

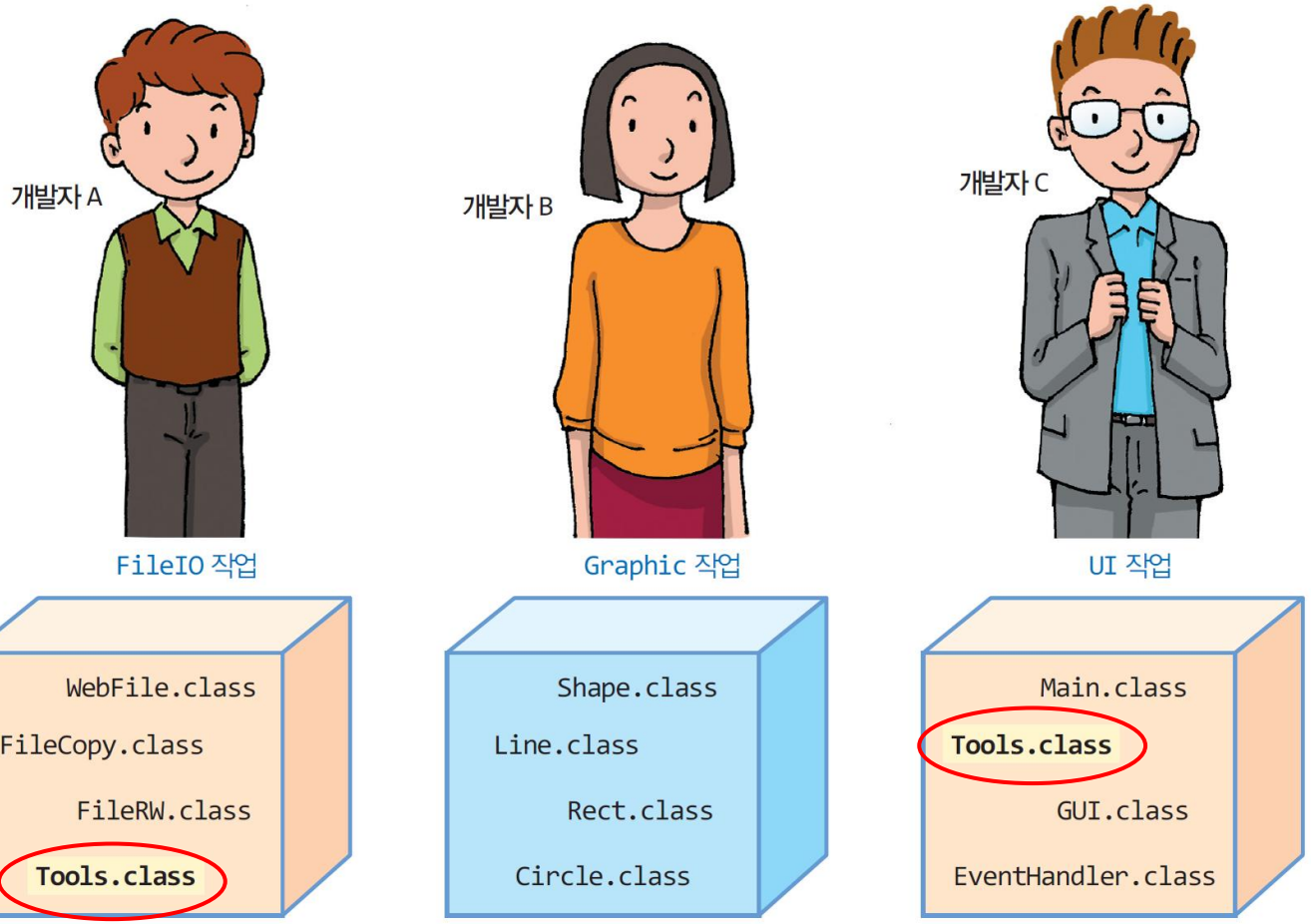
6

모듈과 패키지 개념, 자바 기본 패키지

패키지 개념과 필요성

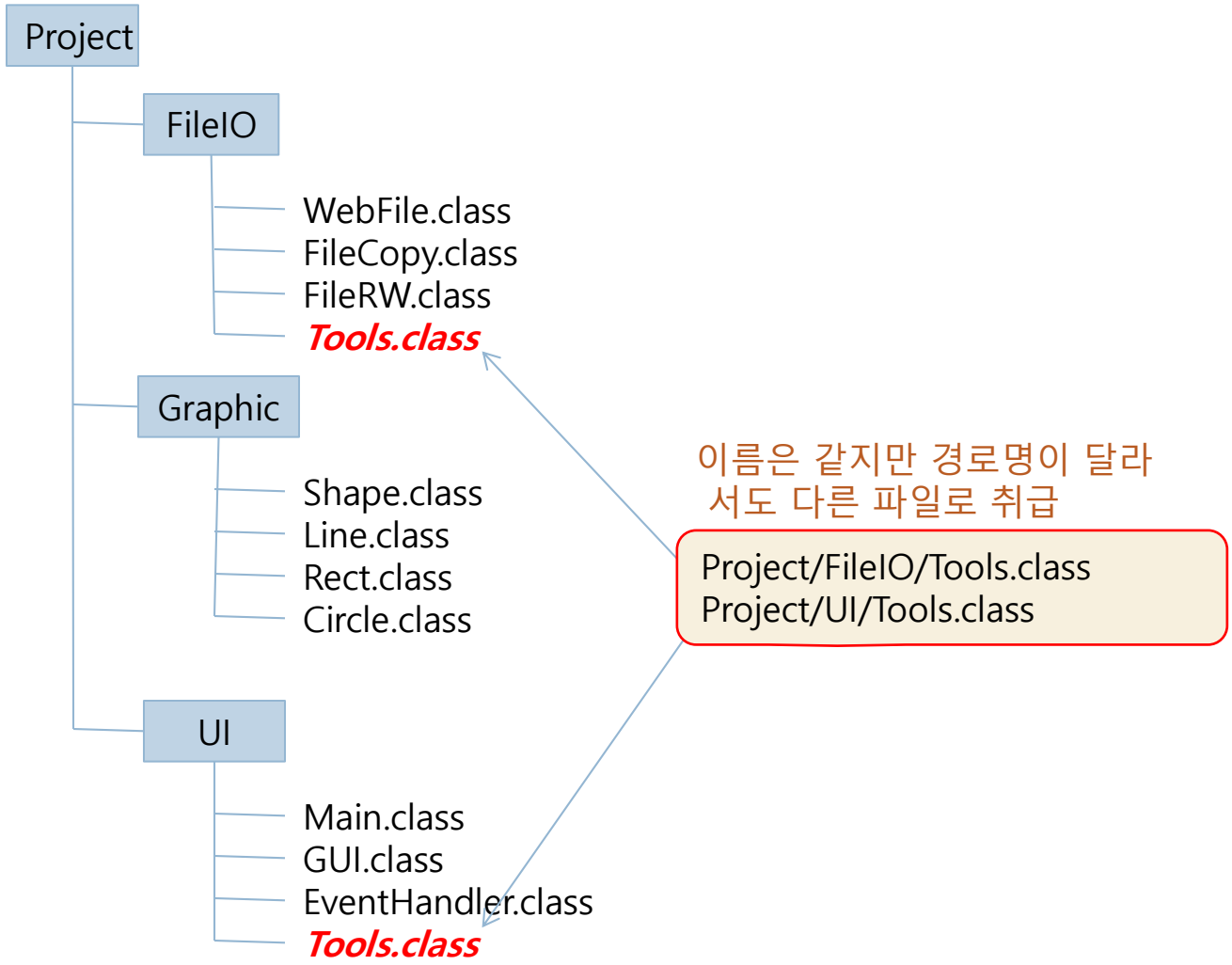


3명이 분담하여 자바 응용프로그램을 개발하는 경우,
동일한 이름의 클래스가 존재할 가능성 있음 -> 합칠 때 오류발생



디렉터리로 각 개발자의 코드 관리(패키지)





자바의 모듈(module)과 패키지(package)

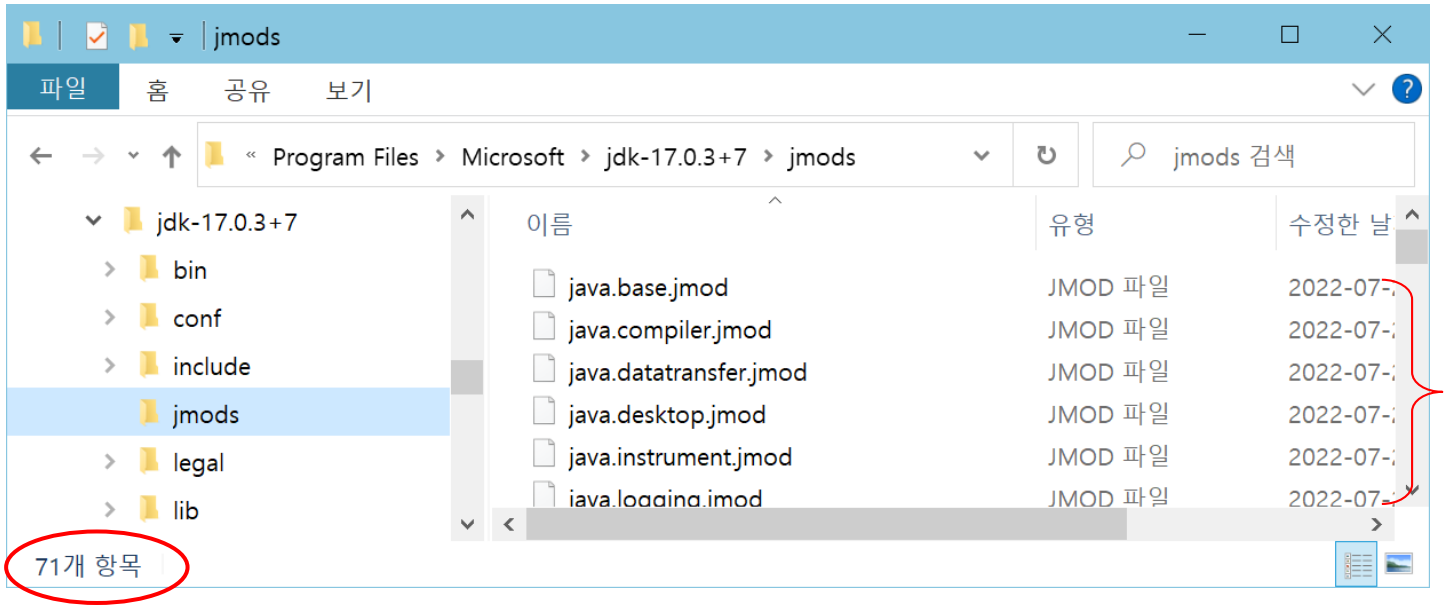


- 패키지
  - 서로 관련된 클래스와 인터페이스의 컴파일 된 클래스 파일들을 하나의 디렉터리에 묶어 놓은 것
- 모듈
  - 여러 패키지와 이미지 등의 자원을 모아 놓은 컨테이너
  - JDK 9부터 자바 API의 모든 클래스들(자바 실행 환경)을 패키지 기반에서 모듈들로 완전히 재구성
  - 응용프로그램 역시 여러 개의 모듈로 분할하여 작성 가능
    - 클래스들은 패키지로 만들고, 다시 패키지를 모듈로 만듦
  - 목적
    - 자바 API를 여러 모듈(99개)로 분할하여 응용프로그램의 실행에 적합한 모듈들로만 실행 환경을 구축할 수 있도록 함
    - 메모리 등의 자원이 열악한 작은 소형 기기에 꼭 필요한 모듈로 구성된 작은 크기의 실행 이미지를 만들기 위함
  - 모듈의 현실
    - Java 9부터 전면적으로 도입, 복잡한 개념, 큰 자바 응용프로그램에는 개발, 유지보수 등에 적합
- 패키지명과 클래스의 경로명
  - 점(.)으로 연결
    - Project.FileIO.Tools.class
    - Project.UI.Tools.class

자바 API의 모듈 파일들



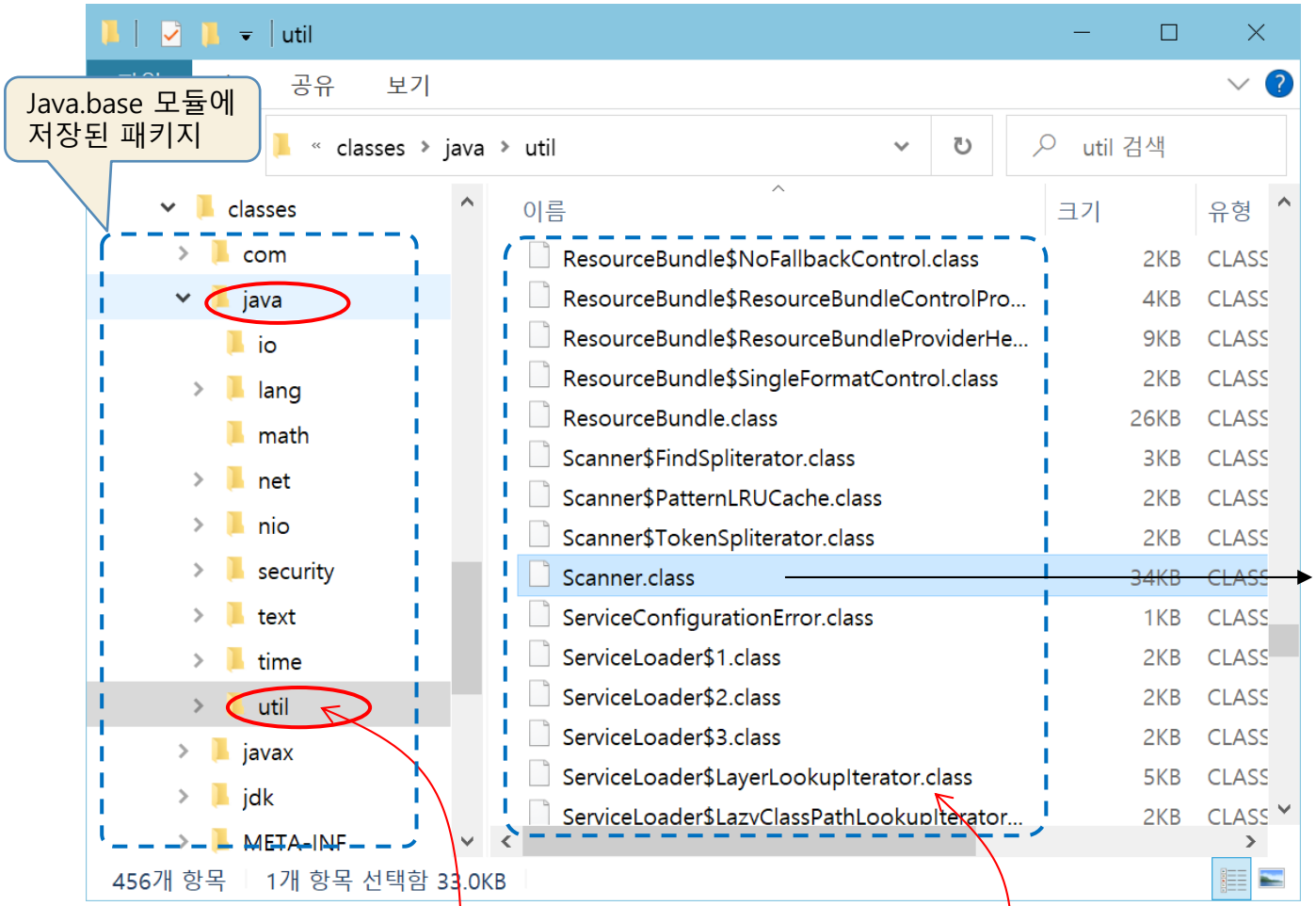
- 자바 JDK에 제공되는 모듈 파일들
  - 자바가 설치된 jmods 디렉터리에 모듈 파일 존재
    - 자바 버전에 따라 100개에 가까운 모듈 파일
    - 모듈 파일(.jmod 파일)은 ZIP 포맷으로 압축된 파일
  - 모듈 파일에는 자바 API의 패키지와 클래스들이 들어 있음
  - Jmod 명령을 이용하여 모듈 파일에 들어 있는 패키지를 풀어 낼 수 있음



