

06

패키지 개념과 자바 패키지 활용

학습 목표

1. 패키지 개념 이해
2. 사용자 패키지 만들기
3. 자바에서 제공하는 표준 패키지
4. Object 클래스 활용
5. 박싱/언박싱을 이해하고 Wrapper 클래스 활용
6. String과 StringBuffer 클래스 활용
7. StringTokenizer 클래스 활용
8. Math 클래스 활용

패키지 개념과 필요성

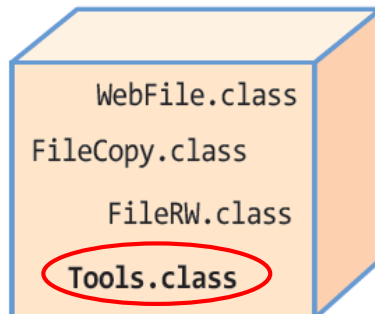
3

- * 3명이 분담하여 자바 응용프로그램을 개발하는 경우,
동일한 이름의 클래스가 존재할 가능성 있음
 - > 합칠 때 오류 발생 가능성
 - > 개발자가 서로 다른 디렉터리로 코드 관리하여 해결



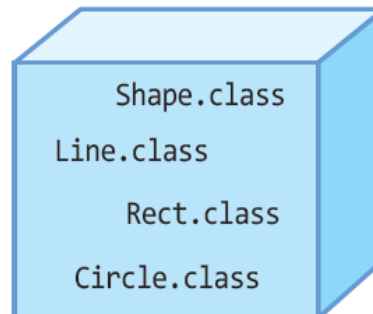
개발자 A

FileIO 작업



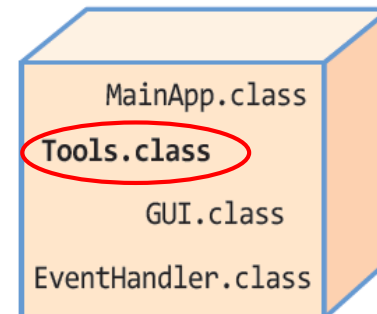
개발자 B

Graphic 작업



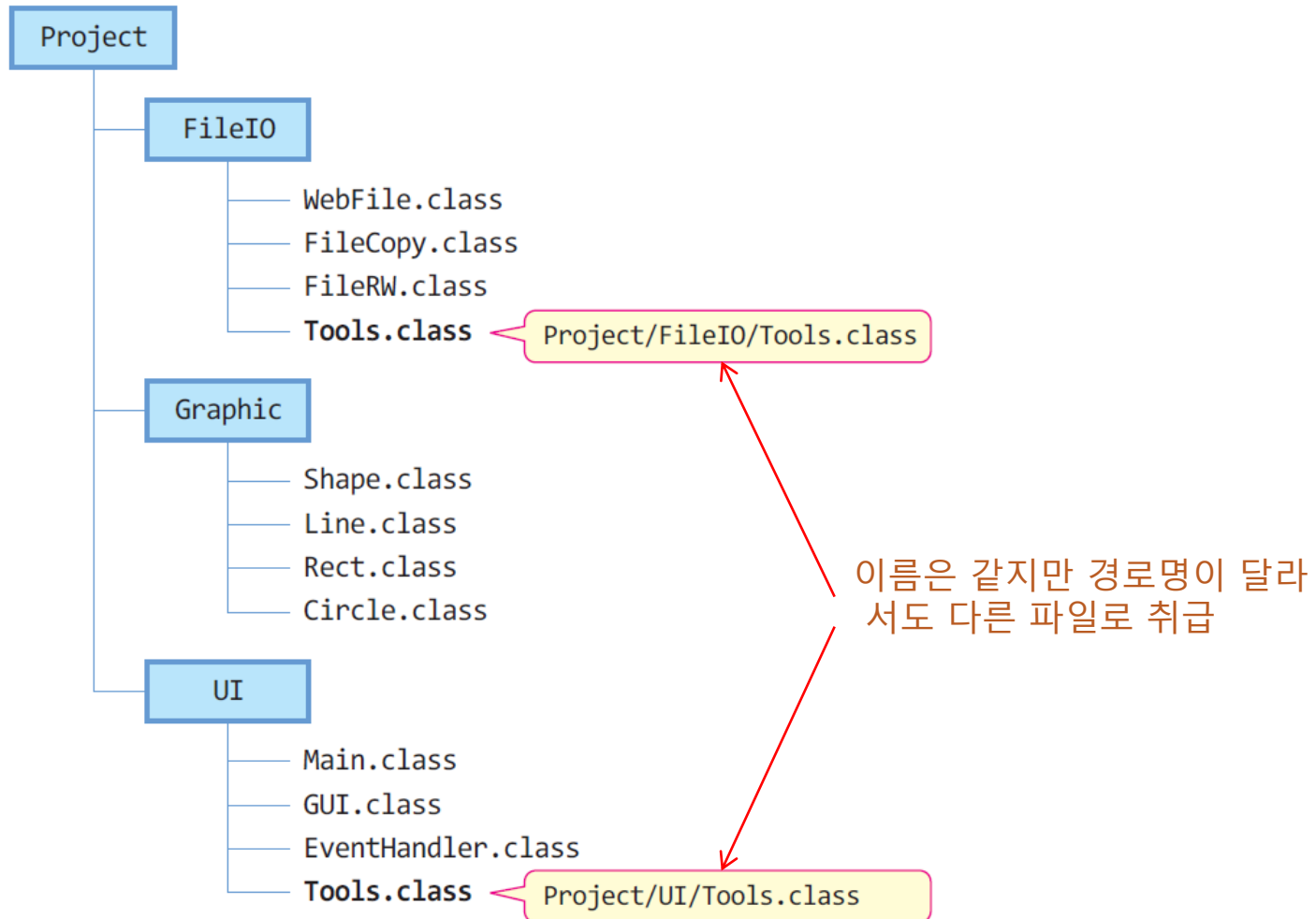
개발자 C

UI 작업



개발자가 서로 다른 디렉터리로 코드 관리

4



자바 패키지

5

□ 패키지(package)

- ▣ 서로 관련된 클래스와 인터페이스를 컴파일한 클래스 파일들을 묶어 놓은 디렉터리
- ▣ 하나의 응용프로그램은 한 개 이상의 패키지로 작성
- ▣ 패키지는 jar 파일로 압축할 수 있음
 - JDK에서 제공하는 표준 패키지는 rt.jar에 압축

□ 클래스 경로명

- ▣ 패키지 이름과 클래스 이름으로 완성



자바 표준 패키지와 클래스 경로명

6

rt.jar

클래스의 이름
(경로명)

java.awt.Color

패키지명

패키지명 : java.awt java.awt 패키지에 속한 클래스

패키지 사용하기, import문

7

□ 다른 패키지에 작성된 클래스 사용

□ import를 이용하지 않는 경우

- 소스에 클래스 이름의 완전 경로명 사용

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

□ 필요한 클래스만 import

- 소스 시작 부분에 클래스의 경로명 import
- import 패키지.클래스
- 소스에는 클래스 명만 명시하면 됨

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

□ 패키지 전체를 import

- 소스 시작 부분에 패키지의 경로명.* import
- import 패키지.*
- 소스에는 클래스 명만 명시하면 됨
- import java.util.*;
 - *java.util* 패키지 내의 모든 클래스만을 지정, 하위 패키지의 클래스는 포함하지 않음

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

패키지 만들기

8

- 클래스 파일(.class)이 저장되는 위치는?
 - ▣ 클래스나 인터페이스가 컴파일되면 클래스 파일(.class) 생성
 - ▣ 클래스 파일은 패키지로 선언된 디렉터리에 저장
- 패키지 선언
 - ▣ 소스 파일의 맨 앞에 컴파일 후 저장될 패키지 지정
 - package 패키지명;

```
package UI; // 아래 Tools를 컴파일하여 UI 패키지(UI 디렉토리)에 저장할 것 지시

public class Tools {
    .....
}
```

