

제6장 패키지

2018년1학기

컴퓨터공학과 김 명 교수

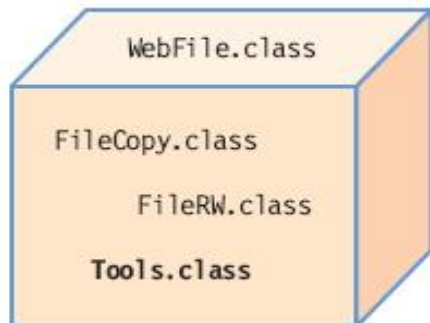
패키지 개념과 필요성

2

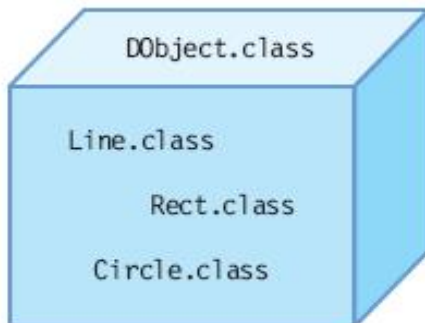
3명이 분담하여 자바 응용프로그램을 개발하는 경우



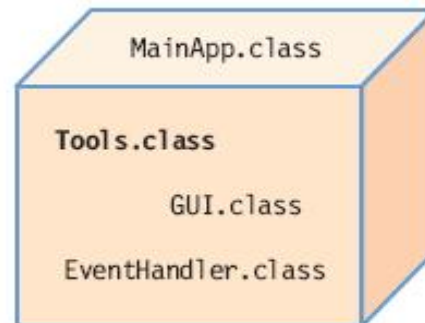
FileIO 작업



Graphic 작업

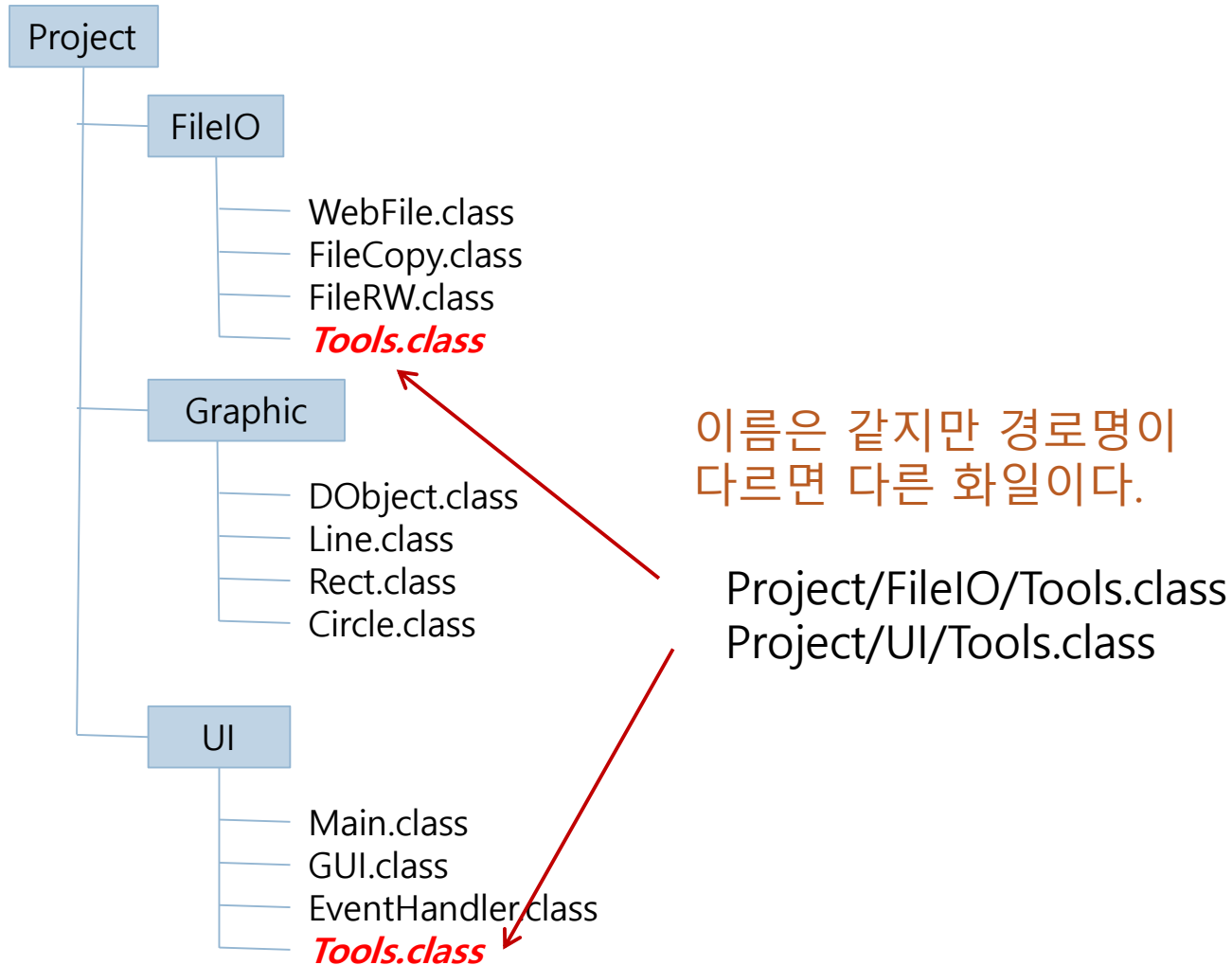


UI 작업



패키지를 사용한 폴더 관리 예제

3



자바의 패키지 (package)

4

- 패키지
 - ▣ 서로 관련된 클래스와 인터페이스의 컴파일된 파일들을 하나의 폴더에 묶어 놓은 것을 말한다.
- 하나의 응용프로그램 (프로젝트) 은 하나의 패키지로 구성될 수도 있고, 여러 개의 패키지로 구성될 수도 있다.

패키지 사용하기, import문 (1/2)

5

□ 다른 패키지 가져다 쓰기

▣ import를 이용하지 않는 경우


- 소스 내에서 매번 전체 패키지 이름과 클래스명을 써주어야 함

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

▣ import 키워드 이용하는 경우

- 소스의 시작 부분에 패키지를 명시하고, 소스 내에서는 클래스 명만 명시.

