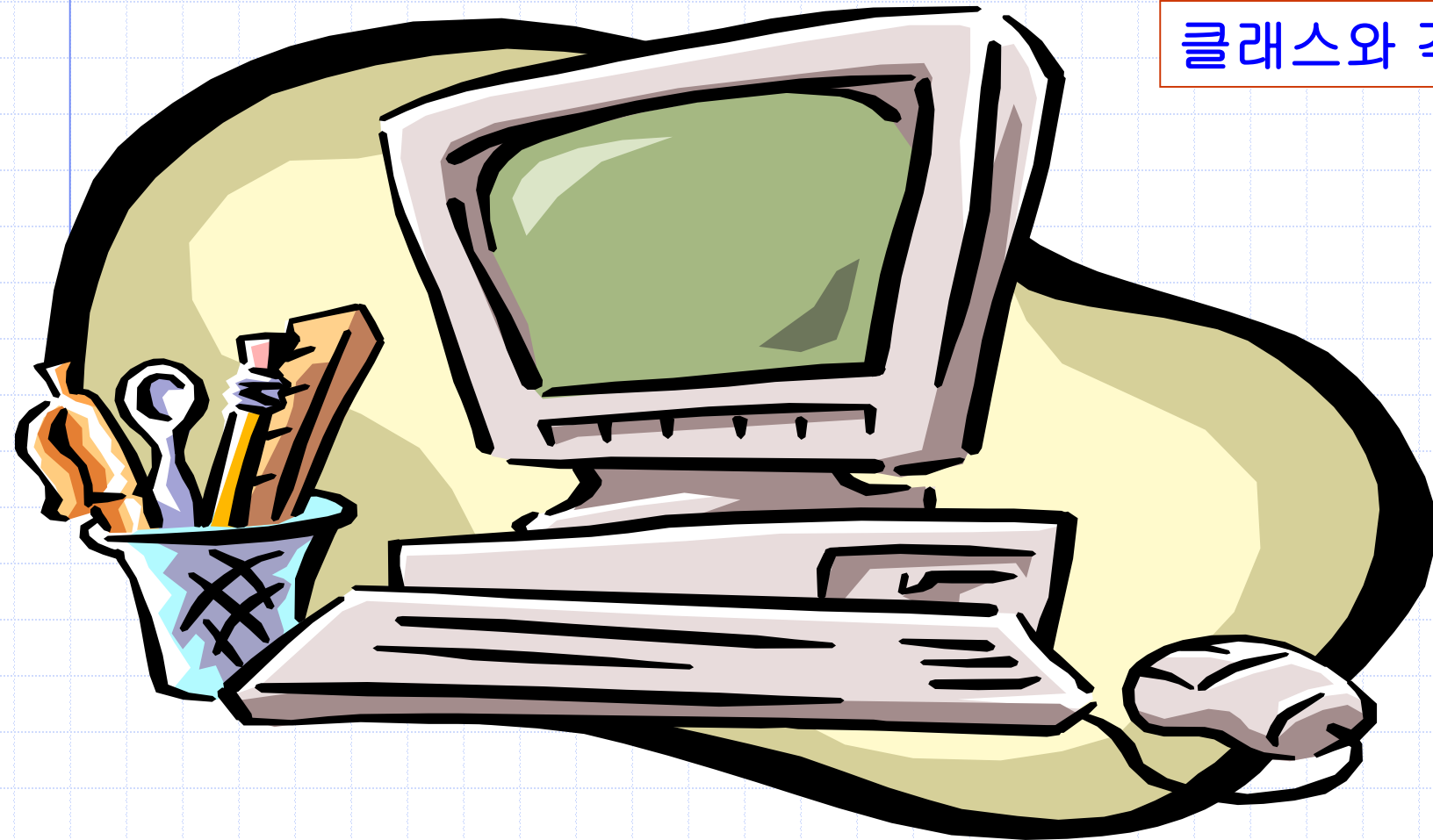
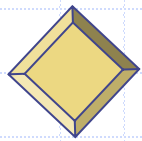


4장=2

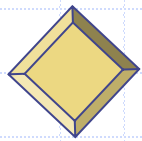
클래스와 객체





4장 클래스와 객체

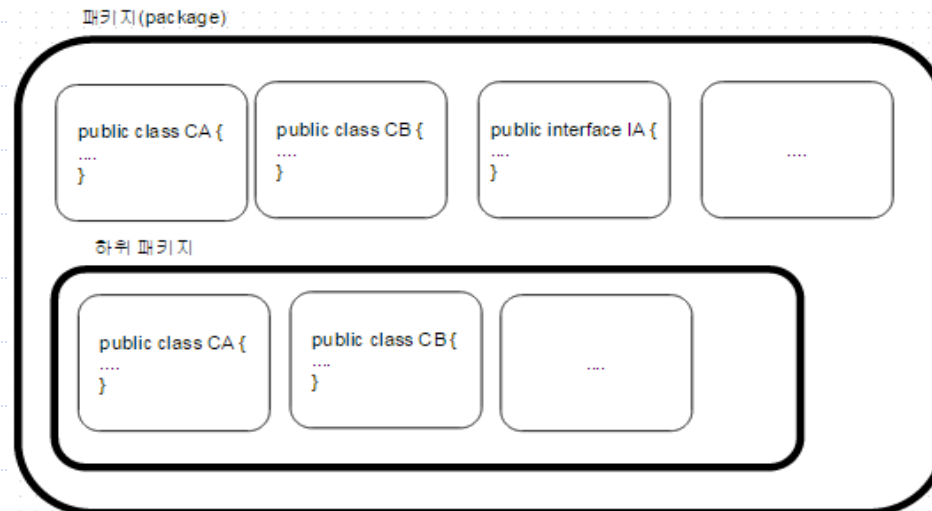
1. 객체지향 프로그래밍의 기본 개념
2. 객체지향 언어의 특징
3. 객체지향 프로그래밍
4. 객체 생성
5. 메소드
6. 가변인자(Varargs)
7. 메소드 중복
8. 패키지과 주요 클래스
9. 객체의 형 변환



8. 패키지 와 주요 클래스

1) 패키지

- ◆ 클래스들의 집합(**클래스 라이브러리**)
- ◆ **관련된 자바 클래스를 한 곳에 모아 놓은 것.**
- ◆ 사용자가 만든 패키지는 동일한 폴더에 들어가 있는 클래스 파일들의 집합이다.
- ◆ 클래스뿐 아니라 **인터페이스(interface)**와 **서브(하위) 패키지**를 구성원으로 갖고 있다.

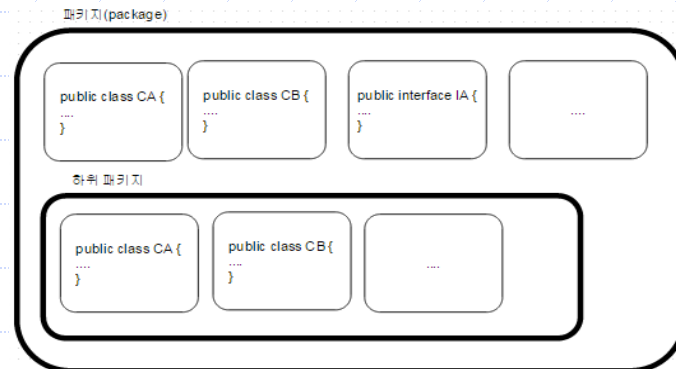


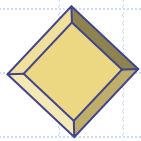
◆ 8. 패키지 와 주요 클래스

1) 패키지

◆ 패키지 사용의 장점

- 동일한 성격 또는 동일한 업무(예, 경리, 판매 등)의 연관된 클래스를 한 단위로 구성.
- 모든 클래스들이 함께 모여 있다면 클래스 이름, 메소드 이름 등이 서로 충돌하게 됨.
 - ◆ 서로 다른 패키지 안에 있는 클래스들 간에는 이름의 충돌이 생기지 않음.
- 패키지 단위로 참조 제어(접근 제어)를 할 수 있다.
 - ◆ public
 - ◆ protected
 - ◆ packaged(디폴트)
 - ◆ private



