

주요클래스

강사 : 강병준

패키지란

● Class의 묶음

- 관련있는 다수의 Class 모음
- PC에서 폴더 개념과 유사



파일명:
JavaTest1.java
저장위치: C:\

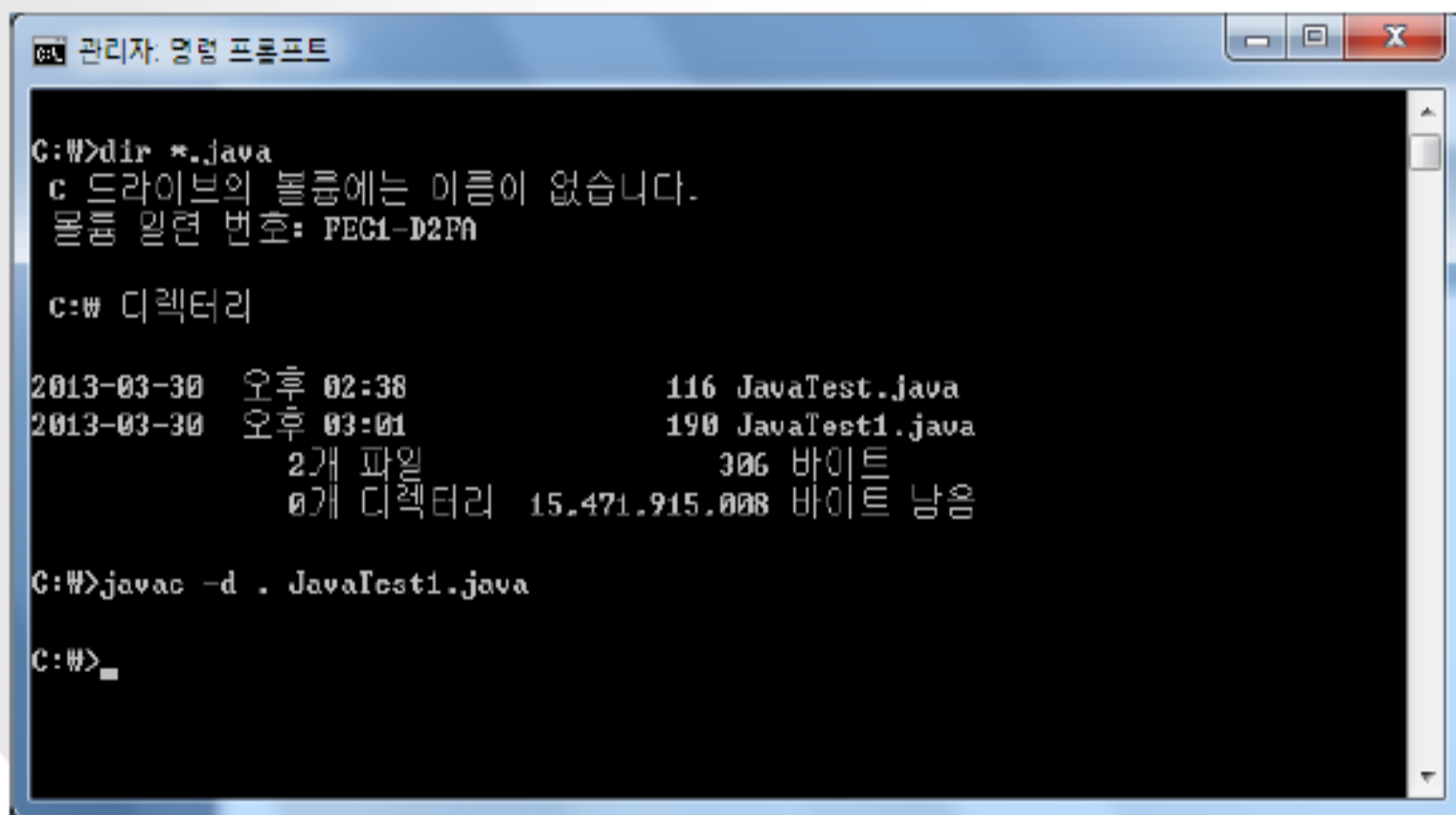
```
JavaTest1.java - 메모장
파일(F) 편집(E) 서식(O) 보기(V)
package test; // test 패키지에 클래스 생성
import java.lang.*;
public class JavaTest1 {
    public static void main(String[] args) {
        System.out.println("Java Test111");
    }
}
```

패키지 선언부

패키지 포함(클래스 사용)

