Power Java

제20장 Package

이번 장에서 학습할 내용

- Package의 개념
- Package로 묶는 방법
- Package 사용
- 기본 Package
- 유틸리티 Package

Package란?

- **Package** : Class들을 묶은 것
- Java 라이브러리도 package로 구성
  - (예) java.net package– 네트워크 관련 라이브러리

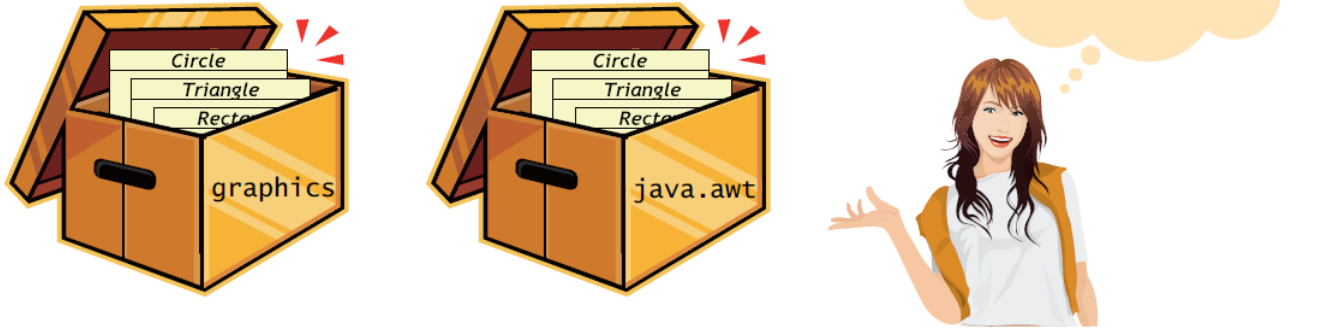


그림20-1. Package의 개념

예제

```
//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?

Package 생성하기

```
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

Q: 만약 package 문을 사용하지 않은 경우에는 어떻게 되는가?

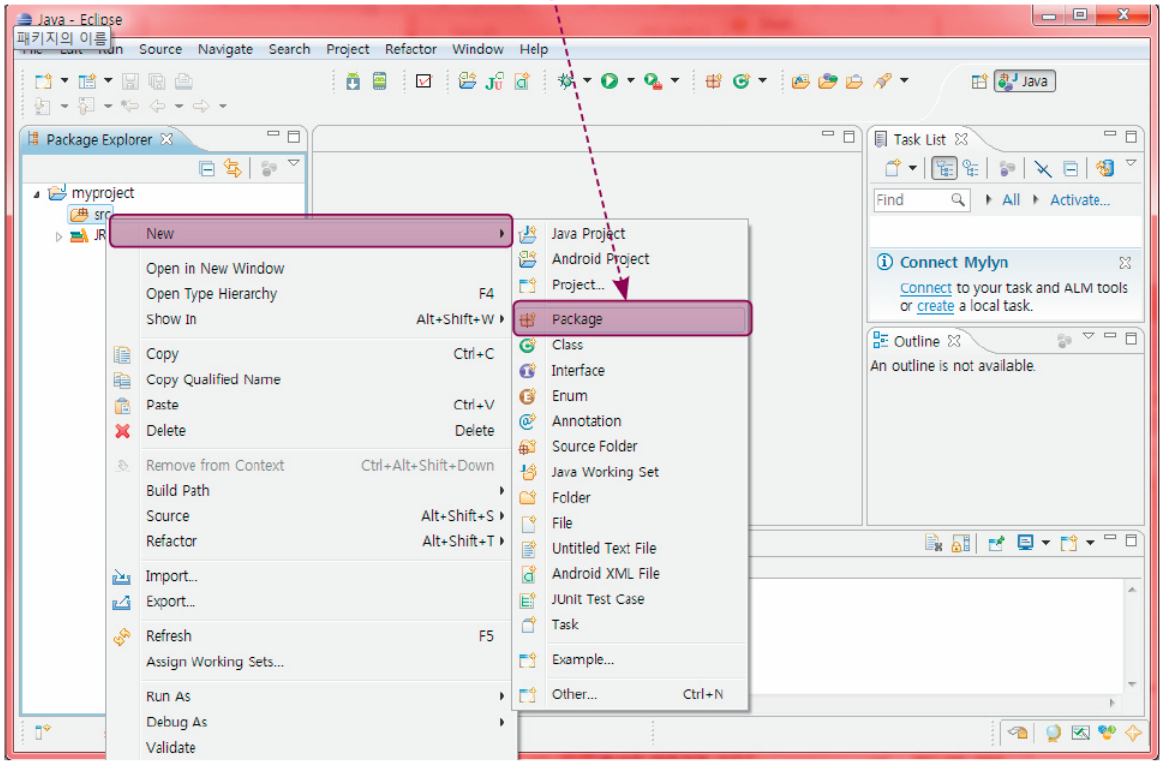
A: **Default package**에 속하게 된다.