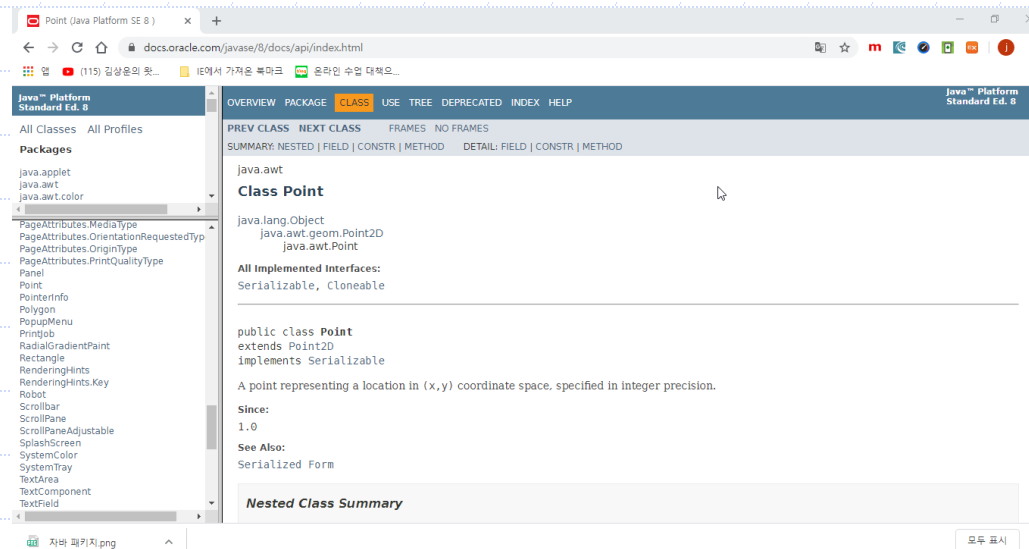


자바 API 문서

◆ 자바에서 제공하는 모든 클래스(인터페이스)에 대한 설명을 제공.

- 각각의 패키지과 패키지에 들어있는 클래스들을 확인 가능.
- 각 클래스에 있는 클래스 변수, 인스턴스 변수, 생성자, 메소드에 대한 상세한 설명.

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

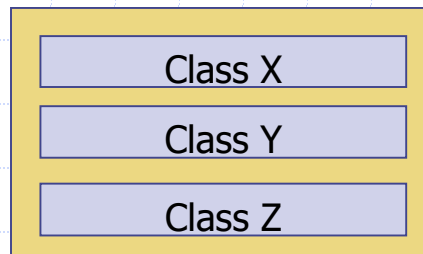


◆ 패키지 사용(import)

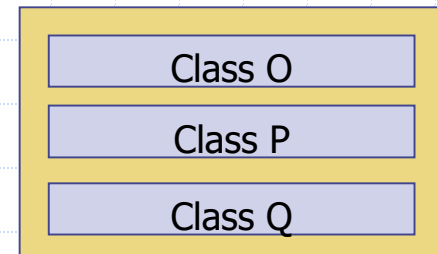
◆ 같은 패키지에서 다른 클래스를 사용하는 방법

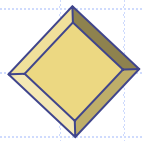
- (1) 클래스 자신이 속한 패키지에 있는 클래스는 클래스 이름만 사용하면 됨.
- (2) **java.lang**은 Object 클래스가 소속한 기본 패키지로 이 패키지 안에 있는 모든 클래스는 클래스 이름만 사용하여 쓸 수 있음. **(default)**
- (3) 위의 (1)(2)가 아닌 패키지에 있는 클래스를 사용할 때는 완전한 이름(full name)이 필요함(“**패키지이름.클래스이름**” 형식으로 사용).
- (4) 패키지 이름을 매번 사용하기 불편하므로 **import 문을 사용**.
 - ◆ import 문을 사용하면 클래스를 참조할 때마다 클래스 이름만 적으면 되므로 편리함.

패키지 A



패키지 B





패키지 사용(import)

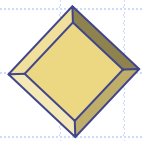
- ◆ 자바 프로그램에서 사용해야 하는 패키지는 import 명령으로 먼저 선언함.
- ◆ 패키지를 선언할 시 : 패키지를 전체를 사용한다고 선언할 수도 있고, 하나의 클래스를 사용한다고 선언할 수가 있음.
 - 어떤 패키지에 있는 모든 public 클래스를 선언하고 싶으면 “*”를 사용.

//java.awt 패키지내의 모든 클래스 사용

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

//java.awt 패키지내의 Graphics 클래스 사용

```
import java.awt.Graphics;
```

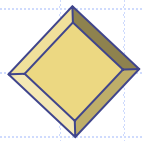


패키지 사용(import)

```
import java.applet.Applet;  
import java.awt.Graphics;
```

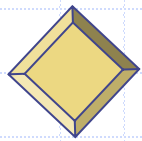
```
public class HelloApplet extends Applet { ①  
    public void paint(Graphics g) { ②  
        g.drawString("Hello world!", 5, 25);  
    }  
}
```

```
public class HelloNoimport extends java.applet.Applet { ③  
    public void paint(java.awt.Graphics g) { ④  
        g.drawString("Hello world!", 5, 25);  
    }  
}
```

시스템 패키지 종류(API)

- ◆ java.applet : Java Applet Package (9 장)
- ◆ java.awt : Java Abstract Windowing Toolkit Package (10,11장)
- ◆ java.awt.datatransfer : Java Data Transfer Package
- ◆ java.awt.event : Java AWT Event Package
- ◆ java.awt.image : Java AWT Image Package
- ◆ java.awt.peer : Java AWT Peer Package
- ◆ java.beans : Java Beans Package
- ◆ java.io : Java Input/Output Package (13 장)
- ◆ java.lang : Java Language Package, 기본 패키지로 import 문 필요 없이 사용가능(default)
- ◆ java.lang.reflect : Java Core Reflection Package
- ◆ java.net : Java Networking Package (14 장)
- ◆ java.rmi : Java Remote Method Invocation Package
- ◆ java.security : Java Security Package
- ◆ java.sql : Java Database Connectivity Package (12 장)
- ◆ java.text : Java Text Packages
- ◆ java.util : Java Utilities Package



사용자 package 만드는 방법



패키지 만드는 방법

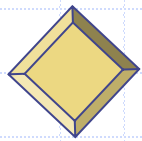
1. 패키지 이름 정하기
2. 폴더 구조 만들기
www 폴더 생성, 안에 naver 폴더, 안에 com 생성
3. 클래스 패키지 안에 포함하기

```
package www.naver.com;  
class Myclass {  
    """"  
}
```



다른 패키지에 있는 클래스 사용

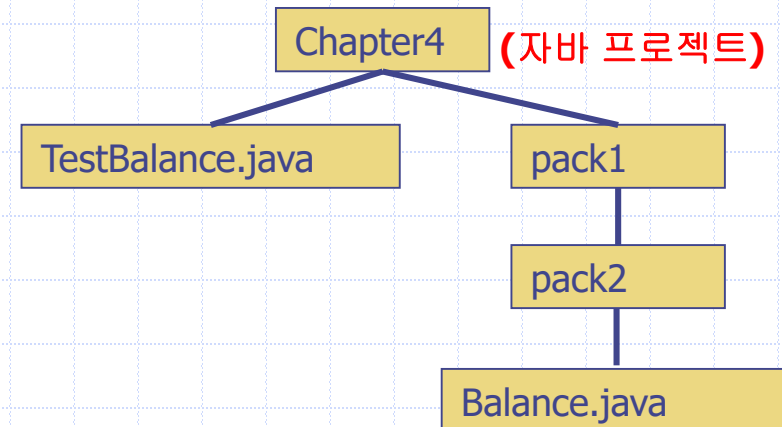
```
import www.naver.com.Myclass;  
class Yourclass {  
    """"  
}
```



사용자 package 만드는 방법

package pack1.pack2; ①

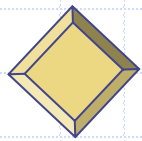
```
public class Balance {  
    String name;  
    double bal;  
    public Balance(String n, double b) {  
        name = n;  
        bal = b;    }  
    public void show() {  
        if(bal<0)  
            System.out.print("-->> ");  
        System.out.println(name + ": $" + bal);  
    }  
}
```



import pack1.pack2.*; ②

```
public class TestBalance {  
    public static void main(String args[]) {  
        Balance test = new Balance("Kim", 55.5);  
        test.show();  
    } }  
}
```

The screenshot shows an IDE console with tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, displaying the output: '<terminated> TestBalance [Java Application] C:\Program Kim: \$55.5'.