Microsoft의 OpenJDK가 설치된 경우, 71개의 모듈 파일들

JDK의 java.base 모듈에 들어 있는 패키지들과 클래스들





클래스의 이름(경로명)

java.util.Scanner

패키지명

패키지 명 : java.util

java.util 패키지에 속한 클래스

패키지 사용하기, import문



- 다른 패키지에 작성된 클래스 사용
  - import를 이용하지 않는 경우
    - 소스 내에서 패키지 이름과 클래스 이름의 전체 경로명을 써주어야 함

```
public class ImportExample {
    public static void main(String[] args) {
        java.util.Scanner scanner =
                new java.util.Scanner(System.in);
        System.out.println(scanner.next());
    }
}
```

  - import를 이용하는 경우
    - 소스의 시작 부분에 사용하려는 패키지 명시
      - 소스에는 클래스 명만 명시하면 됨
    - 특정 클래스의 경로명만 포함
      - *import java.util.Scanner;*
    - 패키지 내의 모든 클래스 포함
      - *import java.util.*;*
      - *는 현재 패키지 내의 클래스만을 의미하며 하위 패키지의 클래스까지 포함하지 않는다.

```
import java.util.Scanner;
public class ImportExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
    }
}
```

```
import java.util.*;
public class ImportExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
    }
}
```

패키지 만들기



- 패키지 선언
  - package 패키지명;
    - 컴파일한 클래스 파일을 패키지명의 디렉터리에 저장하라는 지시
    - 소스 파일의 첫 줄에 선언
- 사례

```
package UI; // Tools 클래스를 컴파일하여 UI 패키지에 저장할 것을 지시

public class Tools {          // 이제 이 클래스의 경로명은 UI.Tools가 된다.
.....
}
```

  - Tools 클래스의 경로명은 UI.Tools
  - 다른 클래스에서 Tools 클래스를 사용하기 위해서는 import UI.Tools

```
package Graphic; // Line 클래스를 Graphic 패키지에 저장

import UI.Tools; // Tools 클래스의 경로명 알림

public class Line {
    public void draw() {
        Tools t = new Tools();
    }
}
```

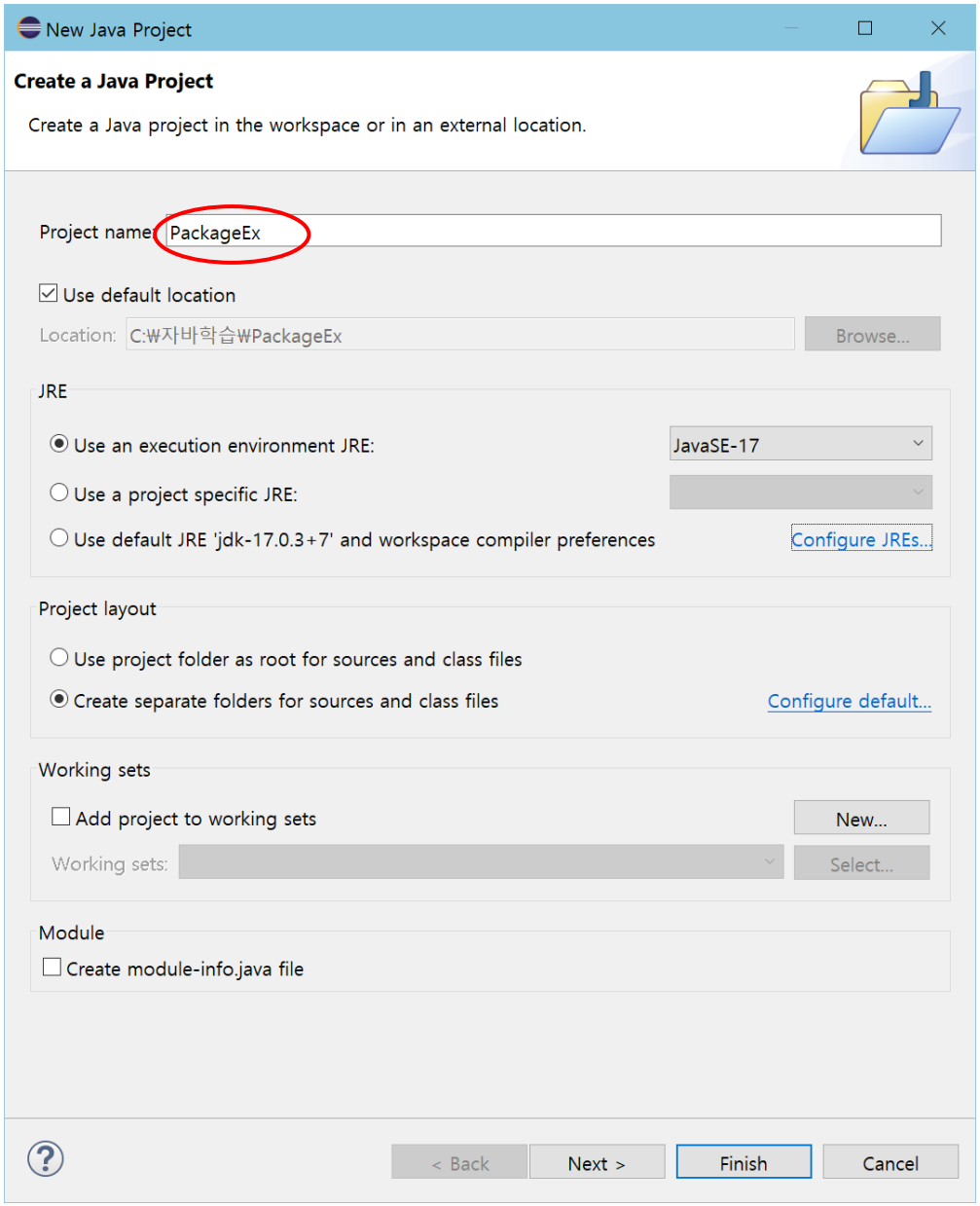
이클립스로 쉽게 패키지 만들기



- 예제로 사용할 샘플 소스(5장의 예제 5-7)

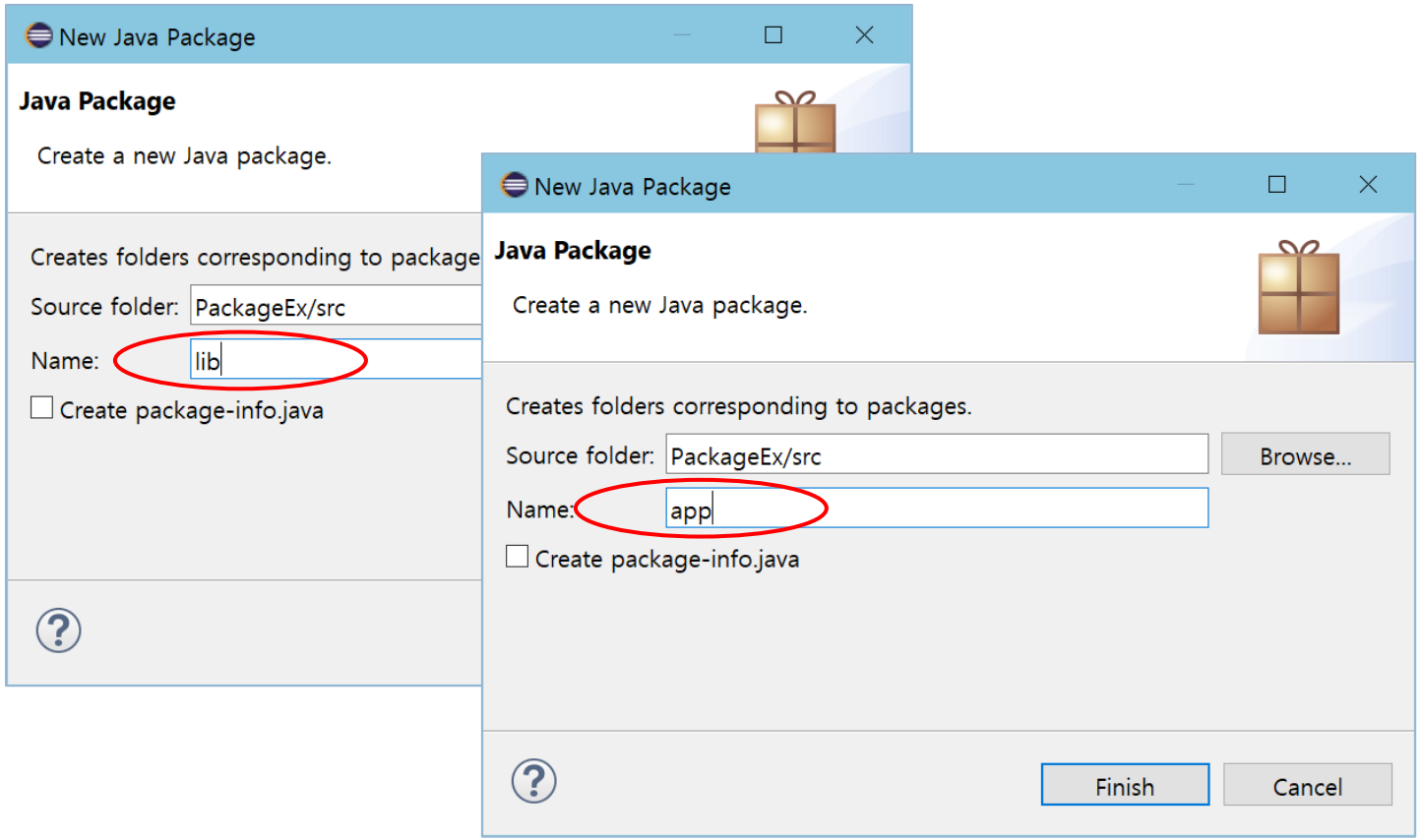
```
abstract class Calculator {
    public abstract int add(int a, int b);
    public abstract int subtract(int a, int b);
    public abstract double average(int[] a);
}

public class GoodCalc extends Calculator {
    public int add(int a, int b) {
        return a+b;
    }
    public int subtract(int a, int b) {
        return a - b;
    }
    public double average(int[] a) {
        double sum = 0;
        for (int i = 0; i < a.length; i++)
            sum += a[i];
        return sum/a.length;
    }
    public static void main(String [] args) {
        Calculator c = new GoodCalc();
        System.out.println(c.add(2,3));
        System.out.println(c.subtract(2,3));
        System.out.println(c.average(new int [] {2,3,4 }));
    }
}
```

프로젝트 작성(프로젝트 이름 : PackageEx)



