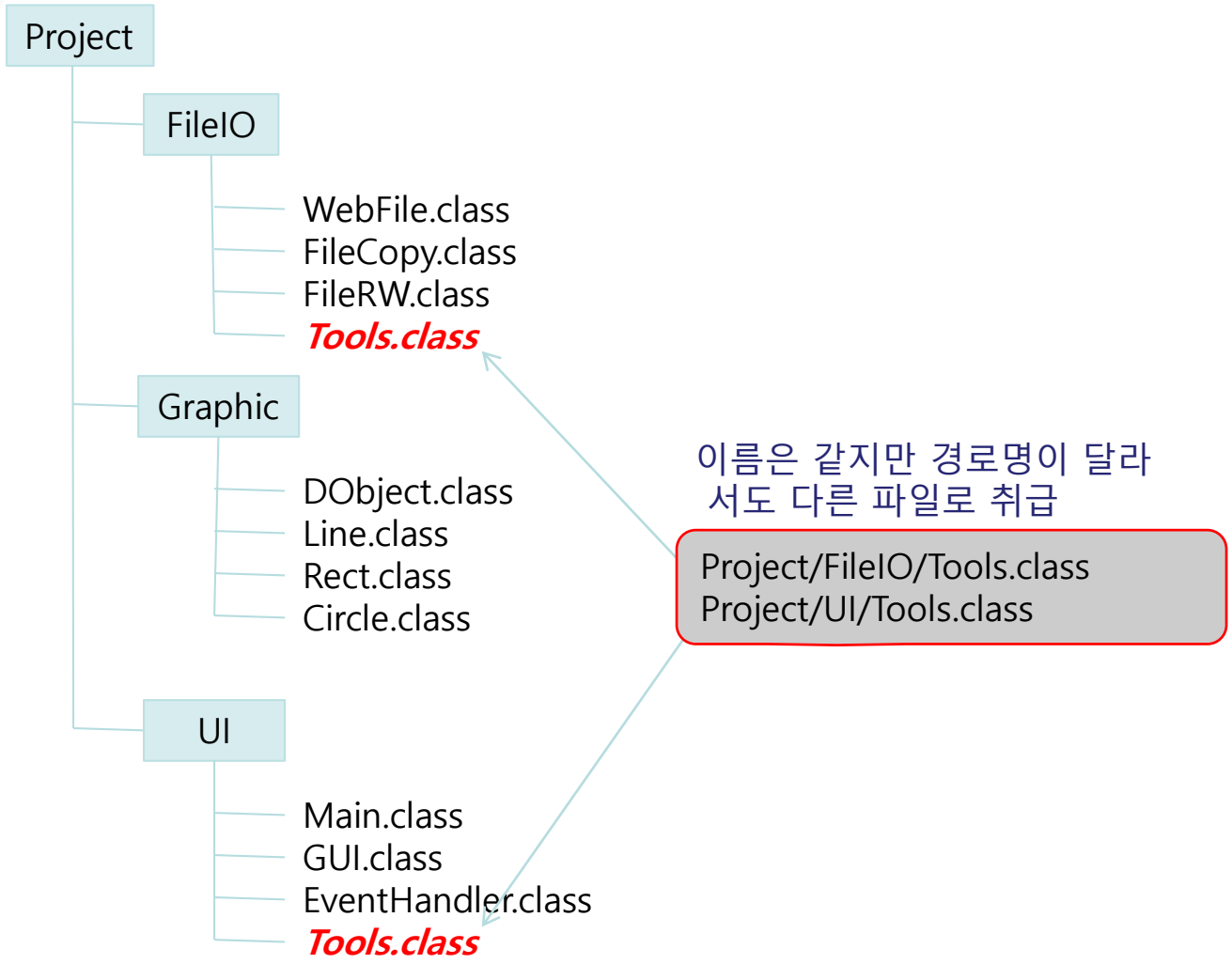
④ 패키지별로 접근에 제약을 가할 수 있다.

패키지 개념과 필요성

3명이 분담하여 자바 응용프로그램을 개발하는 경우,
동일한 이름의 클래스가 존재할 가능성 있음 -> 합칠 때 오류발생



디렉터리로 각 개발자의 코드 관리(패키지)



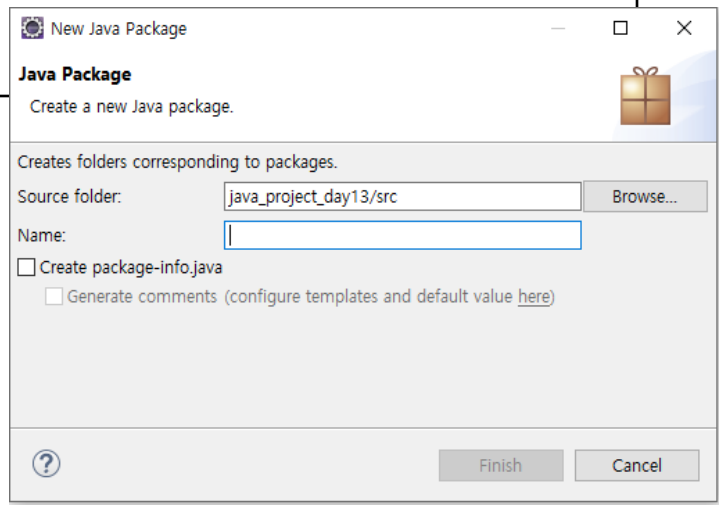
패키지 생성

2) 패키지 생성하기

패키지의 이름을 결정한 후에 소스 코드의 첫 번째 줄에 package문을 사용하면 된다. **패키지를 정의하는 문장은 반드시 소스의 첫 번째 줄에 와야한다.** 하나의 소스 파일에는 패키지 문장이 하나만 있을 수 있다.

```
package 패키지명;

public class 클래스명{
    ...
}
```



패키지 생성

```
package graphics;

public interface Drawable {
…
}
```

Drawable.java

```
package graphics;

public abstract class Shape {
…
}
```

Shape.java

```
package graphics;

public class Circle extends Shape
   implements Drawable{
…
}
```

Circle.java

```
package graphics;

public class Oval extends Shape
   implements Drawable{
…
}
```

Oval.java

패키지 생성

패키지 문을 사용하지 않은 경우에는 디폴트 패키지(default package)에 속하게 된다. 패키지의 이름은 다른 패키지 이름과 구별되어야 한다.

디폴트 패키지

- package 선언문이 없이 만들어진 클래스의 패키지
- 디폴트 패키지는 현재 디렉터리

따라서 패키지의 이름은 아래 내용을 참조하여 지정하는 것이 좋다.

✓ 패키지의 이름은 일반적으로 소문자만을 사용한다.

✓ 인터넷 도메인 이름을 역순으로 사용한다.

✓ 자바 언어 자체의 패키지는 java나 javax로 시작한다.

패키지 사용

- 서로 다른 패키지 안에 들어 있는 클래스나 인터페이스를 사용하는 방법

① 클래스에 패키지 이름을 붙여서 참조한다.

② 개별 클래스를 import한다.

③ 전체 클래스를 import한다.

패키지 사용

1) 클래스에 패키지 이름을 붙여서 참조

같은 패키지 내에 포함되어 있지 않는 클래스를 사용하려면 클래스의 완전한 이름을 써주어야 한다.

```
graphics.Circle obj = new graphics.Circle();
```

패키지 사용

2) 패키지 멤버를 import

외부 패키지의 특정 클래스를 사용하려면 import문장을 사용한다. import문장은 package문장 다음에 위치하여야 한다.

```
import graphics.Circle;
```

```
Circle obj = new Circle();
```

패키지 사용

3) 전체 패키지를 import

하나의 패키지 안에 포함된 모든 클래스를 포함하려면 다음과 같이 *를 사용하면 된다.

```
import graphics.*;
```

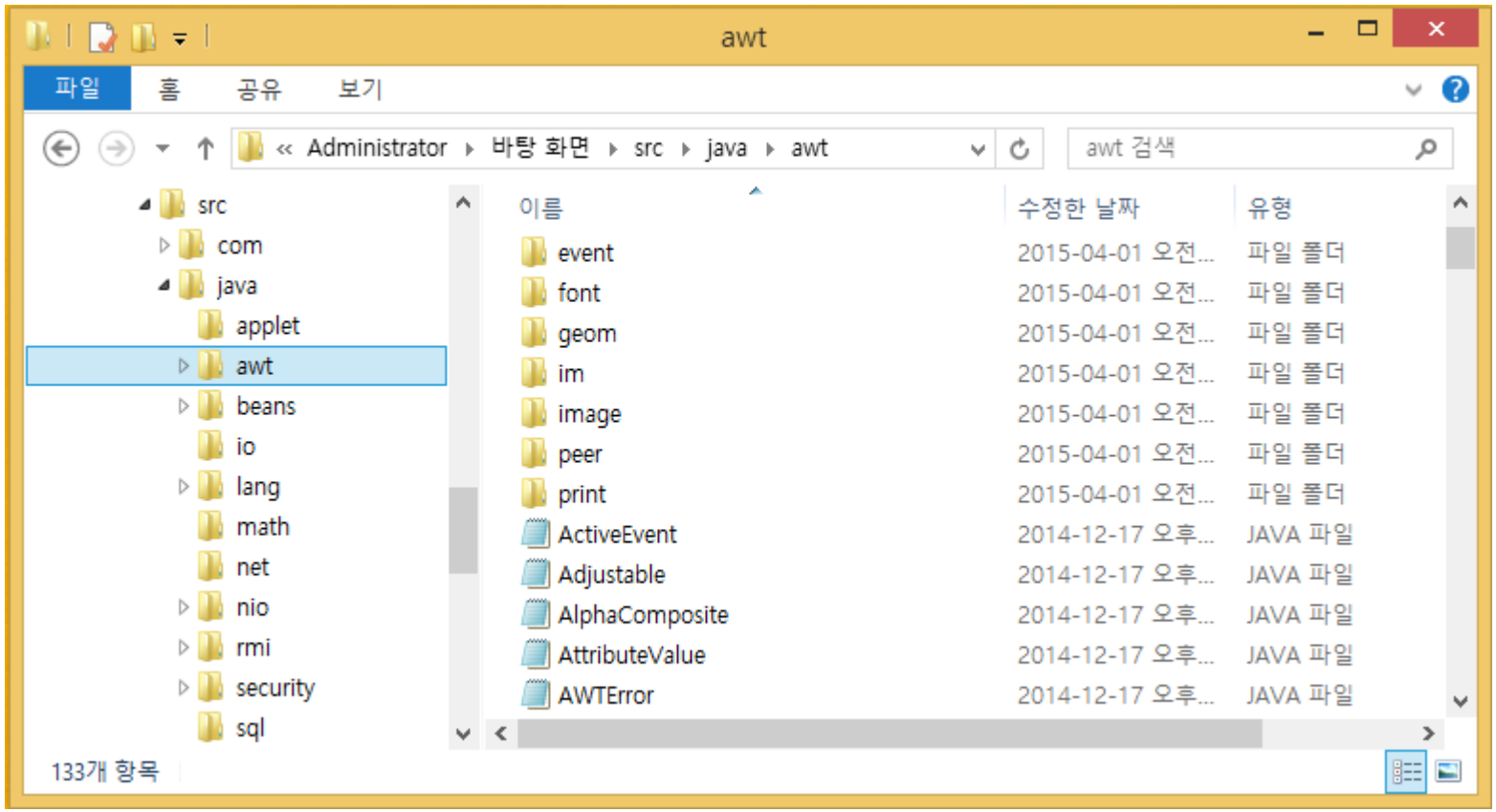
```
Rectangle objRe = new Rectangle();
Circle objC = new Circle();
```