2) java.lang 패키지

- ◆ 프로그램을 개발할 때 가장 많이 사용하는 패키지.
- ◆ java.lang 패키지를 **기본 패키지로** 제공하며(**default**), 이 패키지에 있는 클래스와 인터페이스는 import 없이 사용.
- ◆ Java.lang 패키지
 - **Object** 클래스 - 모든 클래스의 슈퍼 클래스
 - **String** 클래스 - 문자열을 지정하고 여러 정보를 얻을 때 사용
 - **Wrapper** 클래스 - 기본 자료형을 클래스 객체로 만든 묶음.
 - **System** 클래스 - 표준 입출력
 - 등등

[java.lang 패키지의 클래스]

Object, String, StringBuffer, Wrapper, Runtime, Thread, System 등

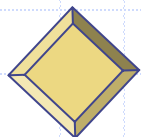
◆ Object 클래스

- ◆ 모든 자바 클래스의 최상위 클래스
- ◆ 자바의 모든 클래스는 Object 클래스로부터 상속 됨
- ◆ 코딩 시에 상속 없는 클래스를 작성했다면 Object 클래스를 상속받는 것으로 간주.
- ◆ Object 클래스의 주요 메소드

메소드	설명
protected Object clone()	객체를 복제
public boolean equals(Object obj)	두 객체의 동등(내용이 동일한지) 비교
public int hashCode()	객체를 식별하는 정수 값인 해시 코드를 반환
protected void finalize()	참조하지 않는 객체를 가베지 콜렉션
public Class getClass()	객체의 클래스 이름을 Class 형으로 반환
public String toString()	객체의 문자열을 반환

equals() 메소드

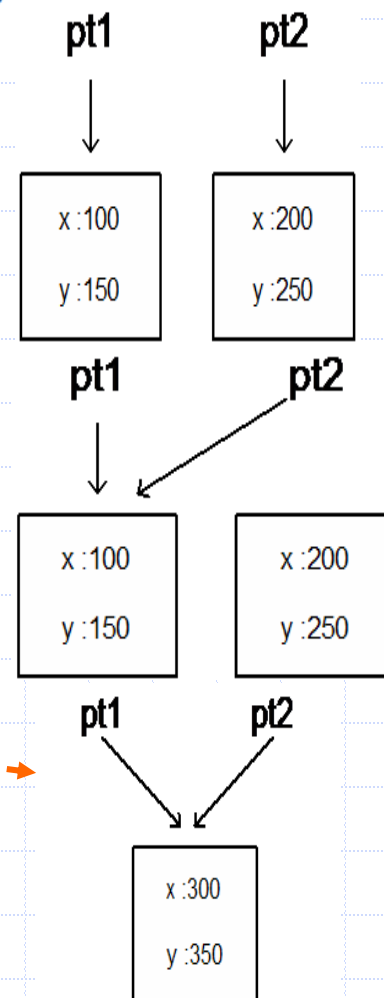
- ◆ 두 객체를 동등 비교할 때 사용
- ◆ 두 객체를 비교하여 논리적으로 동등하면 true를 반환하고, 그렇지 않으면 false를 반환.
- ◆ 객체 참조(객체에 대한 변수)를 비교하는데 사용하는 방법
 - 대입 연산자(=)
 - 등위 연산자(==, !=)
 - equals() 메소드

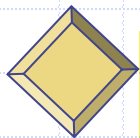


대입 연산자(=) 사용

Problems @ Javadoc Decl
<terminated> ObjRef [Java Appli
pt1 : 100, 150
pt2 : 200, 250
pt1 : 100, 150
pt2 : 100, 150
pt1 : 300, 350
pt2 : 300, 350

```
import java.awt.Point; //①
public class ObjRef{
    public static void main(String[] args) {
        Point pt1,pt2;
        pt1 = new Point(100,150); //②
        pt2 = new Point(200,250); //③
        System.out.println("pt1 : " + pt1.x + "," + pt1.y); //④
        System.out.println("pt2 : " + pt2.x + "," + pt2.y); //⑤
        pt2 = pt1; //⑥
        System.out.println("pt1 : " + pt1.x + "," + pt1.y); //⑦
        System.out.println("pt2 : " + pt2.x + "," + pt2.y); //⑧
        pt1.x = 300; //⑨
        pt1.y = 350; //㉐
        System.out.println("pt1 : " + pt1.x + "," + pt1.y); //㉑
        System.out.println("pt2 : " + pt2.x + "," + pt2.y); //㉒
    }
}
```





등위 연산자(==, !=) 사용



==

- 2개의 객체의 참조가 같은 객체인가를 결정하고 참/거짓 값(true, false)을 반환



!=

- 2개의 객체의 참조가 다른 객체인가를 결정하고 참/거짓 값(true, false)을 반환

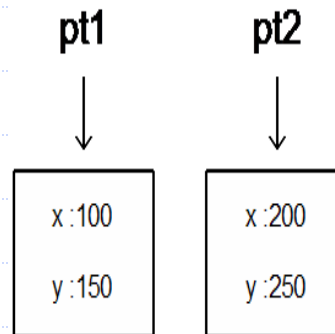


앞 코드에서 ③이 시행된 후에 다음 코드가 나온다면 이 코드의 결과는 “false”가 된다.

```
pt1 = new Point(100,150); //②  
pt2 = new Point(200,250); //③  
if (pt1 == pt2)
```

<< 등위 연산자 >>

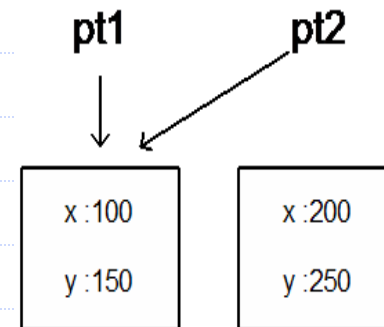
~ 객체 변수(참조)에 저장된 값을 비교.
~ 즉, 주소 값을 비교함.



◆ equals() 메소드 사용

- ◆ 2개의 객체(같은 객체 또는 다른 객체일 수도 있음)가 같은 값(내용)을 가지고 있는지/아닌지를 결정하는데 사용.
- ◆ pt1이 가리키는 객체와 pt2가 가리키는 객체가 가지고 있는 값들이 같은지를 비교하여 같으면 “true” 다르면 “false” 값을 반환.

```
pt1 = new Point(100,150); //②  
pt2 = new Point(200,250); //③  
pt2 = pt1; //⑥  
if (pt1.equals(pt2))
```



<< equals() >>

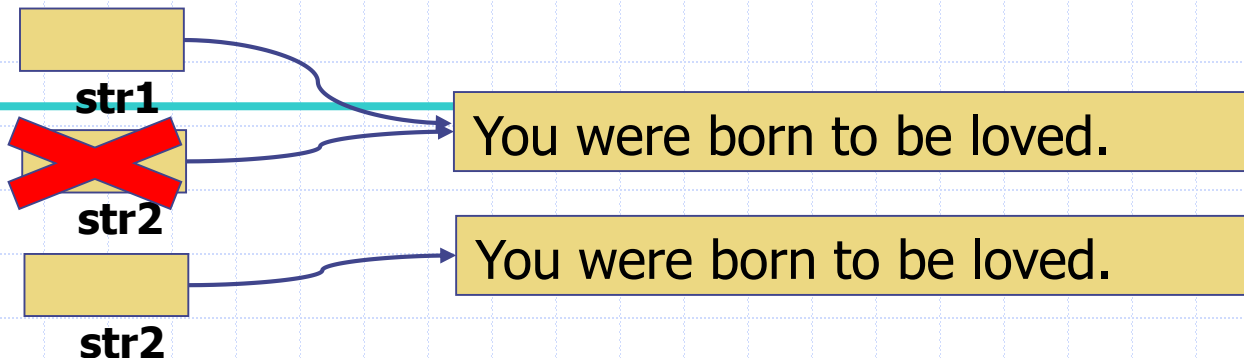
~ 객체 변수(참조)에 가리키는 곳의 값을 비교함.
~ 즉, 주소가 가리키는 곳의 값(x, y 값)

◆ equals() 메소드 사용

```
class EqualsTest {  
    public static void main (String[] args) {  
        String str1, str2; ①  
        str1 = "You were born to be loved."; ②  
        str2 = str1; ③  
        System.out.println("String1: " + str1); ④  
        System.out.println("String2: " + str2); ⑤  
        System.out.println("Same object? " + (str1 == str2)); ⑥  
        str2 = new String(str1); ⑦  
        System.out.println("String1: " + str1); ⑧  
        System.out.println("String2: " + str2); ⑨  
        System.out.println("Same object? " + (str1 == str2));  
        System.out.println("Same value? " + str1.equals(str2));  
    }  
}
```

Problems @ Javadoc Declaration Console

<terminated> EqualsTest [Java Application] C:\Program Files\Java\W
String1: You were born to be loved.
String2: You were born to be loved.
Same object? true
String1: You were born to be loved.
String2: You were born to be loved.
Same object? false
Same value? true