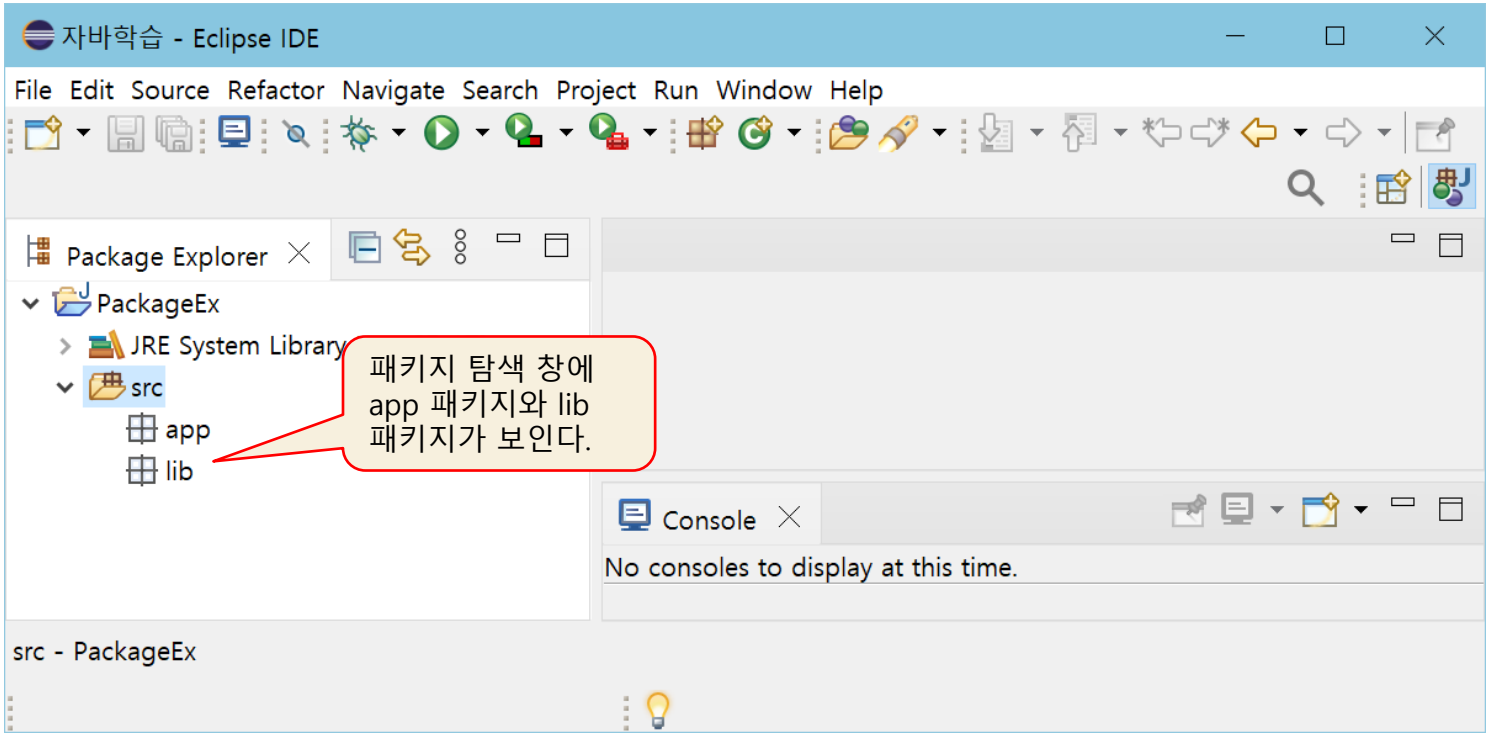
패키지 lib, app 작성



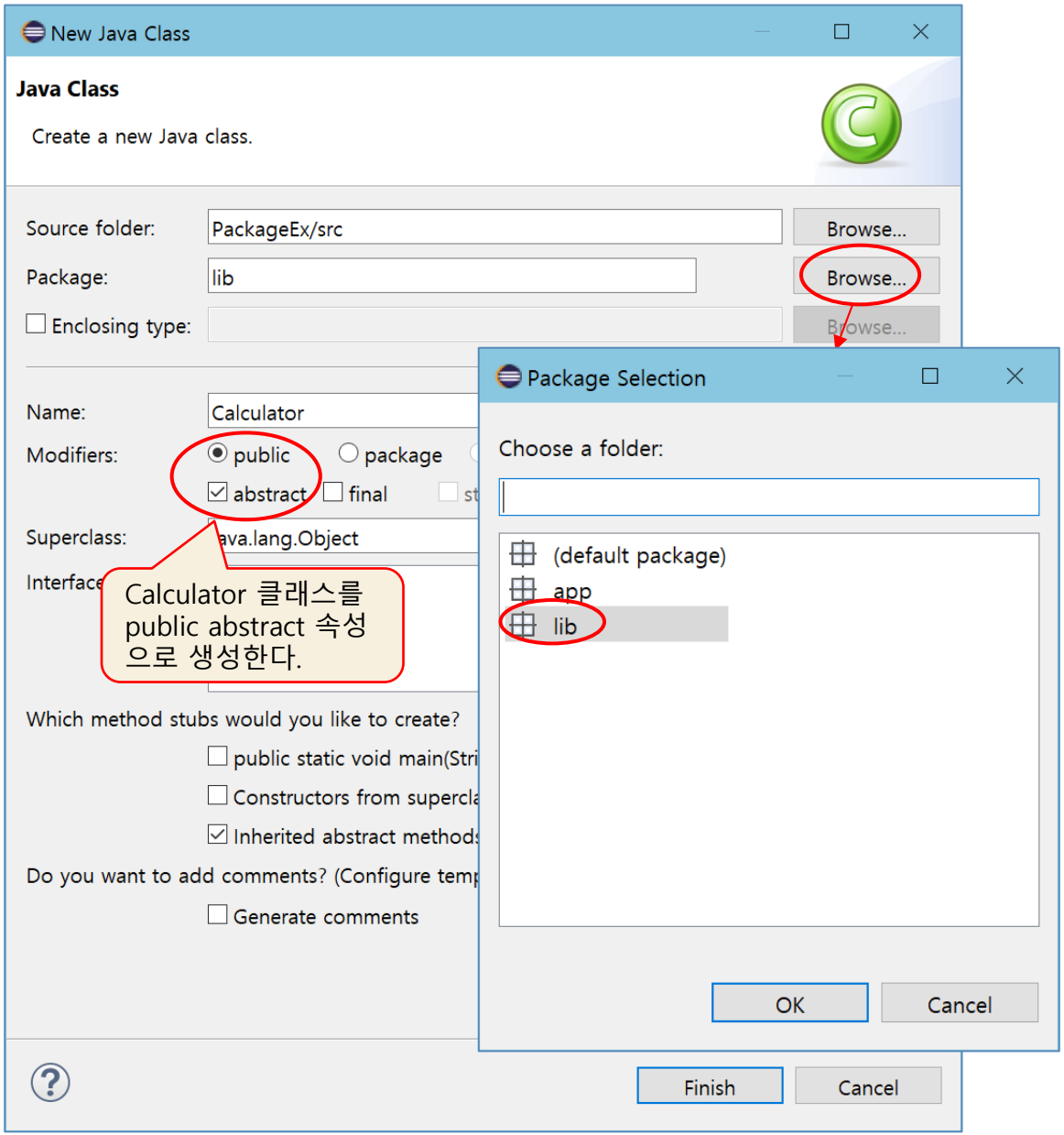


패키지 작성이 완료된 결과

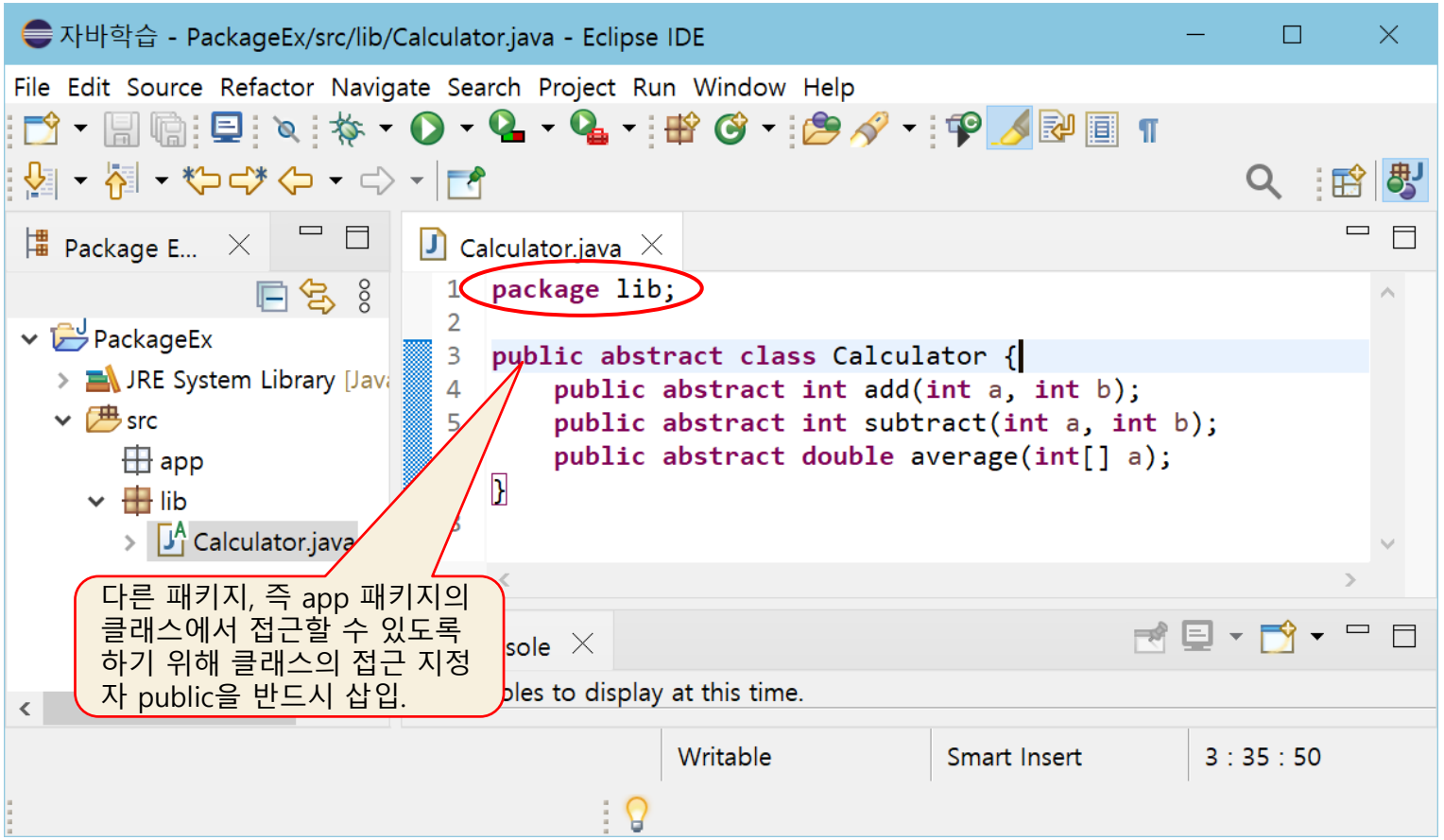




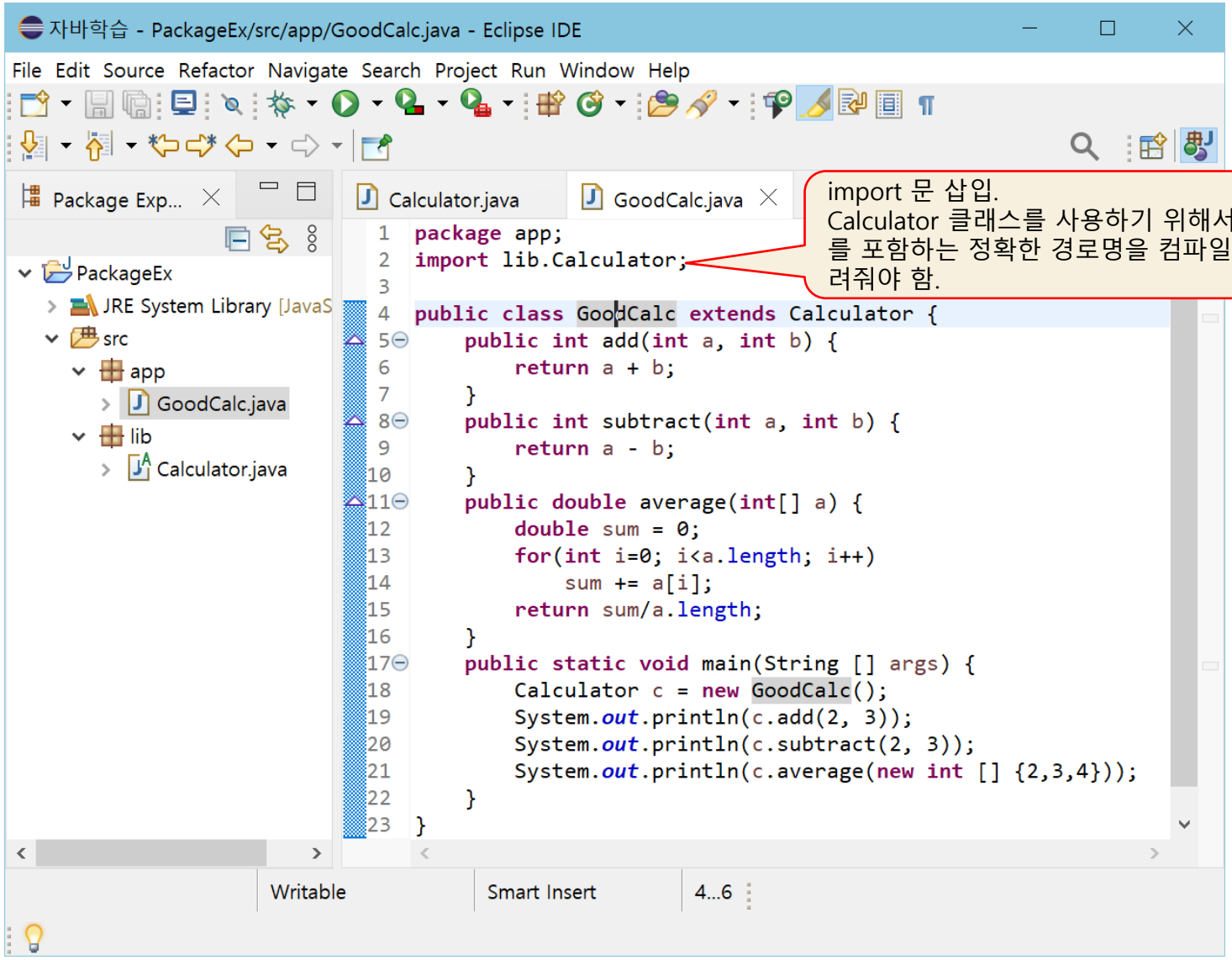
클래스 Calculator 만들기



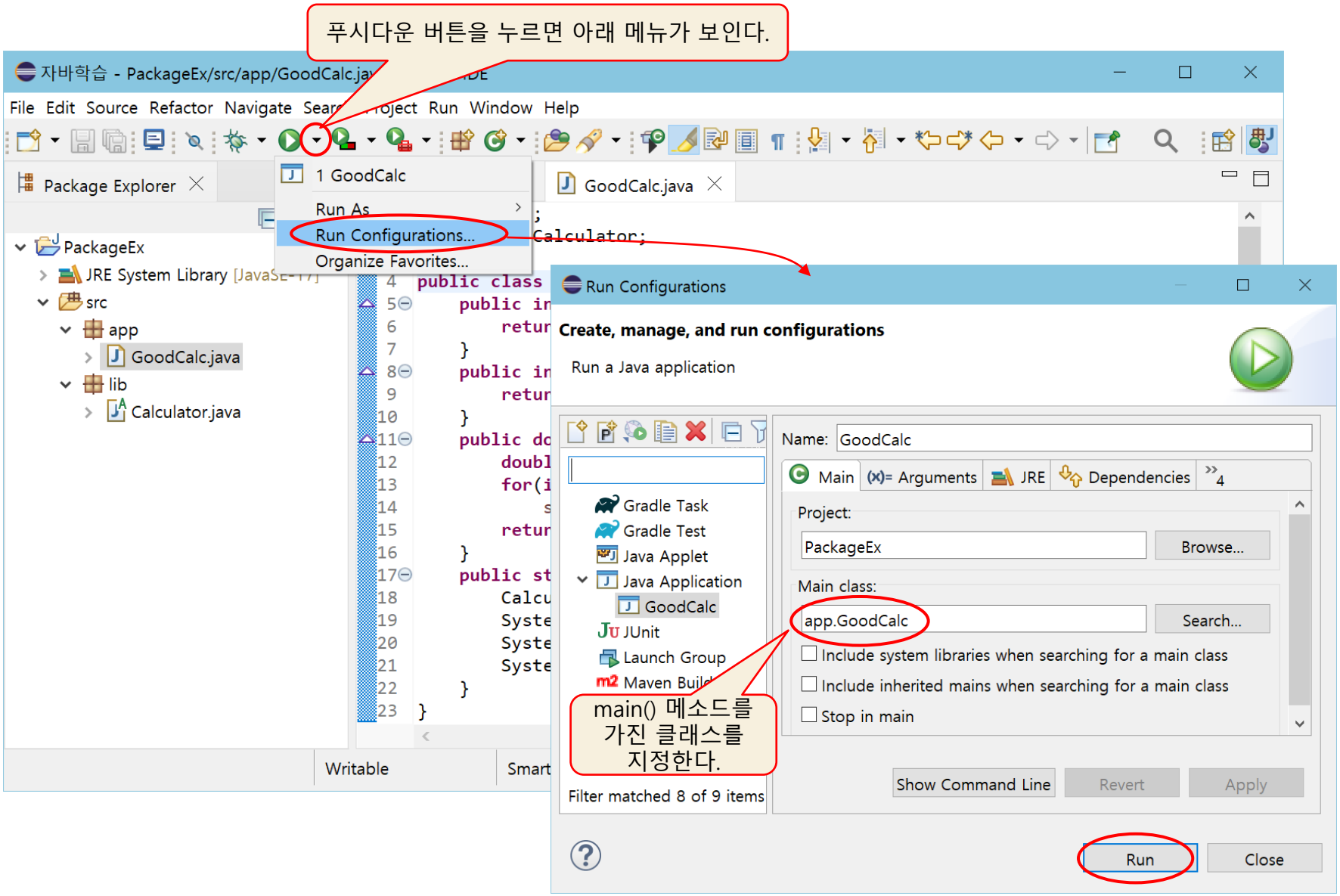
Calculator 소스 수정



GoodCalc.java 작성 후 소스 수정

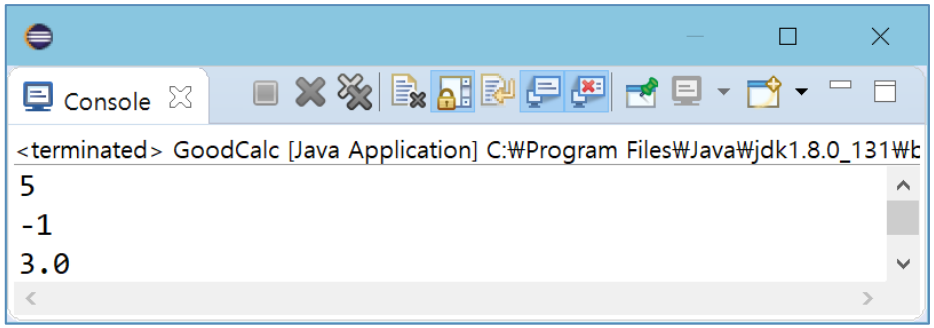


실행을 위한 Run Configurations 작성



프로젝트 PackageEx 실행





디폴트 패키지와 패키지의 특징

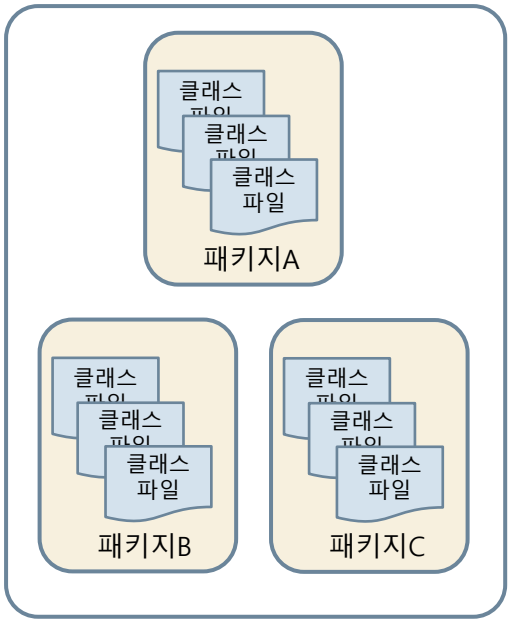


- 디폴트 패키지
  - package 선언문이 없이 만들어진 클래스의 패키지
  - 디폴트 패키지는 현재 디렉터리
- 패키지의 특징
  - 패키지 계층구조
    - 관련된 클래스 파일을 하나의 패키지로 계층화하여 관리 용이
  - 패키지별 접근 제한
    - 패키지 별로 접근 권한 가능
  - 동일한 이름의 클래스와 인터페이스의 사용 가능
    - 서로 다른 패키지에 이름이 같은 클래스와 인터페이스 존재 가능
  - 높은 소프트웨어 재사용성

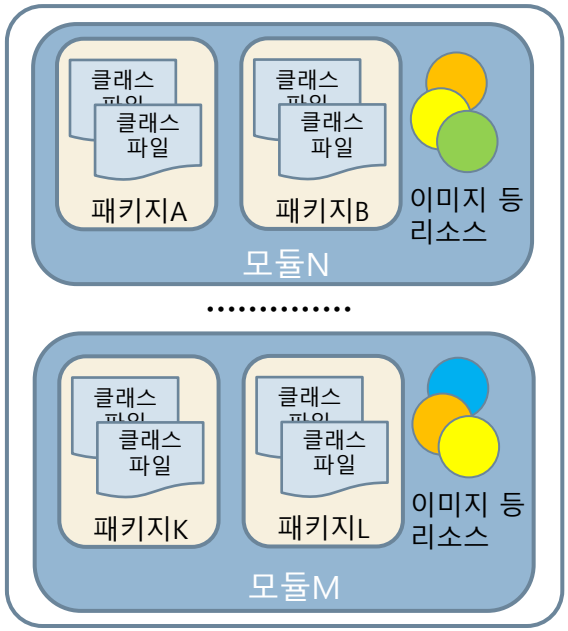
모듈 개념



- 모듈
  - Java 9에서 도입된 개념
  - 패키지와 이미지 등의 리소스를 담은 컨테이너
  - 모듈 파일(.jmod)로 저장



Java 8에서 클래스와 패키지



Java 9 이후 클래스와 패키지, 그리고 모듈

자바 플랫폼의 모듈화



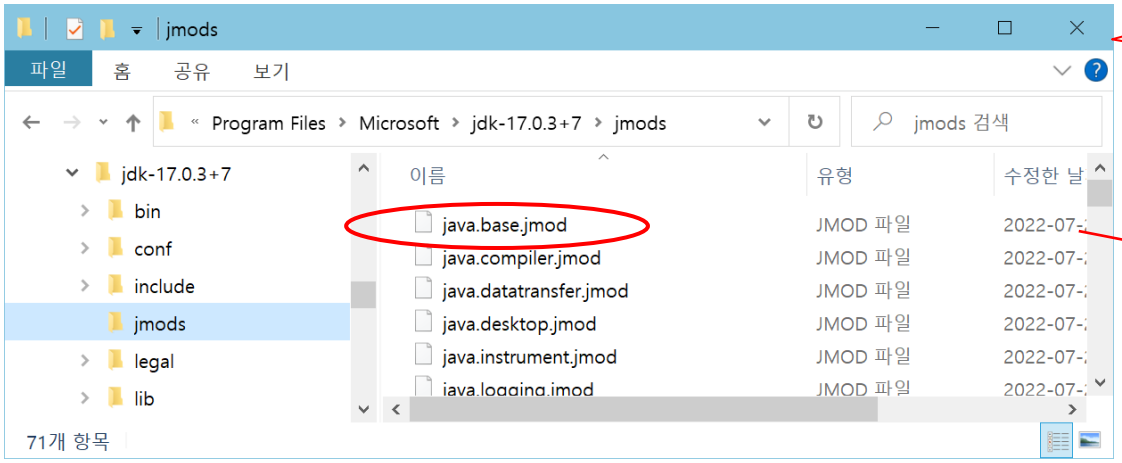
- 자바 플랫폼
  - 자바의 개발 환경(JDK)과 자바의 실행 환경(JRE)을 지칭
    - Java SE(자바 API) 포함
  - 자바 API의 모든 클래스가 여러 개의 모듈로 재구성됨
  - 모듈 파일은 JDK의 jmods 디렉터리에 저장하여 배포

- 모듈 파일로부터 모듈을 푸는 명령