Tools 클래스의
경로명은
UI.Tools가 됨

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

import UI.Tools; // UI.Tools 클래스의 경로명 импорт

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


디폴트 패키지

9

- package 선언문이 없는 자바 소스 파일의 경우
 - ▣ 컴파일러는 클래스나 인터페이스를 디폴트 패키지에 소속시킴
 - ▣ 디폴트 패키지
 - 현재 디렉터리

이클립스로 쉽게 패키지 만들기

10

▣ 예제로 사용할 샘플 소스(5장의 예제 5-5)

```
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);  
}  
  
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 }));  
    }  
}
```

lib 패키지에

app 패키지에



Calculator 클래스는 lib 패키지에
GoodCalc 클래스는 app 패키지에
나누어 저장하는 응용프로그램을
이클립스를 이용하여 만들기

프로젝트 작성(프로젝트 이름 : PackageEx)

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name: PackageEx

☒ Use default location
Location: C:\Temp\PackageEx [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-1.8
☐ Use a project specific JRE: jre8
☐ Use default JRE (currently 'jre8') [Configure JREs...](#)

Project layout

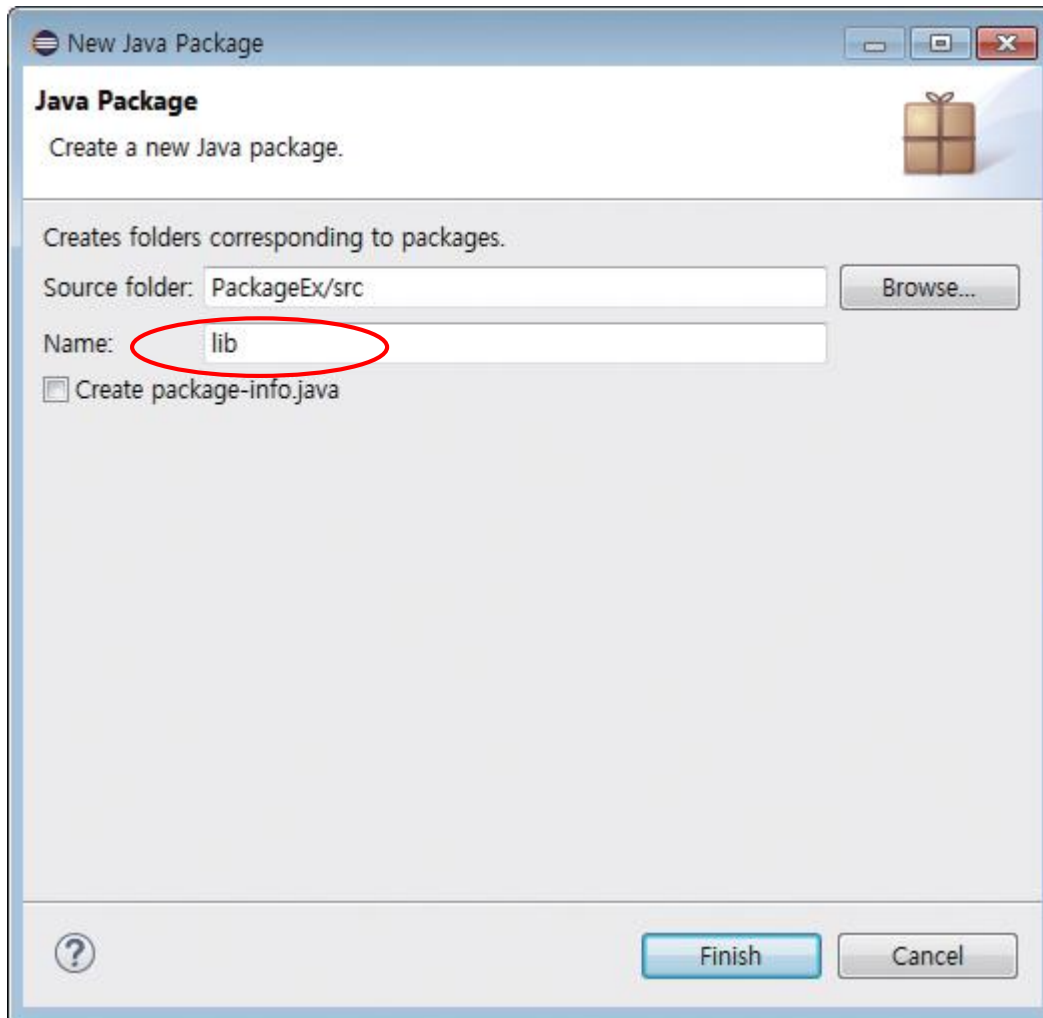
☐ Use project folder as root for sources and class files
☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

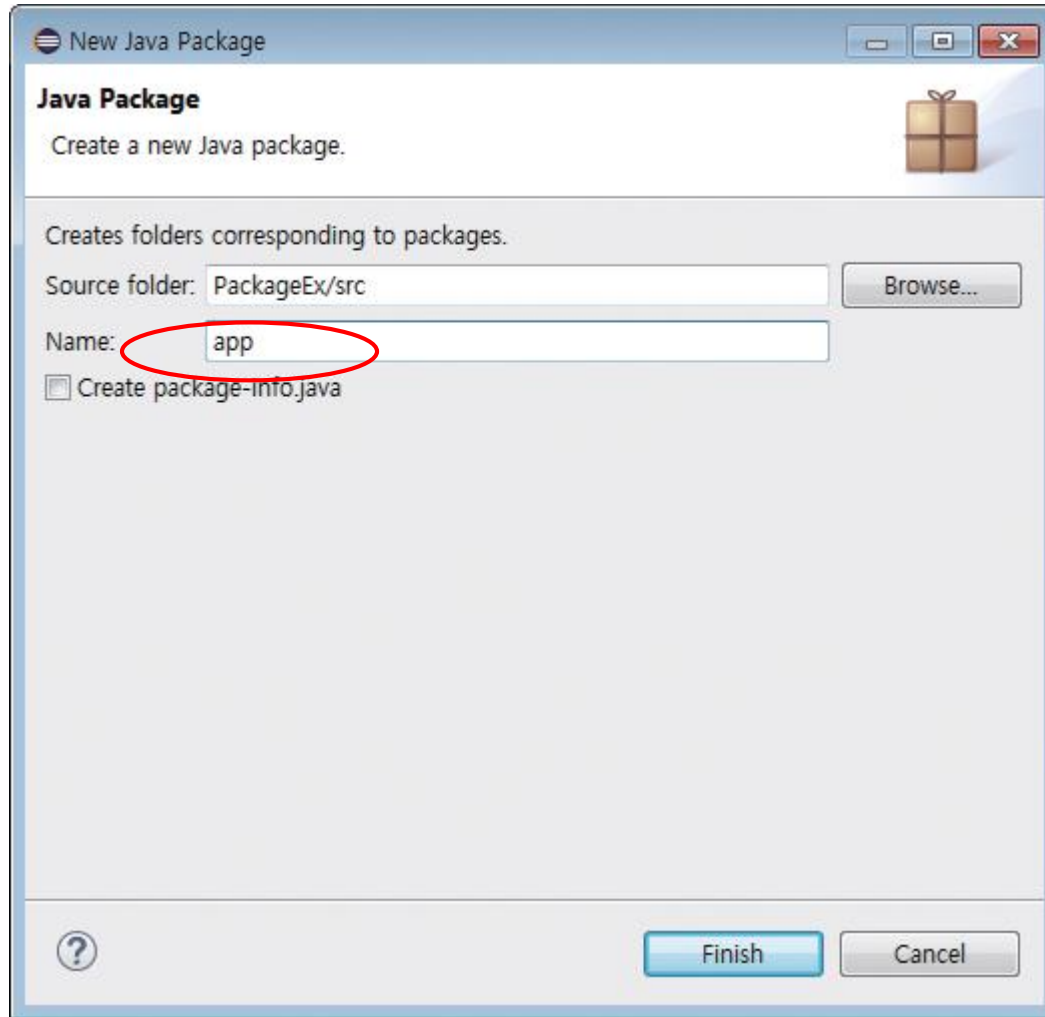
☐ Add project to working sets
Working sets: [Select...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

