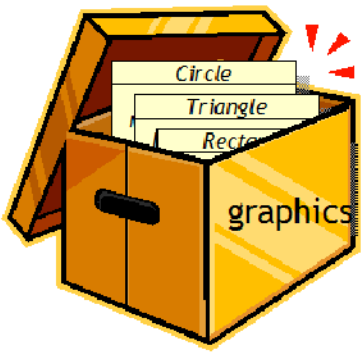
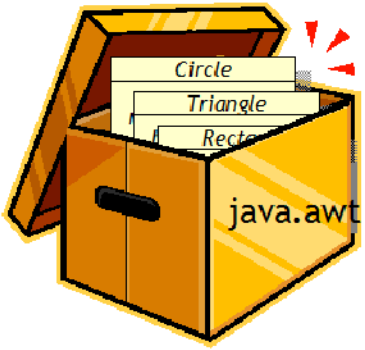
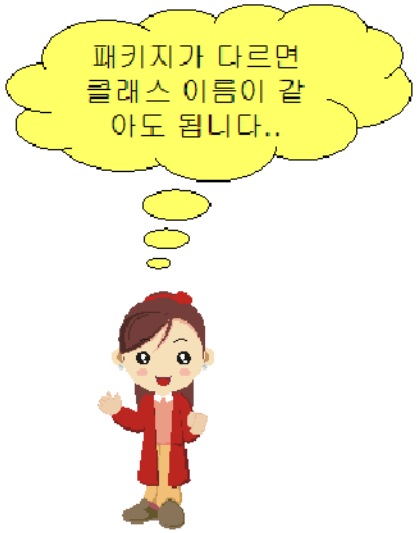
# 패키지란?

- **패키지(package) :** 클래스들을 묶은 것
- 자바 라이브러리도 패키지로 구성
  - (예) java.net 패키지- 네트워크 관련 라이브러리



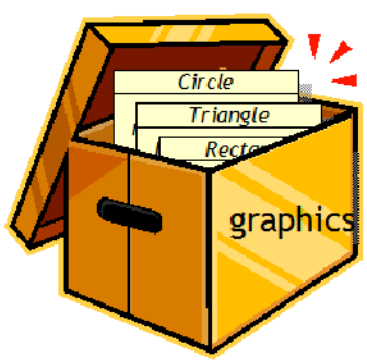
# 패키지의 장점

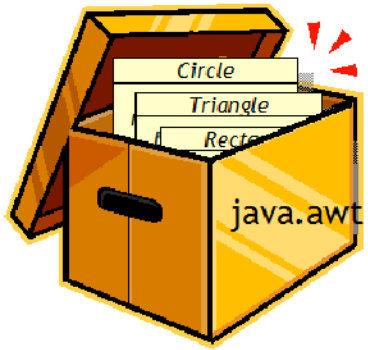
① 관련된 클래스들을 쉽게 파악

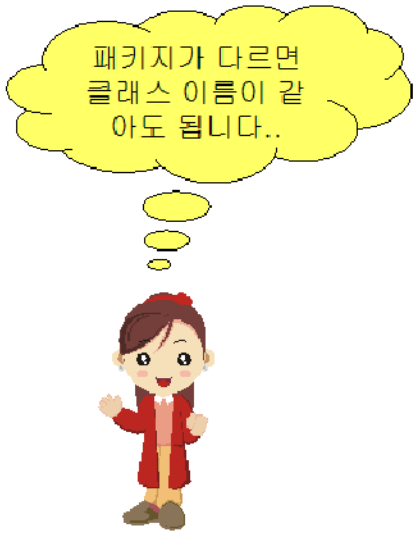
② 원하는 클래스들을 쉽게 찾을 수 있다.

③ 패키지마다 이름 공간을 따로 갖기 때문에 같은 클래스 이름을 여러 패키지가 사용

④ 패키지별로 접근에 제약을 가할 수 있다.



# 패키지 생성하기

```
package business;          // 패키지 선언
public class Order {
    ...
}
```

이 클래스는 business 패키지에 속한다.