  (Microsoft사의 OpenJDK를 설치한 경우)

  jmod extract "C:₩Program Files₩Microsoft₩jdk-17.0.3+7₩jmods₩java.base.jmod"

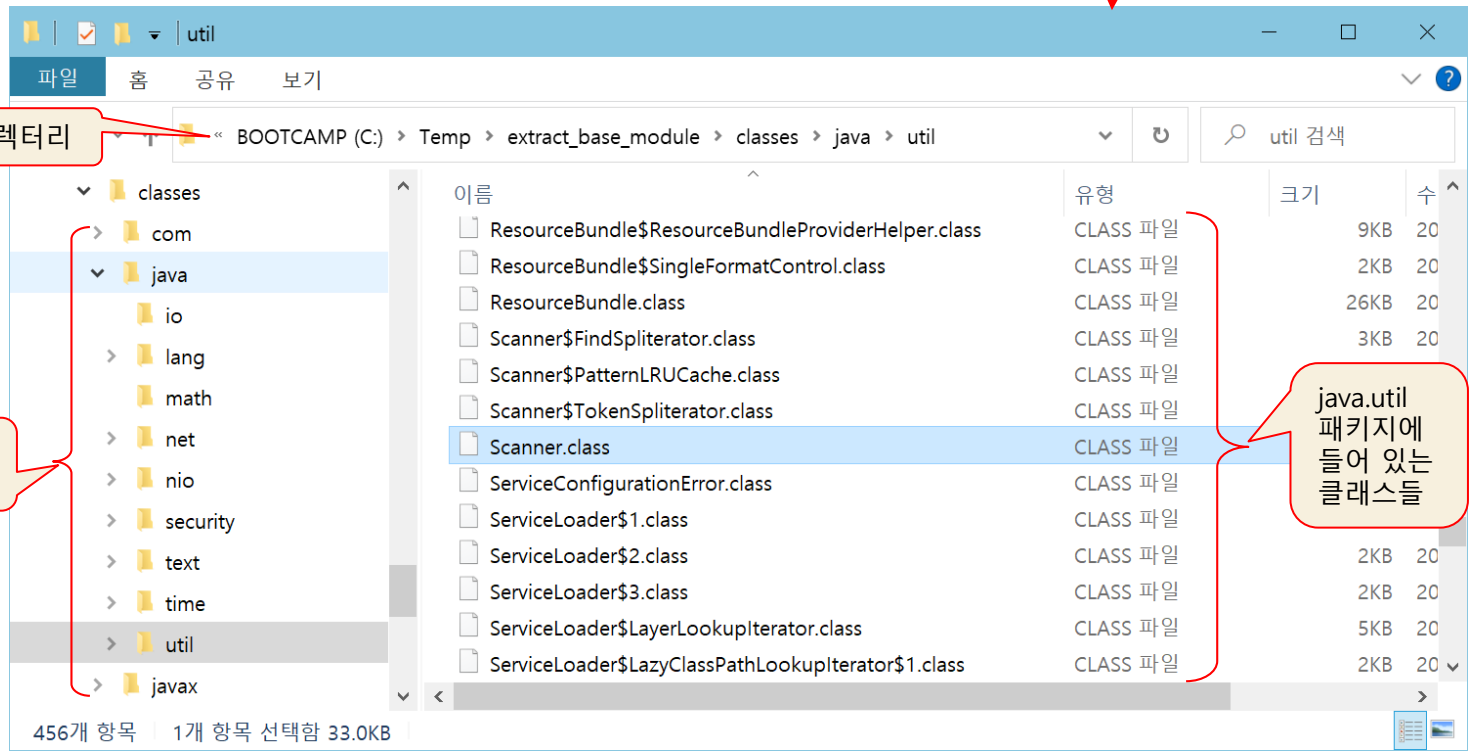
  - 풀어놓고자 하는 디렉터리로 이동 후,
  - jmod 명령을 입력하면
  - 현재 디렉터리에 java.base 모듈이 풀림



jmods 디렉터리에 들어 있는
자바 API의 모듈 파일들

java.base.jmod
모듈 파일 풀기

jmod extract "C:₩Program Files₩Mocrosoft₩jdk-17.0.3+7₩jmods₩java.base.jmod"



base 모듈을 풀어 놓은 디렉터리

java.base 모듈에
들어 있는 패키지들

java.util
패키지에
들어 있는
클래스들

모듈 기반의 자바 실행 환경

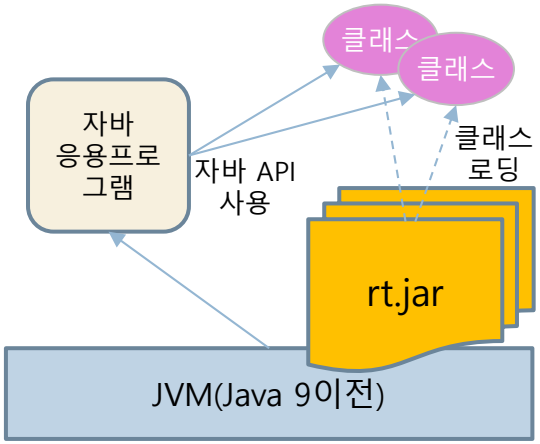


- 자바 실행 환경
  - JRE : 디폴트 자바 실행 환경
    - 자바 모듈(컴파일된 자바 API 클래스들), 자바 가상 기계 등으로 구성

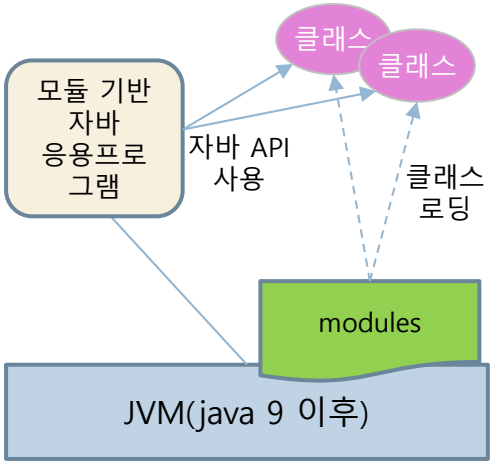
비교 관점	Java 8까지	Java 9 이후
자바 API 클래스의 컨테이너	• rt.jar의 단일체에 자바 API의 패키지들을 모두 담음	• rt.jar를 버렸음 • 자바 API를 수십개의 모듈 파일에 나누어 저장
디폴트 실행 환경	• 실행되는 컴퓨터에 rt.jar 설치 (rt.jar의 크기가 큼)	• modules(JRE의 lib 밑에) 비공개 파일(약109MB)에 모듈 저장 • 응용프로그램 실행 시 modules 파일에서 필요한 모듈 및 클래스 로딩
맞춤형 실행 환경 (custom JRE)	• 없음 • 소형 기기에 rt.jar를 설치할 수 없는 한계	• 맞춤형 실행 환경 구축 가능 • jlink 명령을 이용하여 응용프로그램의 실행에 필요한 모듈들로 실행 환경 구축 가능(홈페이지 설명과 예제 있음) • 메모리가 열악한 소형 기기에도 응용프로그램 실행 가능 *예) DB 액세스가 필요 없고 오직 기본 자바 API만 필요한 소형 IoT 장치에는 java.base 모듈만 있으면 됨*