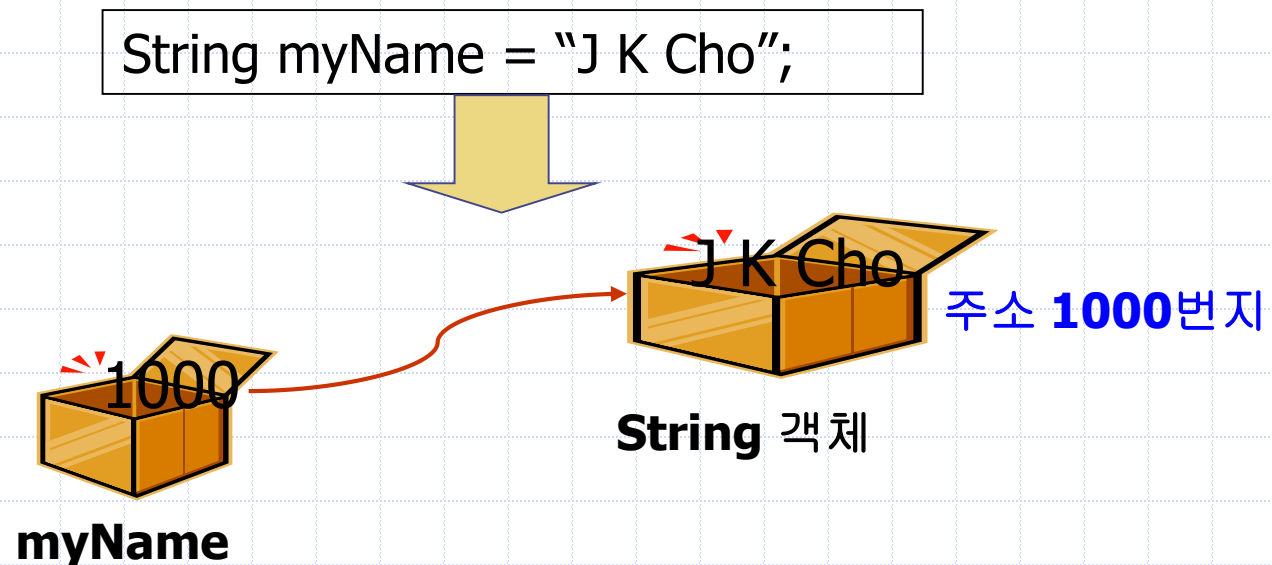


◆ String 클래스

<< 자바 자료형 >>

- 1) 기본 자료형 ~ 8가지 자료형
- 2) 레퍼런스 자료형 ~ 클래스에서 생성되는 모든 자료형(String, 배열, 객체 등)

- ◆ 여러 개의 문자를 나타내는 문자열을 저장하려면 String형 변수를 사용
- ◆ 저장하고자 하는 문자열을 이중 인용부호(“)로 감싸서 표현하여 String형 변수에 저장