패키지 사용

4) 계층 구조의 패키지 포함하기

여기서 한 가지 아주 주의해야 할 사항이 있다.

```
import java.awt.*;
```



패키지 사용

5) 정적 import 문장

클래스 안에 정의된 정적 상수나 정적 메소드를 사용하는 경우에 일반적으로는 클래스 이름을 앞에 적어주어야 한다.

```
int val =Math.max((int)(Math.random()*10)+1, 5);
```

하지만 정적 import 문장을 사용하면 클래스 이름을 생략하여도 된다.

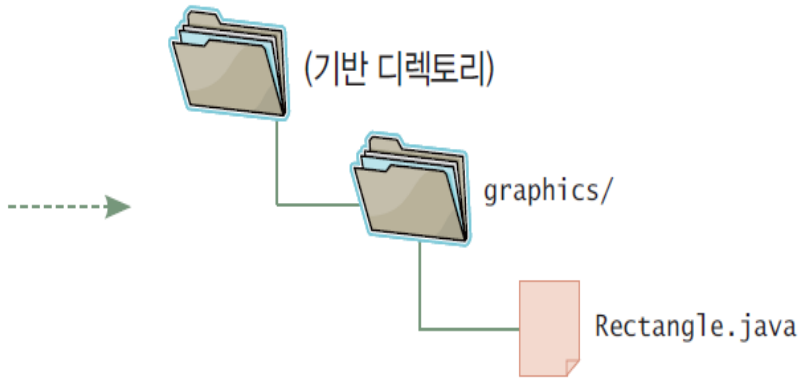
```
import static java.lang.Math.*;
int val =max((int)(random()*10)+1, 5);
```

소스 파일과 클래스 파일 관리

자바에서는 패키지의 계층 구조를 반영한 디렉토리 구조에 이용하여 소스 파일과 클래스 파일을 관리한다.

```
package graphics;
public class Rectangle {
    ...
}
```

Rectangle.java



소스 파일과 클래스 파일 관리

현재 진행하고자 하는 프로젝트명이 onlyone이며, 이에 도메인을 www.onlyone.com이다. 현재 게시판을 작성하고자 한다면 일반적으로 패키지명은 다음과 같이 설정한다.

```
package com.onlyone.board;

public class BoardInsert{ ... }
```

₩com₩onlyone₩board₩BoardInsert.java

이클립스에서는 (기반 디렉토리)/src 밑에 소스 파일이 위치하고 (기반 디렉토리)/bin 밑에 클래스 파일이 위치하게 된다.

소스 파일과 클래스 파일 관리

1) 명령 프롬프트(cmd)에서 패키지 생성할때

com.onlyone.board 패키지를 작성하고 이 패키지 안에 BoardInsert. java를 넣으려면 디렉토리 구조도 똑같이 만들어 주어야 한다.

```
C:₩Temp>javac –d . BoardInsert.java
```

```
package com.onlyone.board;
public class BoardInsert{
        …
}
```

```
C:₩>java com.onlyone.board.BoardInsert
```

소스 파일과 클래스 파일 관리

- 자바 가상 머신은 어떻게 클래스 파일을 찾을까?

가상 머신이 클래스 파일을 찾는 디렉토리들을 클래스 경로(class path)라고 한다.

✓ 자바 가상 머신은 항상 현재 작업 디렉토리부터 찾는다.
✓ 환경 변수인 CLASSPATH에 설정된 디렉토리에서 찾는다.

```
C:₩Temp>javac –d . Exam02.java
```

✓ 자바 가상 머신을 실행할 때 옵션 -classpath를 사용할 수 있다.

```
C:₩>java -classpath C:₩Temp; com.onlyone.test.Exam02
```

소스 파일과 클래스 파일 관리

2) JAR 압축 파일(.jar)

클래스 파일은 또한 JAR(Java archive)파일 형태로 저장될 수 있다. JAR 파일은 여러 개의 클래스 파일을 압축하여서 가지고 있을 수 있다. 이것은 공간을 절약하고 성능을 향상시키게 된다. 전문업체에서 자바 라이브러리를 구매하게 되면 일반적으로 JAR 파일들을 보내게 된다. JAR 파일을 받아서 JAR 파일 안에 들어 있는 클래스 파일을 사용하려면 클래스 경로에 JAR 파일을 포함시키면 된다.

자바에서 지원하는 패키지

패키지(package)는 연관되어 있는 클래스와 인터페이스들을 하나로 묶어 놓은 것이다.

자바에서는 기본적으로 많은 패키지들을 제공하고 프로그래머는 이것들을 이용하여서 편리하게 프로그램을 작성할 수 있다.

자바의 기본 패키지는 java로 시작하며 확장 패키지는 javax로 시작한다.

자바에서 지원하는 패키지

- 자바 API란?
  - 자바에서 기본적으로 제공하는 라이브러리(library)
  - 프로그램 개발에 자주 사용되는 클래스 및 인터페이스 모음

- API 도큐먼트
  - 쉽게 API 찾아 이용할 수 있도록 문서화한 것
  - HTML 페이지로 작성되어 있어 웹 브라우저로 바로 볼 수 있음
    http://docs.oracle.com/javase/8/docs/api/

자바에서 지원하는 패키지

패키지	설명
java.io	입력과 출력 스트림을 위한 클래스
java.lang	자바 프로그래밍 언어에서 필수적인 클래스
java.net	네트워킹 클래스
java.security	보안 프레임워크를 위한 클래스와 인터페이스
java.sql	데이터베이스에 저장된 데이터를 접근하기 위한 클래스
java.util	날짜, 난수 등의 유틸리티 클래스
java.awt	그래픽과 이미지를 위한 클래스
java.swing	스윙 컴포넌트를 위한 클래스

java.lang 패키지

java.lang 패키지

- 자바 프로그램의 기본적인 클래스를 담은 패키지.
- 포함된 클래스와 인터페이스는 import 없이 사용.

다음과 같은 클래스들이 포함된다.

클래스	설 명
Object 클래스	모든 클래스의 최상위 클래스
Math 클래스	각종 수학 메서드(최대값, 최소값, 난수)들을 포함하는 클래스
Wrapper 클래스	기본 자료형(기초형)에 대응된 클래스 • 기본 타입의 데이터를 갖는 객체를 만들 때 사용. • 문자열을 기본 타입으로 변환할 때 사용.

java.lang 패키지

클래스	설 명
String 클래스	문자열을 저장하고 여러 가지 정보를 얻을 때 클래스
System 클래스	표준 입출력 클래스 • 표준 입력 장치(키보드)로부터 데이터를 입력받을 때 사용. • 표준 출력 장치(모니터)로 출력하기 위해서 사용.
Class 클래스	JavaVM에 로드된 클래스에 대한 정보를 얻기 위한 클래스

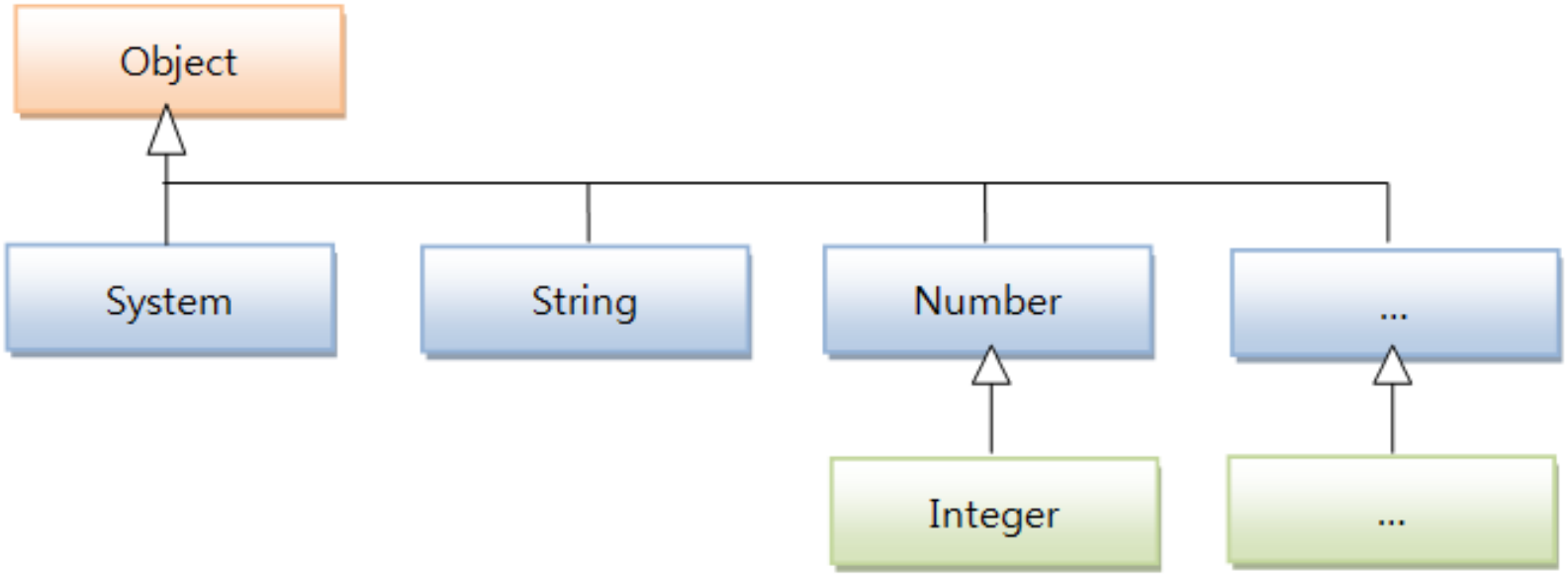
java.util 패키지

다음과 같은 클래스들이 포함된다.