● Package 클래스 컴파일

➤ 명령 수행 : `javac -d . TestJava1.java`



관리자: 명령 프롬프트

```
C:\W>dir *.java
c 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: FEC1-D2FA

c:\w 디렉터리

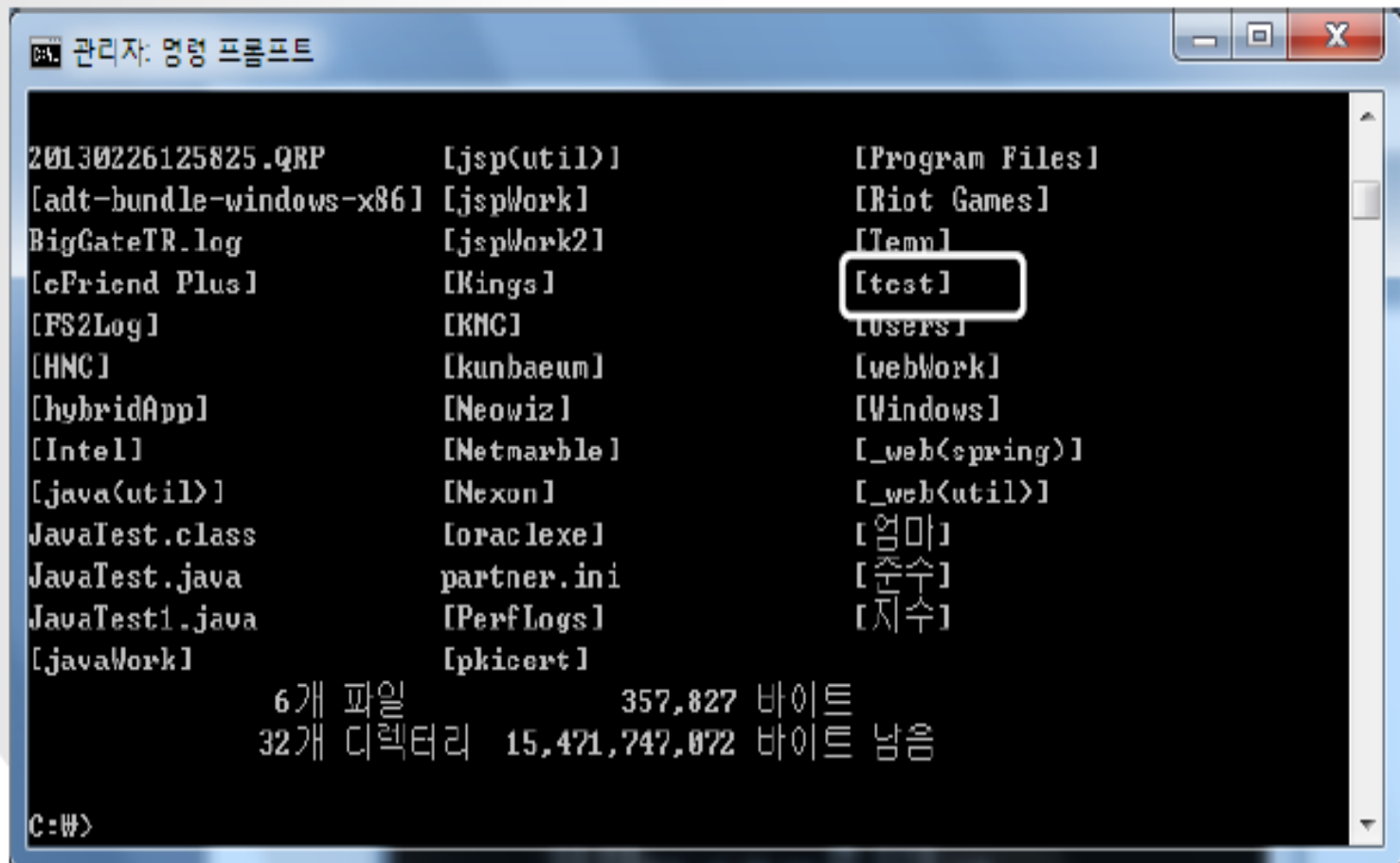
2013-03-30 오후 02:38                116 JavaTest.java
2013-03-30 오후 03:01                190 JavaTest1.java
                2개 파일                306 바이트
                0개 디렉터리 15,471,915,008 바이트 남음

C:\W>javac -d . JavaTest1.java

C:\W>
```