자바 실행 환경 비교

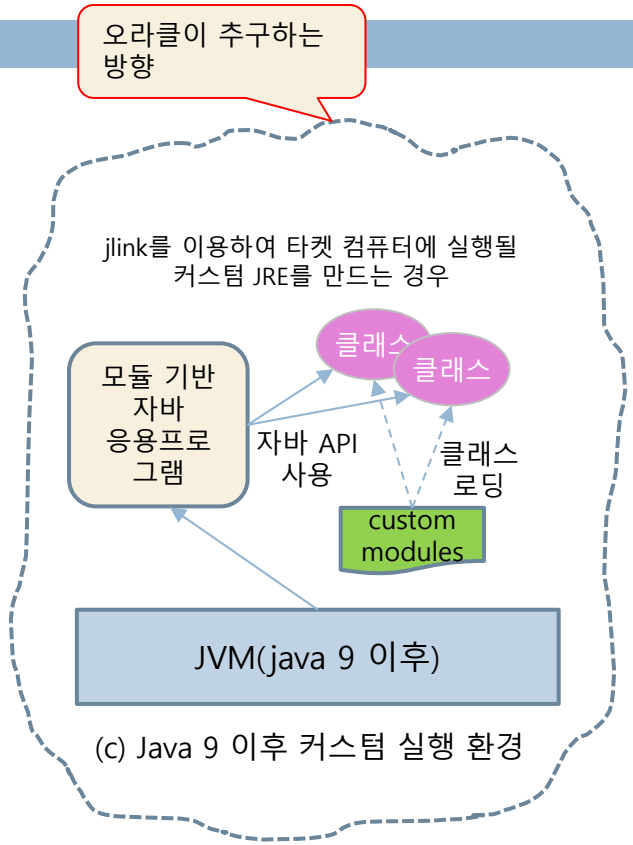




(a) Java 9 이전 실행 환경



(b) Java 9 이후 실행 환경



(c) Java 9 이후 커스텀 실행 환경

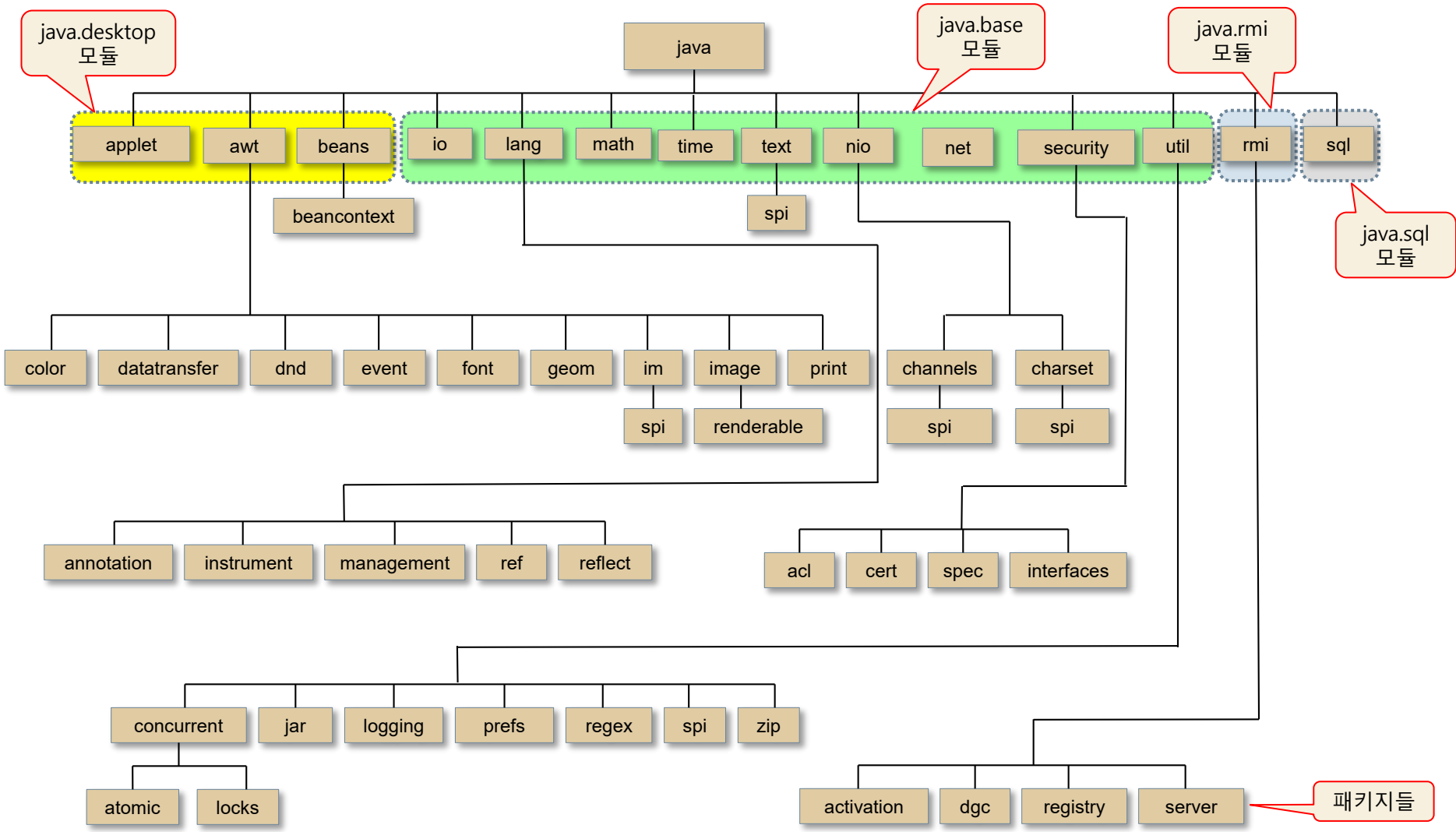
* 홈페이지의 자료에, jlink로 맞춤형 실행 환경을 만드는 예제 있음.
실제로 만들어보면 커스텀 JRE가 생기고, lib 디렉터리 밑에 modules 파일의 크기가 현저히 줄어든 것을 볼 수 있고, 실행 중에 차지하는 메모리의 량도 줄어든 것을 확인할 수 있음

자바 모듈화의 목적



- 가장 큰 목적
  - 자바 컴포넌트들을 필요에 따라 조립하여 사용하기 위함
  - 컴퓨터 시스템의 불필요한 부담 감소
    - 세밀한 모듈화를 통해 필요 없는 모듈이 로드되지 않게 함
    - 소형 IoT 장치에도 자바 응용프로그램이 실행되고 성능을 유지하게 함

자바 모듈과 패키지 구조


java.desktop
모듈
java
java.base
모듈
java.rmi
모듈
applet
awt
beans
io
lang
math
time
text
nio
net
security
util
rmi
sql
beancontext
spi
java.sql
모듈
color
datatransfer
dnd
event
font
geom
im
image
print
channels
charset
spi
renderable
spi
spi
annotation
instrument
management
ref
reflect
acl
cert
spec
interfaces
concurrent
jar
logging
prefs
regex
spi
zip
atomic
locks
activation
dgc
registry
server
패키지들

주요 패키지

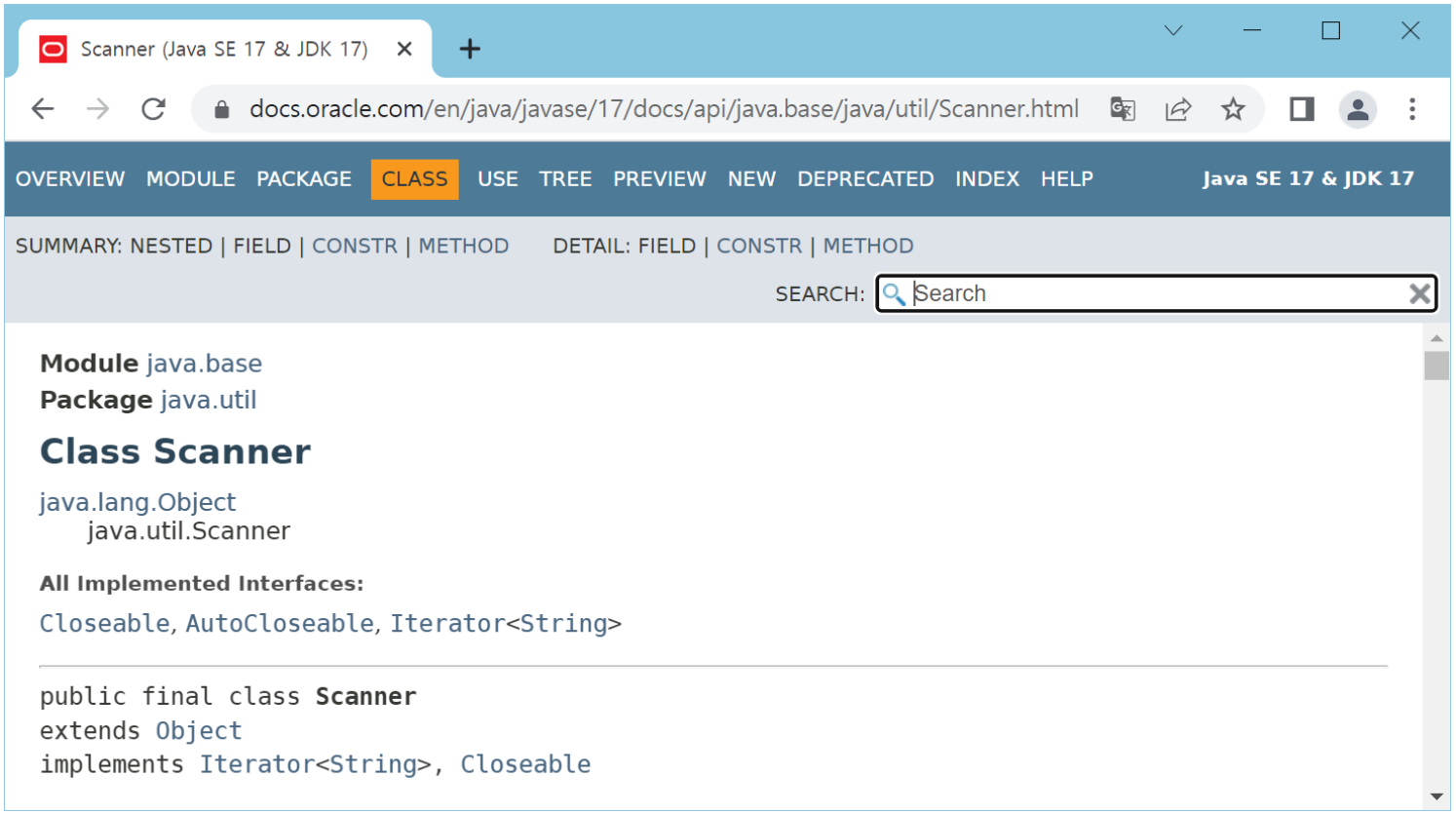


- java.lang
  - 자바 language 패키지
    - 스트링, 수학 함수, 입출력 등 자바 프로그래밍에 필요한 기본적인 클래스와 인터페이스
  - 자동으로 import. import 문 필요 없음
- java.util
  - 자바 유틸리티 패키지
    - 날짜, 시간, 벡터, 해시맵 등과 같은 다양한 유틸리티 클래스와 인터페이스 제공
- java.io
  - 키보드, 모니터, 프린터, 디스크 등에 입출력을 할 수 있는 클래스와 인터페이스 제공
- java.awt
  - 자바 GUI 프로그래밍을 위한 클래스와 인터페이스 제공
- javax.swing
  - 자바 GUI 프로그래밍을 위한 스윙 패키지

자바 API 참조



- 자바 API의 상세 정보
  - Oracle Technology Network에서 온라인제공
    - https://docs.oracle.com/en/java/javase/17/docs/api/index.html



Object 클래스



- 특징
  - java.lang 패키지에 포함
  - 모든 클래스의 수퍼 클래스
    - 모든 클래스에 강제 상속
    - 모든 객체가 공통으로 가지는 객체의 속성을 나타내는 메소드 보유
- 주요 메소드

메소드	설명
boolean equals(Object obj)	obj가 가리키는 객체와 현재 객체를 비교하여 같으면 true 리턴
Class getClass()	현 객체의 런타임 클래스를 리턴
int hashCode()	현 객체에 대한 해시 코드 값 리턴
String toString()	현 객체에 대한 문자열 표현을 리턴
void notify()	현 객체에 대해 대기하고 있는 하나의 스레드를 깨운다.
void notifyAll()	현 객체에 대해 대기하고 있는 모든 스레드를 깨운다.
void wait()	다른 스레드가 깨울 때까지 현재 스레드를 대기하게 한다.