Q: 만약 패키지 문을 사용하지 않은 경우에는 어떻게 되는가?

A: 디폴트 패키지(default package)에 속하게 된다.

# 패키지의 예

```
//Drawable.java 파일로 저장
public interface Drawable {
    ...
}

//Shape.java 파일로 저장
public abstract class Shape {
    ...
}

//Circle.java 파일로 저장
public class Circle extends Shape
    implements Drawable {
    ...
}

//Rectangle.java 파일로 저장
public class Rectangle extends Shape
    implements Drawable {
    ...
}
```

이들 클래스와 인터페이스를 graphics 패키지로 묶으면 어떨까?

# 패키지의 예

```
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

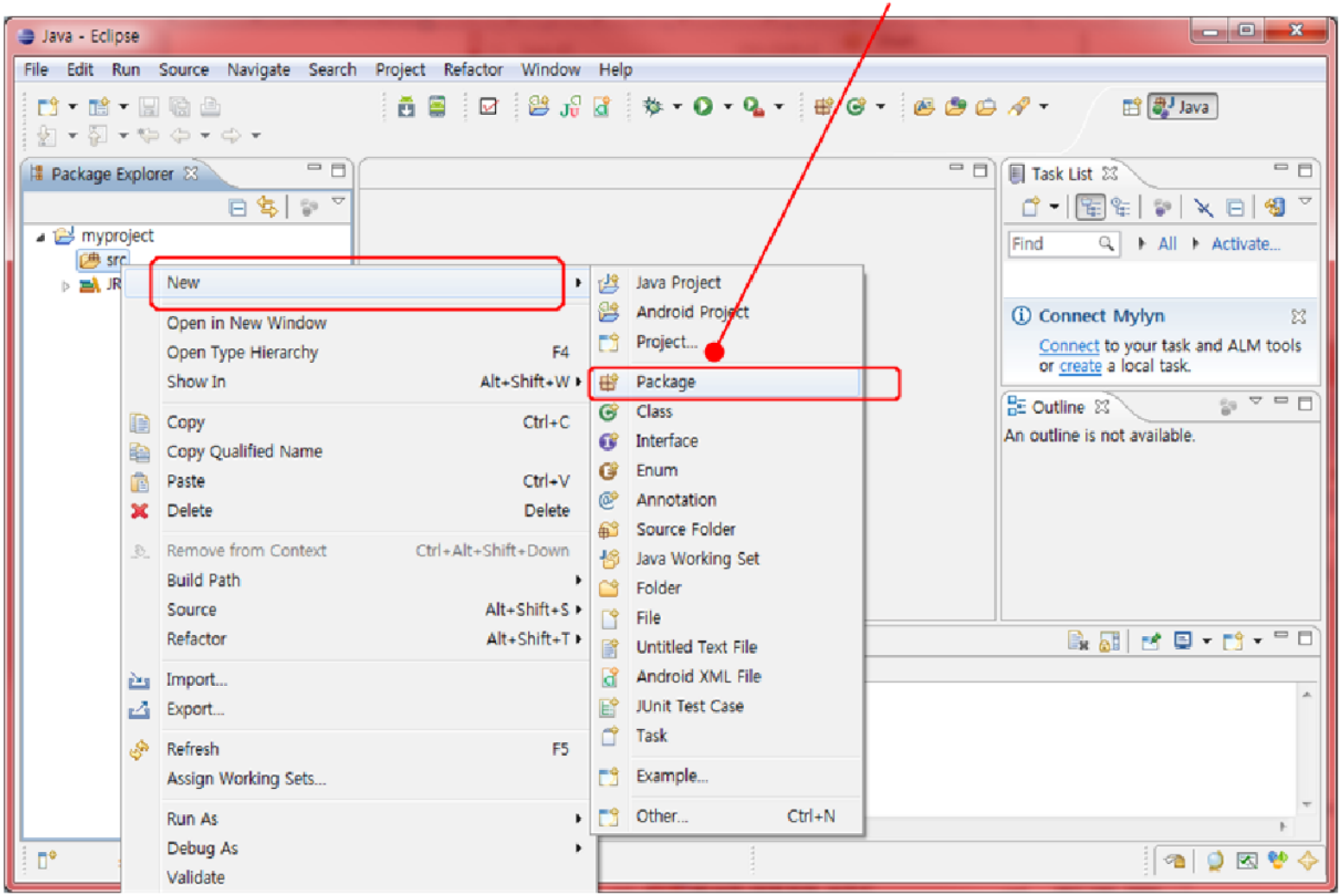
```
package graphics;
public class Rectangle extends Shape
    implements Drawable {
    ...
}
```

Rectangle.java

# 패키지의 이름

- 패키지의 이름은 일반적으로 소문자만을 사용한다.
- 패키지 이름으로 인터넷 도메인 이름의 역순을 사용한다. 예를 들면 com.company.mypackage이다.
- 자바 언어 자체의 패키지는 java나 javax로 시작한다.

# 이클립스에서 패키지 만들기



# 패키지를 사용하는 방법

1. 클래스에 패키지 이름을 붙여서 참조한다.
2. 개별 클래스를 import한다.
3. 전체 패키지를 import한다.