Package의 이름

- Package의 이름은 일반적으로 소문자만을 사용한다.
- Package 이름으로 인터넷 도메인 이름의 역순을 사용한다. 예를 들면 com.company.mypackage이다.
- Java 언어 자체의 package는 java나 javax로 시작한다.

eclipse에서 Package 만들기

- eclipse를 사용하는 경우는 project를 먼저 생성한 후에, src 폴더에 마우스를 두고 오른쪽 버튼을 눌러서 New - Package 메뉴를 선택하면 package를 생성할 수 있다.



Package 사용

- 경로까지 포함하는 완전한 이름으로 참조한다.
- 원하는 package 멤버만을 import한다.
- Package 전체를 import한다.

완전한 이름으로 참조

- graphics package에 있는 Rectangle class의 완전한 이름은 graphics.Rectangle이다.
- 객체를 생성할 때도 원칙적으로 다음과 같이 해야 한다.

```
graphics.Rectangle  myRect = new graphics.Rectangle();
```

Package 멤버를 import

- 외부 package의 특정한 멤버를 import하려면 다음과 같은 문장을 사용한다. import 문장은 package 문장 다음에 위치해야 한다.

```
import graphics.Rectangle;
```

- Class가 포함되었으면 이제부터는 class 이름만 사용해서 참조가 가능하다

```
Rectangle myRect = new Rectangle();
```

전체 Package를 import

- 하나의 package 안에 포함된 모든 class를 포함하려면 다음과 같이 별표 문자(*)를 사용하면 된다

```
import graphics.*;    // package 전체
```

- Package 전체가 포함되면 package 이름을 생략하고 class 이름으로만 참조할 수 있다

```
Circle myCircle = new Circle();
Rectangle myRectangle = new Rectangle();
```

계층 구조의 Package 포함하기

- 여기서 한 가지 아주 주의해야 할 사항이 있다.
- Package는 계층적으로 구성된 것처럼 보인다.
- 예를 들어서 java.awt.*를 포함시키면 java.awt 아래에 있는 모든 package, 즉 java.awt.font와 같은 java.awt로 시작하는 모든 package가 포함될 것이라고 생각하기 쉽다.
- 그러나 java.awt.font package는 java.awt package 안에 포함되지 않는다.
- 만약 java.awt.font의 멤버와 java.awt의 멤버를 동시에 사용하려면 다음과 같이 따로 따로 포함해야 한다.

```
import java.awt.*;       // java.awt package 안의 class 포함
import java.awt.font.*;  // java.awt.font package 안의 class 포함
```

정적 import 문장

- Class 안에 정의된 정적 상수(static constant)나 정적 메소드(static method)를 사용하는 경우에 일반적으로는 class 이름을 앞에 적어 주어야 한다.
- 예를 들면 java.lang.Math class 안에는 PI가 상수로 정의되어 있고, sin(), cos(), tan()와 같은 수많은 정적 메소드들이 정의되어 있다.
- 일반적으로 이들 정적 상수와 정적 메소드를 사용하려면 다음과 같이 class 이름을 앞에 붙여야 한다.

```
double r = Math.cos(Math.PI * theta);
```

- 하지만 정적 import 문장을 사용하면 class 이름을 생략해도 된다.