클래스	설 명
Arrays 클래스	배열을 조작(비교, 복사, 정렬, 찾기) 할 때 사용
Calendar 클래스	운영체제의 날짜와 시간을 얻을 때 사용
Date 클래스	날짜와 시간을 얻을 때 사용
Objects 클래스	객체 비교, 널(null) 여부 등을 조사할 때 사용
StringTokenizer 클래스	특정 문자로 구분된 문자열을 나누고자 할 때 사용
Random 클래스	난수를 얻을 때 사용

Object 클래스

- 모든 클래스의 최상위 클래스
- 모든 클래스는 명시적으로 "extends" 예약어를 지정하지 않아도 컴파일러가 자동으로 java.lang.Object 클래스로부터 상속받도록 컴파일된다.
- 모든 클래스는 묵시적으로 Object 클래스의 필드와 메서드를 모두 상속받게 된다.



패키지 생성

```
public class Car {
…
}
```

컴파일 전

```
public class Car extends Object {
…
}
```

컴파일 후

Object 안에 정의되어 있는 메소드는 다음과 같다.

protected Object clone()	객체 자신의 복사본을 생성하여 반환한다.
public boolean equals(Object obj)	obj가 이 객체와 같은지를 나타낸다.
protected void finalize()	가비지 콜렉터에 의하여 호출된다.
public final Class getClass()	객체의 실행 클래스를 반환한다.
public int hashCode()	객체에 대한 해쉬 코드를 반환한다.
public String toString()	객체의 문자열 표현을 반환한다.

Object 클래스

- 객체 비교(equals() 메소드)

```
public boolean equals(Object obj) { return (this == obj); }
```

- 기본적으로 == 연산자와 동일한 결과 리턴 (주솟값 비교)

```
Object obj1 = new Object();
Object obj2 = new Object();

boolean result = (obj1==obj2);
boolean result = obj1.equals(obj2);
                 기준객체        비교객체
```

논리적 동등 위해 오버라이딩 필요

- 논리적 동등이란? 같은 객체이건 다른 객체이건 상관없이 객체 저장 데이터 동일
- Object의 equals() 메소드 : 재정의하여 논리적 동등 비교할 때 이용

Object 클래스

- 객체 해시코드(hashCode())

객체는 저마다 고유한 값이 있다. 즉 메모리에 생성된 인스턴스의 주솟값을 가지고 일련번호를 만들어 반환하는 메서드가 hashCode()이다. hashCode() 메서드가 반환하는 일련번호를 '해시코드(Hash Code)'라고 하며, 해시 코드는 인스턴스가 메모리에 생성되는 주솟값을 기초로 만들어지는 만큼 서로 다른 인스턴스는 해시코드값이 같을 수가 없다.

- 객체 해시코드란?
  - 객체 식별할 하나의 정수값
  - 객체의 메모리 번지 이용해 해시코드 만들어 리턴
- 개별 객체는 해시코드가 모두 다름

Object 클래스

```
public native int hashCode();
```

- 원칙적으로 자바에서 hashCode() 메서드는 원시 메서드로, JVM에서 원시 코드로 직접 구현된 메서드이다. 즉 native는 메서드에 사용하는 예약어로 이 예약어가 지정된 메서드를 원시 메서드라 하며 이 메서드는 자바가 아닌 C 언어나 다른 플랫폼에 종속적인 다른 언어로 구현되었다는 것을 의미한다. 자바 이외의 프로그램 언어로 작성된 서브(sub) 프로그램을 호출할 때 사용한다. 자바로 구현하기 까다로운 부분을 다른 언어로 구현해서 자바에서 사용하기 위해서이다.

Object 클래스

- 객체 문자 정보(toString())
- 객체가 가진 값을 문자열로 제공한다.

```
public String toString(){
   return
   getClass().getName()+"@"+Integer.toHexString(hashCode());
}
```

getClass()	현재 클래스에 대한 정보를 가진 Class 객체를 반환한다.
getName()	Class의 getName() 메서드는 현재 클래스의 이름을 반환한다.
hashCode()	현재 인스턴스의 해시 코드값(객체를 식별하는데 사용하는 코드)을 반환한다.
toHexString()	인자로 전달된 int값을 16진수 문자열로 변환하여 반환한다.

Object 클래스

```
Object obj = new Object();
System.out.println(obj.toString());
```

[실행 결과]
java.lang.Object@15db9742

- 일반적으로 의미 있는 문자 정보가 나오도록 재정의
- Date 클래스- 현재 시스템의 날짜와 시간 정보 리턴
- String 클래스 - 저장하고 있는 문자열 리턴
- System.out.pritnln(Object) 메소드 : Object의 toString() 의 리턴값 출력

Object 클래스

객체 복제(clone())

- 자신을 복사해서 새로운 객체를 생성한다.(원본 객체의 필드 값과 동일한 값을 가지는 새로운 객체 생성하는 것)

```
protected native Object clone()
                                    throws CloneNotSupportedException;
```

원래의 객체는 보존한 채로 clone() 메서드로 새로운 객체를 생성하여 작업하면 복사본에 변경 작업을 할 수 있다. 본질적으로 이 메서드는 다른 종류의 생성자나 마찬가지다. 복사본 객체는 클래스의 불변 규칙을 지켜야 한다. 즉 복사본 객체가 변경되어도 원본 객체에는 영향을 미쳐서는 안 된다.

Object 클래스

- 복제 종류

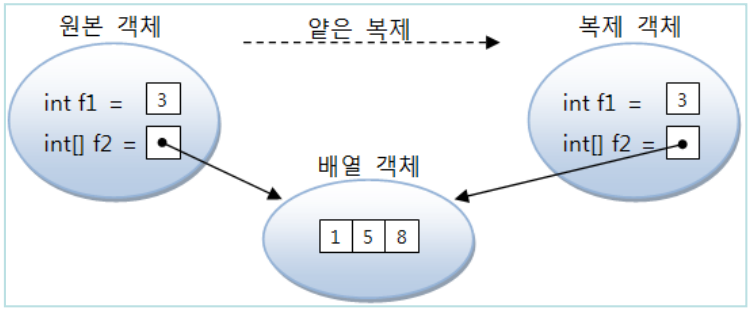
  - 얕은 복제(thin clone): 필드 값만 복제(참조 타입 필드는 번지 공유)

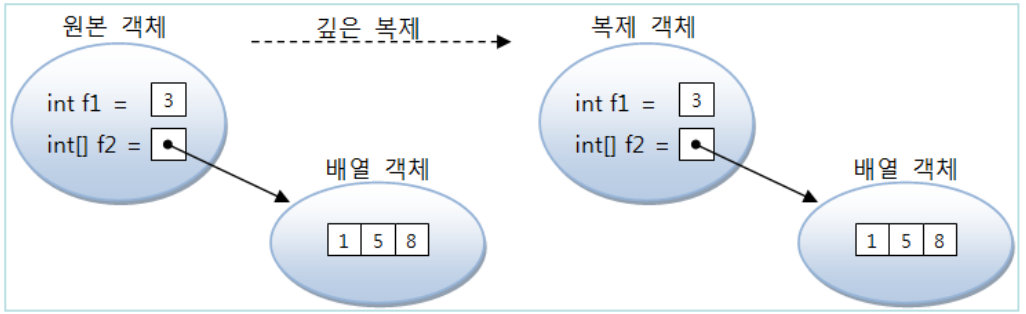
  - 깊은 복제(deep clone): 참조하고 있는 객체도 복제 clone() 메소드 재정의하고 참조 객체도 복제해야 한다.

- Object 클래스로부터 상속받은 클래스에서 clone() 메서드를 사용하려면 java.lang.Cloneable 인터페이스가 반드시 필요하다. 이 인터페이스는 표지 인터페이스(marker interface)로 인터페이스 내부에 정의된 메서드가 없다. 그래서 클래스에서 해당 인터페이스를 구현만 선언하면 별도로 구현하지 않고도 사용할 수 있다.

Object 클래스

- 객체 복제(clone())
- Object의 clone() 메소드
  - 동일한 필드 값을 가진 얕은 복제된 객체 리턴
  - java.lang.Cloneable 인터페이스 구현한 객체만 복제 가능





Object 클래스

Cloneable 인터페이스는 복사가 허용되는 객체라는 것을 알리려는 목적으로 사용하는 인터페이스이다. 이 인터페이스는 Object 클래스의 protected 메서드인 clone() 메서드의 기능 수행에 대해 규정한다. 만약 clone() 메서드가 호출한 객체가 Cloneable 인터페이스 클래스형이라면 Object. clone() 메서드는 이 객체의 모든 속성을 그대로 복사한 복사본을 반환한다. 하지만 Cloneable 인터페이스 클래스형이 아니면 CloneNotSupportedException 예외를 던진다.

이 메서드(clone())를 사용하려면 다음과 같은 제약사항에 주의해야 한다.

- ✓ clone() 메서드는 protected로 선언되어 있으므로 외부에서 호출할 수 없다.
- ✓ Cloneable 인터페이스가 반드시 필요하다.

Object 클래스

- 객체 소멸자(finalize())
- GC(Garbage Collector)는 객체를 소멸하기 직전 객체 소멸자(finalize()) 실행
- Object의 finalize() 는 기본적으로 실행 내용이 없음
- 객체가 소멸되기 전에 실행할 코드가 있다면?

  - Object의 finalize() 재정의

```
@Override
protected void finalize() throws Throwable{
        System.out.println(no + "번 객체의 finalize()가 실행됨.");
}
```

- 될 수 있으면 소멸자는 사용하지 말 것
- GC는 메모리의 모든 쓰레기 객체를 소멸하지 않음
- GC의 구동 시점이 일정하지 않음

Objects 클래스

❖Objects 클래스 (java.util.Objects)

- Objects의 유틸리티 클래스

메서드	설 명
boolean equals (Object a, Object b)	두 객체의 주소 비교
boolean isNull(Object o)	객체가 null 인지 조사
boolean nonNull(Object o)	객체가 null이 아닌지를 조사
requireNonNull(Object obj)	객체가 null인 경우 예외 발생
requireNonNull(Object obj, String message)	객체가 null인 경우 예외 발생 (주어진 예외 메시지 포함)
toString()	객체가 toString() 리턴값 리턴
toString(Object o, String nullDefault)	객체가 toString() 리턴값 리턴. 첫번째 매개값이 null일 경우 두번째 매개값 리턴.