예제 6-1 : Object 클래스로 객체 속성 알아내기



객체 레퍼런스만으로 객체의 클래스명, 해시코드 값, 객체의 문자열을 출력해보자

```
class Point {
   private int x, y;
   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
}
public class ObjectPropertyEx {
   public static void print(Object obj) {
      System.out.println(obj.getClass().getName()); // 클래스 이름
      System.out.println(obj.hashCode()); // 해시 코드 값
      System.out.println(obj.toString()); // 객체를 문자열로 만들어 출력
      System.out.println(obj); // 객체 출력
   }
   public static void main(String [] args) {
      Point p = new Point(2,3);
      print(p);
   }
}
```

```
Point
366712642
Point@15db9742
Point@15db9742
```

객체를 문자열로 변환



- String toString()
  - 객체를 문자열로 반환
  - Object 클래스에 구현된 toString()이 반환하는 문자열

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

- '객체 + 문자열' -> '객체.toString() + 문자열'로자동변환

```
Point p = new Point(2,3);
System.out.println(p);
String s = p + "점";
```

변환

```
System.out.println(p.toString());
String s = p.toString() + "점";
```

```
Point@15db9742점
```

- 개발자는 자신만의 toString() 작성 필요
  - Object의 toString() 오버라이딩

예제 6-2 : Point 클래스에 toString() 작성



Point 클래스에 Point 객체를 문자열로 리턴하는 toString() 메소드를 작성하라.

```
class Point {
   private int x, y;
   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
   public String toString() {
      return "Point(" + x + "," + y + ")";
   }
}

public class ToStringEx {
   public static void main(String [] args) {
      Point p = new Point(2,3);
      System.out.println(p.toString());
      System.out.println(p); // p는 p.toString()으로 자동 변환
      System.out.println(p + "입니다."); // p.toString() + "입니다"로 자동 변환
   }
}
```

Point 객체를 문자열로 리턴하는 toString() 작성

```
Point(2,3)
Point(2,3)
Point(2,3)입니다.
```

객체 비교와 equals()

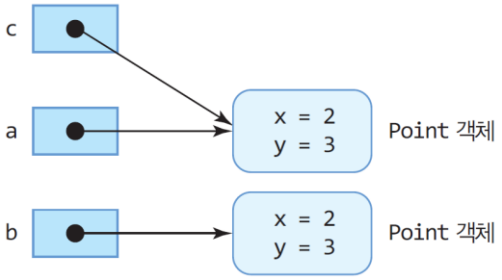


- == 연산자
  - 두 개의 레퍼런스 비교
- boolean equals(Object obj)
  - 객체 내용이 같은지 비교

```
class Point {
    private int x, y;
    public Point(int x, int y) {
        this.x = x; this.y = y;
    }
}
```

```
Point a = new Point(2,3);
Point b = new Point(2,3);
Point c = a;
if(a == b) // false
    System.out.println("a==b");
if(a == c) // true
    System.out.println("a==c");
```

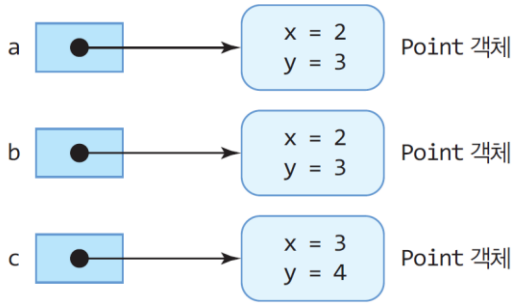
```
a==c
```




```
class Point {
    private int x, y;
    public Point(int x, int y) {
        this.x = x; this.y = y;
    }
    public boolean equals(Object obj) {
        Point p = (Point)obj;
        if(x == p.x && y == p.y)
            return true;
        else return false;
    }
}
```

```
Point a = new Point(2,3);
Point b = new Point(2,3);
Point c =  new Point(3,4);

if(a == b) // false
    System.out.println("a==b");
if(a.equals(b)) // true
    System.out.println("a is equal to b");
if(a.equals(c)) // false
    System.out.println("a is equal to c");
```

```
a is equal to b
```



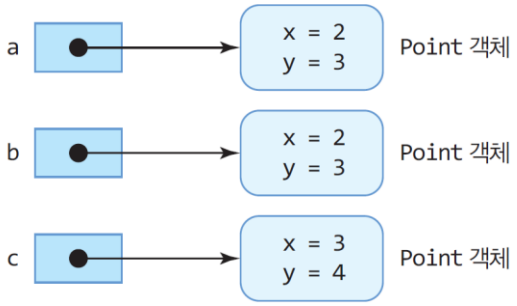
예제 6-3 : Point 클래스에 equals() 작성



Point 클래스에 두 점의 좌표가 같으면 true를 리턴하는 equals()를 작성하라.

```
class Point {
   private int x, y;
   public Point(int x, int y) {
      this.x = x; this.y = y;
   }
   public boolean equals(Object obj) {
      Point p = (Point)obj;
      if(x == p.x && y == p.y) return true;
      else return false;
   }
}
```

```
public class EqualsEx {
   public static void main(String[] args) {
      Point a = new Point(2,3);
      Point b = new Point(2,3);
      Point c = new Point(3,4);

      if(a == b) // false
         System.out.println("a==b");
      if(a.equals(b)) // true
         System.out.println("a is equal to b");
      if(a.equals(c)) // false
         System.out.println("a is equal to c");
   }
}
```




```
a is equal to b
```

예제 6-4 : Rect 클래스와 equals() 만들기 연습



int 타입의 width(너비)와 height(높이) 필드를 가지는 Rect 클래스를 작성하고, 면적이 같으면 두 Rect 객체가 같은 것으로 판별하는 equals()를 작성하라. 생성자에서 너비와 높이를 받아 width, height 필드를 초기화하라.

```
class Rect {
   private int width;
   private int height;
   public Rect(int width, int height) {
      this.width = width;
      this.height = height;
   }
   public boolean equals(Object obj) {
      Rect p = (Rect)obj;
      if (width*height == p.width*p.height)
         return true;
      else
         return false;
   }
}
```

```
public class EqualsEx {
   public static void main(String[] args) {
      Rect a = new Rect(2,3);
      Rect b = new Rect(3,2);
      Rect c = new Rect(3,4);
      if(a.equals(b))
         System.out.println("a is equal to b");
      if(a.equals(c))
         System.out.println("a is equal to c");
      if(b.equals(c))
         System.out.println("b is equal to c");
   }
}
```

```
a is equal to b
```

Wrapper 클래스



- 자바의 기본 타입을 클래스화한 8개 클래스

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

  - 이름이 Wrapper인 클래스는 존재하지 않음
  - 용도
    - 기본 타입의 값을 객체로 다룰 수 있게 함

Wrapper 객체 생성



- 기본 타입의 값으로 Wrapper 객체 생성

```
Integer i = Integer.valueOf(10);
Character c = Character.valueOf('c');
Double f = Double.valueOf(3.14);
Boolean b = Boolean.valueOf(true);
```

```
Integer i = new Integer(10);
Character c = new Character('c');
Double f = new Double(3.14);
Boolean b = new Boolean(true);
```

Java 9부터 생성자를 이용한 Wrapper 객체 생성 폐기

- 문자열로 Wrapper 객체 생성

```
Integer I = Integer.valueOf("10");
Double d = Double.valueOf("3.14");
Boolean b = Boolean.valueOf("false");
```

```
Integer I = new Integer("10");
Double d = new Double("3.14");
Boolean b = new Boolean("false");
```

- Float 객체는 double 타입의 값으로 생성 가능

```
Float f = Float.valueOf((double) 3.14);
```

주요 메소드



- Wrapper 객체들은 거의 유사, 많은 메소드가 static 타입
- Integer 클래스의 주요 메소드

메소드	설명
static int bitCount(int i)	정수 i의 이진수 표현에서 1의 개수 리턴
float floatValue()	float 타입으로 값 리턴
int intValue()	int 타입으로 값 리턴
long longValue()	long 타입으로 값 리턴
short shortValue()	short 타입으로 값 리턴
static int parseInt(String s)	문자열 s를 10진 정수로 변환한 값 리턴
static int parseInt(String s, int radix)	문자열 s를 지정된 진법의 정수로 변환한 값 리턴
static String toBinaryString(int i)	정수 i를 이진수 표현으로 변환한 문자열 리턴
static String toHexString(int i)	정수 i를 16진수 표현으로 변환한 문자열 리턴
static String toOctalString(int i)	정수 i를 8진수 표현으로 변환한 문자열 리턴
static String toString(int i)	정수 i를 문자열로 변환하여 리턴
static Integer valueOf(int i)	정수 i를 담은 Integer 객체 리턴
static Integer valueOf(String s)	문자열 s를 정수로 변환하여 담고 있는 Integer 객체 리턴

Wrapper 활용



- Wrapper 객체로부터 기본 타입 값 알아내기

```
Integer i = Integer.valueOf(10);
int ii = i.intValue(); // ii = 10

Character c = Character.valueOf('c' );
char cc = c.charValue(); // cc = 'c'
```

```
Double f = Double.valueOf(3.14);
double dd = d.doubleValue(); // dd = 3.14

Boolean b = Boolean.valueOf(true);
boolean bb = b.booleanValue(); // bb = true
```

- 문자열을 기본 데이터 타입으로 변환

```
int i = Integer.parseInt("123");                    // i = 123
boolean b = Boolean.parseBoolean("true");           // b = true
double f = Double.parseDouble("3.14" );             // d = 3.14
```

- 기본 타입을 문자열로 변환

```
String s1 = Integer.toString(123);          // 정수 123을 문자열 "123" 으로 변환
String s2 = Integer.toHexString(123);       // 정수 123을 16진수의 문자열 "7b"로 변환
String s3 = Double.toString(3.14);          // 실수 3.14를 문자열 "3.14"로 변환
String s4 = Charater.toString('a');         // 문자 'a'를 문자열 "a"로 변환
String s5 = Boolean.toString(true);         // 불린 값 true를 문자열 "true"로 변환
```

예제 6-5 : Wrapper 클래스 활용



다음은 Wrapper 클래스를 활용하는 예이다. 다음 프로그램의 결과는 무엇인가?

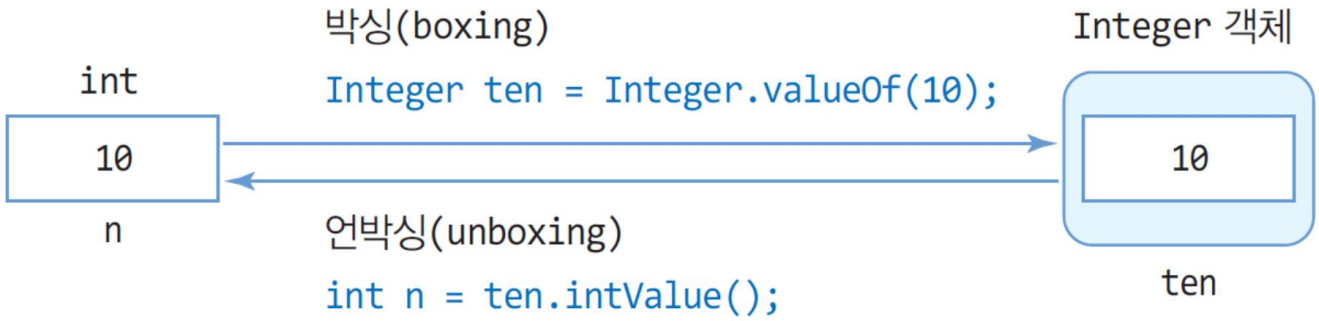
```
public class WrapperEx {
   public static void main(String[] args) {
      System.out.println(Character.toLowerCase('A')); // 'A'를 소문자로 변환
      char c1='4', c2='F';
      if(Character.isDigit(c1)) // 문자 c1이 숫자이면 true
         System.out.println(c1 + "는 숫자");
      if(Character.isAlphabetic(c2)) // 문자 c2가 영문자이면 true
         System.out.println(c2 + "는 영문자");

      System.out.println(Integer.parseInt("-123")); // "-123"을 10진수로 변환
      System.out.println(Integer.toHexString(28)); // 정수 28을 2진수 문자열로 변환
      System.out.println(Integer.toBinaryString(28)); // 28을 16진수 문자열로 변환
      System.out.println(Integer.bitCount(28)); // 28에 대한 2진수의 1의 개수

      Double d = Double.valueOf(3.14);
      System.out.println(d.toString()); // Double을 문자열 "3.14"로 변환
      System.out.println(Double.parseDouble("3.14")); // 문자열을 실수 3.14로 변환

      boolean b = (4>3); // b는 true
      System.out.println(Boolean.toString(b)); // true를 문자열 "true"로 변환
      System.out.println(Boolean.parseBoolean("false")); // 문자열을 false로 변환
   }
}
```

```
4는 숫자
F는 영문자
-123
1c
11100
3
3.14
3.14
true
false
```

박싱과 언박싱



- 박싱(boxing)
  - 기본 타입의 값을 Wrapper 객체로 변환
- 언박싱(unboxing)
  - Wrapper 객체에 들어 있는 기본 타입의 값을 빼내는 것



- 자동 박싱과 자동 언박싱 – JDK1.5부터

```
Integer ten = 10;    // 자동 박싱. Integer ten = Integer.valueOf(10);로 자동 처리
int n = ten;         // 자동 언박싱. int n = ten.intValue();로 자동 처리
```

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
```

String의 특징과 객체 생성



- String - java.lang.String
  - String 클래스는 하나의 문자열 표현

```
// 스트링 리터럴로 스트링 객체 생성
String str1 = "abcd";

// String 클래스의 생성자를 이용하여 스트링 생성
char data[] = {'a', 'b', 'c', 'd'};
String str2 = new String(data);
String str3 = new String("abcd"); // str2와 str3은 모두 "abcd" 스트링
```

  - String 생성자

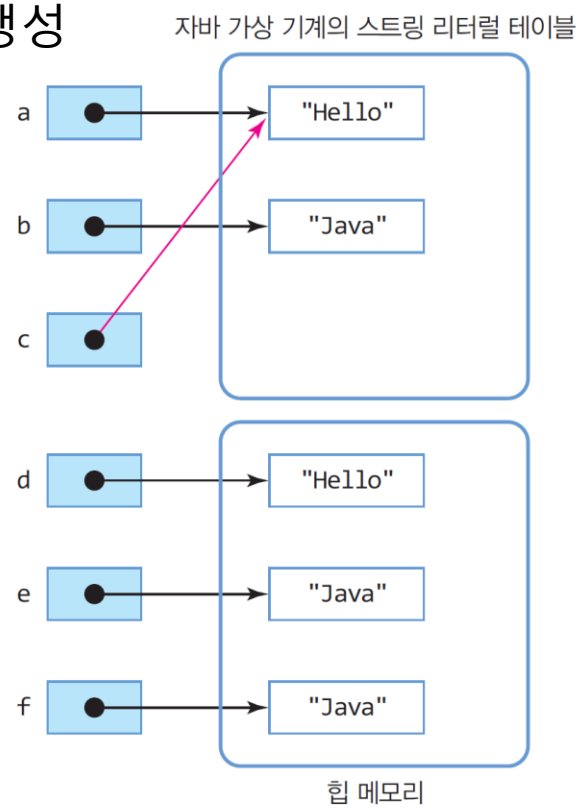
생성자	설명
String()	빈 스트링 객체 생성
String(char[] value)	char 배열에 있는 문자들을 스트링 객체로 생성
String(String original)	매개변수로 주어진 문자열과 동일한 스트링 객체 생성
String(StringBuffer buffer)	매개변수로 주어진 스트링 버퍼의 문자열을 스트링 객체로 생성

스트링 리터럴과 new String()



- 스트링 생성 방법
  - 리터럴로 생성, String s = "Hello";
    - JVM이 리터럴 관리, 응용프로그램 내에서 공유됨
  - String 객체로 생성, String t = new String("Hello");
    - 힙 메모리에 String 객체 생성