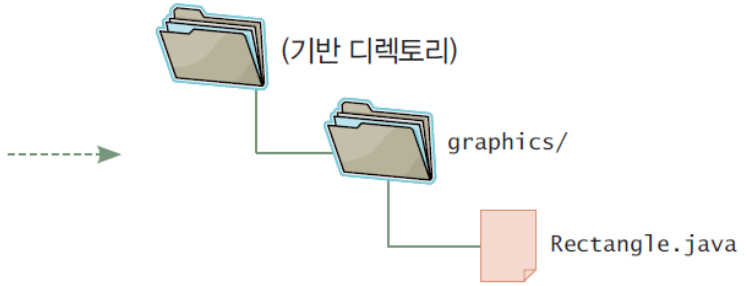
```
import static java.lang.Math.*;
double r = cos(PI * theta);
```

Source 파일과 Class 파일 관리

- Java에서는 package의 계층 구조를 반영한 디렉토리 구조에 source 들을 저장해야 한다.
  - 완전한 class 이름 – graphics.Rectangle
  - 파일의 경로 이름– graphics \ Rectangle.java

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

Rectangle.java

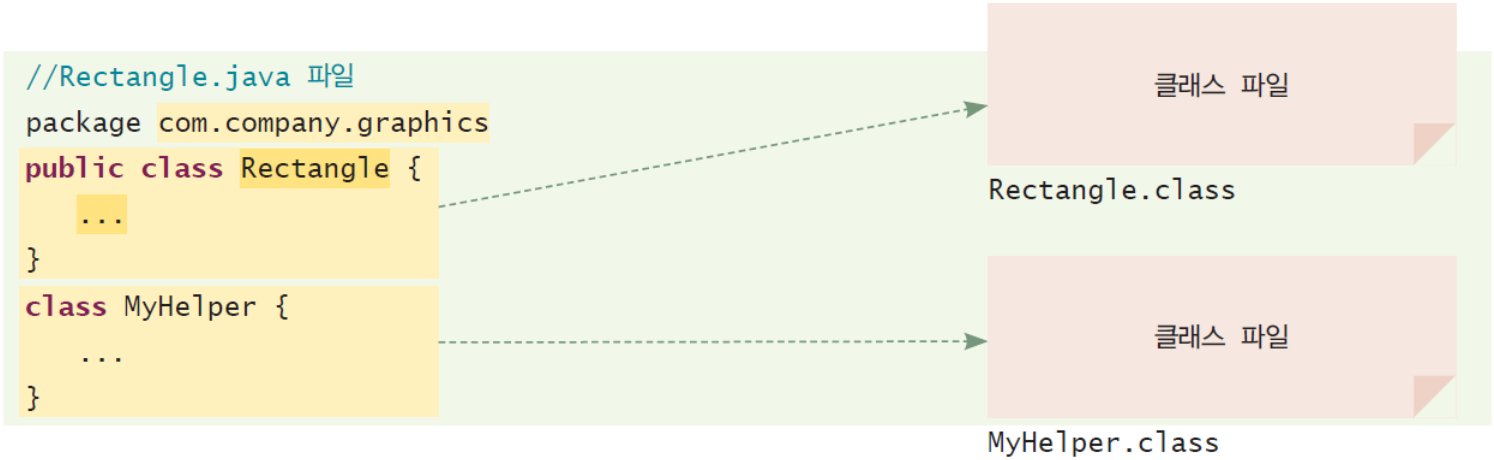


- company라는 회사의 도메인 이름이 com.company라면 graphics package는 다음과 같은 디렉토리에 저장된다.