Objects 클래스

- 널 여부 조사(isNull(), nonNull(), requireNonNull())
  - Objects.isNull(Object obj)
    - obj가 null일 경우 true
  - Objects.nonNull(Object obj)
    - obj가 not null일 경우 true
  - requireNonNull()

메서드	설 명
requireNonNull(Object obj)	not null 이면 obj null 이면 NullPointerException
requireNonNull(Object obj, String message)	not null 이면 obj null 이면 NullPointerException(message)

Objects 클래스

❖객체 문자정보(toString())

- 객체의 문자정보 리턴(사용 방법: Objects.toString())

- toString(Object o, String nullDefault)
- 첫 번째 매개값이 not null - toString ( )으로 얻은 값을 리턴
- null이면 “null” 또는 두 번째 매개값인 nullDefault 리턴

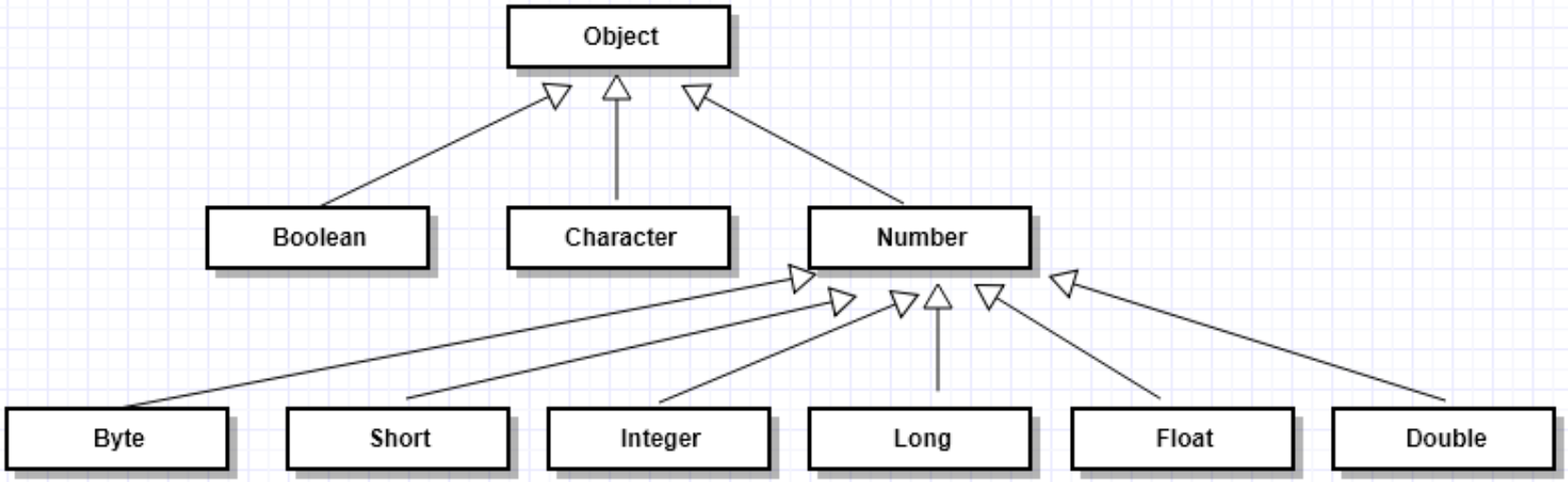
Wrapper 클래스

자바의 자료형은 기본형과 참조형으로 구분된다고 했다. 기본형 8가지 중에서 boolean 형을 제외하고 나머지 7가지 자료형 간에는 형변환이 가능하고, 참조형은 다루는 자료형으로 주로 클래스형이 속하며 참조형 간 형변환은 원칙적으로 지원하지 않지만, 상속 관계인 클래스 간의 형변환만 제한적으로 허용한다고 했다. **기본형과 참조형간의 형변환을 가능하게 해주는 클래스가 래퍼 클래스이다.**

예를 들어, 메소드의 인수로 객체 타입만이 요구되면, 기본 타입의 데이터를 그대로 사용할 수는 없다. 이때에는 기본 타입의 데이터를 먼저 객체로 변환한 후 작업을 수행해야 한다.

Wrapper 클래스

Wrapper는 “감싸다 " 라는 의미로 기본형을 참조형으로 추상화하는 과정에 비유한 말이다.



- Wrapper 객체란?
  - 기본 타입(byte, char, short, int, long, float, double, boolean) 값을 내부에 두고 포장하는 객체

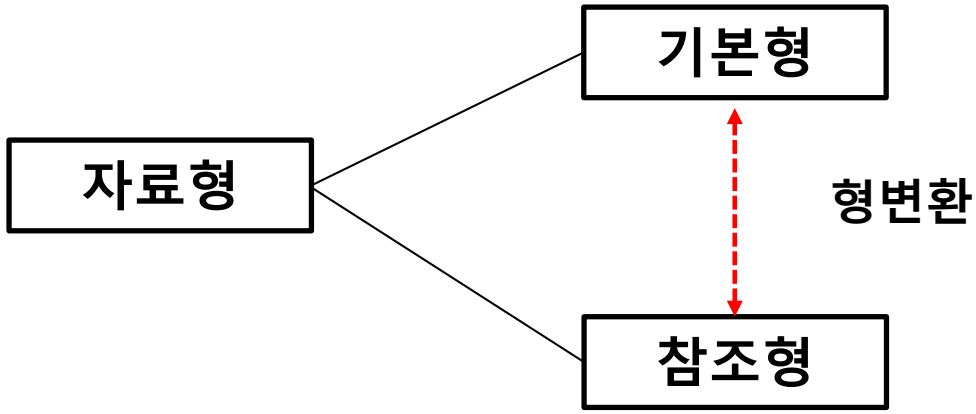
Wrapper 클래스

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

기초 자료형	Wrapper 클래스
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Wrapper 클래스

자료형에서 값을 다루는 기본형과 주소를 다루는 참조형간의 형변환을 박싱(boxing)과 언박싱(unboxing)이라 한다.



- 박싱(Boxing)
  - 값을 다루는 기본형 데이터를 참조형 데이터로 변환하는 것을 의미.
  - 메모리에서 변환관점으로 보면 스택에 저장된 값을 힙으로 복사하는 과정이며 이 과정은 묵시적(자동)으로 이루어진다.
- 언박싱(Unboxing)
  - 반대로 힙에 저장된 값을 스택에 복사하는 과정이다. 즉 힙의 데이터를 기본형으로 변환할 때 언박싱이 일어난다.

Wrapper 클래스 – Boxing과 Unboxing

- 박싱 하는 방법 - 생성자 이용

기본 타입의 값을 줄 경우	문자열을 줄 경우
Byte obj = new Byte(10);	Byte obj = new Byte("10");
Character obj = new Character('가');	
Short obj = new Short(100);	Short obj = new Short("100");
Integer obj = new Integer(1000);	Integer obj = new Integer("1000");
Long obj = new Long(10000);	Long obj = new Long("10000");
Float obj = new Float(2.5F);	Float obj = new Float("2.5F");
Double obj = new Double(5.8);	Double obj = new Double("5.8");
Boolean obj = new Boolean(true);	Boolean obj = new Boolean("true");

Java 9부터 생성자를 이용한 **Wrapper** 객체 생성 폐기.

Wrapper 클래스

- 박싱 하는 방법 - valueOf() 메소드 이용

```
Integer obj = Integer.valueOf(100);
Integer obj = Integer.valueOf("100");
```

- 언박싱 코드
  - 각 포장 클래스마다 가지고 있는 클래스 호출
  - 메서드 형태 : 기본 타입명 + Value()

기본 타입의 값을 줄 경우	
byte num = obj.byteValue();	long num= obj.longValue();
char ch = obj.charValue();	float num = obj.floatValue();
short num = obj. shortValue();	double num = obj.doubleValue();
int num = obj.intValue();	boolean bool = obj.booleanValue();

Wrapper 클래스 – 예제

```
public class BoxingUnBoxingExample {
        public static void main(String[] args) {
                //Boxing
                Integer obj1 = Integer.valueOf(50);

                //Unboxing
                int value1 = obj1.intValue();

                System.out.println(value1);
        }
}
```

Wrapper 클래스

자바는 Wrapper 객체와 기초 자료형 사이의 변환을 자동으로 하여 주는 기능이 있다. 이것을 자동박싱(auto-boxing : 포장 클래스 타입에 기본값이 대입될 경우 발생)과 자동언박싱(auto-unboxing : 기본 타입에 포장 객체가 대입될 경우 발생)이라고 한다.

```
Integer iValue;
iValue = 10;
//정수를 자동으로 Integer 객체로 포장(auto-boxing)

System.out.println(iValue+ 1);
// iValue는 자동으로 int형으로 변환(auto-unboxing)
```

예제 6-6 : 박싱 언박싱

다음 코드에 대한 결과는 무엇인가?

```
public class AutoBoxingUnBoxingEx {
   public static void main(String[] args) {
      int n = 10;
      Integer intObject = n;  // auto boxing
      System.out.println("intObject = " + intObject);

      int m = intObject + 10; // auto unboxing
      System.out.println("m = " + m);
   }
}
```

```
intObject = 10
m = 20
```