```
String a = "Hello";
String b = "Java";
String c = "Hello";
String d = new String("Hello");
String e = new String("Java");
String f = new String("Java");
```

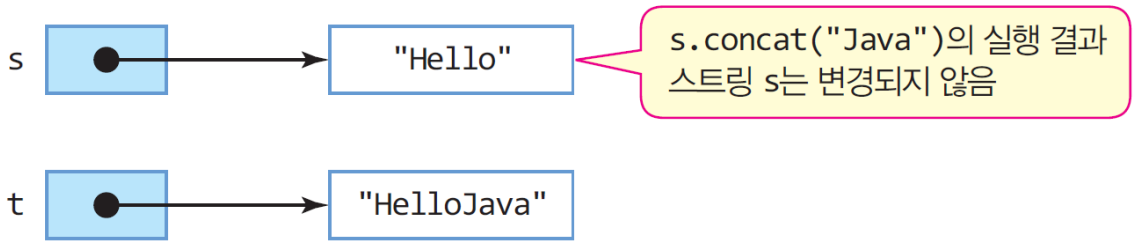


스트링 객체의 주요 특징



- 스트링 객체는 수정 불가능

```
String s = new String("Hello");
String t = s.concat("Java"); // 스트링 s에 "Java"를 덧붙인 새로운 스트링 객체 리턴
```



- 스트링 비교 시 반드시 equals()를 사용
  - == 사용하면 안 됨
  - equals()는 내용을 비교하기 때문

주요 메소드



메소드	설명
char charAt(int index)	index 인덱스에 있는 문자 값 리턴
int codePointAt(int index)	index 인덱스에 있는 유니코드 값 리턴
int compareTo(String anotherString)	두 스트링을 사전 순으로 비교하여 두 스트링이 같으면 0, 현 스트링이 anotherString보다 먼저 나오면 음수, 아니면 양수 리턴
String concat(String str)	현재 스트링 뒤에 str 스트링을 덧붙인 새로운 스트링 리턴
boolean contains(CharSequence s)	s에 지정된 문자들을 포함하고 있으면 true 리턴
int length()	스트링의 길이(문자 개수) 리턴
String replace(Charsequence target, Charsequence replacement)	target이 지정하는 일련의 문자들을 replacement가 지정하는 문자들로 변경한 스트링 리턴
String[] split(String regex)	정규식 regex에 일치하는 부분을 중심으로 스트링을 분리하고, 분리된 스트링들을 배열로 저장하여 리턴
String subString(int beginIndex)	beginIndex 인덱스부터 시작하는 서브 스트링 리턴
String toLowerCase()	소문자로 변경한 스트링 리턴
String toUpperCase()	대문자로 변경한 스트링 리턴
String trim()	스트링 앞뒤의 공백 문자들을 제거한 스트링 리턴

문자열 비교



- int compareTo(String anotherString)
  - 문자열이 같으면 0 리턴
  - 이 문자열이 anotherString 보다 사전에 먼저 나오면 음수 리턴
  - 이 문자열이 anotherString 보다 사전에 나중에 나오면 양수 리턴

```
String java= "Java";
String cpp = "C++";
int res = java.compareTo(cpp);
if(res == 0)
   System.out.println("the same");
else if(res <0)
   System.out.println(java + " < " + cpp);
else
   System.out.println(java + " > " + cpp);
```

"Java" 가 "C++" 보다 사전에 나중에 나오기 때문에 양수 리턴

```
Java > C++
```

- ==는 문자열 비교에는 사용하면 안됨

문자열 연결



- + 연산자로 문자열 연결
  - 피연산자에 문자열이나 객체가 포함되어 있는 경우
    - 객체는 객체.toString()을 호출하여 문자열로 변환하여 연결
    - 기본 타입 값은 문자열로 변환하여 연결

```
System.out.print("abcd" + 1 + true + 3.13e-2 + 'E'+ "fgh" );
// abcd1true0.0313Efgh  출력
```

- String concat(String str)를 이용한 문자열 연결

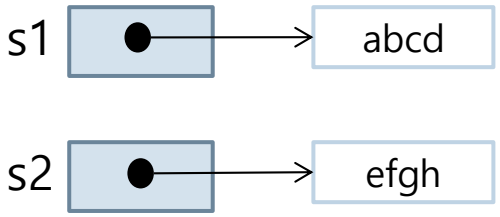
```
"I love ".concat("Java.") 는  "I Love Java." 리턴
```

  - 기존 String 객체에 연결되지 않고 새로운 스트링 객체 리턴
    - 다음 슬라이드에서 설명

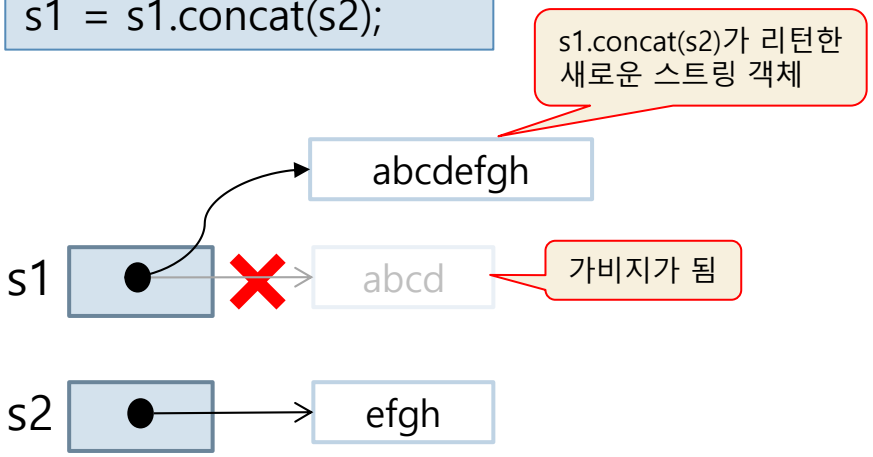
concat()은 새로운 문자열을 생성




```
String s1 = "abcd";
String s2 = "efgh";
```




```
s1 = s1.concat(s2);
```



문자열 내의 공백 제거, 문자열의 각 문자 접근



공백 제거

- String trim()
  - 문자열 앞 뒤 공백 문자(tab, enter, space) 제거한 문자열 리턴

```
String a = "       abcd def       ";
String b = "       xyz₩t";
String c = a.trim(); // c = "abcd def". 문자열 중간에 있는 공백은 제거되지 않음
String d = b.trim(); // d = "xyz". 스페이스와 '₩t' 제거됨"
```

문자열의 문자

- char charAt(int index)
  - 문자열 내의 문자 접근

```
String a = "class";
char c = a.charAt(2); // c = 'a'
```

```
// "class"에 포함된 's'의 개수를 세는 코드
int count = 0;
String a = "class";
for(int i=0; i<a.length(); i++) { // a.length()는 5
   if(a.charAt(i) == 's')
      count++;
}
System.out.println(count); // 2 출력
```

예제 6-7 : String 클래스 메소드 활용



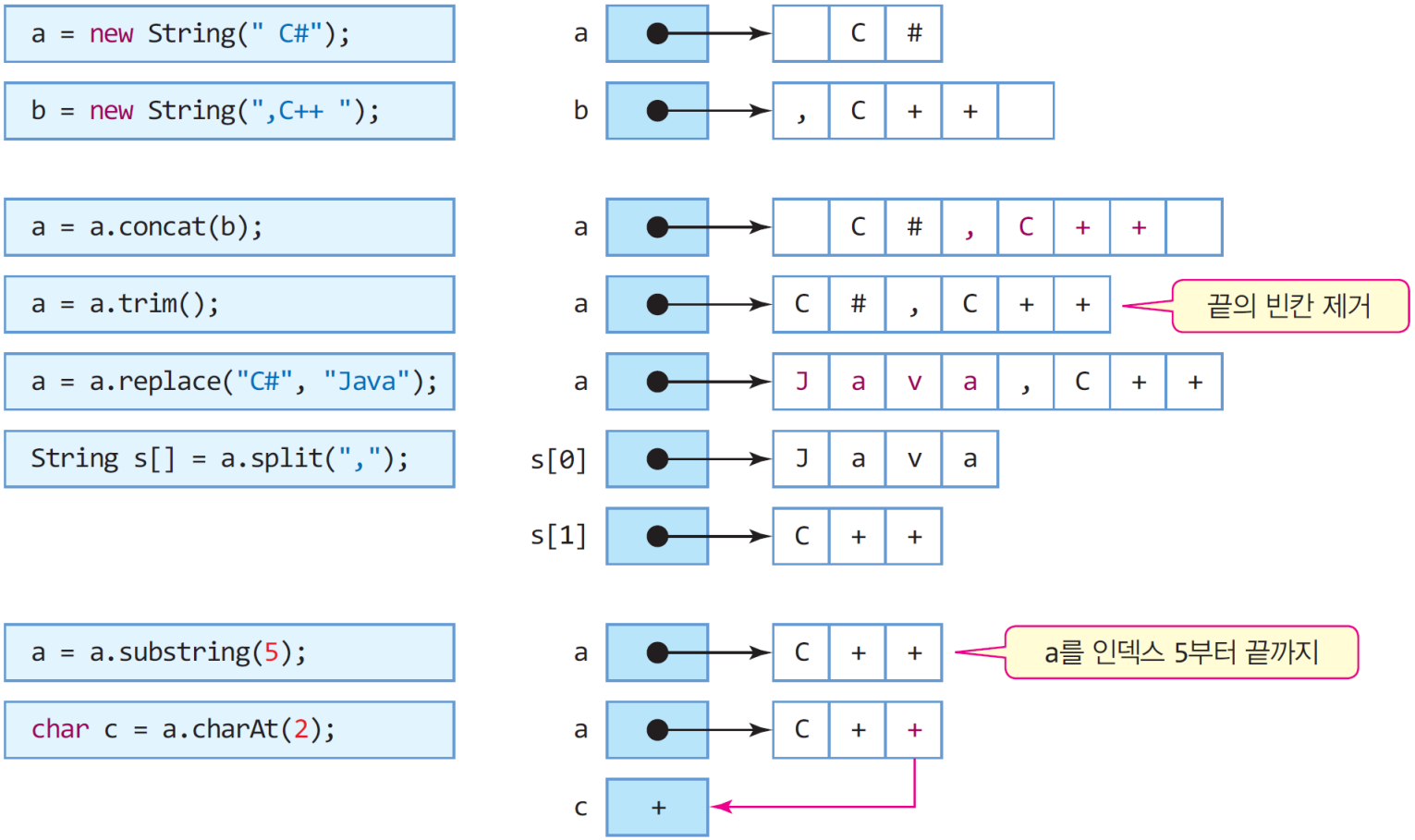
String 클래스의 다양한 메소드를 활용하는 예를 보여라.

```
public class StringEx {
    public static void main(String[] args) {
        String a = new String(" C#");
        String b = new String(",C++ ");

        System.out.println(a + "의 길이는 " + a.length()); // 문자열의 길이(문자 개수)
        System.out.println(a.contains("#")); // 문자열의 포함 관계

        a = a.concat(b); // 문자열 연결
        System.out.println(a);

        a = a.trim(); // 문자열 앞 뒤의 공백 제거
        System.out.println(a);

        a = a.replace("C#","Java"); // 문자열 대치
        System.out.println(a);

        String s[] = a.split(","); // 문자열 분리
        for (int i=0; i<s.length; i++)
            System.out.println("분리된 문자열" + i + ": " + s[i]);

        a = a.substring(5); // 인덱스 5부터 끝까지 서브 스트링 리턴
        System.out.println(a);

        char c = a.charAt(2); // 인덱스 2의 문자 리턴
        System.out.println(c);
    }
}
```

```
 C#의 길이는 3
true
 C#,C++
C#,C++
Java,C++
분리된 문자열0: Java
분리된 문자열1: C++
C++
+
```

예제 실행 과정




a = new String(" C#");
b = new String(",C++ ");
a = a.concat(b);
a = a.trim();
a = a.replace("C#", "Java");
String s[] = a.split(",");
a = a.substring(5);
char c = a.charAt(2);
a
b
a
a
a
s[0]
s[1]
a
a
c
끝의 빈칸 제거
a를 인덱스 5부터 끝까지

StringBuffer 클래스



- 가변 크기의 문자열 저장 클래스
  - Java.lang.StringBuffer
  - String 클래스와 달리 문자열 변경 가능
  - StringBuffer 객체의 크기는 스트링 길이에 따라 가변적
- 생성

```
StringBuffer sb = new StringBuffer("java");
```

생성자	설명
StringBuffer()	초기 버퍼의 크기가 16인 스트링 버퍼 객체 생성
StringBuffer(charSequence seq)	seq가 지정하는 일련의 문자들을 포함하는 스트링 버퍼 생성
StringBuffer(int capacity)	지정된 초기 크기를 갖는 스트링 버퍼 객체 생성
StringBuffer(String str)	지정된 스트링으로 초기화된 스트링 버퍼 객체 생성

주요 메소드



메소드	설명
StringBuffer append(String str)	str 스트링을 스트링 버퍼에 덧붙인다.
StringBuffer append(StringBuffer sb)	sb 스트링 버퍼를 현재의 스트링 버퍼에 덧붙인다. 이 결과 현재 스트링 버퍼의 내용이 변한다.
int capacity()	스트링 버퍼의 현재 크기 리턴
StringBuffer delete(int start, int end)	start 위치에서 end 위치 앞까지 부분 문자열 삭제
StringBuffer insert(int offset, String str)	str 스트링을 스트링 버퍼의 offset 위치에 삽입
StringBuffer replace(int start, int end, String str)	스트링 버퍼 내의 start 위치의 문자부터 end가 지정하는 문자 앞의 서브 스트링을 str로 대치
StringBuffer reverse()	스트링 버퍼 내의 문자들을 반대 순서로 변경
void setLength(int newLength)	스트링 버퍼 내 문자열 길이를 newLength로 재설정, 현재 길이보다 큰 경우 널 문자('')로 채우며 작은 경우는 기존 문자열이 잘린다.

StringBuffer의 메소드 활용 예



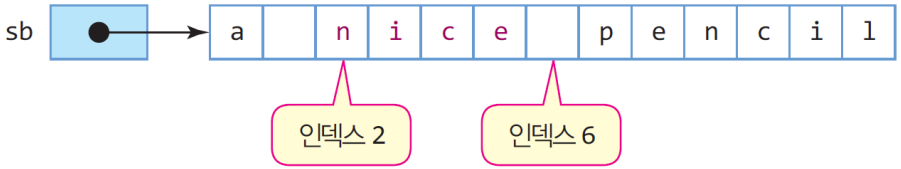