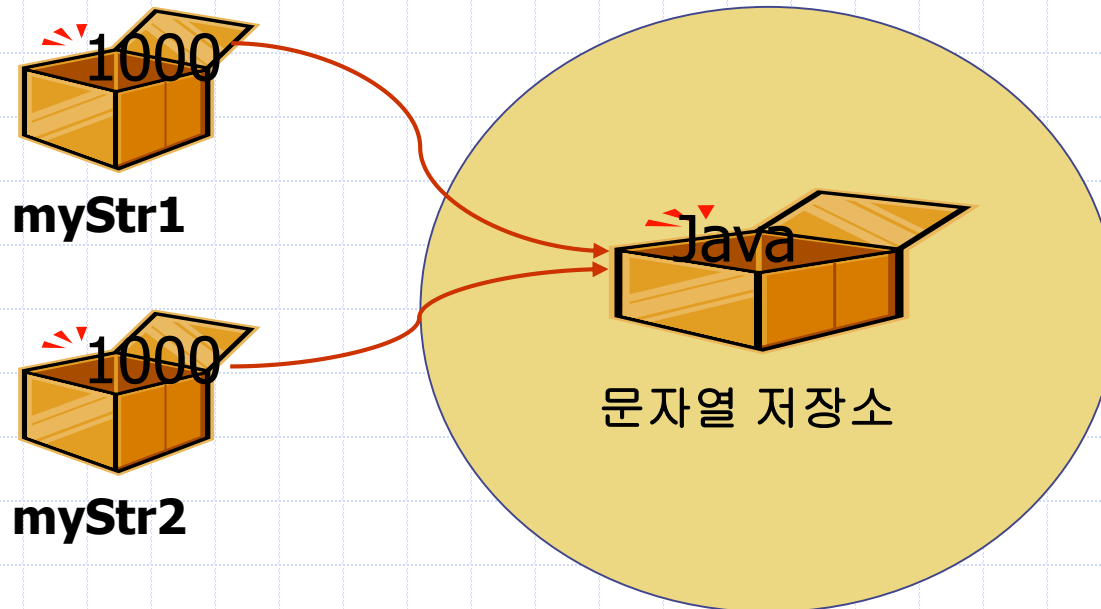


◆ String 클래스

◆ String 클래스 객체 생성 방법

1) 문자열 상수를 지정하는 방법

```
String myStr1 = "Java";  
String myStr2 = "Java";
```

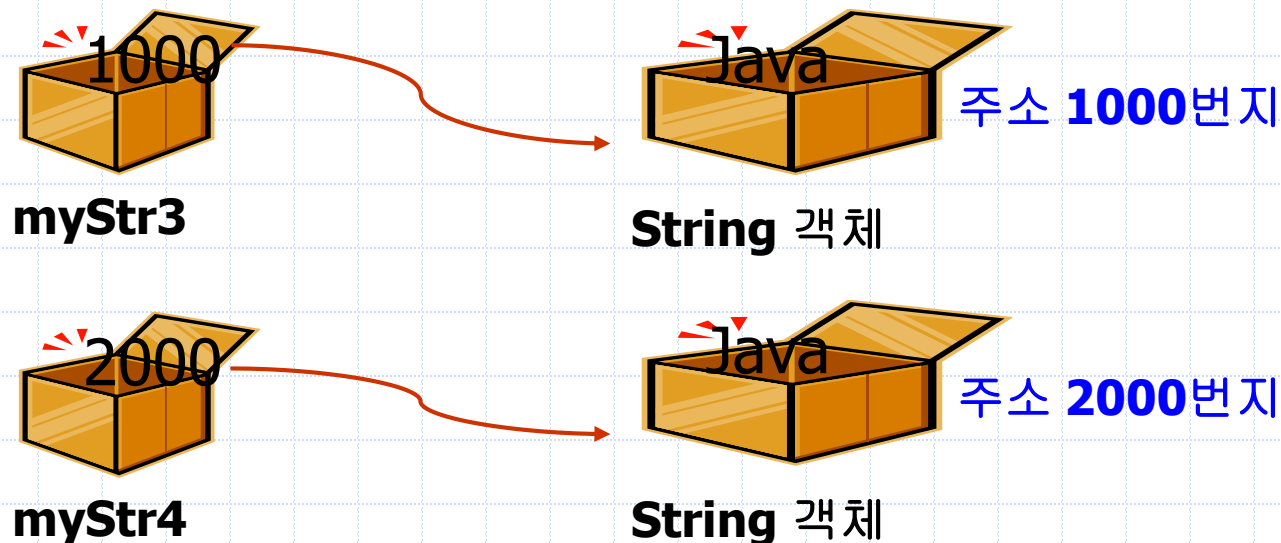


◆ String 클래스

◆ String 클래스 객체를 생성 방법

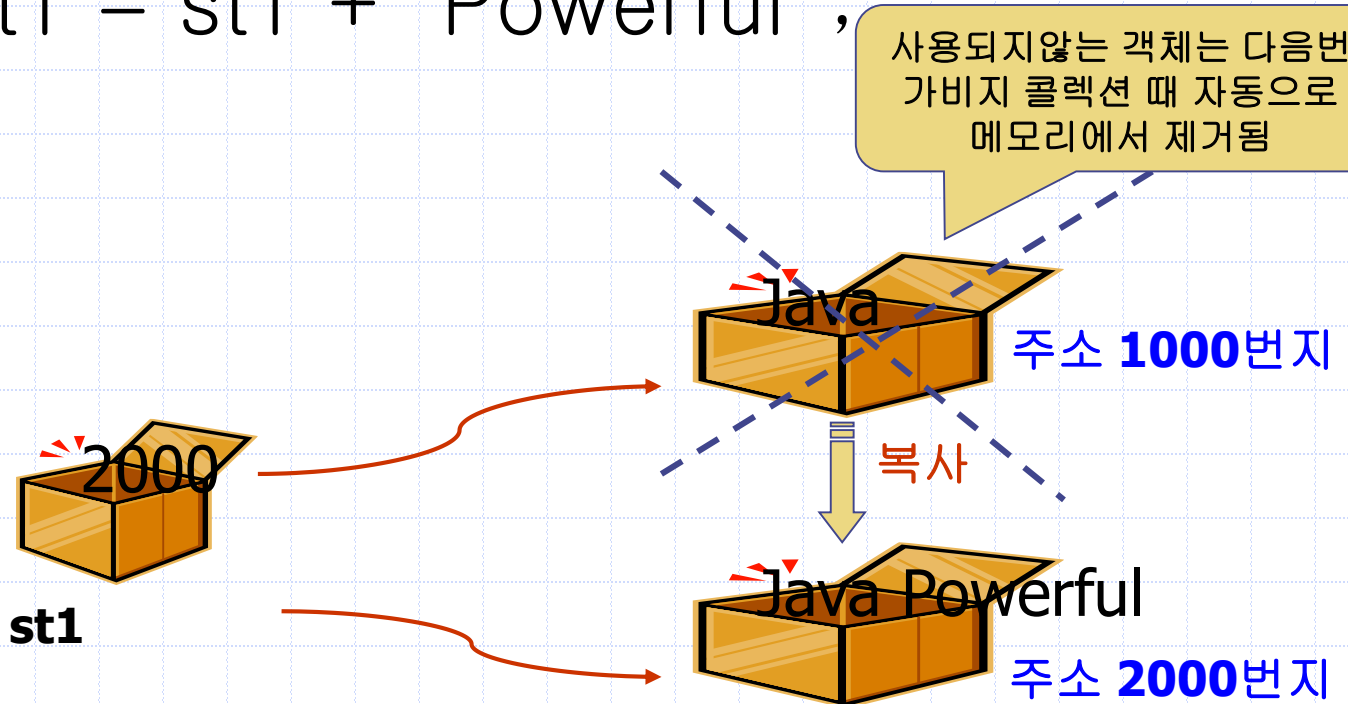
2) String 클래스의 생성자(new)를 이용하는 방법

```
String myStr3 = new String("Java");  
String myStr4 = new String("Java");
```



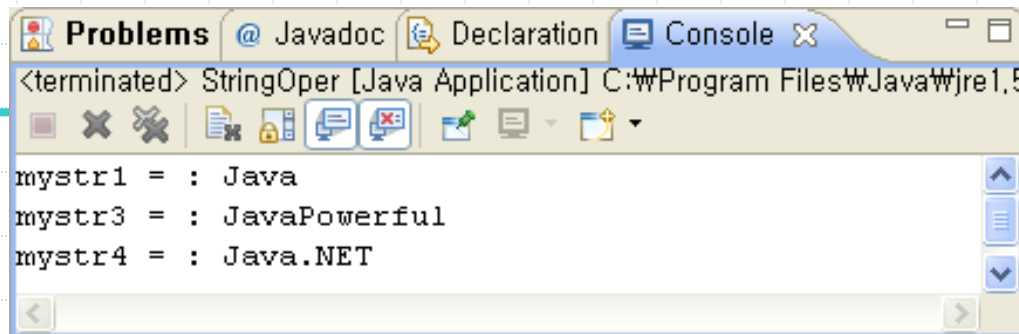
◆ 레퍼런스 자료형의 연산

◆ String st1 = "Java";
st1 = st1 + "Powerful";



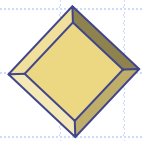
◆ 레퍼런스 자료형의 연산

```
public class StringOper {  
    public static void main(String[] args) {  
        String mystr1 = new String("Java");  
        String mystr2 = new String(".NET");  
        String mystr3 = mystr1;           //①  
        String mystr4 = mystr1 + mystr2;  //②  
        mystr3 = mystr3 + "Powerful";      //③  
  
        System.out.println("mystr1 = : " + mystr1);  
        System.out.println("mystr3 = : " + mystr3);  
        System.out.println("mystr4 = : " + mystr4);  
    }  
}
```



The screenshot shows a Java IDE window with the title bar "Problems @ Javadoc Declaration Console". The console output is as follows:

```
<terminated> StringOper [Java Application] C:\Program Files\Java\jre1.5  
mystr1 = : Java  
mystr3 = : JavaPowerful  
mystr4 = : Java.NET
```



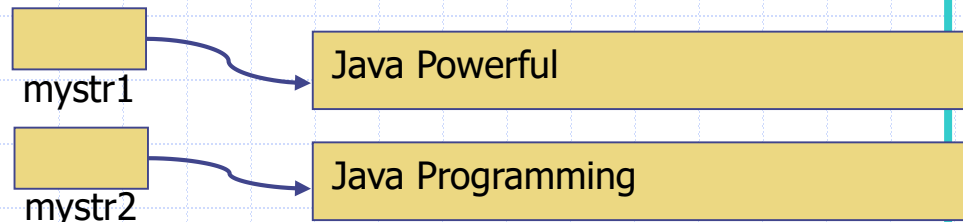
String 클래스의 메소드

메소드	설명
int <code>length()</code>	String 클래스의 문자열 길이를 반환
boolean <code>equals(String str)</code>	저장된 문자열과 str 문자열의 내용이 같은지를 비교
boolean <code>equalsIgnoreCase(String str)</code>	대소문자 구분 없이 , 저장된 문자열과 str 문자열의 내용이 같은지를 비교
String <code>substring(int beginindex)</code>	문자열의 beginindex 위치부터 마지막까지의 문자열을 반환
String <code>concat(String str)</code>	저장된 문자열과 str 문자열을 결합
String <code>replace(char old, char new)</code>	문자열 내의 old 문자를 new 문자로 변경
String <code>toLowerCase()</code>	String 클래스 객체의 문자열을 소문자로 변경
String <code>toUpperCase()</code>	String 클래스 객체의 문자열을 대문자로 변경
char <code>charAt(int index)</code>	index 위치의 char 변수값(문자)을 반환
int <code>indexOf(int ch)</code>	저장된 문자열의 첫 번째 ch 문자의 위치를 반환
int <code>lastIndexOf(int ch)</code>	저장된 문자열의 마지막 ch 문자의 위치를 반환
String <code>trim()</code>	문자열 끝의 공백 문자를 제거

String 클래스의 메소드

```
public class StringMethod {  
    public static void main(String[] args) {  
        int alength, blength;  
        char achar, bchar;  
        String mystr1 = new String("Java Powerful");  
        String mystr2 = new String("Java Programming");  
        String mystr3 = mystr1 + mystr2;
```

```
        alength = mystr1.length(); //①  
        blength = mystr2.length();  
        achar = mystr1.charAt(5); //②  
        bchar = mystr2.charAt(10);
```



```
        System.out.println("mystr1에 저장되어있는 "+mystr1+"의 문자길이는"+alength);  
        System.out.println("mystr2에 저장되어있는 "+mystr2+"의 문자길이는"+blength);  
        System.out.println("mystr1의 6번째 인덱스에 있는 문자는 " + achar);  
        System.out.println("mystr2의 11번째 인덱스에 있는 문자는 " + bchar);  
        System.out.println("mystr1의 저장된 문자를 대문자로 바꿈 : " +  
            mystr1.toUpperCase()); //③  
        System.out.println("mystr2의 저장된 문자중 a를 A로 바꿈 : " +  
            mystr2.replace('a', 'A')); //④  
    }  
}
```

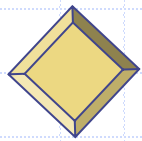
The screenshot shows a Java IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the program. The output text is as follows:

```
<terminated> StringMethod [Java Application] C:\WProgram File  
mystr1에 저장되어있는 Java Powerful의 문자길이는 13  
mystr2에 저장되어있는 Java Programming의 문자길이는 16  
mystr1의 6번째 인덱스에 있는 문자는 P  
mystr2의 11번째 인덱스에 있는 문자는 a  
mystr1의 저장된 문자를 대문자로 바꿈 : JAVA POWERFUL  
mystr2의 저장된 문자중 a를 A로 바꿈 : JAVa ProgrAmming
```

◆ StringBuffer 클래스

- ◆ 동적 문자열을 처리하는 기능을 제공
- ◆ 프로그램 실행 중에 동적으로 문자열의 내용이나 길이를 바꿀 수가 있다.
 - String 클래스처럼 자바 가상머신 내부(문자열 저장소)에 새롭게 문자열을 생성하는 것이 아니라, 메모리 상에서 문자열을 처리하기 때문.
- ◆ StringBuffer 클래스를 생성방법

```
StringBuffer stBuf = new StringBuffer();
```