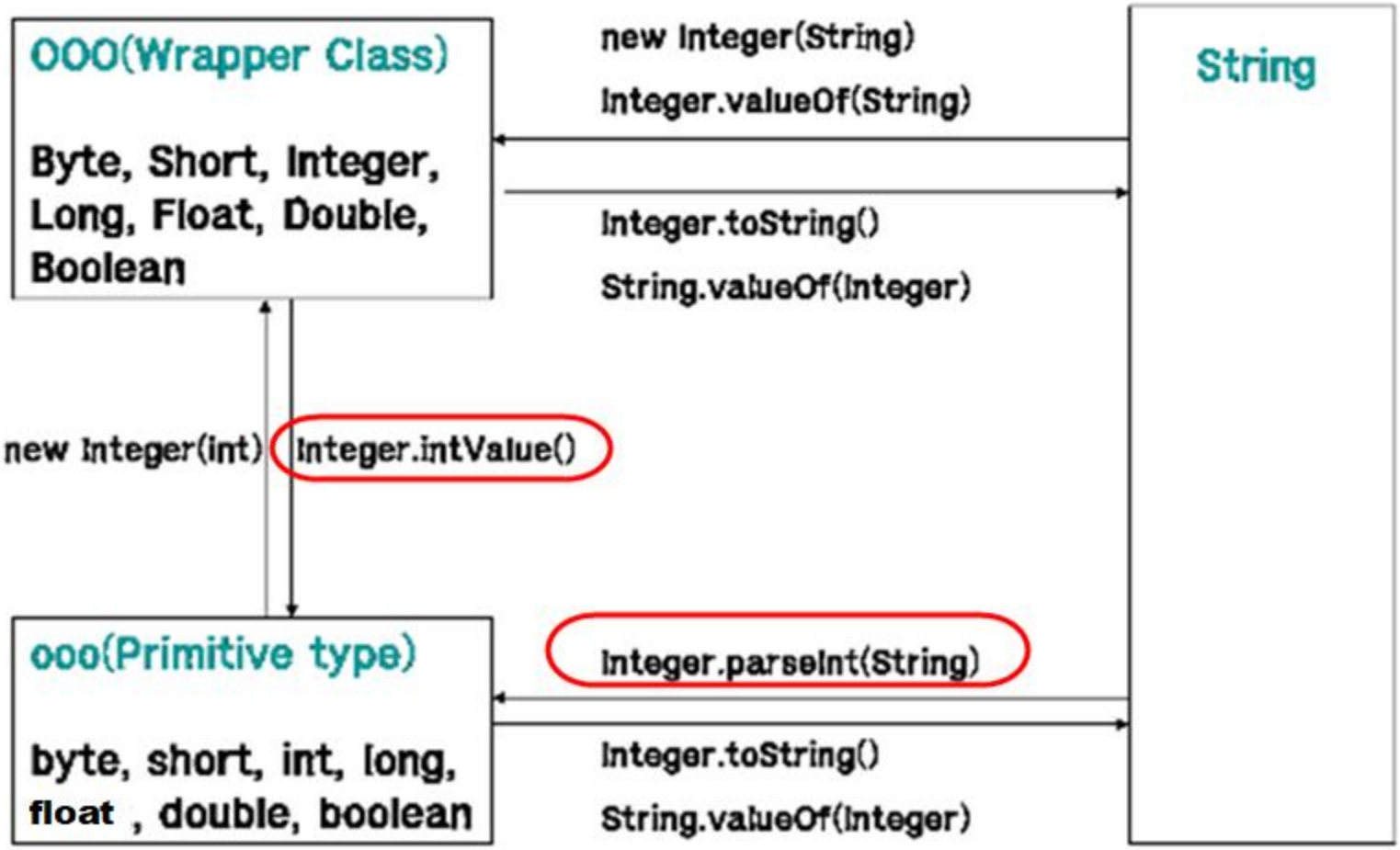
Wrapper 클래스

- 문자열을 기본 타입 값으로 변환
  - parse + 기본타입 명 → 정적 메소드

메소드명	설명
byte num = Byte.parseByte(“10”);	문자열을 byte형으로 반환한다.
short num = Short.parseShort(“100”)	문자열을 short형으로 반환한다.
int num = Integer.parseInt(“10”)	문자열을 int형으로 반환한다.
double num = Double.parseDouble(“2.45”)	문자열을 double형으로 반환한다.
boolean bool = Boolean.parseBoolean(“true”)	문자열을 boolean형으로 반환한다.

Wrapper 클래스

기본 자료형과 문자열 사이의 변환



String 클래스

String 클래스

✓ 자바는 기초 자료형으로 문자열을 지원하지 않는다.

✓ 자바에서는 문자열을 클래스형으로 다룬다

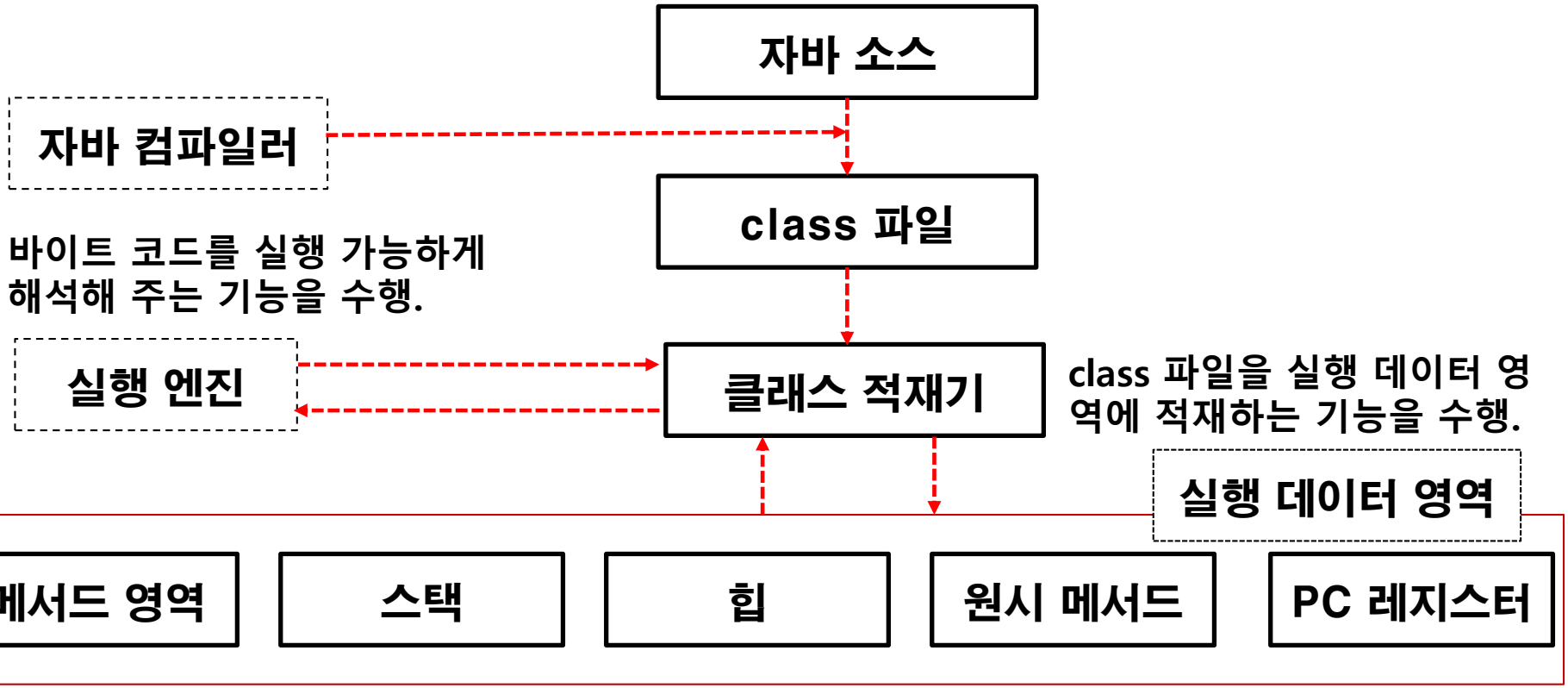
```
public final class String
    implements java.io.Serializable, Comparable<String>,
CharSequence {
    /** The value is used for character storage. */
    private final char value[];
```

실제로는 내부에서 char형의 배열 객체를 다룬다. String 클래스에는 문자열을 저장하고자 문자형 배열 변수(char[]) value를 인스턴스 변수로 정의해 놓고 있다. 객체를 생성할 때 생성자의 매개변수로 입력받는 문자열은 이 인스턴스 변수에 문자형 배열로 저장된다.

String 클래스

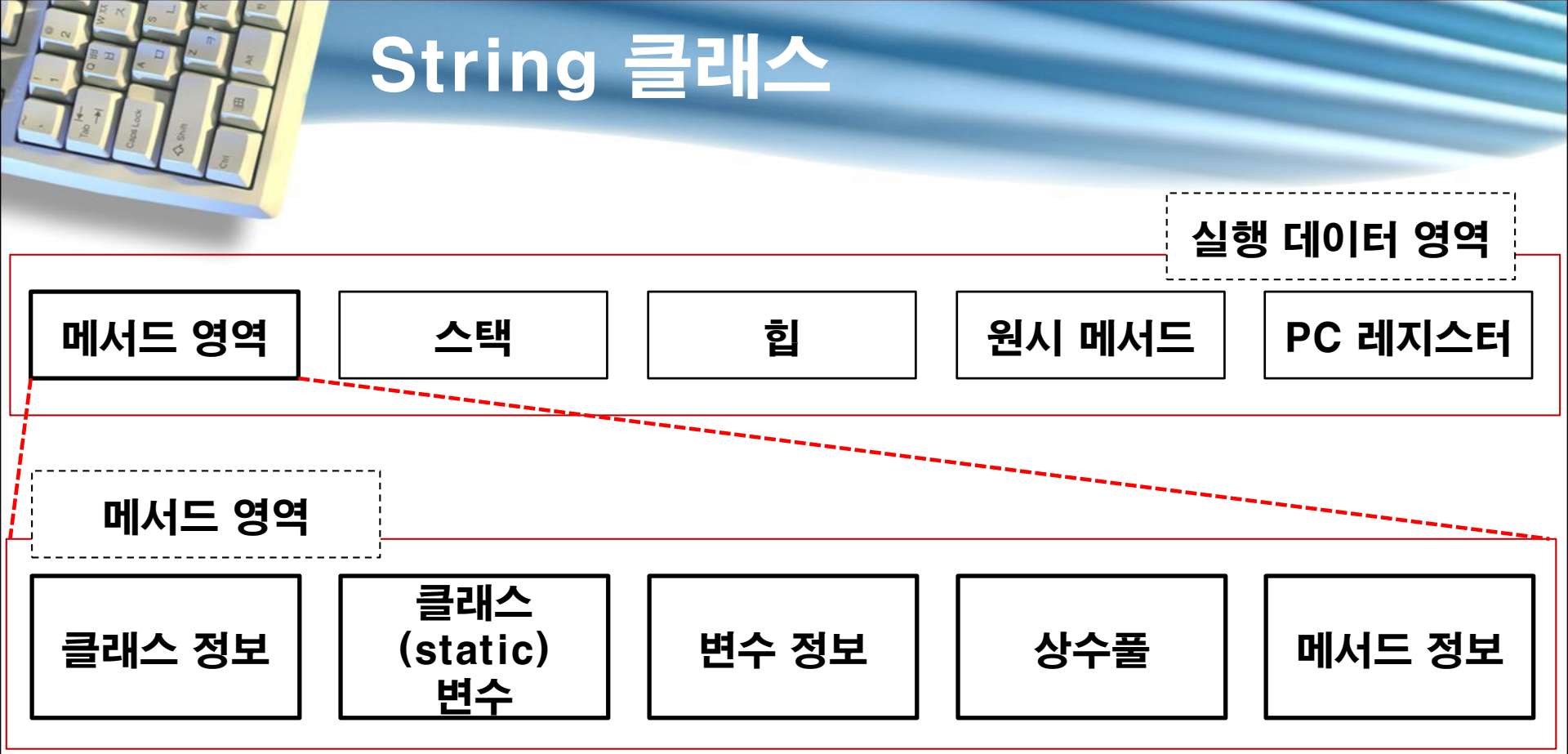
2) 문자열 리터럴을 지정해서 문자열 객체 생성

```
String str = "자바";
```



프로그램을 실행하기 위해 운영체제로부터 할당받은 메모리 공간.