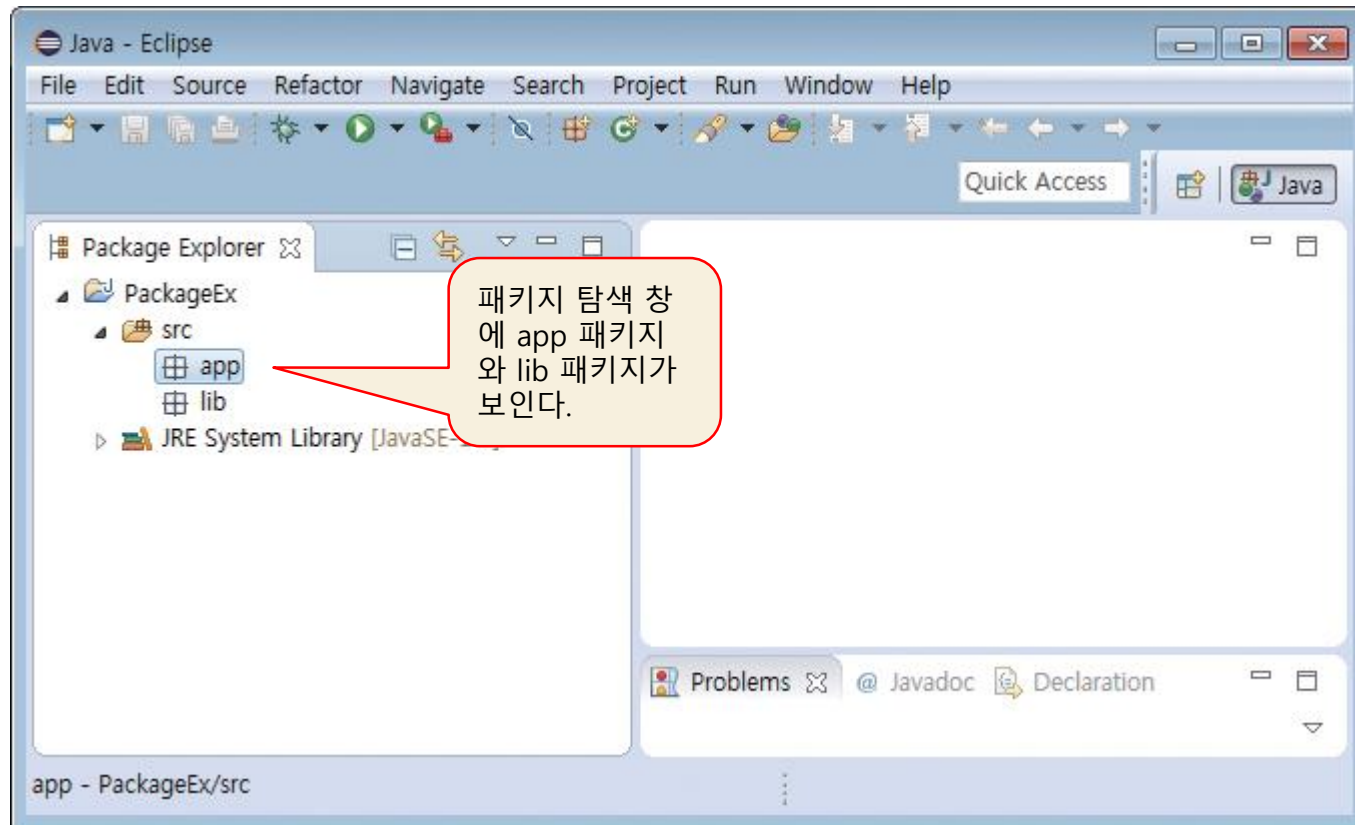
패키지 lib 작성



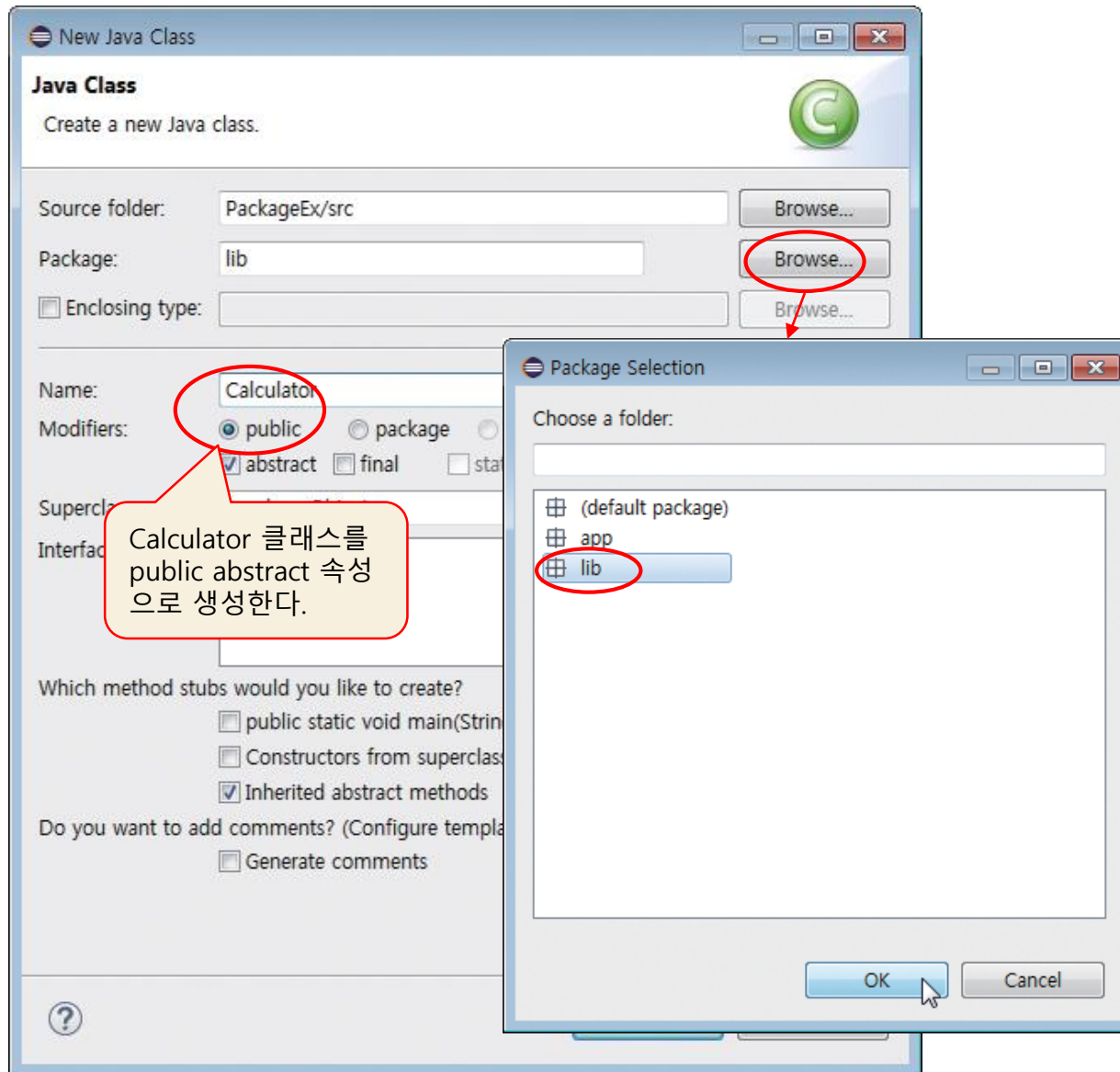
패키지 app 작성



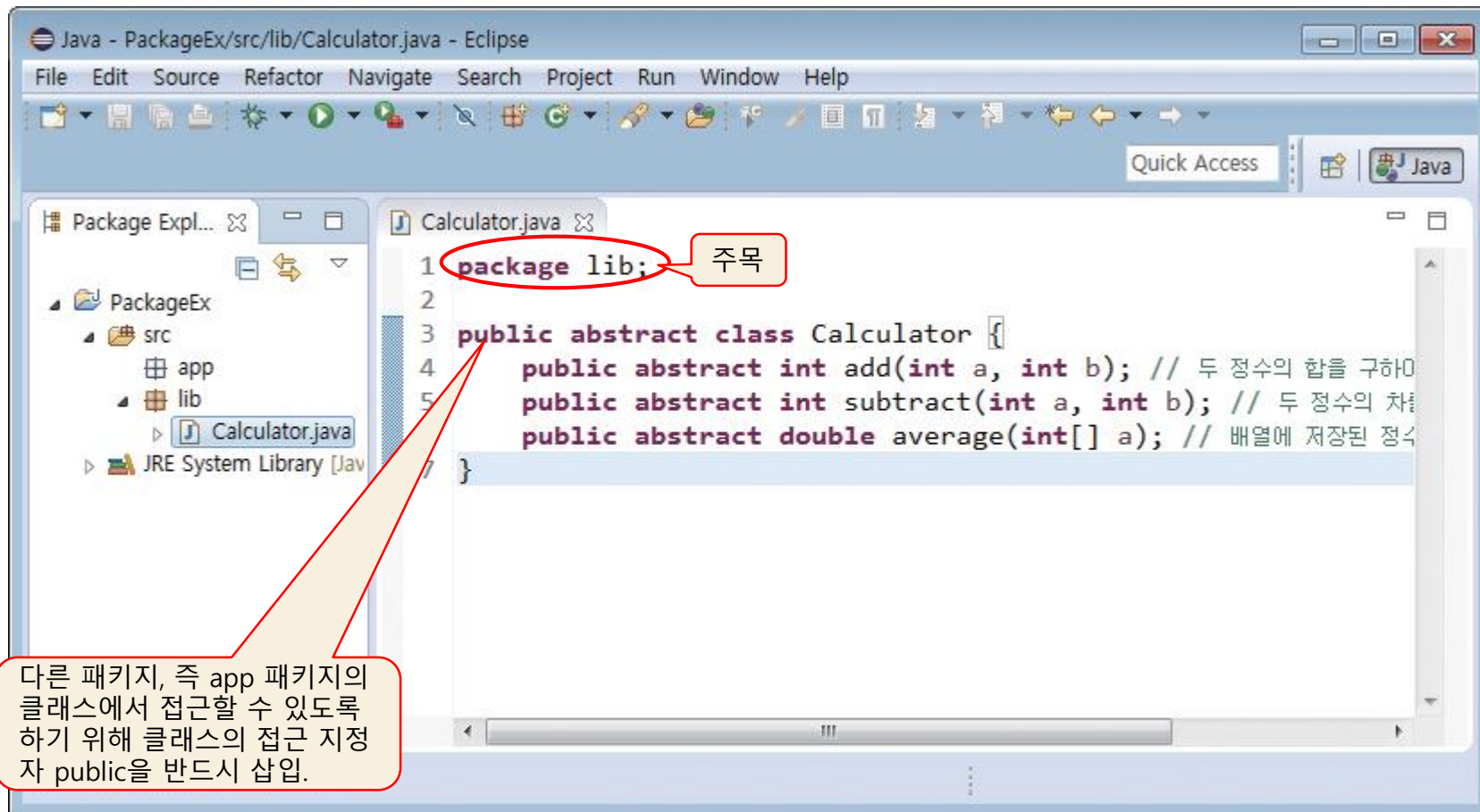
패키지 작성이 완료된 결과



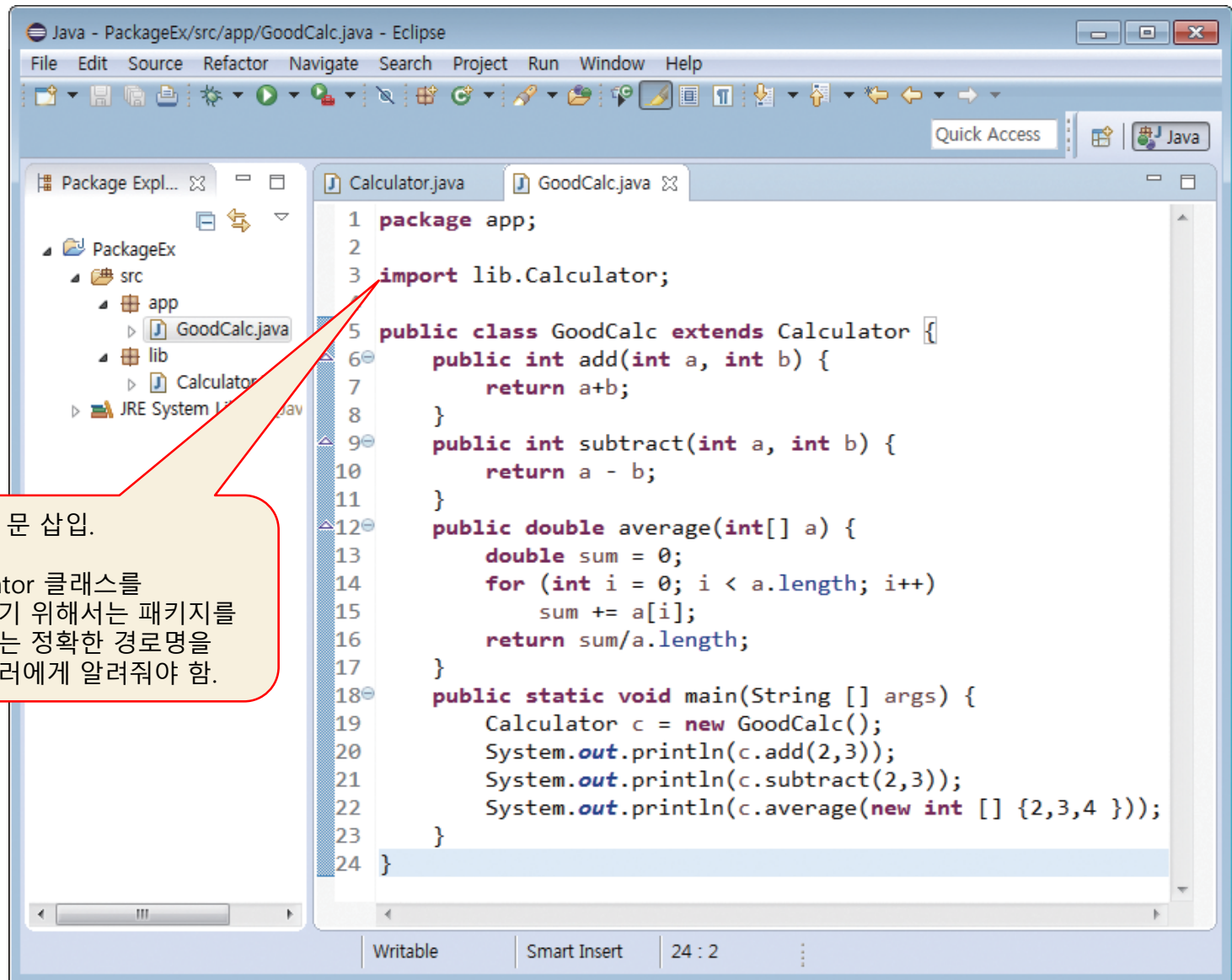