```
StringBuffer sb = new StringBuffer("a");
```



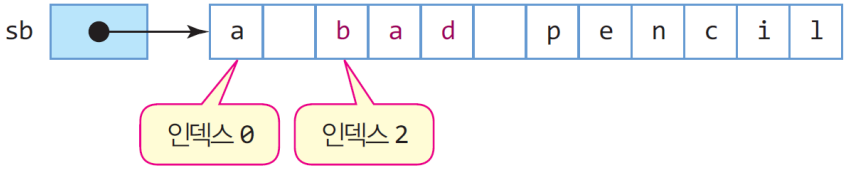

```
sb.append(" pencil");
```




```
sb.insert(2, "nice ");
```



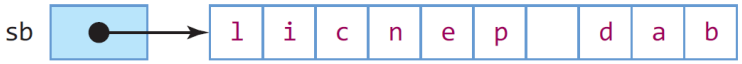

```
sb.replace(2, 6, "bad");
```



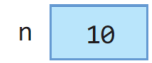

```
sb.delete(0, 2);
```



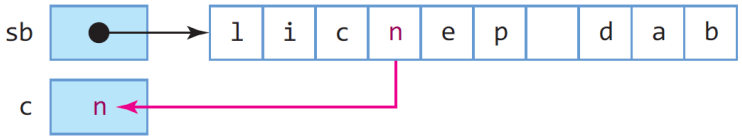

```
sb.reverse();
```




```
int n = sb.length();
```




```
char c = sb.charAt(3);
```




```
sb.setLength(3);
```



예제 6-8 : StringBuffer 클래스 메소드 활용



StringBuffer를 이용하여 문자열을 조작하는 다음 코드의 실행 결과는 무엇인가?

```
public class StringBufferEx {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer("This");

      sb.append(" is pencil"); // 문자열 덧붙이기
      System.out.println(sb);

      sb.insert(7, " my"); // "my" 문자열 삽입
      System.out.println(sb);

      sb.replace(8, 10, "your"); // "my"를 "your"로 변경
      System.out.println(sb);

      sb.delete(8, 13); // "your " 삭제
      System.out.println(sb);

      sb.setLength(4); // 스트링 버퍼 내 문자열 길이 수정
      System.out.println(sb);
   }
}
```

sb.toString()으로 자동 바뀜

```
This is pencil
This is my pencil
This is your pencil
This is pencil
This
```

StringTokenizer 클래스



- java.util.StringTokenizer
  - 하나의 문자열을 여러 문자열 분리
    - 문자열을 분리할 때 사용되는 기준 문자 : 구분 문자(delimiter)
      - 다음 예에서 '&'가 구분 문자

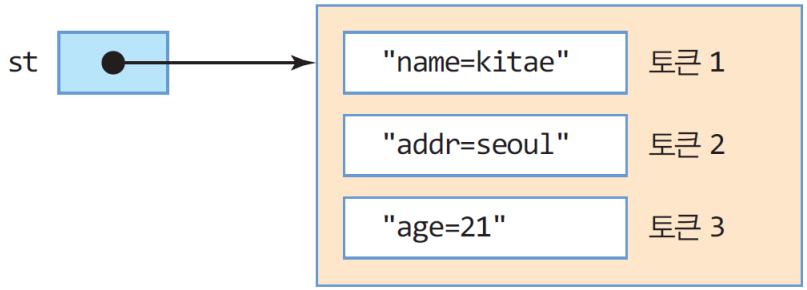
```
String query = "name=kitae&addr=seoul&age=21";
StringTokenizer st = new StringTokenizer(query, "&");
```

    - 토큰(token)
      - 구분 문자로 분리된 문자열
  - String 클래스의 split() 메소드를 이용하여 동일한 구현 가능

StringTokenizer 객체 생성과 문자열 분리

```
String query = "name=kitae&addr=seoul&age=21";
StringTokenizer st = new StringTokenizer(query, "&");
```

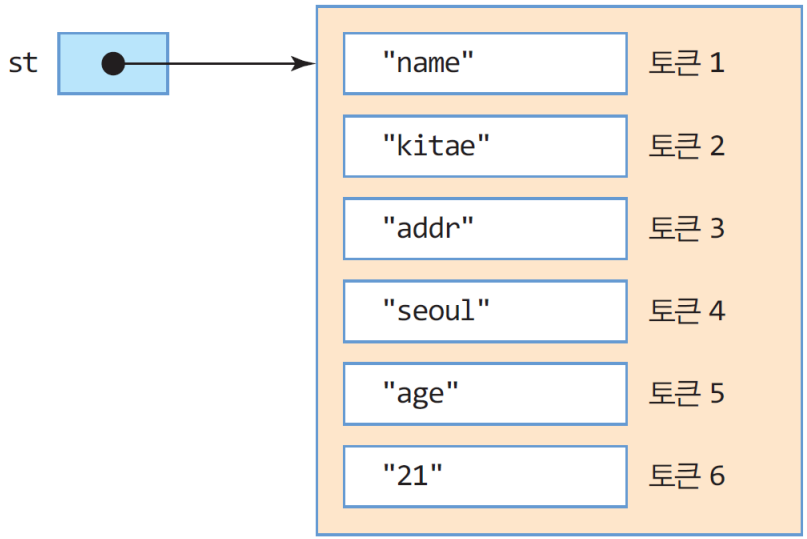
구분 문자 '&'



StringTokenizer 객체

```
StringTokenizer st = new StringTokenizer(query, "&=");
```

구분 문자 '&'와 '='



StringTokenizer 객체

생성자와 주요 메소드



생성자

생성자	설명
StringTokenizer(String str)	str 스트링의 각 문자를 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성
StringTokenizer(String str, String delim)	str 스트링과 delim 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성
StringTokenizer(String str, String delim, boolean returnDelims)	str 스트링과 delim 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성. returnDelims가 true이면 delim이 포함된 문자도 토큰에 포함된다.

주요 메소드

메소드	설명
int countTokens()	스트링 토크나이저가 분리한 토큰의 개수 리턴
boolean hasMoreTokens()	스트링 토크나이저에 다음 토큰이 있으면 true 리턴
String nexToken()	스트링 토크나이저에 들어 있는 다음 토큰 리턴

예제 6-9 : StringTokenizer 클래스 메소드 활용



"홍길동/장화/홍련/콩쥐/팥쥐"문자열을'/'를 구분 문자로 하여 토큰을 분리하여 각 토큰을 출력하라.

```
import java.util.StringTokenizer;

public class StringTokenizerEx {
   public static void main(String[] args) {
      StringTokenizer st = new StringTokenizer("홍길동/장화/홍련/콩쥐/팥쥐", "/");
      while (st.hasMoreTokens())
         System.out.println(st.nextToken());
   }
}
```

```
홍길동
장화
홍련
콩쥐
팥쥐
```

Math 클래스



- 산술 연산 메소드 제공, java.lang.Math
  - 모든 메소드는 static 타입 : 클래스 이름으로 바로 호출해야 함

메소드	설명
static double abs(double a)	실수 a의 절댓값 리턴
static double cos(double a)	실수 a의 cosine 값 리턴
static double sin(double a)	실수 a의 sine 값 리턴
static double tan(double a)	실수 a의 tangent 값 리턴
static double exp(double a)	e^a 값 리턴
static double ceil(double a)	올림. 실수 a보다 크거나 같은 수 중에서 가장 작은 정수를 실수 타입으로 리턴
static double floor(double a)	내림. 실수 a보다 작거나 같은 수 중에서 가장 큰 정수를 실수 타입으로 리턴
static double max(double a, double b)	두 수 a, b 중에서 큰 수 리턴
static double min(double a, double b)	두 수 a, b 중에서 작은 수 리턴
static double random()	0.0보다 크거나 같고 1.0보다 작은 임의의 실수 리턴
static long round(double a)	반올림. 실수 a를 소수 첫째 자리에서 반올림한 정수를 long 타입으로 반환
static double sqrt(double a)	실수 a의 제곱근 리턴

Math 클래스를 활용한 난수 발생



- 난수 발생
  - static double random()
    - 0.0 이상 1.0 미만의 임의의 double 값을 반환
    - 0에서 100사이의 난수 10개 발생시키는 샘플 코드

```
for(int x=0; x<10; x++) {
   int n = (int)(Math.random()*100 + 1); // n은 [1~100] 사이의 랜덤 정수
   System.out.println(n);
}
```

    - *Math.random()*100은 0.0~99.99.. 사이의 실수 리턴*
    - *Math.random()*100+1은 1.0~100.99.. 사이의 실수 값*
    - *(int)(Math.random()*100 + 1)는 소수점이하를 제거하여 1~100 사이의 정수 값*

- java.util.Random 클래스
  - 다양한 형태로 난수 발생 가능

예제 6-10 : Math 클래스 메소드 활용



Math 클래스의 다양한 메소드 활용 예를 보여라.

```
public class MathEx {
   public static void main(String[] args) {
      System.out.println(Math.PI); // 원주율 상수 출력
      System.out.println(Math.ceil(a)); // ceil(올림)
      System.out.println(Math.floor(a)); // floor(내림)
      System.out.println(Math.sqrt(9)); // 제곱근
      System.out.println(Math.exp(2)); // e의 2승
      System.out.println(Math.round(3.14)); // 반올림

      // [1, 45] 사이의 정수형 난수 5개 발생
      System.out.print("이번주 행운의 번호는 ");
      for(int i=0; i<5; i++)
         System.out.print((int)(Math.random()*45 + 1) + " ");
      }
}
```

```
3.141592653589793
4.0
3.0
3.0
7.38905609893065
3
이번주 행운의 번호는 15 31 9 7 5
```

Calendar 클래스



- Calendar 클래스의 특징
  - java.util 패키지
  - 시간과 날짜 정보 저장 관리
    - 년, 월, 일, 요일, 시간, 분, 초, 밀리초, 오전 오후 등
    - Calendar 클래스의 각 시간 요소를 설정하기나 알아내기 위한 필드들

필드	의미	필드	의미
YEAR	년도	DAY_OF_MONTH	한 달의 날짜
MONTH	달(0~11)	DAY_OF_WEEK	한 주의 요일
HOUR	시간(0~11)	AM_PM	오전인지 오후인지 구분
HOUR_OF_DAY	24시간을 기준으로 한 시간	MINUTE	분
SECOND	초	MILLISECOND	밀리초

Calendar 객체 생성 및 날짜와 시간



- Calendar 객체 생성,
  - Calendar now = Calendar.getInstance();
    - now객체는 현재 날짜와 시간 정보를 가지고 생성
    - Calendar는 추상 클래스이므로 new Calendar() 하지 않음
- 날짜와 시간 알아내기

```
int year = now.get(Calendar.YEAR);              // now에 저장된 년도
int month = now.get(Calendar.MONTH) + 1; // now에 저장된 달
```

- 날짜와 시간 설정하기
  - Calendar 객체에 저장
    - Calendar 객체에 날짜와 시간을 저장한다고 컴퓨터의 날짜와 시간을 바꾸는 것은 아님

```
// 이성 친구와 처음으로 데이트한 날짜와 시간 저장
Calendar firstDate = Calendar.getInstance();

firstDate.clear(); // 현재 날짜와 시간 정보를 모두 지운다.
firstDate.set(2016, 11, 25); // 2016년 12월 25일. 12월은 11로 설정
firstDate.set(Calendar.HOUR_OF_DAY, 20); // 저녁 8시로 설정
firstDate.set(Calendar.MINUTE, 30); // 30분으로 설정
```

예제 6-11 : Calendar를 이용하여 현재 날짜와 시간 알아내기/날짜 시간 설정하기




```
import java.util.Calendar;
public class CalendarEx {
    public static void printCalendar(String msg, Calendar cal) {
        int year = cal.get(Calendar.YEAR);

        // get()은 0~30까지의 정수 리턴.
        int month = cal.get(Calendar.MONTH) + 1;
        int day = cal.get(Calendar.DAY_OF_MONTH);
        int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
        int hour = cal.get(Calendar.HOUR);
        int hourOfDay = cal.get(Calendar.HOUR_OF_DAY);
        int ampm = cal.get(Calendar.AM_PM);
        int minute = cal.get(Calendar.MINUTE);
        int second = cal.get(Calendar.SECOND);
        int millisecond = cal.get(Calendar.MILLISECOND);

        System.out.print(msg + year + "/" + month + "/" + day + "/");
        switch(dayOfWeek) {
            case Calendar.SUNDAY : System.out.print("일요일"); break;
            case Calendar.MONDAY : System.out.print("월요일"); break;
            case Calendar.TUESDAY : System.out.print("화요일"); break;
            case Calendar.WEDNESDAY : System.out.print("수요일"); break;
            case Calendar.THURSDAY : System.out.print("목요일"); break;
            case Calendar.FRIDAY: System.out.print("금요일"); break;
            case Calendar.SATURDAY : System.out.print("토요일"); break;
        }
        System.out.print("(" + hourOfDay + "시)");
        if(ampm == Calendar.AM) System.out.print("오전");
        else System.out.print("오후");
        System.out.println(hour + "시 " + minute + "분 " + second + "초 "
            + millisecond +"밀리초");
    }
```

```
public static void main(String[] args) {
        Calendar now = Calendar.getInstance();
        printCalendar("현재 ", now);

        Calendar firstDate = Calendar.getInstance();
        firstDate.clear();
        // 2016년 12월 25일. 12월을 표현하기 위해 month에 11로 설정
        firstDate.set(2016, 11, 25);
        firstDate.set(Calendar.HOUR_OF_DAY, 20); // 저녁 8시
        firstDate.set(Calendar.MINUTE, 30); // 30분
        printCalendar("처음 데이트한 날은 ", firstDate);
    }
}
```

```
현재 2017/3/29/수요일(19시)오후7시 59분 51초 892밀리초
처음 데이트한 날은 2016/12/25/일요일(20시)오후8시 30분 0초 0밀리초
```