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



패키지 사용하기, import문 (2/2)

6

- import 사용
 - ▣ 특정 클래스의 경로명만 포함하는 경우
 - import java.util.Scanner;
 - ▣ 패키지 내의 모든 클래스를 포함시키는 경우
 - import java.util.*;
 - *는 현재 패키지 내의 클래스만을 의미하며 하위 패키지의 클래스까지 포함하지 않는다.

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

패키지의 특징

7

□ 패키지 계층구조

- 클래스나 인터페이스가 너무 많아지면 관리의 어려움
- 관련된 클래스 파일들을 하나의 패키지로 계층화하여 관리

□ 패키지별 접근 제한

- default로 선언된 클래스나 멤버는 동일 패키지 내의 클래스들이 자유롭게 접근하도록 허용

□ 동일한 이름의 클래스와 인터페이스의 사용 가능

- 서로 다른 패키지에 "이름이 같은" 클래스와 인터페이스 존재 가능

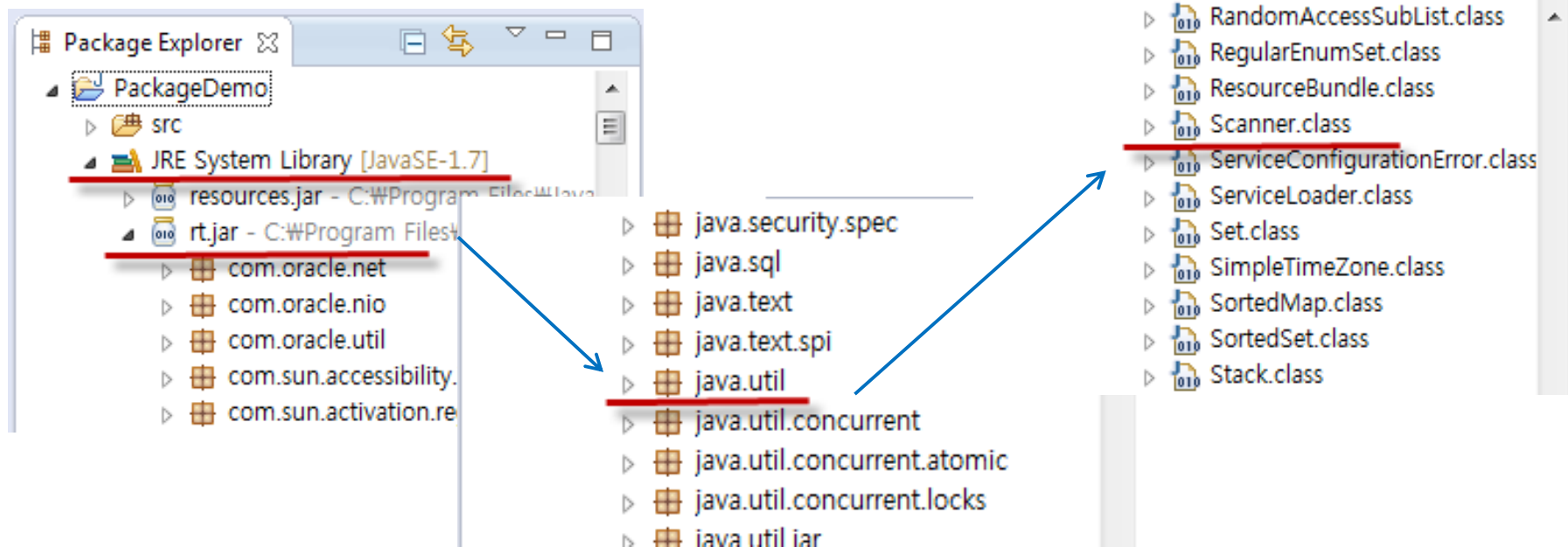
□ 높은 소프트웨어 재사용성

- 오라클에서 제공하는 자바 API는 패키지로 구성되어 있음
- java.lang, java.io 등의 패키지 사용으로 입출력을 간단히 작성할 수 있음