lib 패키지에 클래스 Calculator 만들기



Calculator 소스 작성 후 수정

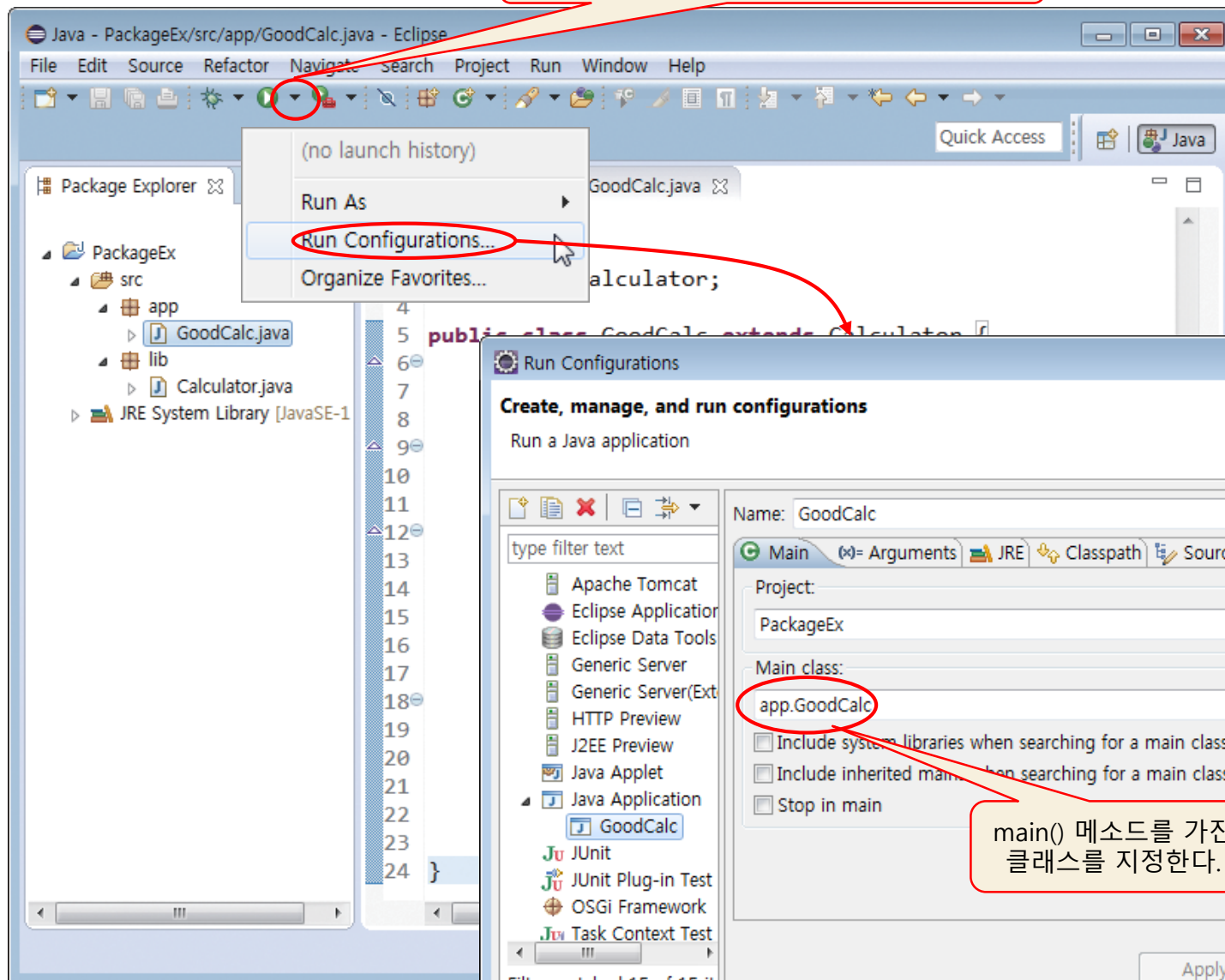


app 패키지에 GoodCalc.java 작성 후 수정



푸시다운 버튼을 누르면 아래 메뉴가 보인다.

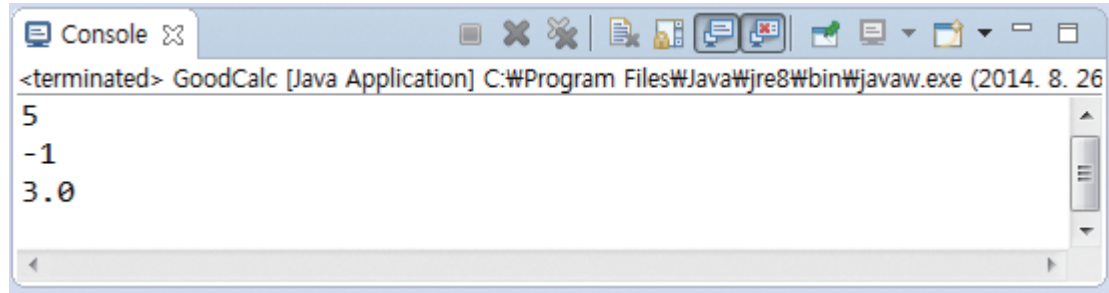
실행을 위한
Run
Configurations