```
1. business.Order myOrder = new business.Order();

2. import business.Order; // business 패키지 안의 Order 클래스 포함

3. import business.*;       // 패키지 전체 포함
```

# 주의할점!!

Q: java.awt.* 문장은 java.awt.font 패키지를 포함하는가?

A: java.awt.font 패키지는 java.awt 패키지 안에 포함되지 않는다.

만약 java.awt.font의 멤버와 java.awt의 멤버를 동시에 사용하려면 다음과 같이 따로 따로 포함하여야 한다.

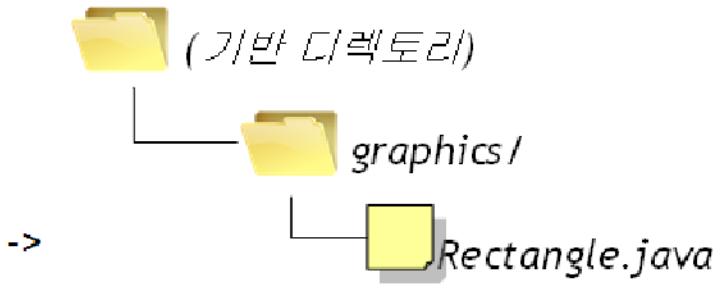
```
import java.awt.*;
import java.awt.font.*;
```

# 패키지와 소스 파일

- 자바에서는 패키지의 계층 구조를 반영한 디렉토리 구조에 소스들을 저장하여야 한다.
  - 클래스 이름 - graphics.Rectangle
  - 파일의 경로 이름- graphics\Rectangle.java

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

Rectangle.java




->

# 패키지와 클래스 파일

- 클래스 파일도 패키지 이름을 반영하는 디렉토리 구조에 저장된다.
  - <출력 디렉토리>\graphics\Rectangle.class
- 그러나 .class 파일 경로는 .java 소스 파일 경로와 달라도 된다.
  - 소스 파일: C:\sources\graphics\Rectangle.java
  - 클래스 파일: C:\classes\graphics\Rectangle.class
- 소스와 클래스 파일을 서로 다른 위치에 저장하려면 자바 가상 기계(JVM)가 이 파일들을 찾을 수 있도록 설정하여야 한다.

# 자바에서 지원하는 패키지

| 패키지 | 설명 |
|---|---|
| java.applet | 애플릿을 생성하는데 필요한 클래스 |
| java.awt | 그래픽과 이미지를 위한 클래스 |
| java.beans | 자바빈즈 구조에 기초한 컴포넌트를 개발하는데 필요한 클래스 |
| java.io | 입력과 출력 스트림을 위한 클래스 |
| java.lang | 자바 프로그래밍 언어에 필수적인 클래스 |
| java.math | 수학에 관련된 클래스 |
| java.net | 네트워킹 클래스 |
| java.nio | 새로운 네트워킹 클래스 |
| java.rmi | 원격 메소드 호출(RMI) 관련 클래스 |
| java.security | 보안 프레임워크를 위한 클래스와 인터페이스 |
| java.sql | 데이터베이스에 저장된 데이터를 접근하기 위한 클래스 |
| java.util | 날짜, 난수 생성기 등의 유틸리티 클래스 |
| javax.imageio | 자바 이미지 I/O API |
| javax.net | 네트워킹 애플리케이션을 위한 클래스 |
| javax.swing | 스윙 컴포넌트를 위한 클래스 |
| javax.xml | XML을 지원하는 패키지 |

# Java.lang 패키지

- Import 문을 사용할 필요가 없이 기본적으로 포함된다.
  - Object 클래스: 기초적인 메소드를 제공하는 모든 클래스의 조상 클래스
  - Math 클래스: 각종 수학 함수들을 포함하는 클래스
  - Wrapper 클래스: Integer와 같이 기초 자료형을 감싸서 제공하는 랩퍼 클래스들
  - String 클래스, StringBuffer 클래스: 문자열을 다루는 클래스
  - System 클래스: 시스템 정보를 제공하거나 입출력을 제공하는 클래스
  - Thread 클래스: 스레드 기능을 제공하는 클래스
  - Class 클래스: 클래스에 대한 정보를 얻기 위한 클래스

# Class 클래스

- Class 객체는 실행 중인 클래스를 나타낸다.
- Class 객체는 자바 가상 기계에 의하여 자동적으로 생성된다.
- Class 객체를 이용하면 객체의 클래스 이름을 출력할 수 있다.