String 클래스



상수풀(constant pool)이라는 영역이 있는데 클래스에서 사용된 상수를 저장하는 곳이다. 상수값은 중복되는 경우 재사용하기 위한 영역이다. 보통 문자 상수와 클래스형, 속성, 메서드도 상수풀에 저장한다.

String 클래스

- 문자열 객체 생성

String 클래스의 생성자를 사용하거나 문자열 리터럴을 지정하여 생성한다.

1) String 클래스의 생성자를 사용해서 문자열 객체 생성

```
String str1 = new String("자바");
String str2 = new String("자바");
```

String 클래스는 문자열 객체를 생성하면서 상수로 인식한다. 그래서 한번 생성되면 문자열을 수정할 수 없다.

2) 문자열 리터럴를 사용해서 문자열 객체 생성

```
String str3 = "자바";
String str4 = "자바";
```

String 클래스

String 클래스의 생성자

생성자	설명
String(String str)	문자열을 갖는 String 객체를 생성한다.
String(char[] value)	문자형 배열을 갖는 String 객체를 생성한다.
String(byte[] value)	바이트 배열을 갖는 String 객체를 생성한다.

String 클래스

– 사용 빈도 높은 메소드

리턴타입	메서드명(매개변수)	설명
char	charAt(int index)	지정된 인덱스 위치에 있는 문자를 리턴한다.
boolean	equals(Object anObj)	전달받은 문자열(anObj)과 String 객체의 문자열이 같은지 비교해 결과를 리턴한다.
byte[]	getBytes()	문자열을 바이트배열(byte[])로 리턴한다
byte[]	getBytes(Charset charset)	주어진 문자셋으로 인코딩한 byte[]로 리턴한다.
int	indexOf(String str)	문자열내에서 주어진 문자열의 위치를 리턴한다 (존재하지 않으면 -1을 리턴)
int	length()	총 문자의 수를 리턴한다.

String 클래스

– 사용 빈도 높은 메소드

리턴타입	메서드명(매개변수)	설명
String	replace(char oldChar, char newChar)	문자열내에서 oldChar를 newChar로 변환하여 리턴한다
String	substring(int beginIndex, int endIndex)	시작인덱스부터 마지막인덱스 전까지 문자열을 리턴한다.
String	toLowerCase()	소문자로 변환한 새로운 문자열을 리턴한다.
String	toUpperCase()	대문자로 변환한 새로운 문자열을 리턴한다.
String	trim()	앞뒤 공백을 제거한 새로운 문자열을 리턴한다.
String	valueOs(int i) valueOs(double d)	기초 자료형의 값을 문자열로 리턴한다.

String 클래스

– 사용 빈도 높은 메소드

리턴타입	메서드명(매개변수)	설명
String	concat(String str)	전달받은 문자열(str)을 해당 메서드를 호출한 String 객체의 문자열 뒤에 합쳐 새로운 문자열을 생성해서 반환한다.
String	contains(String str)	전달받은 문자열(str)이 String 객체의 문자열에 포함되었는지를 검사한다.
String	intern()	문자열을 상수 풀에 등록하고 이미 같은 문자열이 존재하면 해당 문자열의 참조값을 반환한다.

String 클래스

- 바이트 배열로 변환(getBytes())
  - 시스템의 기본 문자셋으로 인코딩된 바이트 배열 얻기

```
byte[] bytes = "문자열".getBytes();
```

  - 특정 문자셋으로 인코딩 된 바이트 배열 얻기

```
byte[] bytes = "문자열".getBytes("EUC-KR");
byte[] bytes = "문자열".getBytes("UTF-8");
```

[참고] 디코딩

```
String str = new String(byte[] bytes, String charsetName);
```

String 클래스

- 문자열 찾기(indexOf())
  - 매개값으로 주어진 문자열이 시작되는 인덱스 리턴
  - 주어진 문자열이 포함되어 있지 않으면 -1 리턴

```
String subject = “자바 프로그래밍”;
Int index = subject.indexOf(“프로그래밍”);
```

자	바		프	로	그	래	밍
0	1	2	3	4	5	6	7

  - 특정 문자열이 포함되어 있는지 여부 따라 실행 코드 달리할 때 사용

String 클래스

- 문자열 변환(valueOf())
  - 기본 타입의 값을 문자열로 변환

static String valueOf(boolean b)	static String valueOf(char c)
static String valueOf(int i)	static String valueOf(logn l)
static String valueOf(double d)	static String valueOf(float f)

- 문자열 길이(length()) – 공백도 문자에 포함.
- 문자열 대치(replace())
  - 첫 번째 매개값인 문자열 찾음
  - 두 번째 매개값인 문자열로 대치

```
String oldStr = "자바 프로그래밍";
String newStr = oldStr.replace("자바", "JAVA");
```

StringBuffer 클래스

- 문자열 결합 연산자 +
  - 원본 문자열이 변경되지 않으니 문자열을 수정할 때마다 메모리에 새로운 문자열이 생성되고, 이전 문자열은 메모리에 남아 쓸데없는 자리만 차지하게 된다.

- StringBuffer, StringBuilder
  - StringBuffer 객체에 저장된 문자열은 수정할 수 있다.
  - String 객체와 다르게 StringBuffer 객체는 + 연산자를 통해 매번 새로운 String 객체를 생성하지 않고, append() 메서드를 통해 String 객체 하나만으로도 문자열을 다룰 수 있다.
  - 버퍼(buffer: 데이터를 임시로 저장하는 메모리)에 문자열 저장
  - 버퍼 내부에서 추가, 수정, 삭제 작업 가능

StringBuffer 클래스

StringBuffer 객체는 내부적으로 문자열을 저장하는 메모리를 가지고 있다. 이 메모리를 버퍼(buffer)라고 한다.

버퍼의 총 크기는 capacity가 나타내고 현재 저장된 문자열의 길이는 length이다. 버퍼의 크기는 자동적으로 조절된다. length()와 capacity()메소드는 각각 문자열의 길이 수와 버퍼의 크기를 반환한다.

```
StringBuffer sb = new StringBuffer(); //16byte의 공간 할당
sb.append("hello");
```

StringBuffer 클래스

StringBuffer만이 제공하는 가장 중요한 메소드는 append()와 insert()이다.

필드	설명
StringBuffer append(String s)	매개변수로 전달받은 값을 추가한다
StringBuffer delete(int start, int end)	매개변수로 전달받은 인덱스 사이의 문자열 삭제한다
StringBuffer insert(int offset, String str)	특정 위치에 문자열을 삽입하게 된다.
StringBuffer replace(int start, int end, String s)	매개변수로 전달받은 범위를 세번째 매개변수의 문자열로 대체한다
StringBuffer reverse()	저장된 문자들의 순서를 역순으로 한다.

StringBuffer 클래스

insert()는 여러 가지 형태로 중복 정의된 메소드로서 문자열의 특정 위치에 여러 종류의 데이터를 문자열로 바꾸어 삽입한다. 역시 중복 정의된 append()는 문자열 버퍼의 끝에 여러 종류의 데이터를 추가하는 메소드이다.

```
StringBuffer sb = new StringBuffer("10 +20 = ");
sb.append(10+20);

sb.insert(0, "수식 ");
System.out.println(sb);
```

StringTokenizer 클래스

❖문자열 분리 방법

- String의 split() 메소드 이용
- java.util.StringTokenizer 클래스 이용

❖String의 split()

- 정규표현식을 구분자로 해서 부분 문자열 분리
- 배열에 저장하고 리턴

```
홍길동&이수홍,박연수,김자바-최명호
```

```
String text = "홍길동&이수홍,박연수,김자바-최명호";
String[] name = text.split("&|,|-")
```

StringTokenizer 클래스

이 클래스(StringTokenizer)는 문자열을 구분자를 사용해 나누어준다. 이때 나누어진 문자열을 토큰이라고 한다. 구분자를 지정하지 않으면 공백이 구분자가 된다.

사용자로부터 받은 텍스트나 파일에 저장된 텍스트를 처리하는 경우에 유용하게 사용할 수 있다. 예를 들어서 날짜를 "2009.12.25"과 같은 문자열로 받아서 "2009" ,"12", "25"로 분리하는데 사용할 수 있다. 생성자에서 문자열을 분리하는데 사용하는 분리자를 지정할 수 있다.

StringTokenizer 클래스

생성자	설명
StringTokenizer(String str)	주어진 문자열을 위한 StringTokenizer 객체를 생성한다.
StringTokenizer(String str, String delim)	주어진 문자열을 위한 StringTokenizer 객체를 생성한다. 분리자로 delim을 사용한다.
StringTokenizer(String str, String delim, boolean returnDelims)	주어진 문자열을 위한 StringTokenizer 객체를 생성한다. 분리자로 delim을 사용한다. 구분자도 토큰으로 포함할지 여부를 나타낸다. 기본값은 false, 출력하지 않는다.