  ....\com\company\graphics\Rectangle.java

Source 파일과 Class 파일 관리

- Source 파일을 컴파일하면 컴파일러는 각 class들을 서로 다른 출력 파일로 저장한다.
- 출력 파일의 이름은 class 이름과 같고, 확장자는 ".class"이다.
- 예를 들어서 source 파일이 다음과 같다고 하자.



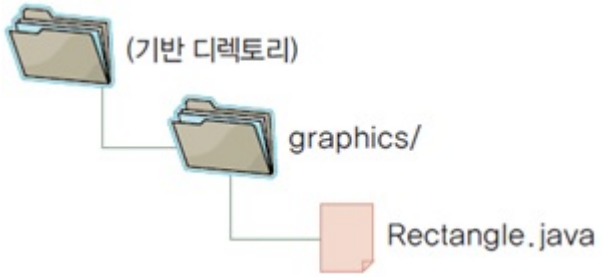
- 컴파일된 파일은 다음과 같이 저장된다.

```
<출력 디렉토리>\com\company\graphics\Rectangle.class
<출력 디렉토리>\com\company\graphics\MyHelper.class

C:\sources\com\company\graphics\Rectangle.java
C:\classes\com\company\graphics\Rectangle.class
```

명령어 버전을 사용할때

- graphics package를 작성하고 이 package 안에 Rectangle. java를 넣으려면 디렉토리 구조도 똑같이 만들어 주어야 한다.



- 컴파일할 때는 작업 디렉토리에서 다음과 같이 컴파일한다.

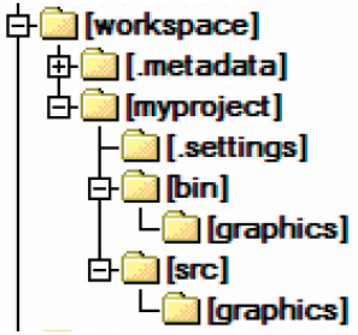
```
C> javac graphics/Rectangle.java
C> java graphics.Rectangle.java
C> javac -d C:\classes\graphics\Rectangle.java
```

eclipse를 사용할 때

- eclipse를 사용해서 package graphics를 생성하면 다음과 같은 디렉토리가 자동으로 생성된다.

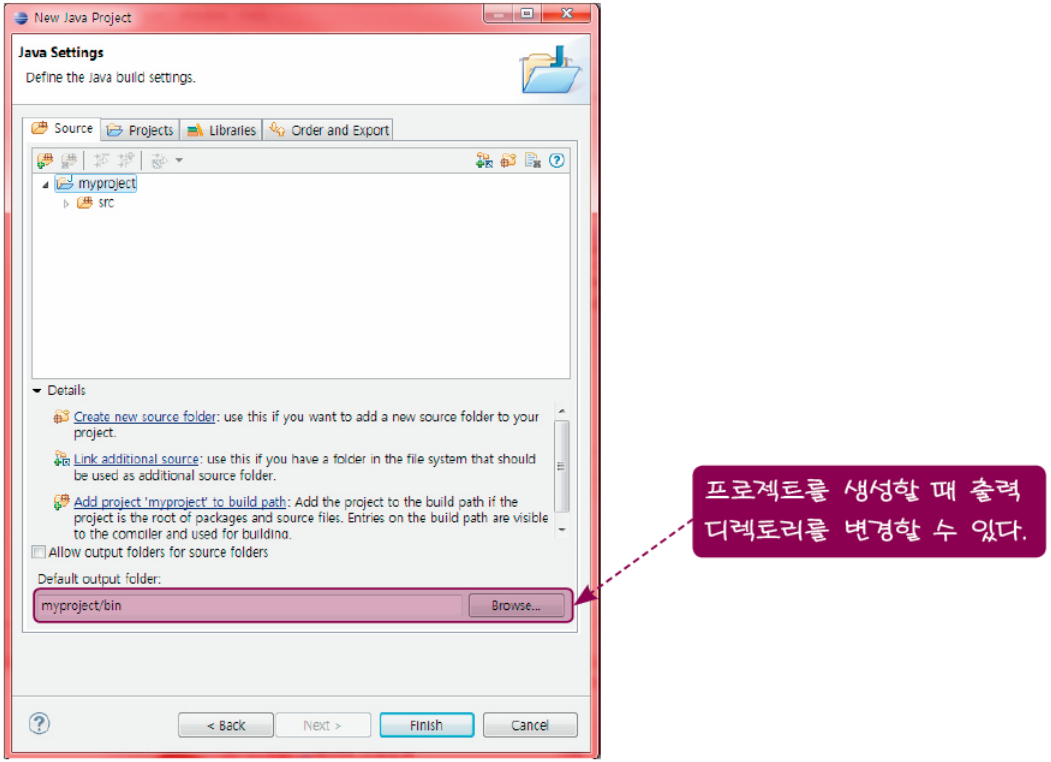
  즉 workspace 아래에 (작업디렉 토리)₩myproject₩src₩graphics 디렉토리에 source 파일들이 저장된다.