자바 JDK의 패키지 구조

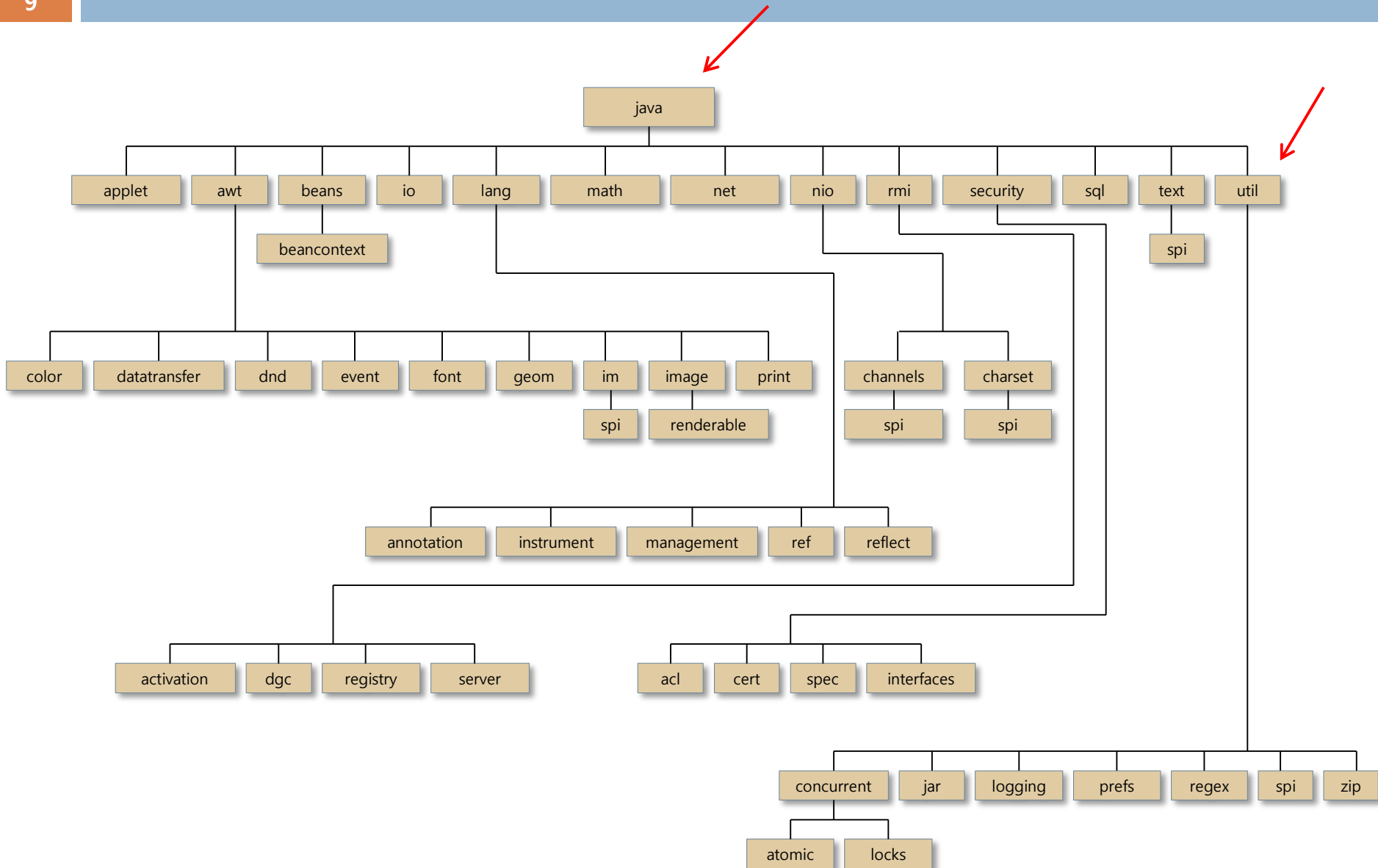
8

- JDK 패키지
 - ▣ 자바에서는 관련된 클래스들을 표준 패키지로 묶어 사용자에게 제공
 - ▣ 자바에서 제공하는 패키지는 C언어의 표준 라이브러리와 유사
 - ▣ JDK의 표준 패키지는 rt.jar에 담겨 있다.
 - C:\Program Files\Java\jdk1.7.0_51\jre\lib\rt.jar



자바 패키지 구조

9



주요 패키지

10

- java.lang : 자바 language 패키지
 - ▣ 스트링, 수학 함수, 입출력 등 기본적인 클래스와 인터페이스
 - ▣ 자동으로 import 됨 - import 문이 필요 없음
- java.util : 자바 유틸리티 패키지
 - ▣ 날짜, 시간, 벡터, 해시 테이블 등과 같은 다양한 유틸리티 클래스와 인터페이스
- java.io : 자바 입출력 관련 패키지
 - ▣ 키보드, 모니터, 프린터, 디스크 등에 입출력을 할 수 있는 클래스와 인터페이스

자바 API 참조 :

<http://docs.oracle.com/javase/8/docs/api/>

11

The screenshot shows a web browser window displaying the Java API documentation for the `Scanner` class. The browser's address bar shows the URL `http://docs.oracle.com/javase/8/docs/api/`. The page has a navigation bar with tabs for `OVERVIEW`, `PACKAGE`, `CLASS` (which is highlighted), `USE`, `TREE`, `DEPRECATED`, `INDEX`, and `HELP`. Below the navigation bar, there are links for `PREV CLASS`, `NEXT CLASS`, `FRAMES`, and `NO FRAMES`. The main content area is titled `Class Scanner` and shows the inheritance hierarchy: `java.lang.Object` and `java.util.Scanner`. It also lists the `All Implemented Interfaces`: `Closeable`, `AutoCloseable`, and `Iterator<String>`. The class signature is `public final class Scanner`, which `extends Object` and `implements Iterator<String>, Closeable`. A description states: "A simple text scanner which can parse primitive types and strings using regular expressions." Another line of text at the bottom says: "A Scanner breaks its input into tokens using a delimiter pattern, which by". On the left side, there is a search bar and a list of classes, with `Scanner` highlighted. A red arrow points to the `Scanner` class in the left sidebar, and another red arrow points to the `CLASS` tab in the navigation bar.

java.time.temporal
java.time.zone
java.util
java.util.concurrent
java.util.concurrent.at
java.util.concurrent.lo
java.util.function

Random
ResourceBundle
ResourceBundle.Cont
Scanner
ServiceLoader
SimpleTimeZone
Spliterators
Spliterators.AbstractD
Spliterators.AbstractI
Spliterators.AbstractL
Spliterators.AbstractS
SplittableRandom
Stack
StringJoiner
StringTokenizer
Timer
TimerTask
TimeZone
TreeMap

OVERVIEW PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3
java.util

Class Scanner

java.lang.Object
java.util.Scanner

All Implemented Interfaces:
Closeable, AutoCloseable, Iterator<String>

public final class Scanner
extends Object
implements Iterator<String>, Closeable

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by

Object 클래스와 주요 메소드

12

- ▣ java.lang 패키지에 포함, 자바 클래스 계층 구조의 최상위에 위치
- ▣ 모든 클래스의 수퍼 클래스

메소드	설명
protected Object clone()	현 객체와 똑같은 객체를 만들어 반환. 오버라이딩 필요
boolean equals(Object obj)	obj가 가리키는 객체와 현재 객체를 비교하여 같으면 true 반환
Class getClass()	현 객체의 런타임 클래스를 반환
int hashCode()	현 객체에 대한 해쉬 코드 값 반환
String toString()	현 객체에 대한 스트링 표현을 반환
void notify()	현 객체에 대해 대기하고 있는 하나의 스레드를 깨운다.
void notifyAll()	현 객체에 대해 대기하고 있는 모든 스레드를 깨운다.
void wait()	현 객체의 다른 스레드가 notify() 또는 notifyAll() 메소드를 호출할 때까지 현 스레드를 대기하게 한다.