StringTokenizer 클래스

❖StringTokenizer 클래스의 메서드

```
String text = "홍길동&이수홍,박연수,김자바-최명호";
StringTokenizer st = new StringTokenizer(text, "&,-")
```

```
while(st.hasMoreTokens()){
    System.out.println(st.nextToken())
}
```

메소드명	설명
int countTokens()	문자열에 존재하는 토큰의 개수를 반환한다.
boolean hasMoreTokens()	나누어진 문자열에서 다음으로 가져올 토큰이 있는지 여부를 검사해서 그 결과를 boolean형으로 반환한다.
String nextToken()	나누어진 문자열에서 다음 토큰을 반환한다.
String nextToken(String delim)	다음 토큰을 반환하고 분리자를 delim으로 변경한다.

StringTokenizer 클래스

```
import java.util.*;

public class StringTest {
        public static void main(String[] args) {
            StringTokenizer st =
                new StringTokenizer("Will Java change
                    my life?"," ");
             System.out.println("문자수"+st.countTokens());
             while (st.hasMoreTokens()) {
                    System.out.println(st.nextToken());
              }
        }
}
```

Math 클래스

Math 클래스는 수학에서 자주 사용하는 상수들과 함수들을 미리 구현해 놓은 클래스이다. Math 클래스의 모든 메서드는 정적 메서드(static method)이므로, 객체를 생성하지 않고도 바로 사용할 수 있다.

Math.E와 Math.PI

Math 클래스에 정의되어 있는 클래스 필드는 다음과 같다.

1. Math.E : 오일러의 수라 불리며, 자연로그(natural logarithms)의 밑(base) 값으로 약 2.718을 의미한다.

2. Math.PI : 원의 원주를 지름으로 나눈 비율(원주율) 값으로 약 3.14159를 의미한다.

Math 클래스

Math 클래스의 정적 메서드

메소드	설명	예제 코드	리턴값
int abs(int a) double abs(double a)	절대값	int v1 = Math.abs(-5); double v2 = Math.abs(-3.14);	v1 = 5 v2 = 3.14
double ceil(double a)	올림값	double v3 = Math.ceil(5.3); double v4 = Math.ceil(-5.3);	v3 = 6.0 v4 = -5.0
double floor(double a)	버림값	double v5 = Math.floor(5.3); double v6 = Math.floor(-5.3);	v5 = 5.0 v6 = -6.0
int max(int a, int b) double max(double a, double b)	최대값	int v7 = Math.max(5, 9); double v8 = Math.max(5.3, 2.5);	v7 = 9 v8 = 5.3
int min(int a, int b) double min(double a, double b)	최소값	int v9 = Math.min(5, 9); double v10 = Math.min(5.3, 2.5);	v9 = 5 v10 = 2.5
double random()	랜덤값	double v11 = Math.random();	0.0<= v11<1.0
double rint(double a)	가까운 정수의 실수값	double v12 = Math.rint(5.3); double v13 = Math.rint(5.7);	v12 = 5.0 v13 = 6.0
long round(double a)	반올림값	long v14 = Math.round(5.3); long v15 = Math.round(5.7);	v14 = 5 v15 = 6

java.util 패키지

Random 클래스의 객체는 난수를 발생하는데 사용된다.

생성자	설명
Random()	새로운 난수 발생 객체를 생성한다.

메소드명	설명
int nextInt(int n)	0과 n전까지의 int형의 난수를 발생한다.
double nextDouble()	double형의 0.0과 1.0사이의 난수를 발생한다.

java.util 패키지

```
import java.util.Random;

public class RandomTest {
	public static void main(String[] args)
	{
		Random random = new Random();
		for (int i = 0; i < 10; i++)
			System.out.println(random.nextInt(100));
	}
}
```

Arrays 클래스

Arrays(java.util.Arrays) 클래스는 배열을 다루는 다양한 메소들을 가지고 있다. Arrays 클래스의 모든 메소드는 정적 메소드(static method)이므로, 객체를 생성하지 않고도 바로 사용할 수 있다.

메소드명	설명
static void sort(int[] a)	전달받은 배열의 모든 요소를 오름차순으로 정렬한다(지정된 int형의 배열을 정렬한다.)
static void fill(int[] a, int val)	주어진 val값을 가지고 배열을 채운다.
static boolean equals(int[] a, int[] b)	주어진 두 개의 배열이 같으면 true를 반환한다
static List asList(Object[] a)	주어진 배열을 고정 길이의 리스트로 변환한다.
static int[] copyOf(int[] original, int newLength)	전달받은 배열의 전체 또는 특정 길이만큼을 새로운 배열로 복사하여 반환한다.

Arrays 클래스

```
import java.util.Arrays;

public class ArraysTest {
        public static void main(String[] args) {
                int[] array = { 9, 4, 5, 6, 2, 1 };
                Arrays.sort(array);
                printArray(array);

                Arrays.fill(array, 8);
                printArray(array);
        }
```

Arrays 클래스

```
        private static void printArray(int[] array) {
                System.out.print("[");
                for (int i = 0; i < array.length; i++)
                        System.out.print(array[i] + " ");
                System.out.println("]");
        }
}
```

StringTokenizer 클래스

1. 문자열을 처리하는 프로그램을 작성하여 보자.(StringSort)

(1) StringTokenizer 클래스를 이용하여서 사용자로부터 받은 문자열을 단어로 분리(, 공백)한다. 분리된 단어와 단어의 개수를 출력한다.

(2) 단어들을 문자열 배열에 넣고 이 배열을 Arrays의 sort() 메소드를 이용하여 정렬한다.

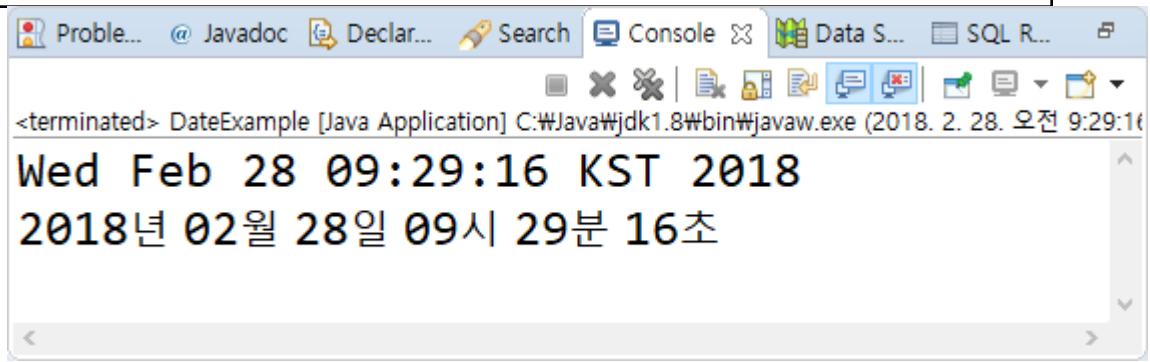
Date, Calendar 클래스

❖Date 클래스

- 날짜를 표현하는 클래스
- 날짜 정보를 객체간에 주고 받을 때 주로 사용

```
Date now = new Date();
String strNow1 = now.toString();
System.out.println(strNow1);

SimpleDateFormat sdf = new SimpleDateFormat(
                    "yyyy년 MM월 dd일 hh시 mm분 ss초");
String strNow2 = sdf.format(now);
System.out.println(strNow2);
```



Date, Calendar 클래스

❖Calendar 클래스

- Calendar 클래스를 이용하여 날짜와 시간에 관한 처리를 하는 **추상 클래스**
- **OS에 설정된 시간대(TimeZone) 기준의 Calendar 객체 얻기**

```
Calendar now = Calendar.getInstance();
```

- **다른 시간대의 Calendar 객체 얻기**

```
TimeZone timeZon =
TimeZone.getTimeZone("America/Los_Angeles");
Calendar now = Calendar.getInstance(timeZon);
```

Date, Calendar 클래스

❖Calendar 클래스

- 날짜 및 시간 정보 얻기

```
Calendar now = Calendar.getInstance();

int year = now.get(Calendar.YEAR);                  //년도
int month = now.get(Calendar.MONTH)+1;              //월
int day = now.get(Calendar.DAY_OF_MONTH);           //일
int week = now.get(Calendar.DAY_OF_WEEK);           //요일
int amPm = now.get(Calendar.AM_PM);                 //am(0)pm(1)
int hour = now.get(Calendar.HOUR);                  //시간
int minute = now.get(Calendar.MINUTE);              //분
int second = now.get(Calendar.SECOND);              //초
```

Format 클래스

❖형식(Format) 클래스

- 숫자와 날짜를 원하는 형식의 문자열로 변환
- 종류
  - 숫자 형식: DecimalFormat
  - 날짜 형식: SimpleDateFormat
  - 매개변수화된 문자열 형식: MessageFormat

❖숫자 형식 클래스(DecimalFormat)

- 적용할 패턴 선택해 생성자 매개값으로 지정 후 객체 생성

```
DecimalFormat df = new DecimalFormat("#,###.0");
String result = df.format(1234567.89);
```

패턴문자	의미
0	숫자 한자리를 표시한다. 숫자가 없는 경우 0을 표시한다.
#	숫자 한자리를 표시한다. 숫자가 없는 경우 아무것도 표시하지 않는다.

Format 클래스

❖ 날짜 형식 클래스(SimpleDateFormat)

```
SimpleDateFormat sdf = new SimpleDateFormat (
                                    "yyyy년 MM월 dd일");
String strNow = sdf.format(new Date());
```

패턴문자	의미	패턴문자	의미
y	년	H	시(0~23)
M	월	h	시(1~12)
d	일	K	시(0~11)
D	전체일(1~365)	k	시(1~24)
E	요일	m	분
a	오전/오후	s	초
w	년도의 몇번째 주	S	밀리초(1/1000)
W	월의 몇번째 주		

System 클래스

- System 클래스 용도
  - 프로그램 종료, 키보드로부터 입력, 모니터 출력, 메모리 정리, 현재 시간 읽기
  - 시스템 프로퍼티 읽기, 환경 변수 읽기

System 클래스

메소드명	설명
static long currentTimeMillis()	밀리초 단위로 현재 시각을 반환한다.
static void exit(int status)	현재 실행 중인 자바 가상 머신을 중단한다.
static String getenv(String name)	지정된 환경 변수의 값을 얻는다.
static void String getProperty(String key)	키에 의해서 지정된 시스템의 특정을 얻는다.