  또 (작업디렉토리)₩myproject₩bin₩graphics 디렉토리에 컴파일된 class 파일들이 저장된다.



eclipse를 사용할 때

- 만약 컴파일된 class 파일의 디렉토리를 변경하려면, project를 생성할 대화상자에서 folder를 지정하면 된다.



CLASSPATH시스템 변수 설정

- 실행에 필요한 class 파일들이 저장되는 디렉토리를 class 경로(class path)라고 한다.
- Class 경로가 C:\classes이고 package 이름이 com.company.graphics 라면 컴파일러와 JVM이 class 파일을 찾는 디렉토리는 다음과 같다.

```
C:\classes\com\company\graphics
```

CLASSPATH 시스템 변수 설정

- Java에서 class 경로를 지정하는 방법은 다음과 같은 2가지이다.
  - 환경 변수인 CLASSPATH를 설정한다. 현재 설정된 CLASSPATH 변수를 화면에 표시하려면, 보조 프로 그램의 명령 프롬프트에서 다음과 같은 명령어를 사용한다.

    ```
    C:\> set CLASSPATH=C:\classes;C:\lib;
    ```

  - JVM을 실행할 때 옵션 -classpath를 사용할 수 있다.

    ```
    C> java –classpath C:\classes;C:\lib;.\graphics.Rectangle
    ```

- 만약 class 경로가 위와 같이 지정되었다고 가정하면 JVM은 다음과 같은 순서로 class 파일을 탐색한다.
  - `C:\classes\graphics\Rectangle.class`
  - `C:\lib\graphics\Rectangle.class`
  - `.\graphics\Rectangle.class`

JAR 압축 파일

- Class 파일은 또한 JAR(Java archive) 파일 형태로 저장될 수 있다.
- Class 파일을 디렉토리의 계층 구조를 유지한 채로 압축 가능
- Class 경로에 지정해 주면 된다.

```
C:\> set CLASSPATH=C:\classes;C:\lib;C\test.jar;.
```

Java에서 지원하는 Package

- **Package**는 연관되어 있는 class와 interface들을 하나로 묶어 놓은 것이다.
- Java의 기본 package는 java로 시작하며 확장 package는 javax로 시작한다.