객체의 속성

13

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

```
public class ObjectProperty {  
    public static void main(String[] args) {  
        Point p = new Point(1, 2);  
        System.out.println(p);  
        System.out.println(p.toString());  
        System.out.println(p.getClass());  
        System.out.println(p.getClass().getName());  
        System.out.println(p.hashCode());  
    }  
}
```

실행결과

Point@659e0bfd
Point@659e0bfd
class Point
Point
1704856573

객체를 문자열로 변환

14

□ String toString()

- 객체를 텍스트 형태로 표현한 문자열로 반환
- 반환되는 문자열 : 클래스 이름@객체의 hash code
- 객체와 문자열이 + 연산이 되는 경우 객체의 toString() 메소드를 호출

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

변환

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

Point@c17164점

새로운 toString() 만들기

15

```
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 ObjectProperty {
    public static void main(String [] args) {
        Point a = new Point(2,3);
        System.out.println(a.toString());
    }
}
```

실행결과

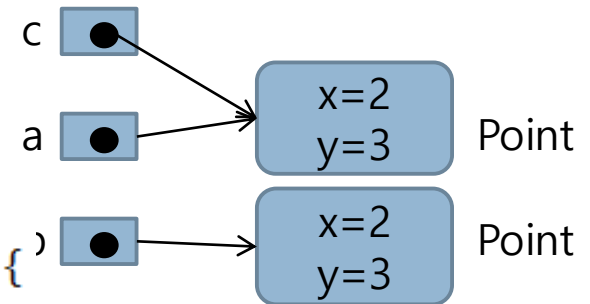
Point(2,3)

객체 비교 (1/2)

16

- 객체 레퍼런스의 동일성 비교 : == 연산자 이용
- 객체 내용의 동일성 비교 : boolean equals(Object obj) 이용

```
class Point {  
    int x, y;  
    public Point(int x, int y) {  
        this.x = x; this.y = y;  
    }  
}  
  
public class ObjectProperty {  
    public static void main(String [] args) {  
        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");  
    }  
}
```



실행결과

a==c

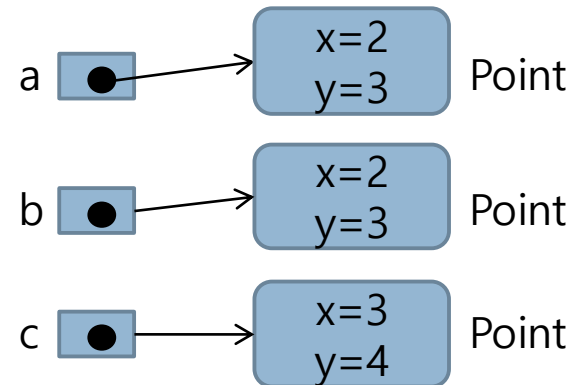
객체 비교 (2/2)

17

```
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;
    }
}

public class ObjectProperty {
    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) Rect 클래스 만들고 equals() 만들기

18

- int 타입의 width, height의 필드를 가지는 Rect 클래스를 작성하고, 두 Rect 객체의 면적이 같으면 두 객체가 같은 것으로 판별하도록 equals()를 작성하시오.

```
class Rect {  
    int width;  
    int 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;  
    }  
}
```

(예제) Rect 클래스 만들고 equals() 만들기

19

```
public class EqualsDemo {  
    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");  
    }  
}
```

실행결과

a is equal to b

Wrapper 클래스

20

- 자바 기본 데이터 타입을 클래스화한 8개 클래스

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

- 용도
 - ▣ 기본 데이터 타입을 사용할 수 없고 객체만 사용하는 컬렉션에 기본 데이터 타입을 Wrapper 클래스로 만들어 사용

Wrapper 객체 생성

21

- 기본 데이터 값을 생성자 인자로 하는 경우

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

- 해당 데이터 값을 나타내는 문자열을 생성자 인자로 하는 경우

- ▣ Boolean, Short, Byte, Integer, Long, Double, Float 경우

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

- Float는 double 타입의 값을 생성자의 인자로 사용 가능

```
Float f = new Float((double) 3.14);
```

Integer 클래스의 주요 메소드

22

메소드	설명
<code>static int bitCount(int i)</code>	이진수 표현에서 1의 개수를 반환
<code>float floatValue()</code>	float 타입으로 변환된 값 반환
<code>int intValue()</code>	int 타입으로 변환된 값 반환
<code>long longValue()</code>	long 타입으로 변환된 값 반환
<code>short shortValue()</code>	short 타입으로 변환된 값 반환
<code>static int parseInt(String s)</code>	스트링을 10진 정수로 변환된 값 반환
<code>static int parseInt(String s, int radix)</code>	스트링을 지정된 진법의 정수로 변환된 값 반환
<code>static String toBinaryString(int i)</code>	이진수 표현으로 변환된 스트링 반환
<code>static String toHexString(int i)</code>	16진수 표현으로 변환된 스트링 반환
<code>static String toOctalString(int i)</code>	8진수 표현으로 변환된 스트링 반환
<code>static String toString(int i)</code>	정수를 스트링으로 변환하여 반환

Wrapper 활용 (1/2)

23

- Wrapper 객체로부터 기본 데이터 타입 알아내기

```
public class WrapperClassDemo1 {  
    public static void main(String[] args) {  
        Integer i = new Integer(10);  
        int ii = i.intValue();  
        Character c = new Character('c');  
        char cc = c.charValue();  
        Float f = new Float(3.14);  
        float ff = f.floatValue();  
        Boolean b = new Boolean(true);  
        boolean bb = b.booleanValue();  
  
        System.out.println(ii);  
        System.out.println(cc);  
        System.out.println(ff);  
        System.out.println(bb);  
    }  
}
```

실행결과

10
c
3.14
true

Wrapper 활용 (2/2)

24

- 문자열과 기본 데이터 타입 사이의 변환

```
public class WrapperClassDemo1 {  
    public static void main(String[] args) {  
        int i = Integer.parseInt("123");  
        boolean b = Boolean.parseBoolean("true");  
        float f = Float.parseFloat("3.14");  
  
        System.out.println(i); System.out.println(f);  
        System.out.println(b); System.out.println();  
  
        String s1 = Integer.toString(123);  
        String s2 = Integer.toHexString(123);  
        String s3 = Float.toString(3.14f);  
        String s4 = Character.toString('a');  
        String s5 = Boolean.toString(true);  
  
        System.out.println(s1); System.out.println(s2);  
        System.out.println(s3); System.out.println(s4);  
        System.out.println(s5);  
    }  
}
```

실행결과