System 클래스

- 프로그램 종료(exit())
  - 기능 - 강제적으로 JVM 종료
  - int 인수값을 지정하도록 - 종료 상태 값
    정상 종료일 경우 0, 비정상 종료일 경우 0 이외 다른 값. 어떤 값 주더라도 종료

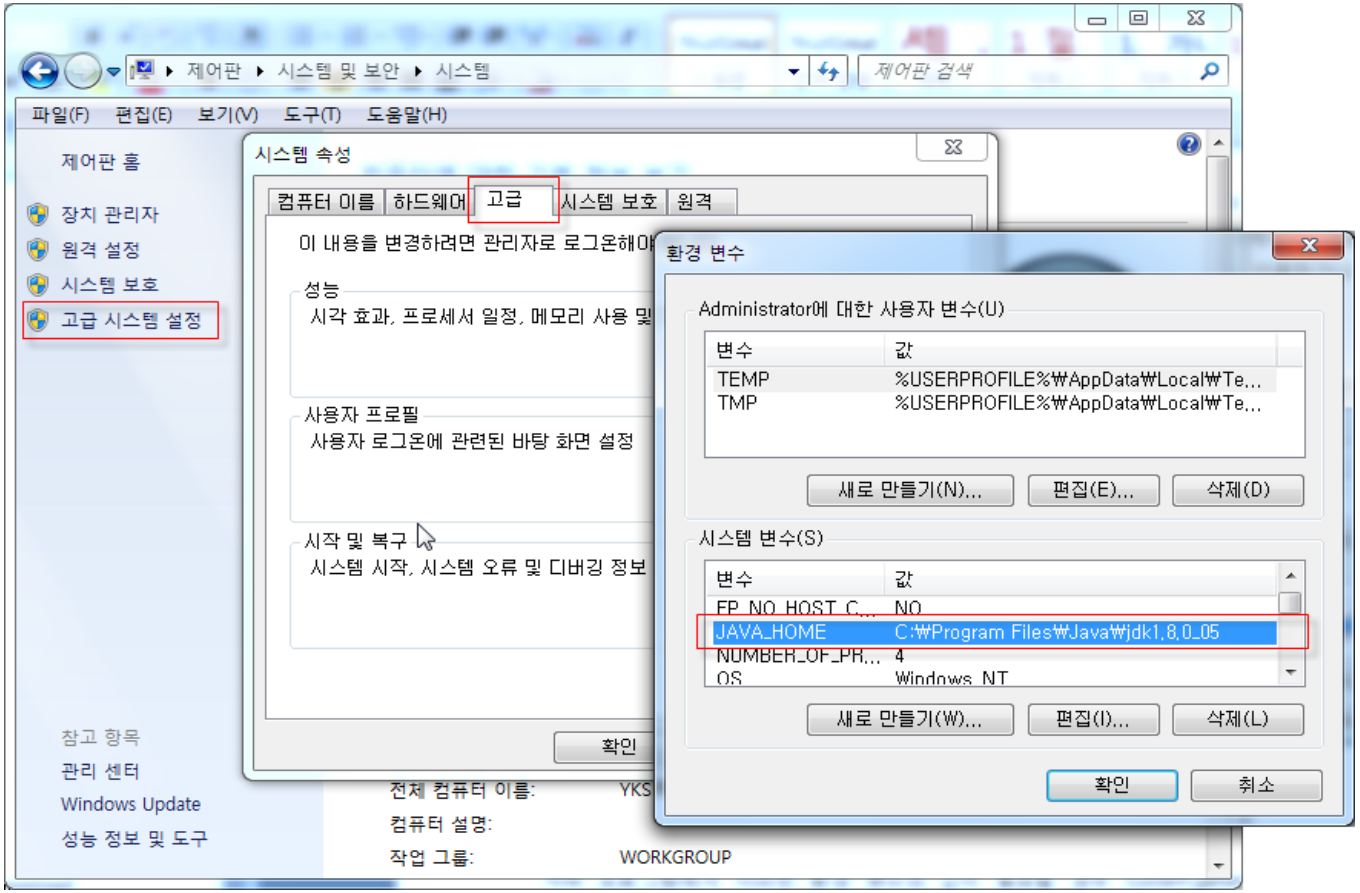
- 현재 시각 읽기
  - 현재 시간을 읽어 밀리 세컨드(currentTimeMillis() -> 1/1000초) 단위의 long값 리턴

```
long time = System.currentTimeMillis();
```

  - 주로 프로그램 실행 소요 시간 구할 때 이용

System 클래스

- 환경 변수 읽기(getenv())
  - 운영체제가 제공하는 환경 변수 값 (문자열) 을 읽음



System 클래스

- 시스템 프로퍼티 읽기(getProperty())
  - 시스템 프로퍼티란?

    JVM이 시작할 때 자동 설정되는 시스템의 속성값.
  - 대표적인 키와 값

키(key)	설명	값(value)
java.version	자바의 버전	1.7.0_25
java.home	사용하는 JRE 의 파일 경로	<jdk 설치경로>₩jre
os.name	Operating system name	Windows 7
file.separator	File separator ("/" on UNIX)	₩
user.name	사용자의 이름	사용자계정
user.home	사용자의 홈 디렉토리	C:₩Users₩사용자계정
user.dir	사용자가 현재 작업 중인 디렉토리 경로	다양

  - 시스템 프로퍼티 읽어오는 법

```
String value = System.getProperty(String key);
```

System 클래스

System 클래스는 실행 시스템과 관련된 속성과 메소드를 제공한다. 콘솔에 출력하는 메소드인 System.out.println() 이 바로 System클래스가 제공하는 메소드 중의 하나이다.

System 클래스 안에 들어 있는 out은 PrintStream 타입의 객체로 정적 변수로 선언되어 있으며 모니터를 나타낸다. System 클래스의 in은 InputStream 타입의 객체로 역시 정적 변수로서 키보드를 나타낸다.

필드	설명
static PrintStream err	표준 오류 출력 스트림
static InputStream in	표준 입력 스트림
static PrintStream out	표준 출력 스트림

Class 클래스

- Class 클래스
  - 클래스와 인터페이스의 메타 데이터 관리
    - 메타데이터: 클래스의 이름, 생성자 정보, 필드 정보, 메소드 정보
    - 생성자가 존재하지 않는다.
- Class 객체 얻기(getClass(), forName())
  - 객체로부터 얻는 방법

```
Car car = new Car();
Class c1 = car.getClass();
```

  - 문자열로부터 얻는 방법

```
Class c2 = Class.forName("exam_api_class.Car");
```

Class 클래스

- **Class 클래스의 주요 메서드에 대해서 알아보면 다음과 같다.**

메서드	설명
String getName()	클래스의 이름을 리턴한다.
Package getPackage()	클래스의 패키지 정보를 패키지 클래스 타입으로 리턴한다.
Field[] getFields()	public으로 선언된 변수 목록을 Field 클래스 배열 타입으로 리턴한다.
Field getField(String name)	public으로 선언된 변수를 Field 클래스 타입으로 리턴한다.
Field[] getDeclaredFields()	해당 클래스에서 정의된 변수 목록을 field 클래스 배열 타입으로 리턴한다.
Field getDeclaredField(String name)	name과 동일한 이름으로 정의된 변수를 Field 클래스 타입으로 리턴한다.

Class 클래스

- **Class 클래스의 주요 메서드에 대해서 알아보면 다음과 같다.**

메서드	설명
Method[] getMethods()	public으로 선언된 모든 메소드 목록을 Method 클래스 배열 타입으로 리턴한다. 해당 클래스에서 사용 가능한 상속받은 메소드도 포함된다.
Method getMethod(String name, Class... parameterTypes)	지정된 이름과 매개변수 타입을 갖는 메소드를 Method 클래스 타입으로 리턴한다.
Method[] getDeclaredMethods()	해당 클래스에서 선언된 모든 메소드 정보를 리턴한다.
Method getDeclaredMethod(String name, Class... parameterTypes)	지정된 이름과 매개변수 타입을 갖는 해당 클래스에서 선언된 메소드를 Method 클래스 타입으로 리턴한다.

Class 클래스

- **Class 클래스의 주요 메서드에 대해서 알아보면 다음과 같다.**

메서드	설명
Constructor[] getConstructors()	해당 클래스에 선언된 모든 public 생성자의 정보를 Constructor 배열 타입으로 리턴한다.
Constructor[] getDeclaredConstructors()	해당 클래스에서 선언된 모든 생성자의 정보를 Constructor 배열 타입으로 리턴한다.
int getModifiers()	해당 클래스의 접근자(modifier) 정보를 int 타입으로 리턴한다.
String toString()	해당 클래스 객체를 문자열로 리턴한다.

Class 클래스

- Method 클래스
  - Method 클래스를 이용하여 메소드에 대한 정보를 얻을 수 있다.
  - Method 클래스에는 생성자가 없어서, Method 클래스의 정보를 얻기 위해서 Class 클래스의 getMethods() 메소드를 사용하거나 getDeclaredMethod() 메소드를 통해서 인스턴스를 얻을 수 있다.

Class 클래스

Method 클래스의 주요 메서드에 대해서 알아보면 다음과 같다.

메서드	설명
Class<?> getDeclaringClass()	해당 메소드가 선언된 클래스 정보를 리턴한다.
Class<?> getReturnType()	해당 메소드의 리턴 타입을 리턴한다.
Class<?>[] getParameterTypes()	해당 메소드를 사용하기 위한 매개변수의 타입들을 리턴한다.
String getName()	해당 메소드의 이름을 리턴한다.
int getModifiers()	해당 메소드의 접근자 정보를 리턴한다.
Class<?>[] getExceptionTypes()	해당 메소드에 정의되어 있는 예외 타입들을 리턴한다.
Object invoke(Object obj, Object...args)	해당 메소드를 수행한다.
String toGenericString()	타입 매개변수를 포함한 해당 메소드의 정보를 리턴한다.

Class 클래스

- Field 클래스
  - Field 클래스는 클래스에 있는 변수들의 정보를 제공하기 위해서 사용한다.
  - Method 클래스와 마찬가지로 생성자가 존재하지 않으므로 Class 클래스의 getField() 메소드나 getDeclaredFields() 메소드를 통해서 인스턴스를 얻을 수 있다.

Class 클래스

Field 클래스의 주요 메서드에 대해서 알아보면 다음과 같다.

메서드	설명
int getModifiers()	해당 변수의 접근자 정보를 리턴한다.
String getName()	해당 변수의 이름을 리턴한다.
String toString()	해당 변수의 정보를 리턴한다.


Thank You