작성



main() 메소드를 가진
클래스를 지정한다.

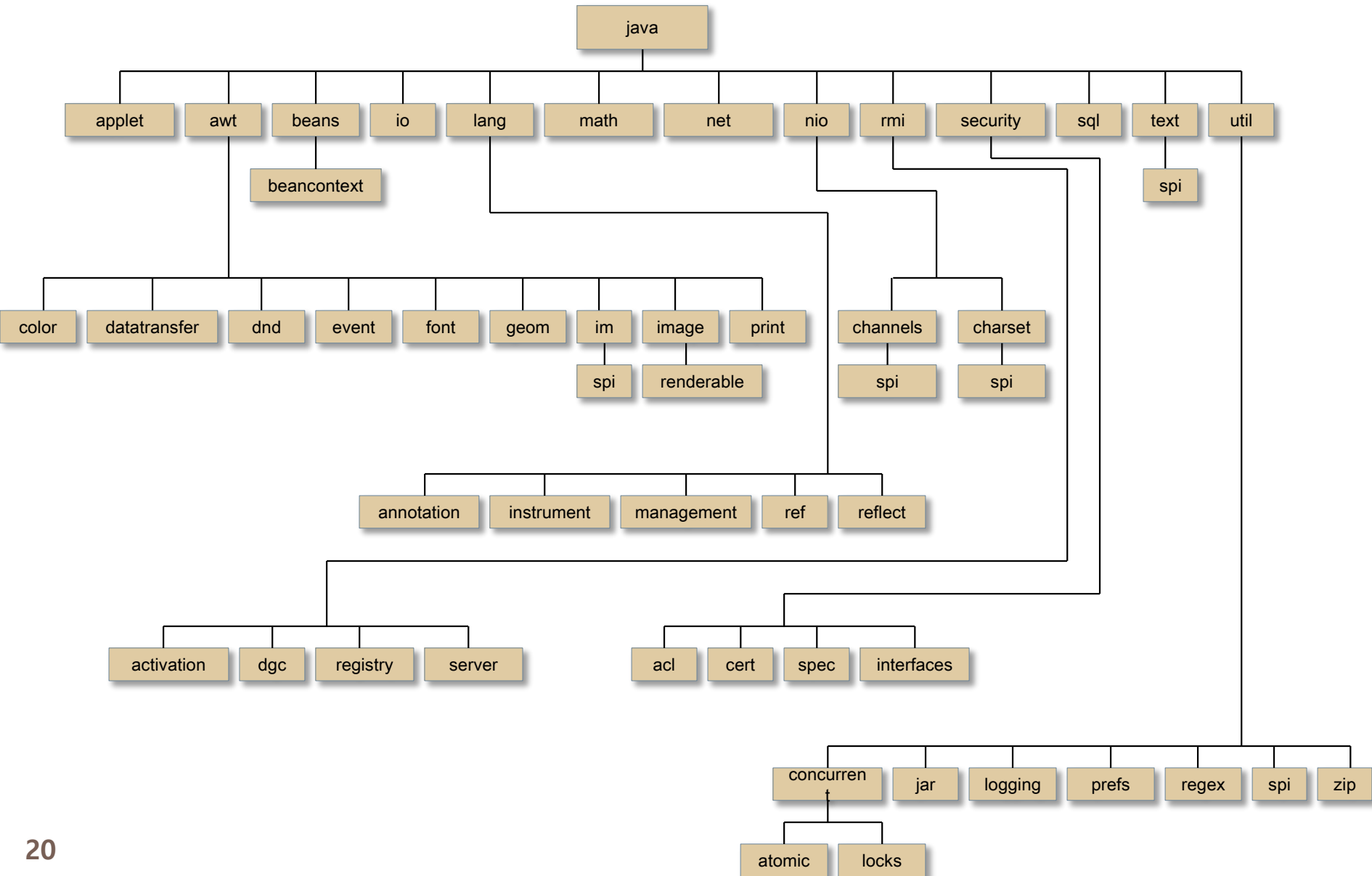
프로젝트 PackageEx 실행

19



```
<terminated> GoodCalc [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (2014. 8. 26)
5
-1
3.0
```