● Package 폴더

➤ 명령 수행 : `dir/d` → [test] 폴더 생성



```
관리자: 명령 프롬프트

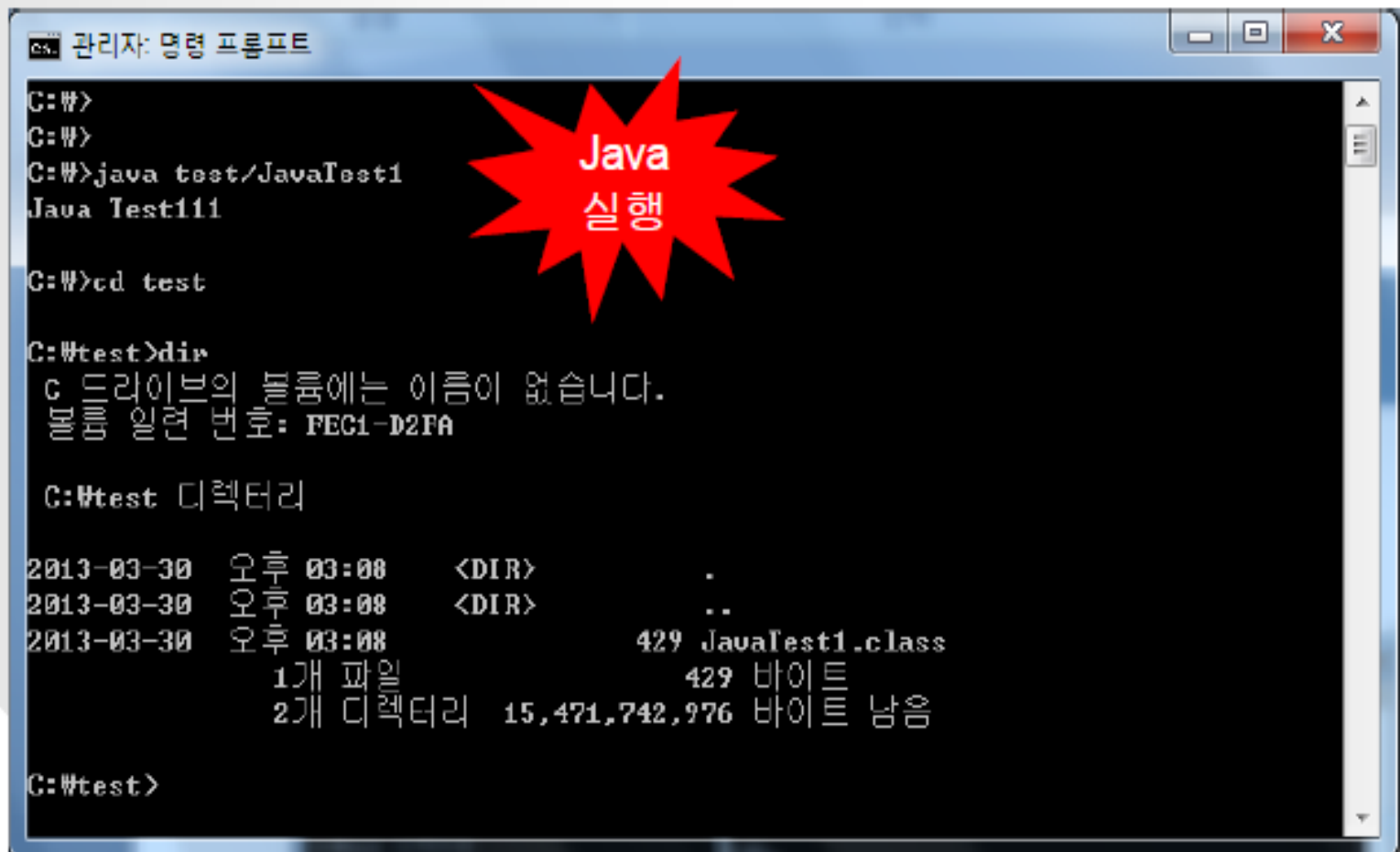
20130226125825.QRP      [jsp<util>]           [Program Files]
[adt-bundle-windows-x86] [jspWork]             [Riot Games]
BigGateTR.log          [jspWork2]           [Temp]
[cFriend Plus]         [Kings]              [test]
[FS2Log]               [KNC]               [Users]
[HNC]                 [kunbaeum]          [webWork]
[hybridApp]           [Neowiz]            [Windows]
[Intel]               [Netmarble]         [_web<spring>]
[java<util>]          [Nexon]             [_web<util>]
JavaTest.class        [oraclex]           [엄마]
JavaTest.java         partner.ini         [준수]
JavaTest1.java        [PerfLogs]          [지수]
[javaWork]            [pkicert]

        6 개 파일                357,827 바이트
       32 개 디렉터리    15,471,747,072 바이트 남음

C:\>
```

● Package 클래스 실행

➤ 명령 수행 : `java test/JavaTest1`



```
관리자: 명령 프롬프트
C:\>
C:\>
C:\>java test/JavaTest1
Java Test111
C:\>cd test
C:\test>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: FEC1-D2FA

C:\test 디렉터리

2013-03-30 오후 03:08 <DIR>          .
2013-03-30 오후 03:08 <DIR>          ..
2013-03-30 오후 03:08                429 JavaTest1.class
                1개 파일                429 바이트
                2개 디렉터리 15,471,742,976 바이트 남음

C:\test>
```

패키지 개요와 종류

▶ 패키지란?

비슷한 종류의 클래스나 인터페이스들을 묶어 패키지화 한다

▶ JDK에서 많은 패키지 제공

- ☞ `java.lang` : 자바 프로그램의 기본적인 기능을 제공. 명시적으로 지정하지 않아도 모든 자바 프로그램에 포함되는 패키지
- ☞ `java.util` : 유용한 유틸리티 클래스를 제공
- ☞ `java.io` : 입출력 기능을 제공하는 패키지
- ☞ `java.net` : 네트워킹과 관련된 기능을 제공하는 패키지.
telnet, ftp, http와 같은 프로토콜을 사용할 수 있는 클래스를 제공
- ☞ `java.awt` : 그래픽 사용자 인터페이스(GUI)를 구축하기 위한 다양한 컴포넌트를 제공하는 패키지
- ☞ `java.awt.event` : AWT 컴포넌트들의 이벤트를 제어하는 패키지
- ☞ `java.applet` : 애플릿 프로그램과 연관된 클래스를 제공하는 패키지

패키지의 사용

- ▶ import 문을 사용하여 패키지 포함

```
import java.util.Date;
```

```
......
```

```
Date date = new Date();    // java.util.Date 클래스만을 사용
```

```
......
```

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

```
.....
```

```
Date date = new Date();
```

```
// java.util 패키지의 모든 클래스를 사용
```

```
Random random = new Random();
```

```
Stack stack = new Stack();
```

```
Hashtable hashtable = new Hashtable();
```

사용자 패키지의 작성 및 사용

- ▶ 사용자가 작성한 클래스를 패키지로 만들어 사용할 수 있다
- ▶ 작성된 클래스를 패키지로 지정하기 위해서는 프로그램의 첫 라인에 다음과 같이 지정하여야 한다

```
package package-name;
```

- ▶ 컴파일 할 때

```
javac -d . 프로그램명.java
```


자바 API Documentation

자바 API란

자바 Application Programming Interface의 약어로서 자바 프로그램을 작성할 수 있도록 썬 마이크로시스템사에서 제공하는 클래스

자바 API Documentation

자바 클래스들에 대한 도움말

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

<https://docs.oracle.com/en/java/javase/11/docs/api/index.html>

자바 API Documentation 문서 다운받아 설치하기

www.oracle.org에 접속하여 화면 위쪽 [Downloads] 항목 중에서 "Java SE"를 선택한다. 화면에서 Documentation을 찾아서 [Download]를 클릭한다.