```
123  
3.14  
true  
  
123  
7b  
3.14  
a  
true
```


Wrapper 클래스의 활용

25

다음은 Wrapper 클래스를 활용하는 예이다. 다음 프로그램의 결과는 무엇인가?

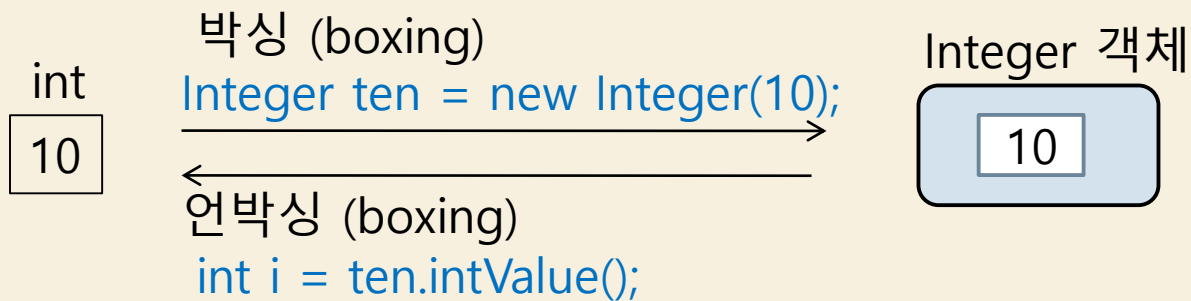
```
public class WrapperClassDemo {  
    public static void main(String[] args) {  
        Integer i = new Integer(10);  
        char c = '4';  
        Double d = new Double(3.1234566);  
        System.out.println(Character.toLowerCase('A'));  
  
        if (Character.isDigit(c))  
            System.out.println(Character.getNumericValue(c));  
  
        System.out.println(Integer.parseInt("-123"));  
        System.out.println(Integer.toBinaryString(28));  
        System.out.println(Integer.toHexString(28));  
        System.out.println(i.doubleValue());  
        System.out.println(d.toString());  
        System.out.println(Double.parseDouble("44.13e-6"));  
    }  
}
```

```
a  
4  
-123  
16  
11100  
3  
1c  
10.0  
3.1234566  
4.413E-5
```

박싱과 언박싱

26

- 박싱(boxing)
 - ▣ 기본 데이터 타입을 Wrapper 클래스로 변환하는 것
- 언박싱(unboxing)
 - ▣ 반대의 경우를 언박싱이라고 한다.



Auto boxing & unboxing

27

- JDK 5.0 이상부터 지원
- Auto boxing
 - ▣ 기본 데이터 타입을 자동으로 Wrapper 클래스로 변환
- Auto unboxing
 - ▣ Wrapper 클래스를 자동으로 기본 데이터 타입으로 변환

```
// JDK5.0이상에서  
Integer ten = 10; // 자동 박싱  
int i = ten; // 자동 언박싱
```

박싱 언박싱의 예

28

다음 코드에 대한 결과는 무엇인가?

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

실행결과

```
intObject = 10  
i = 20
```

String의 생성과 특징

29

- String - java.lang.String
 - ▣ String 클래스는 하나의 스트링을 표현한다

```
public class StringDemo1 {  
    public static void main(String[] args) {  
        // 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");  
        System.out.println(str1);  
        System.out.println(str2 + ", " + str3);  
    }  
}
```

실행결과

```
abcd  
abcd, abcd
```

String의 생성과 특징

30

□ String 생성자

생성자	설명
String()	빈 스트링 객체 생성
String(byte[] bytes)	플랫폼의 기본 문자 집합을 이용하여 바이트 배열을 디코딩하여 스트링 객체 생성
String(String original)	인자로 주어진 스트링과 똑같은 스트링 객체를 생성
String(StringBuffer buffer)	스트링 버퍼에 포함된 일련의 문자들을 나타내는 스트링 객체 생성

스트링 리터럴과 new String()

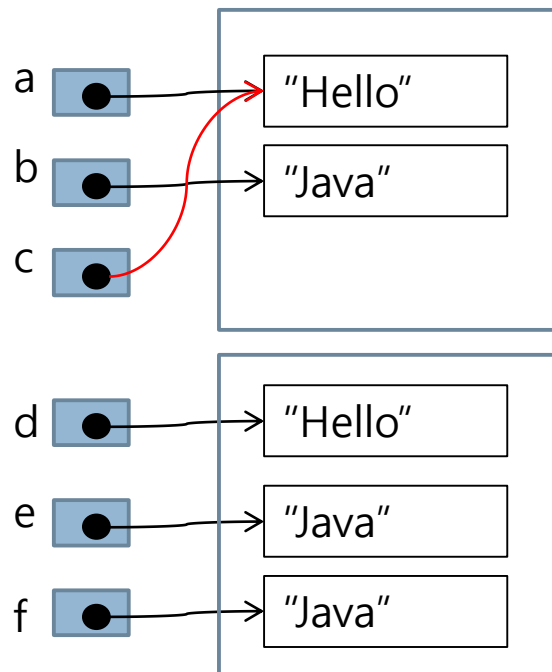
31

□ 스트링 생성

- 단순 리터럴로 생성, String s = "Hello";
 - JVM이 리터럴 관리, 응용프로그램 내에서 공유됨
- String 객체로 생성, String t = new String("Hello");
 - 힙에 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");
```