JDK 표준 자바 패키지 구조 : rt.jar



JDK의 주요 패키지

21

▣ java.lang

- 스트링, 수학 함수, 입출력 등 자바 프로그래밍에 필요한 기본적인 클래스와 인터페이스
- 자동으로 import 됨 - import 문 필요 없음

▣ java.util

- 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

- GUI 프로그램을 작성하기 위한 AWT 패키지

▣ javax.swing

- GUI 프로그래밍을 작성하기 위한 스윙 패키지

Object 클래스

22

□ 특징

- 모든 자바 클래스는 반드시 Object를 상속받도록 자동 컴파일
 - 모든 클래스의 슈퍼 클래스
 - 모든 클래스가 상속받는 공통 메소드 포함

□ 주요 메소드

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

객체 속성

23

- Object 클래스는 객체의 속성을 나타내는 메소드 제공
 - ▣ hashCode() 메소드
 - 객체의 해시코드 값을 리턴하며, 객체마다 다름
 - ▣ getClass() 메소드
 - 객체의 클래스 정보를 담은 Class 객체 리턴
 - Class 객체의 getName() 메소드는 객체의 클래스 이름 리턴
 - ▣ toString() 메소드
 - 객체를 문자열로 리턴

예제 6-1 : Object 클래스로 객체 속성 알아내기

24

Object 클래스를 이용하여 객체의 클래스명, 해시 코드 값, 객체의 문자열을 출력해보자.

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

public class ObjectPropertyEx {
    public static void main(String [] args) {
        Point p = new Point(2,3);
        System.out.println(p.getClass().getName()); // 클래스 이름
        System.out.println(p.hashCode());           // 해시 코드 값
        System.out.println(p.toString());            // 객체의 문자열
    }
}
```

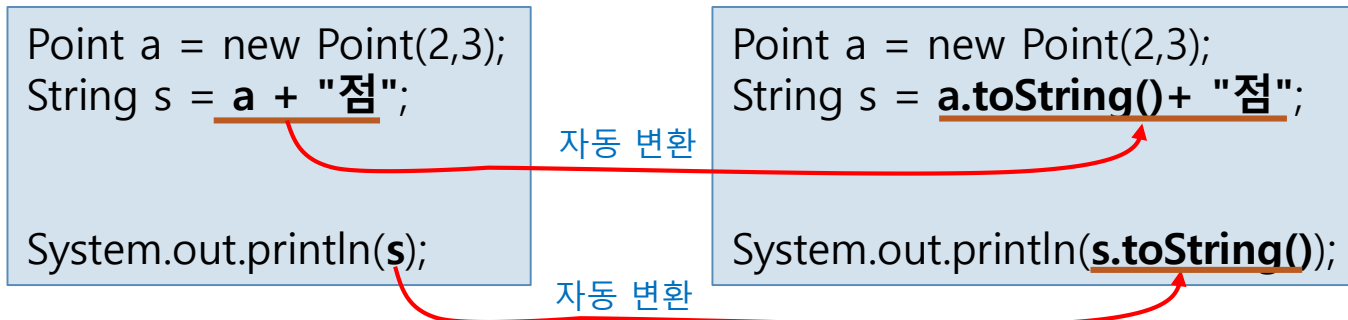
Point
22279806
Point@153f67e

해시 코드의 16진수 값.
이 값은 실행할 때마다
달라질 수 있음.

toString() 메소드, 객체를 문자열로 변환

25

- 각 클래스는 toString()을 오버라이딩하여 자신만의 문자열 리턴 가능
 - 객체를 문자열로 반환
 - 원형
 - public String toString();
- 컴파일러에 의한 toString() 자동 변환
 - '객체 + 문자열' -> '객체.toString() + 문자열'로 자동 변환
 - 객체를 단독으로 사용 하는 경우 -> 객체.toString()으로 자동 변환



예제 6-2 : Point 클래스에 toString() 작성

26

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

```
class Point {  
    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 a = new Point(2,3);  
        System.out.println(a.toString());  
        System.out.println(a); // a는 a.toString()으로 자동 변환됨  
    }  
}
```

Point(2,3)
Point(2,3)

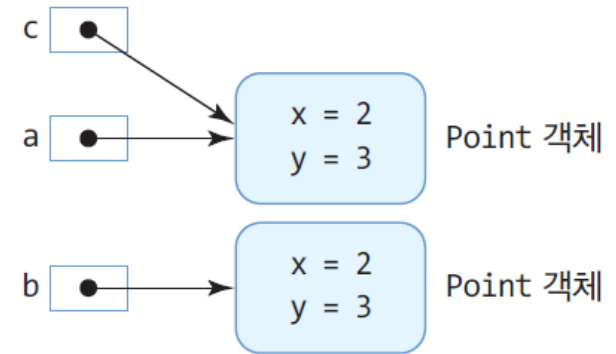
객체 비교(==)와 equals() 메소드

27

- == 연산자
 - ▣ 객체 레퍼런스 비교

```
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



- boolean equals(Object obj)
 - ▣ 두 객체의 내용물 비교
 - ▣ 객체의 내용물을 비교하기 위해 클래스의 멤버로 작성

예제 6-3 : Point 클래스의 equals() 작성

28

Point 클래스에 x, y 점 좌표가 같으면 true를 리턴하는 equals()를 작성하라.

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

이 Point 객체와 객체 p가
같은지 비교

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

a is equal to b

예제 6-4 : Rect 클래스와 equals() 메소드 만들기 연습

29

int 타입의 **width**(너비), **height**(높이) 필드를 가지는 **Rect** 클래스를 작성하고, 면적이 같으면 두 **Rect** 객체가 같은 것으로 판별하는 **equals()**를 작성하라.

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

이 사각형과 객체 p의 면적 비교

```
public class RectEx {  
    public static void main(String[] args) {  
        Rect a = new Rect(2,3); // 면적 6  
        Rect b = new Rect(3,2); // 면적 6  
        Rect c = new Rect(3,4); // 면적 12  
  
        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");  
    }  
}
```

a와 b는 면적이 같으므로
equals()는 같다고 판단

a is equal to b

Wrapper 클래스

30

- Wrapper 클래스
 - ▣ 자바의 기본 타입을 클래스화한 8개 클래스를 통칭

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

- 용도
 - ▣ 객체만 사용할 수 있는 컬렉션 등에 기본 타입의 값을 사용하기 위해 -> Wrapper 객체로 만들어 사용

Wrapper 클래스의 객체 생성

31

□ Wrapper 객체로 생성하는 방법

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

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

주요 메소드

32

- ▣ 가장 많이 사용하는 Integer 클래스의 주요 메소드
 - 다른 Wrapper 클래스의 메소드는 이와 유사

| 메소드 | 설명 |
|---|---|
| <code>static int bitCount(int i)</code> | 정수 <code>i</code> 의 이진수 표현에서 1의 개수 리턴 |
| <code>float floatValue()</code> | <code>float</code> 타입으로 값 리턴 |
| <code>int intValue()</code> | <code>int</code> 타입으로 값 리턴 |
| <code>long longValue()</code> | <code>long</code> 타입으로 값 리턴 |
| <code>short shortValue()</code> | <code>short</code> 타입으로 값 리턴 |
| <code>static int parseInt(String s)</code> | 스트링 <code>s</code> 를 10진 정수로 변환한 값 리턴 |
| <code>static int parseInt(String s, int radix)</code> | 스트링 <code>s</code> 를 지정된 진법의 정수로 변환한 값 리턴 |
| <code>static String toBinaryString(int i)</code> | 정수 <code>i</code> 를 이진수 표현으로 변환한 스트링 리턴 |
| <code>static String toHexString(int i)</code> | 정수 <code>i</code> 를 16진수 표현으로 변환한 스트링 리턴 |
| <code>static String toOctalString(int i)</code> | 정수 <code>i</code> 를 8진수 표현으로 변환한 스트링 리턴 |
| <code>static String toString(int i)</code> | 정수 <code>i</code> 를 스트링으로 변환하여 리턴 |