```
void printClassName(Object obj) {
   System.out.println("The class of " + obj + " is " + obj.getClass().getName());
}
```

# Math 클래스

| 필드 | 설명 |
|---|---|
| static double E | 자연 로그의 밑수(the base of the natural logarithms.) |
| static double PI | 파이값 |

| 메소드 | 설명 |
|---|---|
| static double abs(double a) | 절대값 |
| static double acos(double a) | arc cosine, 반환값의 범위는 0.0에서 *pi*. |
| static double asin(double a) | arc sine, 반환값의 범위는 -*pi*/2에서 *pi*/2. |
| static double atan(double a) | arc tangent, 반환값의 범위는 -*pi*/2에서 *pi*/2. |
| static double atan2(double y, double x) | 직교 좌표계 (x, y)를 극좌표계 (r, *theta*)로 변환할 때 theta를 반환 |
| static double cos(double a) | cosine |
| static double cosh(double x) | hyperbolic cosine |
| static double exp(double x) | $e^x$ |
| static double hypot(double x, double y) | sqrt($x^2$ +$y^2$) |
| static double log(double a) | natural logarithm (base *e*) |
| static double log10(double a) | base 10 logarithm |

# Math 클래스

| | |
|---|---|
| static double **max**(double a, double b) | 큰수 |
| static double **min**(double a, double b) | 작은 수 |
| static double **pow**(double a, double b) | $a^b$ |
| static double random() | 0.0과 1.0사이의 난수를 반환 |
| static double **sin**(double a) | sine |
| static double **sinh**(double x) | hyperbolic sine |
| static double **sqrt**(double a) | 제곱근 |
| static double **tan**(double a) | tangent |
| static double **tanh**(double x) | hyperbolic tangent |
| static double **toDegrees**(double angrad) | 라디안을 디그리로 변환 |
| static double **toRadians**(double angdeg) | 디그리를 라디안으로 변환 |

# 예제

```
public class MathTest {
        public static void main(String[] args){
                double x = Math.PI;
                System.out.println(Math.sin(x));
                System.out.println(Math.random());
        }
}
```

실행 결과

```
1.2246467991473532E-16
0.2454637294993175
```

# StringBuffer 클래스

- String 클래스는 주로 상수 문자열, 즉 변경이 불가능한 문자열을 나타낸다.
- StringBuffer와 StringBuilder 클래스는 변경 가능한 문자열을 나타낸다.