JVM의
스트링 리터럴 테이블

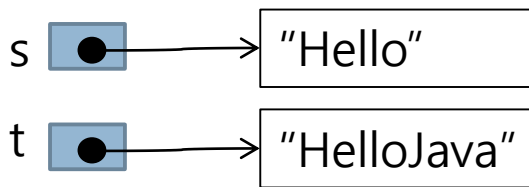
힙 메모리

스트링 객체의 주요 특징

32

- 스트링 객체는 수정할 수 없다.

```
String s = new String("Hello");  
String t = s.concat("Java");
```



s.concat("Java")의 실행 결과
스트링 s는 변경되지 않음

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

주요 메소드

33

메소드	설명
char charAt(int index)	지정된 인덱스에 있는 문자값을 반환
int indexOf(int ch)	ch 문자가 있는 인덱스 리턴. 없으면 -1리턴
int indexOf(int ch, int fromIndex)	fromIndex 위치부터 끝까지 문자 ch 탐색. 인덱스 리턴. 없으면 -1리턴
String concat(String str)	지정된 스트링을 현재 스트링 뒤에 덧붙인 스트링 반환
boolean contains(CharSequence s)	지정된 일련의 문자들을 포함하고 있으면 true 반환
int length()	스트링의 길이 반환
String replace(CharSequence target, CharSequence replacement)	target 지정하는 일련의 문자들을 replacement가 지정하는 문자들로 변경한 스트링 반환
String[] split(String regex)	정규식에 일치하는 부분을 중심으로 스트링 분리하여 스트링 배열로 반환
String subString(int beginIndex)	지정된 인덱스부터 시작하는 서브 스트링 반환
String toLowerCase()	스트링을 소문자로 변경한 스트링 반환
String toUpperCase()	스트링을 대문자로 변경한 스트링 반환
String trim()	스트링 앞뒤의 공백문자들을 제거한 스트링 반환

문자열 비교

34

- int compareTo(String anotherString)

```
public class StringDemo1 {  
    public static void main(String[] args) {  
        String a = "a";  
        String b = "b";  
        String c = "c";  
  
        System.out.println(a.compareTo(b));  
        System.out.println(b.compareTo(b));  
        System.out.println(c.compareTo(b));  
    }  
}
```

실행결과

-1
0
1

(예)	apple	apricot	banana	cherry	grape	orange
	15	20	50	75	90	120

문자열 연결

35

- ▣ 자바에서 스트링 연결 연산자 + 지원
 - + 연산에서 문자열이 인자로 포함되어 있으면 산술 연산이 아닌 문자열 연결 연산으로 간주.
- ▣ 객체가 문자열 연결 연산에 포함되어 있는 경우
 - toString() 메소드를 호출하여 객체를 문자열로 변환한 후 문자열 연결
- ▣ 기본 데이터 타입
 - 그대로 문자열로 변환된 후에 문자열 연결

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

문자열 연결

36

- String 클래스의 concat() 메소드를 이용하여 스트링 연결
문법: String concat(String str)

```
public class StringDemo1 {  
    public static void main(String[] args) {  
        String s1 = "abcd".concat("-efg");  
        String s2 = s1.concat("-opqr");  
        System.out.println(s1);  
        System.out.println(s2);  
    }  
}
```

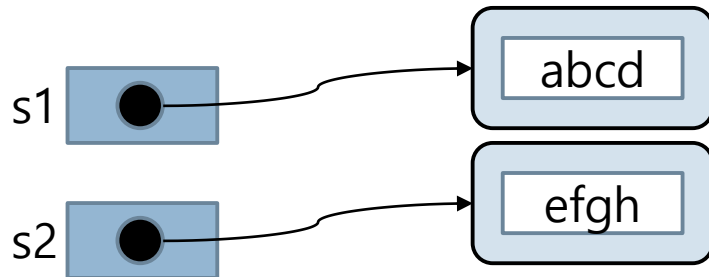
실행결과

```
abcd-efg  
abcd-efg-opqr
```

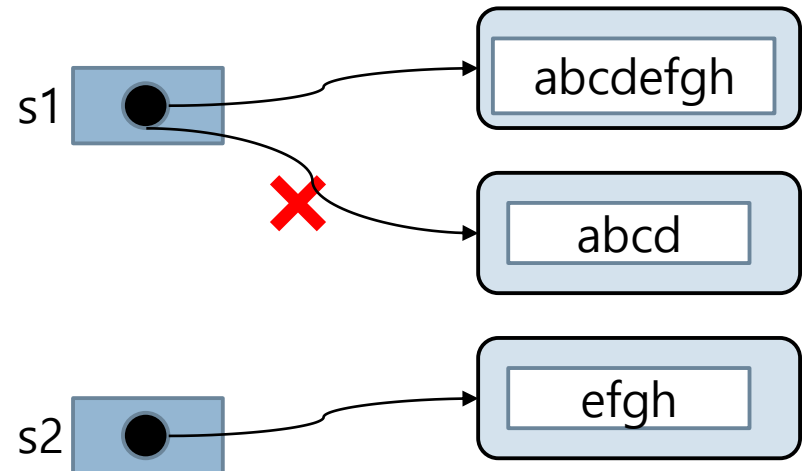
concat()은 새로운 문자열을 생성

37

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



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



문자열 내의 공백 제거, 문자열의 각 문자 접근

38

□ 공백 제거

▣ String trim()

- 스트링 앞 뒤 공백 문자(tab, enter, space) 제거한 스트링 리턴

```
String a = " abcd def ";  
String b = "\txyz\t";  
String c = a.trim(); // c = "abcd def"  
String d = b.trim(); // d = "xyz"
```

문자열 내의 공백 제거, 문자열의 각 문자 접근

39

□ 문자열의 문자

▣ char charAt(int index)

■ 스트링에 포함된 각 문자 접근

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

// “class”에 몇 개의 ‘s’가 포함되었는지 알아내는 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
```

String 클래스 메소드 활용

40

```
public class StringDemo {  
    public static void main(String[] args) {  
  
        String a = new String(" abcd");  
        String b = new String(",efg");  
  
        // 문자열 연결  
        a = a.concat(b);  
        System.out.println(a);  
  
        // 공백 제거  
        a = a.trim();  
        System.out.println(a);  
  
        // 문자열 대치  
        a = a.replace("ab", "12");  
        System.out.println(a);  
    }  
}
```

실행결과

```
abcd,efg  
abcd,efg  
12cd,efg  
분리된 0번 문자열: 12cd  
분리된 1번 문자열: efg  
d,efg  
e
```


String 클래스 메소드 활용

41

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

// 서브 스트링
a = a.substring(3);
System.out.println(a);

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

실행결과

```
abcd,efg
abcd,efg
12cd,efg
분리된 0번 문자열: 12cd
분리된 1번 문자열: efg
d,efg
e
```

예제 실행 과정

42