Wrapper 클래스의 활용

33

□ Wrapper 객체에 들어 있는 기본 타입 값 알아내기

```
Integer i = new Integer(10);  
int ii = i.intValue();           // ii = 10  
Character c = new Character('c');  
char cc = c.charValue();         // cc = 'c'  
Boolean b = new Boolean(true);  
boolean bb = b.booleanValue();   // bb = true
```

□ 문자열을 기본 타입으로 변환

```
int i = Integer.parseInt("123");           // i = 123  
boolean b = Boolean.parseBoolean("true");  // b = true  
double d = Double.parseDouble("3.141592"); // d = 3.141592
```

□ 기본 타입 값을 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 3.141592를 문자열 "3.141592"로 변환  
String s4 = Character.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

34

다음은 Wrapper 클래스를 활용하는 예이다. 다음 프로그램의 결과는 무엇인가?

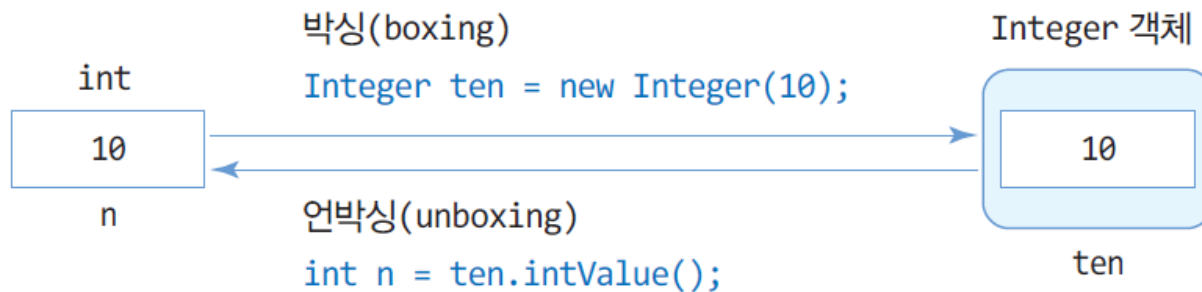
```
public class WrapperEx {  
    public static void main(String[] args) {  
        // Character 사용  
        System.out.println(Character.toLowerCase('A')); // 'A'를 소문자로 변환  
        char c1='4', c2='F';  
        if(Character.isDigit(c1)) // 문자 c1이 숫자이면 true  
            System.out.println(c1 + "는 숫자");  
        if(Character.isAlphabetic(c2)) // 문자 c2가 영문자이면 true  
            System.out.println(c2 + "는 영문자");  
  
        // Integer 사용  
        System.out.println(Integer.parseInt("28")); // 문자열 "28"을 10진수로 변환  
        System.out.println(Integer.toString(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  
        Integer i = new Integer(28);  
        System.out.println(i.doubleValue()); // 정수를 double 값으로 변환. 28.0  
  
        // Double 사용  
        Double d = new Double(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 사용  
        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  
    }  
}
```

a
4는 숫자
F는 영문
자
28
28
11100
3
28.0
3.14
3.14
true
false

박싱과 언박싱

35

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



- 자동 박싱과 자동 언박싱
 - JDK 1.5부터 박싱과 언박싱은 자동으로 이루어지도록 컴파일됨

```
Integer ten = 10;    // 자동 박싱. Integer ten = new Integer(10);과 동일
int n = ten;         // 자동 언박싱. int n = ten.intValue();와 동일
```

String의 생성과 특징

36

□ String

- ▣ String 클래스는 문자열을 나타냄
- ▣ 스트링 리터럴(문자열 리터럴)은 String 객체로 처리됨
- ▣ 스트링 객체의 생성 사례

```
String str1 = "abcd";
```

```
char data[] = {'a', 'b', 'c', 'd'};
```

```
String str2 = new String(data);
```

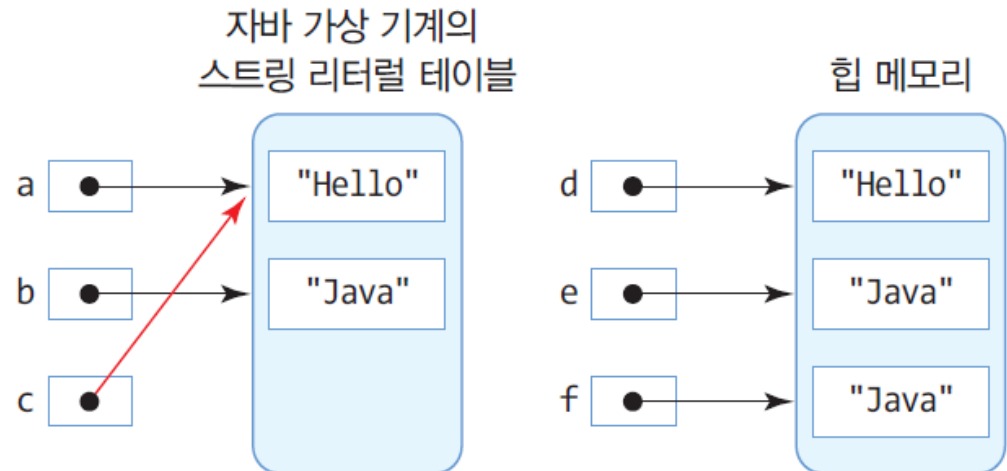
```
String str3 = new String("abcd"); // str2와 str3은 모두 "abcd" 스트링
```

스트링 리터럴과 new String()

37

- 스트링 리터럴
 - ▣ 자바 가상 기계 내부에서 리터럴 테이블에 저장되고 관리됨
 - ▣ 응용프로그램에서 공유됨
 - 스트링 리터럴 사례) String s = "Hello";
- new String()으로 생성된 스트링
 - ▣ 스트링 객체는 힙에 생성
 - ▣ 스트링은 공유되지 않음

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



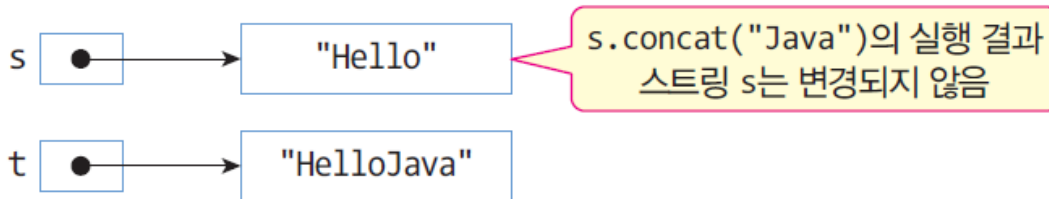
스트링 객체의 주요 특징

38

□ 스트링 객체는 수정 불가능

- ▣ 리터럴 스트링이든 new String()을 생성했든 객체의 문자열 수정 불가능
- ▣ 예)

```
String s = new String("Hello");    // s의 스트링은 수정 불가능  
String t = s.concat("Java");       // 스트링 s에 "Java"를 덧붙인 스트링 리턴
```