```
StringBuffer sb = new StringBuffer(); // 기본적으로 16바이트의 공간이 할당된다.

sb.append("Hello");                   //  6바이트가 사용된다.
```

# StringBuffer의 메소드

| 메소드 | 설명 |
| --- | --- |
| StringBuilder append(String s)<br>StringBuilder append(char[] str)<br>StringBuilder append(char[] str, int offset, int len)<br>... | 인수는 먼저 문자열로 변환되어서 문자열에 붙여진다. |
| StringBuilder delete(int start, int end)<br>StringBuilder deleteCharAt(int index) | 지정된 문자를 삭제한다. |
| StringBuilder insert(int offset, char[] str)<br>StringBuilder insert(int index, char[] str, int offset, int len)<br>... | 첫번째 매개 변수는 데이터가 삽입될 위치의 바로 앞을 나타낸다. 두번째 매개 변수는 문자열로 변환된 후에 삽입된다. |
| StringBuilder replace(int start, int end, String s)<br>void setCharAt(int index, char c) | 지정된 위치의 문자를 변경한다. |
| StringBuilder reverse() | 저장된 문자들의 순서를 역순으로 한다. |

# System 클래스

- System 클래스는 실행 시스템과 관련된 속성과 메소드를 제공

| 필드 | 설명 |
|---|---|
| static PrintStream **err** | 표준 오류 출력 스트림("standard" error output stream) |
| static InputStream **in** | 표준 입력 스트림("standard" input stream) |
| static PrintStream **out** | 표준 입력 스트림("standard" output stream) |

| 메소드 | 설명 |
|---|---|
| static void **arraycopy**(Object src, int srcPos, Object dest, int destPos, int length) | 지정된 소스 배열을 목적지 배열로 복사한다. |
| static long **currentTimeMillis**() | 밀리초 단위로 현재 시각을 반환한다. |
| static void **exit**(int status) | 현재 실행중인 자바 가상 기계를 중단한다. |
| static String **getenv**(String name) | 지정된 환경 변수의 값을 얻는다. |
| static String **getProperty**(String key) | 키에 의해서 지정된 시스템의 특성을 얻는다. |
| static long **nanoTime**() | 나노초 단위로 현재 시각을 반환한다. |

# System 클래스

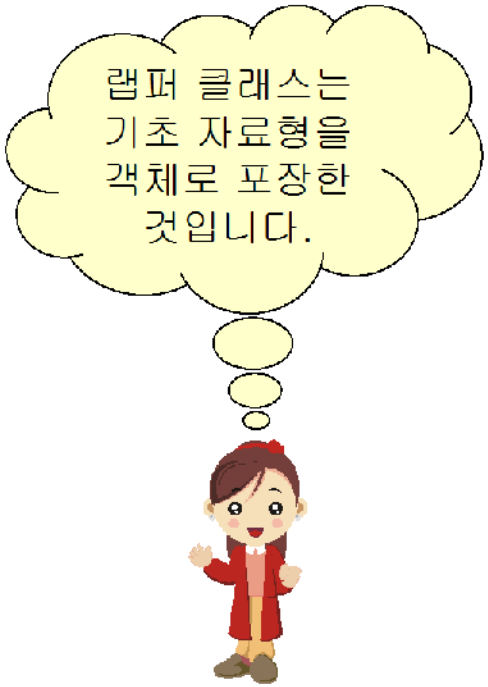
```
public class SystemTest {
        public static void main(String[] args) {
                System.out.println(System.currentTimeMillis());
                System.out.println(System.nanoTime());
                System.exit(0);
        }
}
```

실행 결과