StringBuffer 클래스

메소드	설명
int length()	StringBuffer 클래스의 문자열 길이를 반환 (실제 저장된 문자의 수)
int capacity()	할당된 문자 배열의 크기를 반환(사전에 할당된 문자 배열의 크기)
StringBuffer append(String str)	저장되어있는 문자열 뒤에 str 문자열 추가
StringBuffer insert(int off, String str)	off로 정해진 위치에 Str 문자열 추가
String toString()	저장되어있는 문자열을 String형으로 변경 (StringBuffer 클래스로 작업을 했어도 String 클래스 객체로 요구되는 메소드에 인자를 넘겨줄 때에는 toString() 메소드를 이용해야 함)
StringBuffer reverse()	저장되어있는 문자열을 반대로 변경
void setCharAt(int index, char ch)	문자열에서 index 위치의 문자를 ch 문자로 변경
void setLength(int index)	문자열의 크기 설정

◆ StringBuffer 클래스

```
public class StringBufferProg {  
    public static void main(String[] args) {  
        StringBuffer str = new StringBuffer("Java Programming"); //①  
        StringBuffer str2;
```

```
        str2 = str.insert(5, "JSP "); //②
```

```
        System.out.println(str);  
        System.out.println(str2);
```

```
        str.append(" Good "); //③
```

```
        str.append('A'); //④
```

```
        System.out.println(str);
```

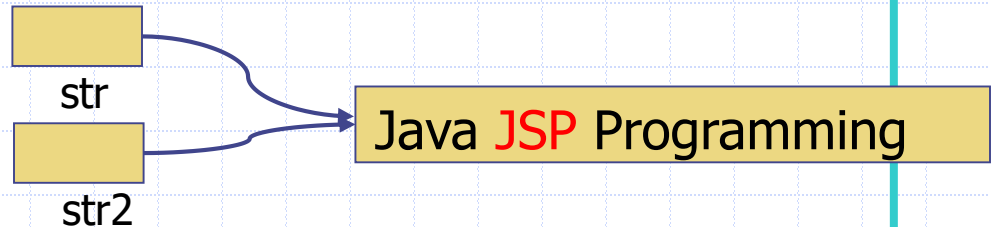
```
        str2.reverse(); //⑤
```

```
        System.out.println(str2);
```

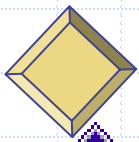
```
        str.setLength(10); //⑥
```

```
        System.out.println(str);
```

```
    }  
}
```



```
Java JSP Programming  
Java JSP Programming|  
Java JSP Programming Good A  
A dooG gnimmargorP PSJ avaJ  
A dooG gni
```



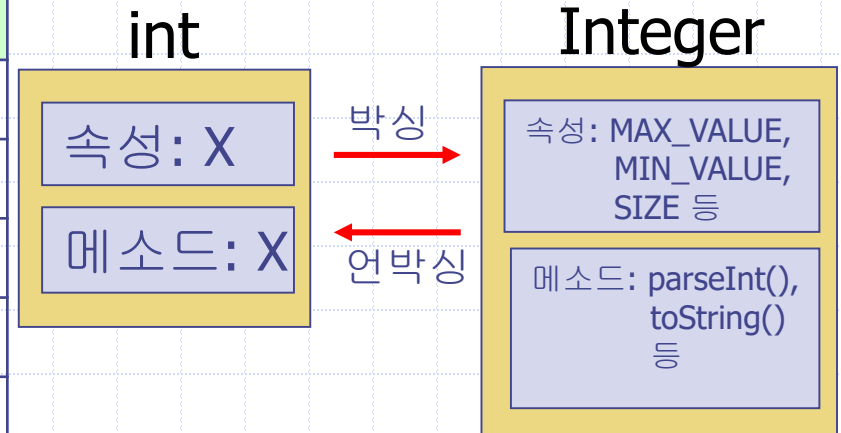
Wrapper 클래스

◆ 기본 자료형을 클래스 객체로 만든 묶음.

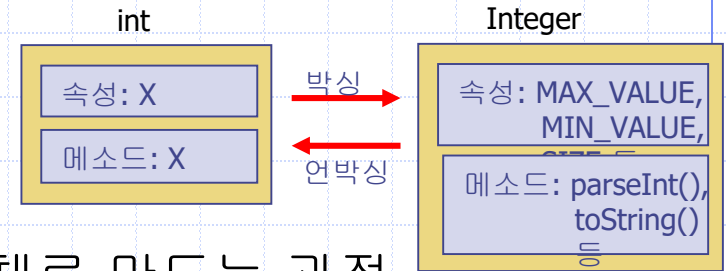
- 포장(Wrapper) 객체
- 기본 데이터를 클래스로 포장하고, 클래스가 가지고 있는 변수 및 메소드를 사용하여 효율적으로 처리하고자 하는 목적으로 사용.

◆ 기본 자료형에 대응되는 클래스들.

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



◆ 박싱과 언박싱



◆ 박싱(Boxing)

- 기본 자료형의 값을 Wrapper 객체로 만드는 과정

◆ 언박싱(Unboxing)

- Wrapper 객체에서 기본 자료형의 값을 얻어내는 과정

◆ int를 생성자를 이용하여 박싱하는 방법.

```
Integer boxint = new Integer(200); //기본 타입의 값을 줄 경우  
Integer boxint = new Integer("200"); //문자열의 값을 줄 경우
```

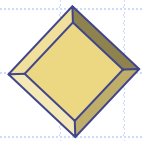
◆ 생성자를 이용하지 않고 각 Wrapper 클래스마다 가지고 있는 메소드인 **valueOf()**를 사용하는 방법.

```
Integer boxint = Integer.valueOf(200); //기본 타입의 값  
Integer boxint = Integer.valueOf("200"); //문자열의 값
```

◆ 박싱과 언박싱

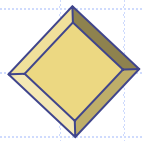
- ◆ Wrapper 객체로부터 int 값으로 언박싱하는 방법
 - 언박싱하기 위해서는 각각의 Wrapper 클래스마다 가지고 있는 **XxxValue()** 메소드를 사용하면 됨.
 - ◆ (**Xxx**는 기본 자료형의 타입이름이 들어가며, int형은 **intValue()**이고 char형은 **charValue()** 이다.)
 - ◆ **byteValue()**, **shortValue()**, **longValue()**, **floatValue()**, **doubleValue()**, **booleanValue()**

```
int number = boxint.intValue();
```



자동 박싱과 자동 언박싱

- ◆ 박싱, 언박싱 과정은 자동으로 됨(V 5.0 부터).
 - 기본 자료형의 값을 직접 박싱, 언박싱하지 않아도 자동으로 박싱, 언박싱이 일어남.
- ◆ 자동 박싱 : 기본 자료형의 값을 Wrapper 클래스 타입에 대입할 경우에 발생함.
 - `Integer boxint = 200;` //자동 박싱
- ◆ 자동 언박싱 : Wrapper 객체의 값이 기본 자료형으로 대입할 경우에 발생함.
 - `int number = boxint;` //자동 언박싱



자동 박싱과 자동 언박싱

```
public class BoxUnboxEx {  
    public static void main(String[] args) {  
        Integer boxint = new Integer(200); ①  
        Integer autobint = 200; //자동박싱  
        Double boxdouble = Double.valueOf(35.267);  
  
        int number = boxint.intValue(); ③  
        int abnumber = autobint; //자동언박싱  
        double dnumber = boxdouble.doubleValue(); ④  
  
        System.out.println("언박싱 정수 값 : " + number);  
        System.out.println("자동언박싱정수값: " + abnumber);  
        System.out.println("언박싱 실수 값 : " + dnumber);  
        System.out.println("정수를 2진수로 : " + Integer.toBinaryString(boxint)); ⑤  
        System.out.println("정수를 16진수로 : " + Integer.toHexString(number));  
        System.out.println("Not a Number : " + Double.isNaN(dnumber) ); ⑦  
    }  
}
```

Problems @ Javadoc Declaratic

<terminated> BoxUnboxEx [Java Applicati

언박싱 정수 값 : 200
자동언박싱정수값 : 200
언박싱 실수 값 : 35.267
정수를 2진수로 : 11001000
정수를 16진수로 : c8
Not a Number : false

<< 래퍼 클래스(8 종류) >>

~ 속성 : BYTES, SIZE, TYPE, MAX_VALUE, MIN_VALUE

~ 메소드 : parseInt(), toString(), compare(), max(), min() 등

등

◆ 3) java.util 패키지

- ◆ 프로그램 개발에서 유용한 역할을 하는 클래스들을 모음.
- ◆ 주로 많이 쓰는 클래스
 - **Calendar 클래스** - 운영체제의 날짜와 시간을 얻을 때 사용
 - **Date 클래스** - 날짜와 시간 저장
 - **Random 클래스** - 난수를 얻을 때 사용
 - **StringTokenizer 클래스** - 특정 문자로 구분된 문자열을 뽑아낼 때 사용
 - Array, Objects 등

◆ Random 클래스

- ◆ 난수(임의의 값)을 발생시켜주는 클래스
- ◆ java.lang 패키지의 Math 클래스의 random() 메소드 :
 - 0.0 ~ 1.0 사이의 double 난수를 발생시킴.
- ◆ Random 클래스:
 - int, long, float, double, boolean 형의 난수를 발생시킴.
- ◆ Random 클래스가 제공하는 메소드

메소드	설명
int nextInt()	int 타입의 난수를 반환.
int nextInt(int n)	0부터 (n-1) 까지의 int 타입의 난수를 반환.
long nextLong()	long 타입의 난수를 반환.
float nextFloat()	float 타입의 난수를 반환.
double nextDouble()	double 타입의 난수를 반환.
boolean nextBoolean()	boolean 타입의 난수를 반환.