```
a = new String(" abcd");
```

```
b = new String(",efg");
```

```
a = a.concat(b);
```

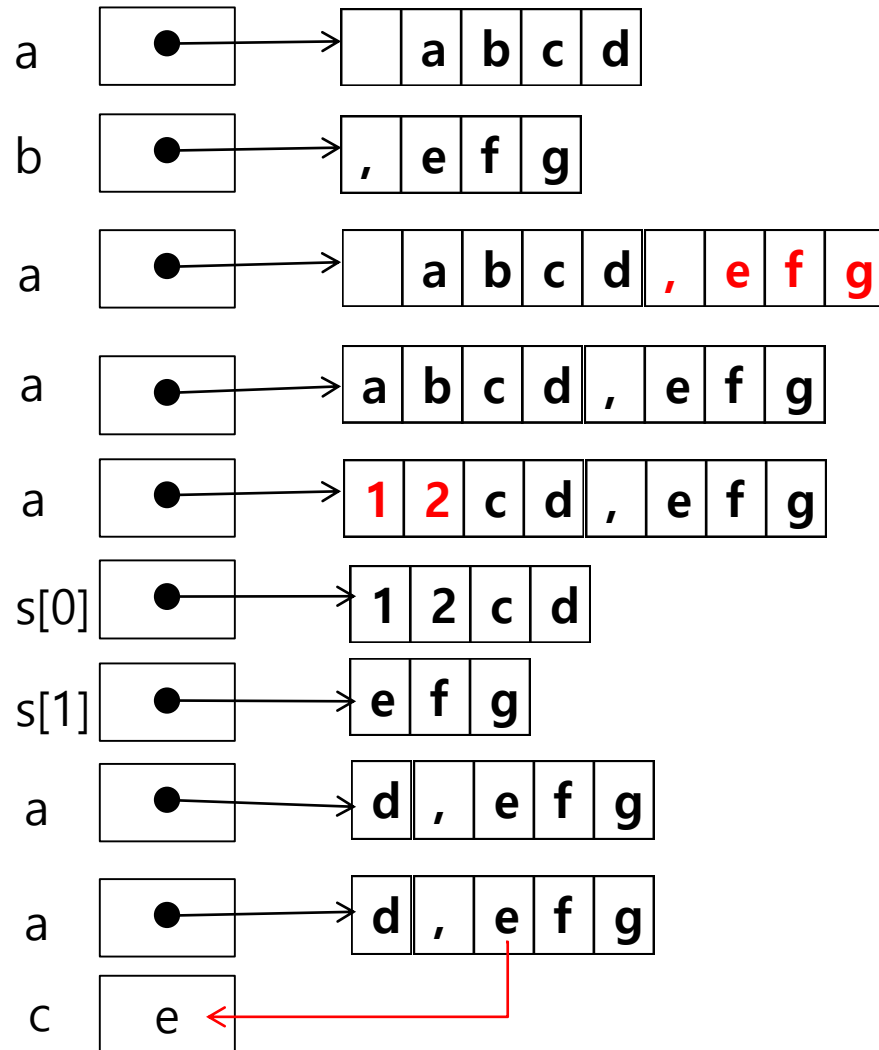
```
a = a.trim();
```

```
a = a.replace("ab","12");
```

```
String s[] = a.split(",");
```

```
a = a.substring(3);
```

```
char c = a.charAt(2);
```



StringBuffer 클래스

43

- java.lang.StringBuffer
 - ▣ String클래스 객체와는 달리 객체 생성 후 스트링 값 변경 가능
 - ▣ append와 insert 메소드를 통해 스트링 조작
 - ▣ StringBuffer 객체의 크기는 스트링 길이에 따라 가변적

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

StringBuffer 클래스의 생성자

44

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

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

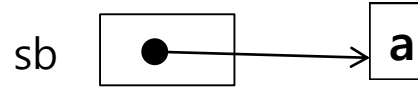
주요 메소드

45

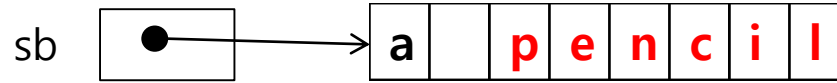
메소드	설명
StringBuffer append(String str)	지정된 스트링을 스트링 버퍼에 덧붙인다.
StringBuffer append(StringBuffer sb)	지정된 스트링 버퍼를 스트링 버퍼에 덧붙인다.
int capacity()	현재 스트링 버퍼의 크기 반환
StringBuffer delete(int start, int end)	지정된 서브 스트링을 스트링 버퍼에서 제거
StringBuffer insert(int offset, String str)	지정된 스트링을 스트링 버퍼의 특정 위치에 삽입
StringBuffer replace(int start, int end, String str)	스트링 버퍼 내의 서브 스트링을 지정된 스트링으로 대체
StringBuffer reverse()	스트링 버퍼 내의 문자들을 반대 순서로 변경
void setLength()	스트링 버퍼 내 저장된 문자열 길이를 설정. 현재 길이보다 큰 경우 널 문자로 채우며 작은 경우는 문자열이 잘린다.

StringBuffer의 메소드 활용 예

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



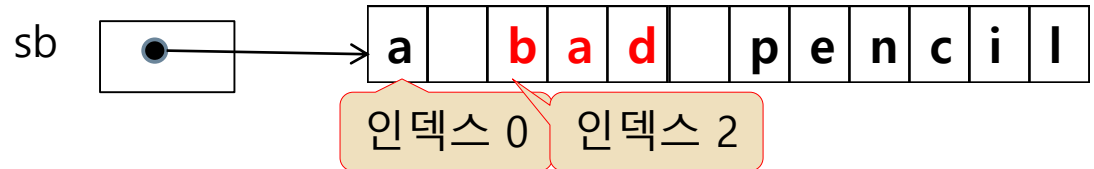
```
sb.append(" pencil");
```



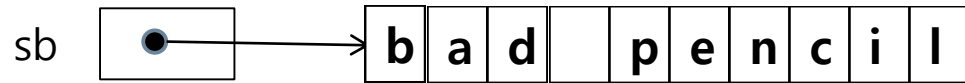
```
sb.insert(2, "nice ");
```



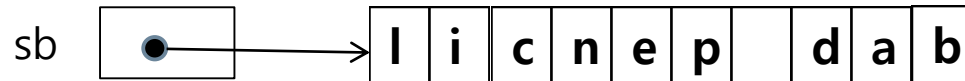
```
sb.replace(2, 6, "bad");
```



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



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

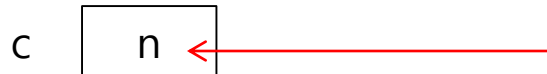