```
1245152017843
101355767783335
```

# Wrapper 클래스

- 기초 자료형을 객체로 포장시켜주는 클래스
  - (예) Integer obj = **new** Integer(10);




| 기초 자료형 | 랩퍼 클래스 |
|---|---|
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| char | Character |
| boolean | Boolean |
| void | Void |

# Integer 클래스가 제공하는 메소드

| 반환값 | 메소드 이름 | 설명 |
|---|---|---|
| static int | intValue() | int형으로 반환한다. |
| static double | doubleValue() | double형으로 반환한다. |
| static float | floatValue() | float형으로 반환한다. |
| static int | parseInt(String s) | 문자열을 int형으로 변환한다. |
| static String | toBinaryString(int i) | int형의 정수를 2진수 형태의 문자열로 변환한다. |
| static String | toHexString(int i) | int형의 정수를 16진수 형태의 문자열로 변환한다. |
| static String | toOctalString(int i) | int형의 정수를 8진수 형태의 문자열로 변환한다. |
| static String | toString(int i) | int형의 정수를 10진수 형태의 문자열로 변환한다. |
| static Integer | valueOf(String s) | 문자열 s를 Integer 객체로 변환한다. |
| static Integer | valueOf(String s, in radix) | 문자열 s를 radix진법의 Integer 객체로 변환한다. |

# 문자열 <-> 기초 자료형

기초 자료형 -> 문자열

```
String s1 = Integer.toString(10);
String s2 = Integer.toString(10000);
String s3 = Float.toString(3.14);
String s4 = Double.toString(3.141592);
```

문자열 -> 기초 자료형

```
int i = Integer.parseInt("10");
long l = Long.parseLong("10000");
float f = Float.parseFloat("3.14");
double d = Double.parseDouble("3.141592");
```

# 오토 박싱(auto-boxing)

- Wrapper 객체와 기초 자료형 사이의 변환을 자동으로 수행한다.

```
Integer box = new Integer(10);
System.out.println(box + 1);          // box는 자동으로 int형으로 변환
```

# java.util 패키지

- 여러 가지 유틸리티 클래스 들을 제공한다.
  - collections framework,
  - legacy collection classes,
  - event model,
  - date time facilities,
  - internationalization,
  - string tokenizer,
  - random-number generator,
  - bit array

# Random 클래스

| 생성자 |
| --- |
| Random()<br>새로운 난수 발생기 객체를 생성한다. |
| Random(long seed)<br>주어진 시드를 사용하는 새로운 난수 발생기 객체를 생성한다. |

| 메소드 | 설명 |
| --- | --- |
| protected int next(int bits) | 다음 난수를 발생한다. |
| boolean nextBoolean() | booelan 형의 난수를 발생한다. |
| void nextBytes(byte[] bytes) | byte 형의 난수를 발생하여서 주어진 배열을 채운다. |
| double nextDouble() | double형의 0.0과 1.0 사이의 난수를 발생한다. |
| float nextFloat() | float형의 0.0과 1.0 사이의 난수를 발생한다. |
| double nextGaussian() | 평균이 0.0이고 표준 편차가 1.0인 가우시안 분포(정규 분포)에서 다음 난수를 발생한다. |
| int nextInt() | int 형의 난수를 발생한다. |
| int nextInt(int n) | 0과 n 상의 int형의 난수를 발생한다. |
| long nextLong() | long 형의 난수를 발생한다. |
| void setSeed(long seed) | 시드를 설정한다. |

# 예제

```
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));
        }
}
```

실행 결과

```
79
60
98
....
96
```

# Arrays 클래스

| 메소드 | 설명 |
|---|---|
| static List asList(Object[] a) | 주어진 배열을 고정 길이의 리스트로 변환한다. |
| static int binarySearch(int[] a, int key) | 주어진 값을 int형의 배열에서 이진 탐색한다. |
| static int binarySearch(Object[] a, Object key) | Object 타입의 배열에서 key를 이진 탐색하는 메소드 |
| static int[] copyOf(int[] original, int newLength) | 주어진 배열을 새로운 크기의 배열에 복사한다. |
| static int[] copyOfRange(int[] original, int from, int to) | 배열에서 주어진 구간의 값들을 새로운 배열로 복사한다. |
| static boolean equals(int[] a, int[] a2) | 주어진 두개의 배열이 같으면 true를 반환한다. |
| static void fill(int[] a, int val) | 주어진 val 값을 가지고 배열을 채운다. |
| static void sort(int[] a) | 지정된 int형의 배열을 정렬한다. |

# 예제