클릭하여 다운로드한다.

다운로드한 압축 파일을 풀어준다. 압축 파일을 푼 후에 "index.html"을 찾아 웹 브라우저에 띄운다.

자바 API Documentation

<팁> 자바 API Documentation을 선 사이트에서 직접 사용하기

웹 브라우저를 띄우고 자바 API Documentation으로 연결한다.

이 주소를 즐겨찾기에 추가해 두고 클래스나 메소드에 대해서 궁금증이 생길 때마다 참고하면 매우 편리하다.

The screenshot shows the Java™ Platform, Standard Edition 6 API Specification page. The browser window title is "http://java.sun.com - Overview Java Platform SE 6 - Microsoft Internet Explorer". The address bar shows "http://java.sun.com/javase/6/docs/api/". The page has a navigation bar with links: Overview, Package, Class, Use, Tree, Deprecated, Index, Help. The main content area is titled "Java™ Platform, Standard Edition 6 API Specification" and contains the text: "This document is the API specification for version 6 of the Java™ Platform, Standard Edition." Below this is a "See:" section with a link to "Description". The "Packages" section is partially visible, showing "Provides the classes".

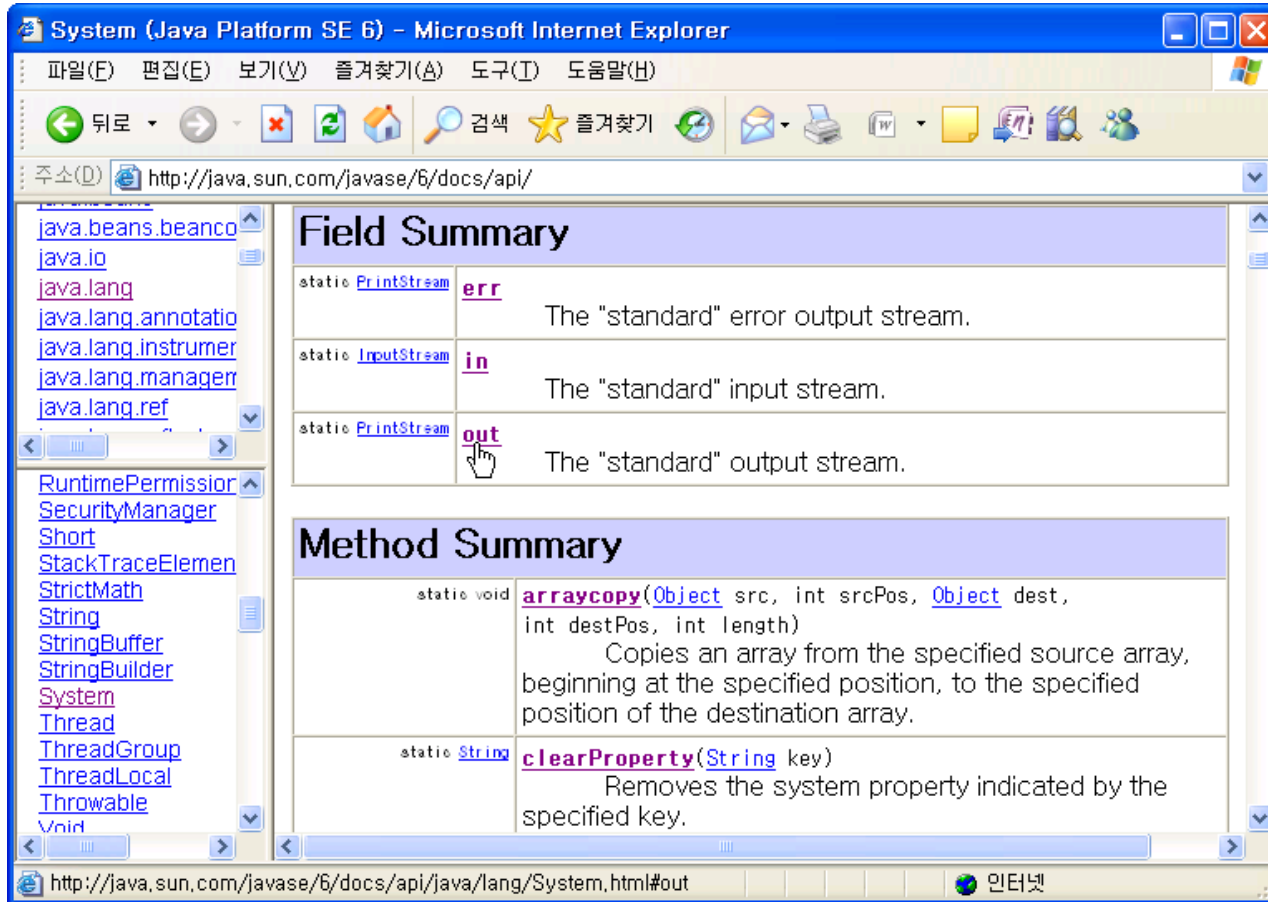
Annotations on the left side of the screenshot:

- 패키지 창 (Package List) pointing to the "Packages" section on the left.
- 클래스 창 (Class List) pointing to the "All Classes" section on the left.
- 설명 창 (Main Content) pointing to the main content area on the right.