패키지	설명
java.applet	애플릿을 생성하는 데 필요한 클래스
java.awt	그래픽과 이미지를 위한 클래스
java.beans	자바빈즈 구조에 기초한 컴포넌트를 개발하는 데 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

Java에서 지원하는 Package

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 Package

- java.lang package는 Java 프로그래밍에 필수적인 class들을 가지고 있기 때문에 import 문을 사용 하지 않아도 자동으로 포함된다. 다음과 같은 class들이 포함된다.
  - **Object class:** 기초적인 method를 제공하는 모든 class의 조상 class
  - **Math class:** 각종 수학 함수들을 포함하는 class
  - **Wrapper class:** Integer와 같이 기초 자료형을 감싸서 제공하는 랩퍼 class들
  - **String class, StringBuffer class:** 문자열을 다루는 class
  - **Thread class:** 스레드 기능을 제공하는 class
  - **Class class:** 객체를 생성한 class에 대한 정보를 얻기 위한 class

Math Class

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)

Math Class

`static double log(double a)`	natural logarithm (base e)
`static double log10(double a)`	base 10 logarithm
`static double max(double a, double b)`	큰 수
`static double min(double a, double b)`	작은 수
`static double pow(double a, double b)`	a^b
`static double random()`	0.0과 1.0 사이의 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

Math Class

```
01  public class MathTest {
02      public static void main(String[] args){
03          double x = Math.PI;
04          System.out.println(Math.sin(x));
05          System.out.println(Math.random());
06      }
07  }
```

실행결과

```
1.2246467991473532E-16
0.2454637294993175
```

Class Class

- Class 객체는 실행 중인 class를 나타낸다.
- Class 객체는 JVM에 의하여 자동적으로 생성된다.
- Class 객체를 이용하면 객체의 class 이름을 출력할 수 있다.

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

System Class

- System class는 실행 시스템과 관련된 속성과 method를 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 실행 중인 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 Class

```
01  public class SystemTest {
02      public static void main(String[] args) {
03          System.out.println(System.currentTimeMillis());
04          System.out.println(System.nanoTime());
05          System.exit(0);
06      }
07  }
```

실행결과

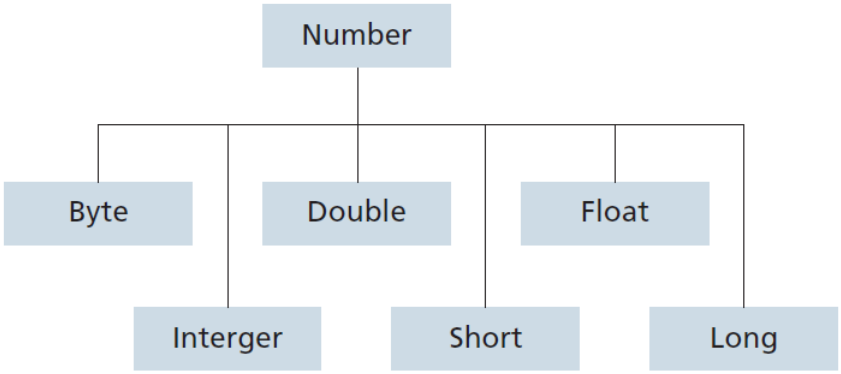
```
1245152017843
101355767783335
```

Wrapper Class

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

Wrapper Class

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



Integer Class가 제공하는 Method

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

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

```
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;
box = 10;                          // 정수를 자동으로 Integer 객체로 포장한다(boxing).
System.out.println(box + 1);       // box는 자동으로 int형으로 변환(unboxing)
```

StringBuffer Class

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

```
StringBuffer s = new StringBuffer("Happiness depends upon ourselves");
```

```
StringBuffer sb = new StringBuffer();     // 16바이트의 공간이 할당된다.
sb.append("Hello");                       // 6바이트가 사용된다.
```

StringBuffer의 Method

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

java.util Package

생성자	설명
Random()	새로운 난수 발생기 객체를 생성한다.
Random(longseed)	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 사이의 int형의 난수를 발생한다.
long nextLong()	long 형의 난수를 발생한다.
void setSeed(long seed)	시드를 설정한다.

java.util Package

```
01  import java.util.Random;
02
03  public class RandomTest {
04     public static void main(String[] args)
05     {
06        Random random = new Random();
07        for (int i = 0; i < 10; i++)
08           System.out.println(random.nextInt(100));
09     }
10  }
```

0에서 100 사이의 난수를 발생한다.