```
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);
                System.out.println(Arrays.binarySearch(array,9));
                Arrays.fill(array,8);
                printArray(array);
        }
         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("]");
        }
}
```

실행 결과

```
[1 2 4 5 6 9 ]
5
[8 8 8 8 8 8 ]
```

# Date 클래스

```
import java.util.*;

public class DateTest {
	public static void main(String[] args) {
		Date d = new Date();
		System.out.println(d);
		System.out.println(1900+d.getYear());
		System.out.println(d.getMonth()+1);
		System.out.println(d.getDate());

		d.setHours(12);
		d.setMinutes(34);
		d.setSeconds(0);
		System.out.println(d);
	}
}
```

실행 결과

```
Mon Jun 15 11:19:31 KST 2009
2009
6
15
Mon Jun 15 12:34:00 KST 2009
```

# Calendar 클래스

| 메소드 | 설명 |
|---|---|
| abstract void **add**(int field, int amount) | 지정된 필드에 시간을 더하거나 뺀다. |
| boolean **after**(Object when) | 현재의 객체가 주어진 시간보다 뒤이면 true를 반환한다. |
| boolean **before**(Object when) | 현재의 객체가 주어진 시간보다 앞이면 true를 반환한다. . |
| void **clear**(int field) | 지정된 필드를 정의되지 않은 상태로 변경한다. |
| int **compareTo**(Calendar anotherCalendar) | 두개의 Calendar 객체를 비교한다. |
| boolean **equals**(Object obj) | 두개의 Calendar 객체가 같으면 true를 반환한다. |
| int **get**(int field) | 주어진 필드의 값을 반환한다. |
| static Calendar **getInstance**() | 현재 시각을 나타내는 Calendar 객체를 반환한다. |
| Date **getTime**() | Calendar 객체를 Date 객체로 반환한다. |
| void **set**(int year, int month, int date, int hourOfDay, int minute, int second) | 각 필드의 값을 설정한다. |
| void **setTime**(Date date) | Date 객체의 값으로 Calendar 객체를 설정한다. |

# 예제

```
import java.util.*;

public class CalendarTest {
        public static void main(String[] args) {
                Calendar d = Calendar.getInstance();
                System.out.println(d);
                System.out.println(d.get(Calendar.YEAR)+"년");
                System.out.println(d.get(Calendar.MONTH)+1 +"월");
                System.out.println(d.get(Calendar.DATE)+"일");

                d.set(Calendar.HOUR, 12);
                d.set(Calendar.MINUTE, 34);
                d.set(Calendar.SECOND, 0);
                System.out.println(d);
        }
}
```

실행 결과

```
java.util.GregorianCalendar[...,YEAR=2009,MONTH=5,WEEK_OF_YEAR=25,...]
2009년
6월
15일
java.util.GregorianCalendar[...,YEAR=2009,MONTH=5,WEEK_OF_YEAR=25,...]
```

# StringTokenizer 클래스

- 문자열을 분석하여서 토큰으로 분리시켜 주는 기능을 제공

| 생성자 |
| --- |
| **StringTokenizer**(String str)<br>주어진 문자열을 위한 StringTokenizer 객체를 생성한다. |
| **StringTokenizer**(String str, String delim)<br>주어진 문자열을 위한 StringTokenizer 객체를 생성한다. 분리자로 delim을 사용한다. |
| **StringTokenizer**(String str, String delim, boolean returnDelims)<br>주어진 문자열을 위한 StringTokenizer 객체를 생성한다. 분리자로 delim을 사용한다. returnDelims이 true이면 분리자를 포함하여 반환한다. |

| 메소드 | 설명 |
| --- | --- |
| int **countTokens**() | 문자열에 존재하는 토큰의 개수를 반환한다. |
| boolean **hasMoreTokens**() | 다음 토큰을 가지는지를 반환한다. |
| String **nextToken**() | 다음 토큰을 반환한다. |
| String **nextToken**(String delim) | 다음 토큰을 반환하고 분리자를 delim으로 변경 |

# 예제

```
import java.util.*;

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

        }
}
```

/

실행 결과

```
Will
Java
change
my
life?
```