자바 API Documentation

<실습하기> 자바 API Documentation 사용하기

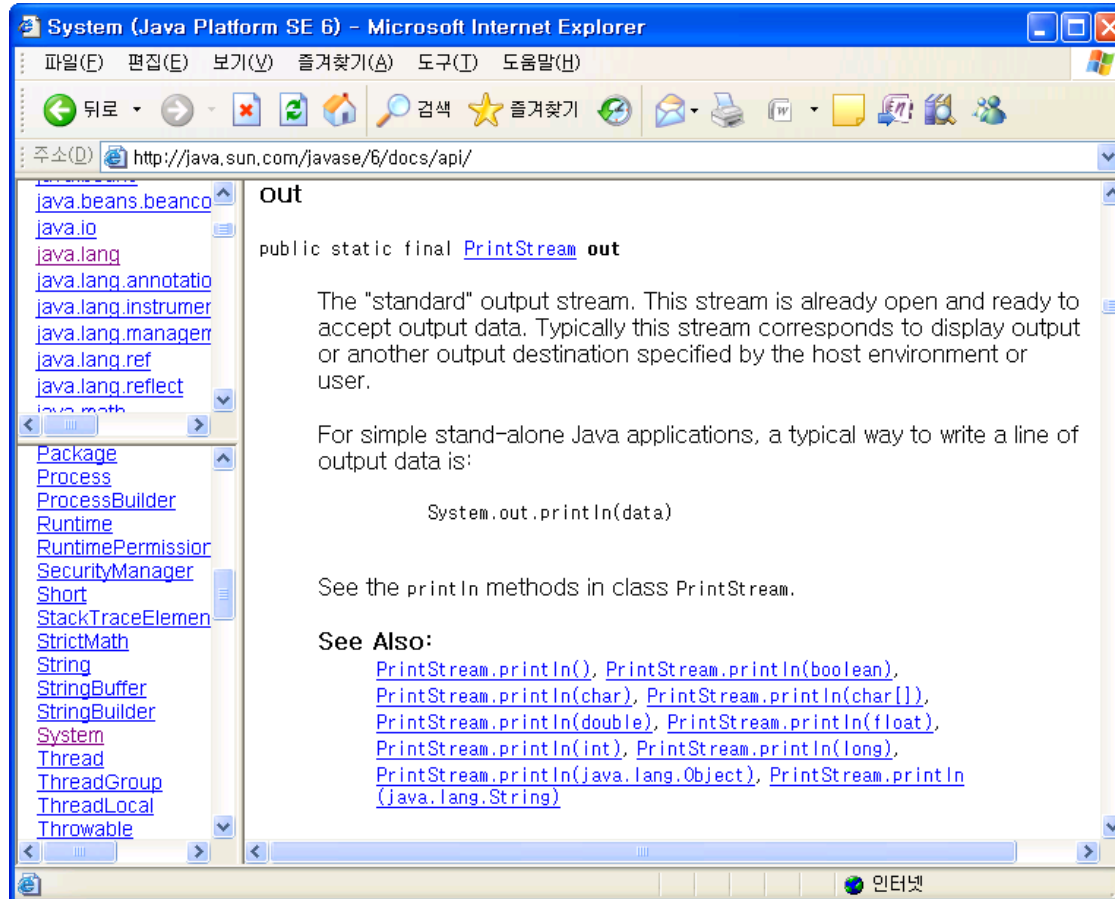
java.lang 패키지의 System.out.println 메소드를 찾기



자바 API Documentation

<실습하기> 자바 API Documentation 사용하기

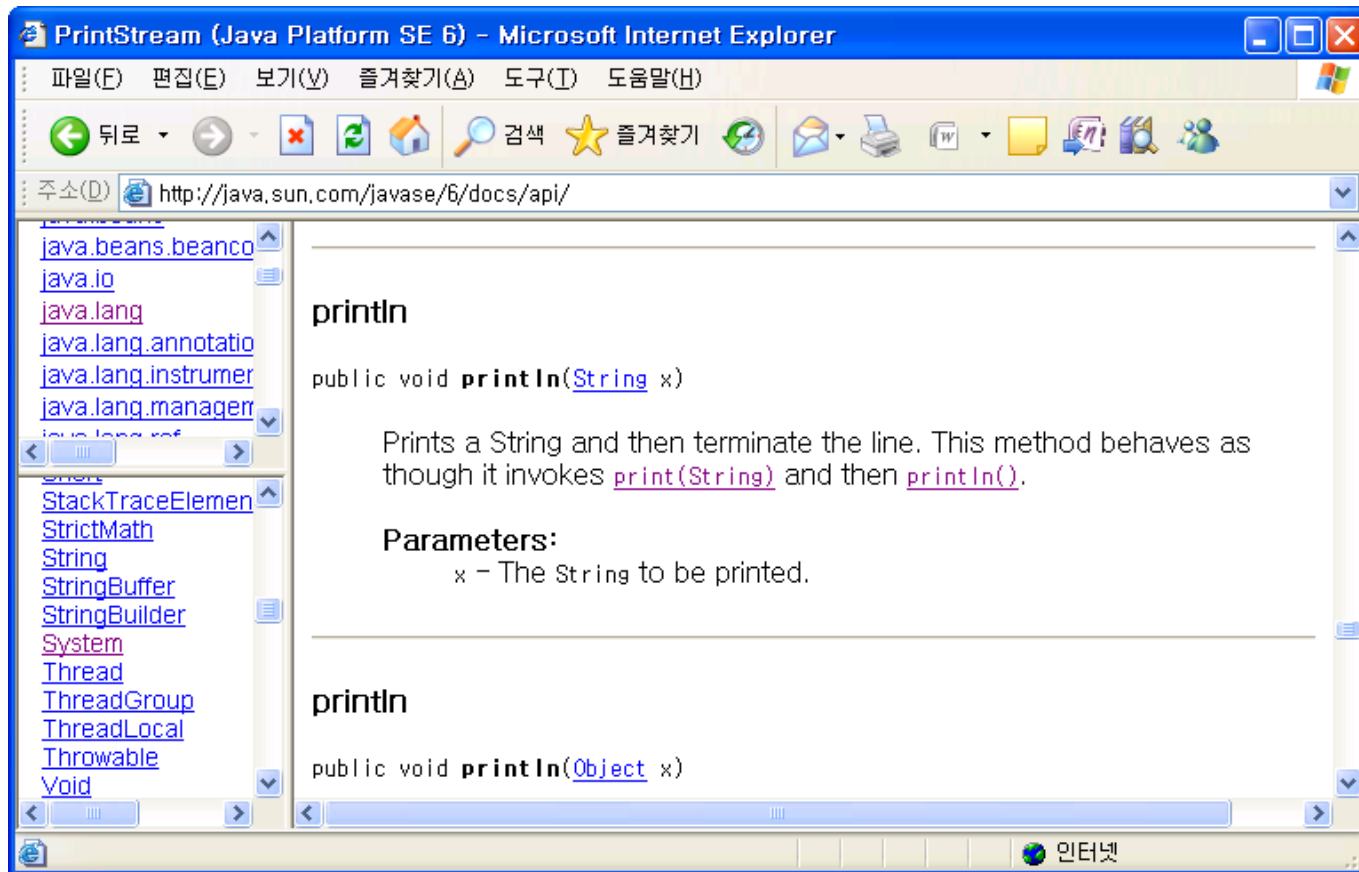
out 필드에 대한 설명과 함께 println 메소드들의 목록이 나타난다.