◆ Random 클래스

- ◆ Ex) 난수 발생은 Random 객체를 생성하고, 원하는 자료형의 메소드를 사용하여 얻는다.
 - '0 <= num < 45'의 정수 난수 값을 얻는다.

```
Random random = new Random();  
int num = random.nextInt(45);    //0~44까지의 정수
```

Random 클래스

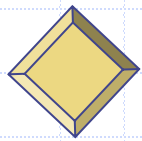
```
import java.util.*;
```

```
public class RandomExam {  
    public static void main(String[] args) {  
        Random ran = new Random();  
        int num = ran.nextInt(100);  
        System.out.println("0 부터 99 사이의 임의의 값 출력 : " + num);  
        float flt = ran.nextFloat();  
        System.out.println("0.0 부터 1.0 사이의 임의의 값 출력 : " + flt);  
        boolean bln = ran.nextBoolean();  
        System.out.println("임의의 논리값 출력 : " + bln);  
        System.out.println("*** 복권번호 출력 ***");  
        for(int i=0; i<6; i++) {  
            num = ran.nextInt(46);  
            System.out.println((i+1) + " 번째 복권번호 : " + num);  
        }  
    }  
}
```

Problems @ Javadoc Declaration Console

<terminated> RandomExam [Java Application] C:\WProgram

1 부터 99 사이의 임의의 값 출력 : 59
0.0 부터 1.0 사이의 임의의 값 출력 : 0.54338324
임의의 논리값 출력 : true
*** 복권번호 출력 ***
1 번째 복권번호 : 34
2 번째 복권번호 : 17
3 번째 복권번호 : 38
4 번째 복권번호 : 42
5 번째 복권번호 : 2
6 번째 복권번호 : 40



StringTokenizer 클래스

- ◆ 문자열을 파싱해서 토큰으로 만드는 것
 - 파싱 - 구분자에 따라 문자열을 나누는 작업
 - 토큰 - 전체 문자열을 구분되는 문자열(또는 문자)을 기준으로 쪼갠 단위이고, 이때 구분되는 문자열(또는 문자)를 구분자라고 한다.

Ex) 2008/02/15

- 문자열 파싱 : “2008”, “02”, “15” 이라는 문자열인 3개의 토큰으로 나누어지며, 문자열을 구분하는 구분자는 “/”이다.

- ◆ StringTokenizer 클래스의 객체를 생성방법

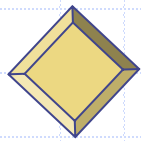
```
StringTokenizer st = new StringTokenizer(파싱하려는문자열,  
                                           구분자);
```

- ◆ Ex) 클래스 작성 방법

```
StringTokenizer st = new StringTokenizer("2008/02/15", "/");
```

◆ StringTokenizer 클래스

- ◆ 문자열 파싱을 위하여
`java.util.StringTokenizer` 클래스 제공
 - 프로그램 맨 앞에 import문으로 선언.
 - 파싱하고자 하는 문자열을 인자로 받아서 생성되며 자동으로 파싱작업을 해줌.
 - StringTokenizer는 객체를 생성한 후에는 메소드들을 사용해서 토큰을 만들 수 있음.



StringTokenizer 클래스

메소드	설명
String nextToken()	파싱해서 구한 토큰을 반환
String nextToken(String delim)	새로운 구분자인 delim을 써서 구한 토큰을 반환
boolean hasMoreTokens()	파싱된 문자열이 nextToken() 메소드를 실행한 후에 아직 넘겨주지 않은 토큰이 있는지 여부를 반환
int countTokens()	파싱한 결과로 구한 토큰이 모두 몇 개인지 반환

//앞의 예제의 변수 st에서 토큰을 모두 출력하려면 hasMoreTokens(),
//nextToken() 메소드를 사용하여 작성

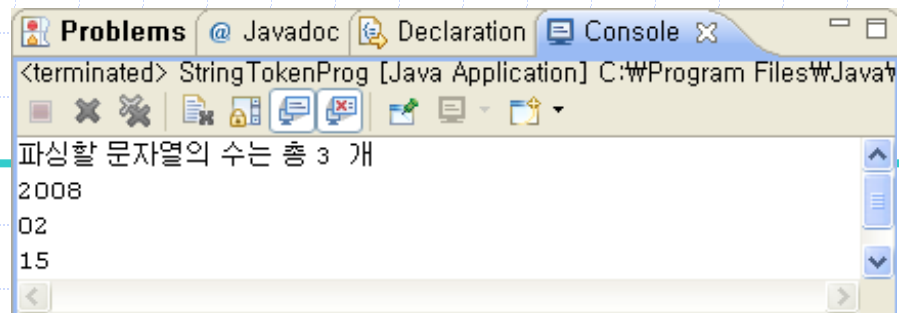
```
StringTokenizer st = new StringTokenizer("2008/02/15", "/");  
...  
while(st.hasMoreTokens()) {  
    System.out.println(st.nextToken());  
}
```


◆ StringTokenizer 클래스

```
import java.util.StringTokenizer;    //①

public class StringTokenProg {
    public static void main(String[] args) {
        StringTokenizer str = new StringTokenizer("2008/02/15", "/"); //②
        int count;
        count = str.countTokens(); //③
        System.out.println("파싱할 문자열의 수는 총 " + count + " 개");

        while(str.hasMoreTokens()) {    //④
            System.out.println(str.nextToken());    //⑤
        }
    }
}
```



◆ 9. 객체의 형 변환(casting)

◆ 원시 유형(기본 자료형)간에 형 변환

■ 자동(묵시적) 형 변환(Promotion)

- ◆ 작은 크기의 형이 큰 크기의 형에 저장될 때.
- ◆ $\text{byte}(1) < \text{short}(2) < \text{int}(4) < \text{long}(8) < \text{float}(4) < \text{double}(8)$
- ◆ $\text{char} \rightarrow \text{int}$, $\text{char} \rightarrow \text{byte}(x)$

■ 강제(명시적) 형 변환(Casting)

- ◆ $\text{char} \rightarrow \text{byte}$ (강제)
- ◆ 일반 형식 `(data type) value`

- ◆ Ex)

```
int x=3, y=2;
float z;
z = (float)(x/y);
```

◆ 9. 객체의 형 변환(casting)

◆ 객체 간에 형 변환

- 슈퍼클래스 형을 서브클래스 형으로 변환.

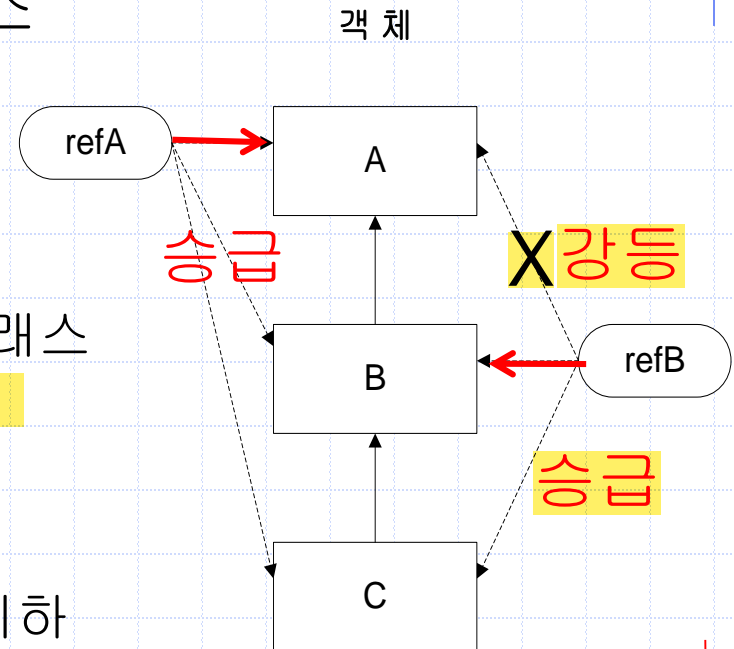
- 규칙

- ◆ 객체간에 상속 관계여야함.
- ◆ 서브클래스의 객체를 슈퍼클래스의 객체로 형 변환 가능(승급, Promotion)

refA = refB;

* 필요할 경우 컴파일 에러를 피하기 위해 형 변환 연산자를 사용.