실행결과

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

Arrays Class

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 두 개의 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.

Arrays Class

```
01  import java.util.Arrays;
02
03  public class ArraysTest {
04      public static void main(String[] args) {
05          int[] array = { 9, 4, 5, 6, 2, 1 };
06          Arrays.sort(array);<----------------------------------------------배열 정렬
07          printArray(array);
08          System.out.println(Arrays.binarySearch(array,9));
09          Arrays.fill(array,8);<--------------------------------------------배열을 8로 채운다.
10          printArray(array);
11
12      }
13
14      private static void printArray(int[] array) {
15          System.out.print("[");
16          for(int i=0 ;i< array.length;i++)
17              System.out.print(array[i]+" ");
18          System.out.println("]");
19
20      }
21  }
```

실행결과

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

Date Class

```
01  import java.util.*;
02
03  public class DateTest {
04      public static void main(String[] args) {
05          Date d = new Date();
06          System.out.println(d);
07          System.out.println(1900+d.getYear());
08          System.out.println(d.getMonth()+1);
09          System.out.println(d.getDate());
10
11          d.setHours(12);
12          d.setMinutes(34);
13          d.setSeconds(0);
14          System.out.println(d);
15      }
16  }
```

현재 날짜로 Date 객체가 생성된다.

객체로부터 연, 월, 일을 읽어서 출력한다.

실행결과

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

Calendar Class

- Calendar class는 추상 class로서 날짜와 시간에 대한 정보를 가지고 있고 특정 시각을 연도, 월, 일 등으로 변환하는 method도 가지고 있다. 시각은 1970년 1월 1일 00:00:00.000 GMT부터 흘러온 시 간으로 나타낸다.
- 현재 시각을 나타내는 객체를 얻으려면 다음과 같이 하면 된다

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

AM	AM_PM	APRIL	AUGUST	DATE
DAY_OF_MONTH	DAY_OF_WEEK	DAY_OF_WEEK_IN_ MONTH	DECEMBER	ERA
FEBRUARY	FRIDAY	HOUR	HOUR_OF_DAY	JANUARY
JULY	JUNE	MARCH	MAY	MILLISECOND
MINUTE	MONDAY	MONTH	NOVEMBER	OCTOBER
PM	SATURDAY	SECOND	SEPTEMBER	SUNDAY
THURSDAY	TUESDAY	WEDNESDAY	WEEK_OF_MONTH	WEEK_OF_yEAR
YEAR	ZONE_OFFSET			

Calendar Class

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)	두 개의 Calendar 객체를 비교한다.
boolean equals(Object obj)	두 개의 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. en Date.

예제

```
01  import java.util.*;
02
03  public class CalendarTest {
04     public static void main(String[] args) {
05        Calendar d = Calendar.getInstance();
06        System.out.println(d);
07        System.out.println(d.get(Calendar.YEAR)+"년");
08        System.out.println(d.get(Calendar.MONTH)+1 +"월");
09        System.out.println(d.get(Calendar.DATE)+"일");
10
11        d.set(Calendar.HOUR, 12);
12        d.set(Calendar.MINUTE, 34);
13        d.set(Calendar.SECOND, 0);
14        System.out.println(d);
15     }
16  }
```

현재 날짜로 Calendar 객체가 생성된다.

실행결과

```
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 Class

- 문자열을 분석해서 토큰으로 분리시켜 주는 기능을 제공

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. 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

```
01  import java.util.*;
02  public class StringTest {
03     public static void main(String[] args) {
04        StringTokenizer st = new StringTokenizer("Will Java change my life?");
05        while (st.hasMoreTokens()) {
06           System.out.println(st.nextToken());
07        }
08
09     }
10  }
```

문자열을 토큰으로 분리하여 출력한다.

실행결과

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

Q & A