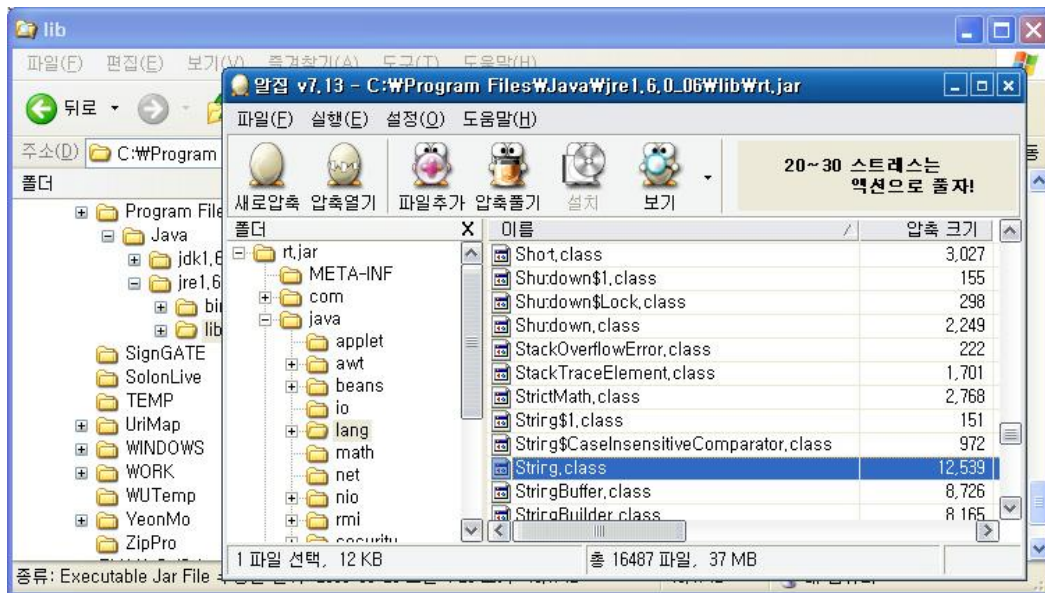
자바 API Documentation

<실습하기> 자바 API Documentation 사용하기

이중 하나를 클릭하여 들어가 보면 `println` 메소드에 대한 설명이 나타난다.



클래스의 위치 정보



Java.base.jar 압축 파일의 내용을 살펴보면 String 클래스 하나만 단독으로 존재하는 것이 아니라 굉장히 다양한 클래스 파일들이 압축되어 있음을 확인할 수 있다.

자바에서는 이렇게 같은 종류의 클래스를 여러 개 묶어서 디렉터리에 저장해 두고 사용하는데 이러한 클래스의 묶음을 패키지(Package)라고 한다.

클래스의 위치 정보

우리는 지금까지 사용해 왔던 "String"은 원칙적으로는 다음과 같이 String 클래스 앞에 java.lang을 덧붙여 사용해야 한다.

String 클래스 앞의 java.lang이 바로 패키지 이름이다.

```
java.lang.String name="홍길동";
```

이 패키지에 이름은 String.class 파일이 존재하는 물리적인 디렉터리 명이다.

import 문

String 클래스를 사용할 때마다 일일이 패키지 이름까지 기술해야 한다면 클래스 이름이 너무 길어서 무척 불편하다.

```
java.lang.String name;  
java.lang.String tel;  
java.lang.String email;
```

```
import java.lang.String;  
String name;  
String tel;  
String email;
```


import 문

```
import java.lang.*;
```

위와 같이 선언하면 패키지에 포함된 클래스 모두를 사용할 수 있다는 의미가 된다.

하지만, 이전 예제를 작성할 때에는

`import java.lang.*;` 문장을 기술하지 않고도 `String` 클래스를 사용하여 왔다.

이는 `java.lang` 패키지가 자동으로 포함되기 때문이다.