(classname) 객체



refA : 객체 A의 객체 참조 변수
refB : 객체 B의 객체 참조 변수

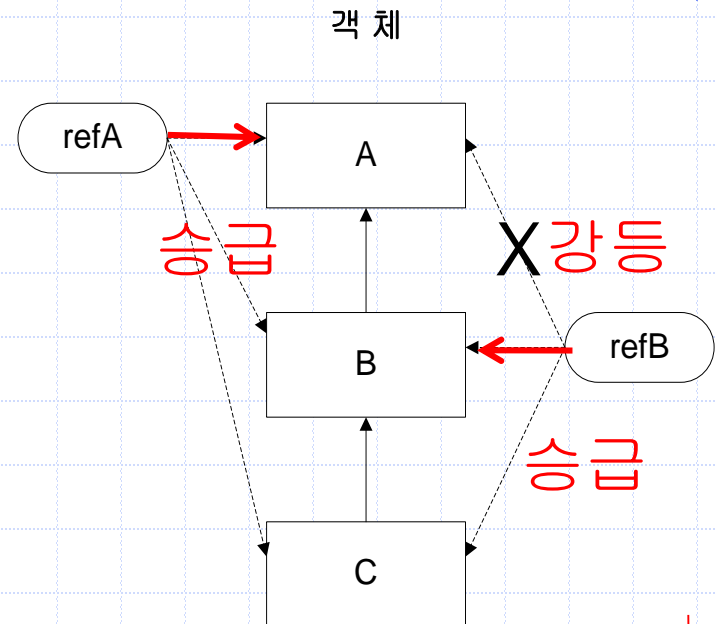
◆ 9. 객체의 형 변환(casting)

◆ 객체 간에 형 변환

- 슈퍼클래스 형을 서브클래스 형으로 변환.
- 규칙
 - ◆ 슈퍼클래스의 객체를 서브클래스의 객체로 형 변환 불가능 (강등, Demotion)

refB = refA;

* 강등은 허용되지 않음.

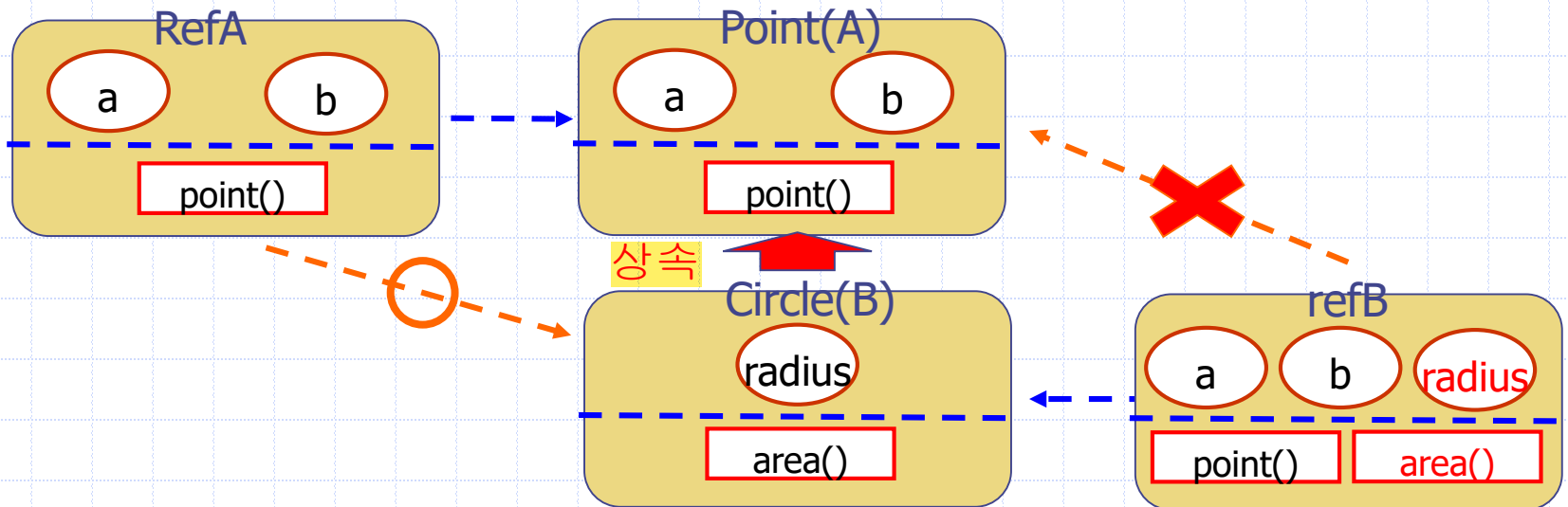


refA : 객체 A의 객체 참조 변수
refB : 객체 B의 객체 참조 변수

◆ 9. 객체의 형 변환(casting)

■ 강등이 허용되지 않는 이유

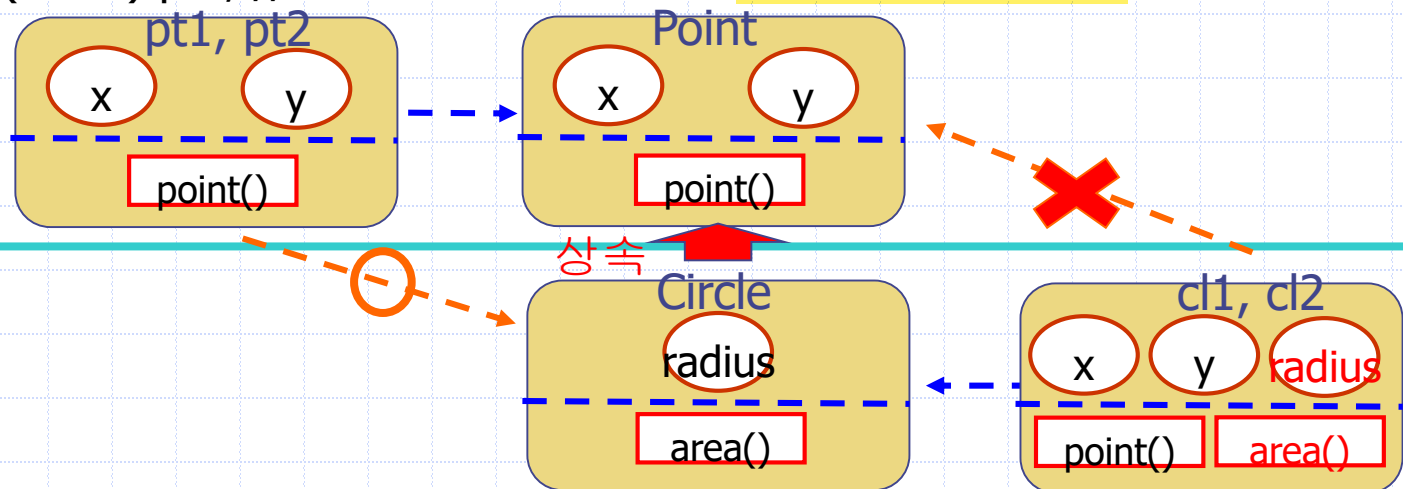
- ◆ 슈퍼클래스의 객체는 서브클래스의 객체에 정의된 변수들이 없고,
 - ◆ 서브클래스에 있는 메소드를 실행할 수가 없기 때문.
- * 강제로 형 변환을 하면 컴파일은 되나 실행 시점에 예외가 발생함.



9. 객체의 형 변환(casting)

```
import java.awt.Point;
public class Circle extends Point {
    int radius;
    public static void main(String args[ ]) {
        Point pt1, pt2; //슈퍼클래스
        Circle cl1, cl2; //서브클래스
        pt1 = new Point();
        pt2 = new Point();
        cl1 = new Circle();
        pt1 = cl1; //① Point pt1=new Circle(); 정상
        cl2 = pt2; //② Circle cl2=new Point(); 컴파일 오류
        cl2 = (Circle) pt2; // ③
    }
}
```

코드	컴파일	실행
①pt1=cl1;	정상	정상
②cl2=pt2;	컴파일에러	실행 안됨
③cl2=(Circle)pt2;	문제없음	실행시점 예외발생



◆ 실습문제



◆ 1, 2, 3, 4, 5, 6번 실습

◆연습문제



◆ 1) 1, 2, 3번

◆ 2) 1, 2, 3, 4번

◆ 3, 4번 실습