□ 스트링 비교

- ▣ 두 스트링을 비교할 때 반드시 equals()를 사용하여야 함
 - equals()는 내용을 비교하기 때문

주요 메소드

39

| 메소드 | 설명 |
|--|---|
| <code>char charAt(int index)</code> | <code>index</code> 인덱스에 있는 문자 값 리턴 |
| <code>int codePointAt(int index)</code> | <code>index</code> 인덱스에 있는 유니코드 값 리턴 |
| <code>int compareTo(String anotherString)</code> | 두 스트링을 사전적 순서를 기준으로 비교. 두 스트링이 같으면 0, 현 스트링이 <code>anotherString</code> 보다 먼저 나오면 음수, 아니면 양수 리턴 |
| <code>String concat(String str)</code> | <code>str</code> 스트링을 현재 스트링 뒤에 덧붙인 스트링 리턴 |
| <code>boolean contains(CharSequence s)</code> | <code>s</code> 에 지정된 문자들을 포함하고 있으면 <code>true</code> 리턴 |
| <code>int length()</code> | 스트링의 길이(문자 개수) 리턴 |
| <code>String replace(CharSequence target, CharSequence replacement)</code> | <code>target</code> 이 지정하는 일련의 문자들을 <code>replacement</code> 가 지정하는 문자들로 변경한 스트링 리턴 |
| <code>String[] split(String regex)</code> | 정규식 <code>regex</code> 에 일치하는 부분을 중심으로 스트링을 분리하고 분리된 스트링을 배열에 저장하여 리턴 |
| <code>String substring(int beginIndex)</code> | <code>beginIndex</code> 인덱스부터 시작하는 서브 스트링 리턴 |
| <code>String toLowerCase()</code> | 소문자로 변경한 스트링 리턴 |
| <code>String toUpperCase()</code> | 대문자로 변경한 스트링 리턴 |
| <code>String trim()</code> | 스트링 앞뒤의 공백 문자들을 제거한 스트링 리턴 |

String 활용

40

▣ 스트링 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++

▣ 공백 제거, String trim()

- 키보드나 파일로부터 스트링을 입력 시, 스트링 앞 뒤 공백이 끼는 경우가 많다. -> trim()을 이용하면 스트링 앞 뒤에 있는 공백 제거

```
String a = " xyzWt";  
String b = a.trim();    // b = "xyz". 빈 칸과 'Wt' 제거됨
```


예제 6-6 : String을 활용하여 문자열 다루기

41

```
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);  
    }  
}
```

3

true

a = " C#, C++ "

a = "C#,C++"

a = "Java,C++"

s[0] = "Java"
s[1] = "C++"

a = "C++"

+

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

StringBuffer 클래스

42

- ▣ 가변 스트링을 다루는 클래스
- ▣ StringBuffer 객체 생성

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

- ▣ String 클래스와 달리 문자열 변경 가능
 - 가변 크기의 버퍼를 가지고 있어 문자열 수정 가능
 - 문자열의 수정이 많은 작업에 적합

- ▣ 스트링 조작 사례

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

sb.append(" is pencil.");           // sb = "This is pencil."
sb.insert(7, " my");               // sb = "This is my pencil."
sb.replace(8, 10, "your");         // sb = "This is your pencil."
System.out.println(sb);           // "This is your pencil." 출력
```

StringTokenizer 클래스

43

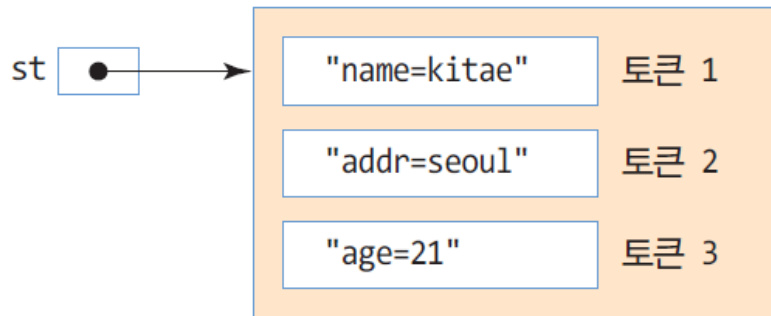
- 구분 문자를 기준으로 문자열을 분리하는 클래스
 - 구분 문자(delimiter) : 문자열을 구분할 때 사용되는 문자
 - 토큰(token) : 구분 문자로 분리된 문자열

예)

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

구분 문자 '&'

st



StringTokenizer 객체

```
int count = st.countTokens();
```

토큰 개수 알아내기. count = 3

```
String token = st.nextToken();
```

다음 토큰 얻어내기.
st = "name=kitae"

StringTokenizer로 문자열 분리 사례

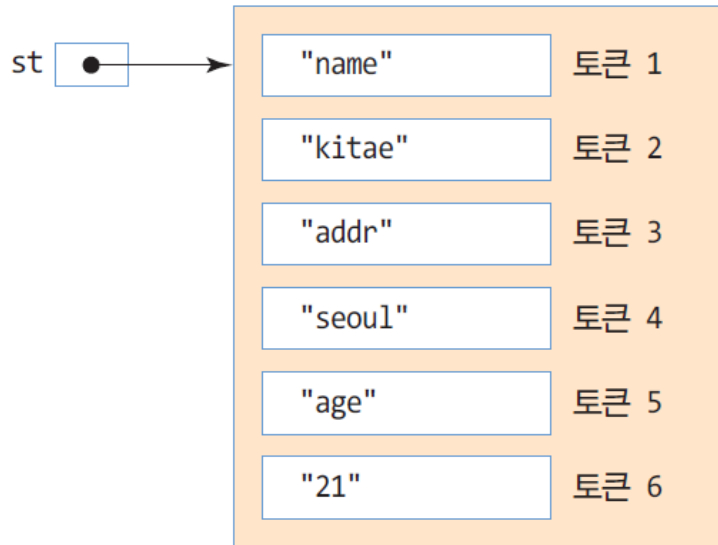
44

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

주목

구문 문자는 2개
'&'와 '='

st



StringTokenizer 객체

int count = st.countTokens();

토큰 개수 알아내기. count = 6

String token = st.nextToken();

다음 토큰 얻어내기.
st = "name"

예제 6-7 : StringTokenizer를 이용한 문자열 분리

45

"name=kitae&addr=seoul&age=21"를 '&'문자를 기준으로 분리하는 코드를 작성하라.

```
import java.util.StringTokenizer;
public class StringTokenizerEx {
    public static void main(String[] args) {
        String query = "name=kitae&addr=seoul&age=21";
        StringTokenizer st = new StringTokenizer(query, "&");

        int n = st.countTokens();           // 분리된 토큰 개수
        System.out.println("토큰 개수 = " + n);

        while(st.hasMoreTokens()) {
            String token = st.nextToken();    // 토큰 얻기
            System.out.println(token);          // 토큰 출력
        }
    }
}
```

```
토큰 개수 = 3
name=kitae
addr=seoul
age=21
```

Math 클래스

46

- ▣ 기본 산술 연산 메소드를 제공하는 클래스
- ▣ 모든 메소드는 static으로 선언
 - 클래스 이름으로 호출 가능
- ▣ Math.random() 메소드로 난수 발생
 - random()은 0보다 크거나 같고 1.0보다 작은 실수 난수 발생
 - 1에서 100까지의 랜덤 정수 10개를 발생시키는 코드 사례

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

* java.util.Random 클래스를 이용하여 난수 발생 가능

```
Random r = new Random();  
int n = r.nextInt(); // 음수, 양수, 0 포함, 자바의 정수 범위 난수 발생  
int m = r.nextInt(100); // 0에서 99 사이(0과 99 포함)의 정수 난수 발생
```

예제 6-8 : Math 클래스 활용

47

Math 클래스의 메소드 활용 예를 보인다.

```
public class MathEx {  
    public static void main(String[] args) {  
        System.out.println(Math.abs(-3.14));           // 절댓값 구하기  
        System.out.println(Math.sqrt(9.0));             // 9의 제곱근 = 3  
        System.out.println(Math.exp(2));                // e2  
        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) + " "); // 난수 발생  
    }  
}
```

3.14

3.0

7.38905609893065

3

이번주 행운의 번호는 14 44 21 36 17