ArrayList 클래스명 앞에 패키지 명시하기

```
001: public class Ch09Ex02 {  
002:     public static void main(String[] args) {  
003:         java.util.ArrayList list = new java.util.ArrayList();  
004:  
005:         list.add("하나");  
006:         list.add("둘");  
007:         list.add("셋");  
008:         list.add("넷");  
009:  
010:         for(int i=0; i<list.size(); i++)  
011:             System.out.println( i + " 번째 요소는 " + list.get(i));  
012:     }  
013: }
```

import 구문으로 패키지 지정하기

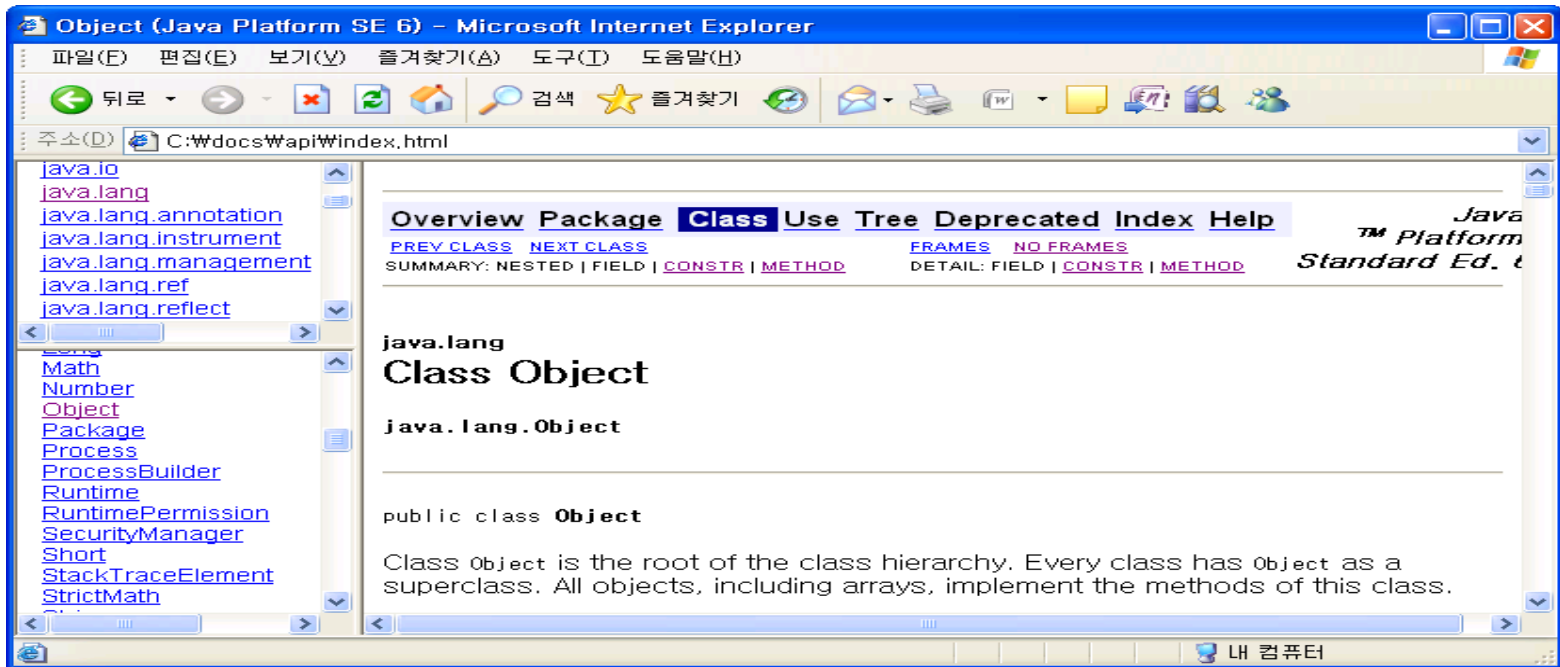
```
001: import java.util.*;
002:
003: public class Ch09Ex03 {
004:     public static void main(String[] args) {
005:         ArrayList list = new ArrayList();
006:
007:         list.add("하나");
008:         list.add("둘");
009:         list.add("셋");
010:         list.add("넷");
011:
012:         for(int i=0; i<list.size(); i++)
013:             System.out.println( i + " 번째 요소는 " + list.get(i));
014:     }
015: }
```

자바 주요 클래스

Object 클래스

모든 자바 클래스의 최상위 클래스

자바의 모든 클래스는 자바의 최상위 클래스인 `java.lang.Object` 클래스로부터 상속을 받도록 설계되어있다



Object 클래스

생성자

public Object()

중요 메소드

protected Object clone()

객체를 복사하는데 사용한다.

public boolean equal(Object obj)

두 객체의 내용이 동일한지 알아볼 때 사용한다.

public int hashCode()

자바에서 객체를 식별하는 정수 값인 해시 코드를 반환한다.

protected void finalize()

객체를 더 이상 사용하지 않을 때 쓰레기 수집 기능을 수행한다.

public Class getClass()

객체의 클래스 이름을 Class 형으로 반환한다.

public String toString()

객체의 문자열을 반환한다.

public void notify()

대기 중인 스레드를 하나 다시 시작한다.

public void notifyall()

대기 중인 모든 스레드를 다시 시작한다.

public void wait()

스레드의 작동을 중지하고 대기 상태로 만든다.

equals(Object obj)

- 객체 자신과 주어진 객체(obj)를 비교한다. 같으면 true, 다르면 false.
- Object클래스에 정의된 equals()는 참조변수 값(객체의 주소)을 비교한다.

```
public boolean equals(Object obj) {  
    return (this==obj);  
}
```

- equals()를 오버라이딩해서 인스턴스변수의 값을 비교하도록 바꾼다.