```
int n = sb.length();
```

n = 10



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



StringBuffer 클래스 메소드 활용

47

```
public class StringBufferDemo {  
    public static void main(String[] args) {  
        StringBuffer sb = new StringBuffer("This");  
        System.out.println(sb.hashCode());  
        sb.append(" is pencil"); // 문자열 덧붙이기  
        System.out.println(sb);  
  
        sb.insert(7, " my"); // 문자열 삽입  
        System.out.println(sb);  
  
        sb.replace(8, 10, "your"); // 문자열 대체  
        System.out.println(sb);  
  
        sb.setLength(5); // 스트링 버퍼 내 문자열 길이 설정  
  
        System.out.println(sb);  
        System.out.println(sb.hashCode());  
    }  
}
```

14576877
This is pencil
This is my pencil
This is your pencil
This
14576877

StringTokenizer 클래스

48

□ java.util.StringTokenizer

- 하나의 스트링을 구분자로 분리하여 토큰 형태로 파싱

- 스트링을 구분할 때 사용되는 문자들을 구분 문자(delimiter)라고 함

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

- 위의 예에서 '&'가 구분 문자

- 토큰(token)

- 구분 문자로 분리된 스트링

- String 클래스의 split 메소드를 이용하여 동일한 구현 가능

생성자와 주요 메소드

49

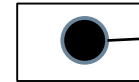
생성자	설명
<code>StringTokenizer(String str)</code>	지정된 스트링으로 초기화된 스트링 토큰라이저 생성
<code>StringTokenizer(String str, String delim)</code>	지정된 스트링과 구분 문자로 초기화된 스트링 토큰라이저 생성
<code>StringTokenizer(String str, String delim, boolean returnDelims)</code>	지정된 스트링과 구분 문자로 초기화된 스트링 토큰라이저 생성. <code>returnDelims</code> 가 <code>true</code> 이면 구분 문자로 지정된 문자도 분리된 토큰에 포함된다.

메소드	설명
<code>int countTokens()</code>	스트링에 남아 토큰 수 반환
<code>boolean hasMoreTokens()</code>	스트링에 토큰이 남아 있으면 <code>true</code> 반환
<code>String nextToken()</code>	다음 토큰 반환
<code>String nextToken(String delim)</code>	지정된 분리자에 대한 다음 토큰 반환

StringTokenizer 객체 생성과 문자열 파싱

50

st



StringTokenizer 객체

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

"name=kitae"

토큰1

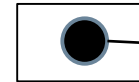
"addr=seoul"

토큰2

"age=21"

토큰3

st



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

"name"

토큰 1

"kitae"

토큰 2

"addr"

토큰 3

"seoul"

토큰 4

"age"

토큰 5

"21"

토큰 6

StringTokenizer 객체

StringTokenizer 클래스 메소드 활용

51

“홍길동/장화/홍련/콩쥐/팥쥐” 문자열을 “/”를 구분 문자로 하여
토큰을 분리하여 각 토큰을 출력하시오.

```
import java.util.StringTokenizer;

public class StringTokenizerDemo {
    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

52

- `java.lang.Math`
 - ▣ 기본적인 산술 연산을 수행하는 메소드 제공
 - ▣ 모든 멤버 메소드는 `static`으로 정의됨
 - ▣ 객체를 만들어서 사용할 필요 없음

주요 메소드

53

메소드	설명
static double abs(double a)	절대값 반환
static double cos(double a)	cosine 값 반환
static double sin(double a)	sine 값 반환
static double tan(double a)	tangent 값 반환
static double exp(double a)	e^a 값 반환
static double ceil(double a)	지정된 실수보다 크거나 같은 수 중에서 가장 작은 정수를 실수 타입으로 반환
static double floor(double a)	지정된 실수보다 작거나 같은 수 중에서 가장 큰 정수를 실수 타입으로 반환

주요 메소드

54

메소드	설명
<code>static double max(double a, double b)</code>	두 수 중에서 큰 수 반환
<code>static double min(double a, double b)</code>	두 수 중에서 작은 수 반환
<code>static double random()</code>	0.0보다 크거나 같고 1.0보다 작은 임의의 수 반환
<code>static double rint(double a)</code>	지정된 실수와 가장 근접한 정수를 실수 타입으로 반환
<code>static double round(double a)</code>	지정된 실수를 소수 첫째 자리에서 반올림한 정수를 실수 타입으로 반환
<code>static double sqrt(double a)</code>	제곱근을 반환

Math 클래스를 활용한 난수 발생

55

- 난수 발생 : static double random()
 - ▣ 0.0 이상 1.0 미만의 임의의 double 값을 반환.

```
for(int x=0; x<10; x++) {  
    // [0.0 ~ 99.9999] 실수 발생  
    double d = Math.random()*100;  
  
    // Math.round(d)는 d에 가장 가까운 정수를 리턴  
    int n = (int)(Math.round(d));    System.out.println(n);  
}
```

- ▣ 위의 코드에서 round() 메소드는 Math.round(55.3)은 55.0을 리턴하며,
Math.round(55.9)는 56.0을 리턴
- java.util의 Random 클래스를 이용하면 좀 더 다양한 형태로
난수 발생 가능

Math 클래스 메소드 활용

56

```
public class MathDemo {  
    public static void main(String[] args) {  
        double a = -2.78987434;  
  
        System.out.println(Math.abs(a));           // 절대값 구하기  
        System.out.println(Math.ceil(a));          // ceil  
        System.out.println(Math.floor(a));         // floor  
        System.out.println(Math.sqrt(9.0));        // 제곱근  
        System.out.println(Math.exp(1.5));         // exp  
        System.out.println(Math rint(3.141592));   // rint  
  
        // [1, 45] 사이의 난수 발생  
        System.out.print("이번주 행운의 번호는");  
        for (int i=0; i<5; i++)  
            System.out.print(Math.round(1 + Math.random() * 44) + " ");  
        System.out.println("입니다.");  
    }  
}
```

2.78987434

-2.0

-3.0

3.0

4.4816890703380645

3.0

이번주 행운의 번호는 35 42 18 31 33