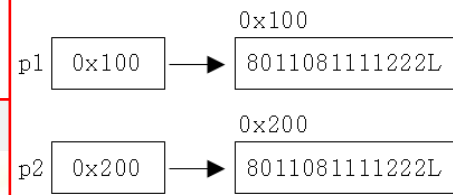
```
class Person {  
    long id;  
  
    public boolean equals(Object obj) {  
        if(obj!=null && obj instanceof Person) {  
            return id == ((Person)obj).id;  
        } else {  
            return false;  
        }  
    }  
  
    Person(long id) {  
        this.id = id;  
    }  
}
```

obj가 Object타입이므로 id값을 참조하기 위해서는 Person타입으로 형변환이 필요하다.

타입이 Person이 아니면 값을 비교할 필요도 없다.

```
Person p1 = new Person(8011081111222L);  
Person p2 = new Person(8011081111222L);
```

```
System.out.println(p1==p2);  
System.out.println(p1.equals(p2));
```



```
class EqualsEx1{
    public static void main(String[] args)    {
        Value v1 = new Value(10);
        Value v2 = new Value(10);
        if (v1.equals(v2)) {
            System.out.println("v1과 v2는 같습니다.");
        } else {
            System.out.println("v1과 v2는 다릅니다.");
        }
        v2 = v1;
        if (v1.equals(v2)) {
            System.out.println("v1과 v2는 같습니다.");
        } else { System.out.println("v1과 v2는 다릅니다.");
        }
    } // main
}

class Value {
    int value;
    Value(int value) {        this.value = value;        }
}
```

```

class Person {
    long id;
    public boolean equals(Object obj) {
        if(obj!=null && obj instanceof Person) {
            return id ==((Person)obj).id;
        }
        // obj가 Object타입이므로 id값을 참조하기 위해서는 Person타입으로 형변환이 필요하다.
        } else {    return false; // 타입이 Person이 아니면 값을 비교할 필요도 없다.
        }
    }
    Person(long id) {        this.id = id;    }
}

```

```

class EqualsEx2 {
    public static void main(String[] args) {
        Person p1 = new Person(8011081111222L);
        Person p2 = new Person(8011081111222L);
        if(p1==p2) {        System.out.println("p1과 p2는 같은 사람입니다.");
        } else {            System.out.println("p1과 p2는 다른 사람입니다."); }

        if(p1.equals(p2)) {
            System.out.println("p1과 p2는 같은 사람입니다.");
        } else {    System.out.println("p1과 p2는 다른 사람입니다."); }
    }
}

```


hashCode()

- 객체의 해시코드(int타입의 정수)를 반환하는 메서드(해시함수)
다량의 데이터를 저장&검색하는 해싱기법에 사용된다.
- Object클래스의 hashCode()는 객체의 내부주소를 반환한다.

```
public class Object {  
    ...  
    public native int hashCode();  
}
```

- equals()를 오버라이딩하면, hashCode()도 같이 오버라이딩 해야한다.
equals()의 결과가 true인 두 객체의 hash code는 같아야하기 때문

```
String str1 = new String("abc");  
String str2 = new String("abc");  
System.out.println(str1.equals(str2)); // true  
System.out.println(str1.hashCode());   // 96354  
System.out.println(str2.hashCode());   // 96354
```

- System.identityHashCode(Object obj)는 Object클래스의 hashCode()와 동일한 결과를 반환한다.

```
System.out.println(System.identityHashCode(str1)); // 3526198  
System.out.println(System.identityHashCode(str2)); // 7699183
```

toString()

- 객체의 정보를 문자열(String)로 제공할 목적으로 정의된 메서드

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

오버라이딩

```
class Card {  
    String kind;  
    int number;  
  
    Card() {  
        this("SPADE", 1);  
    }  
    Card(String kind, int number) {  
        this.kind = kind;  
        this.number = number;  
    }  
}
```

```
public String toString() {  
    // Card인스턴스의 kind와 number를 문자열로 반환한다.  
    return "kind : " + kind + ", number : " + number;  
}
```

```
class CardToString  
{  
    public static void main(String[] args)  
    {  
        Card c1 = new Card();  
        Card c2 = new Card();  
  
        System.out.println(c1.toString());  
        System.out.println(c2.toString());  
    }  
}
```

[실행결과]

Card@47e553
Card@20c10f

[실행결과]

kind : SPADE, number : 1
kind : SPADE, number : 1

```
class Card {
    String kind;
    int number;
    Card() {
        this("SPADE", 1);
    }
    Card(String kind, int number) {
        this.kind = kind;
        this.number = number;
    }
}

class CardToString {
    public static void main(String[] args) {
        Card c1 = new Card();
        Card c2 = new Card();

        System.out.println(c1.toString());
        System.out.println(c2.toString());
    }
}
```

```
class Card {
    String kind;
    int number;
    Card() {
        this("SPADE", 1);
    }
    Card(String kind, int number) {
        this.kind = kind;
        this.number = number;
    }
    public String toString() {
        // Card인스턴스의 kind와 number를 문자열로 반환한다.
        return "kind : " + kind + ", number : " + number;
    }
}

class CardToString2 {
    public static void main(String[] args) {
        Card c = new Card("HEART", 10);
        System.out.println(c.toString());
    }
}
```

clone()

- 객체 자신을 복제(clone)해서 새로운 객체를 생성하는 메서드
- Cloneable인터페이스를 구현한 클래스의 인스턴스만 복제할 수 있다.
- Object클래스에 정의된 clone()은 인스턴스변수의 값만을 복제한다.
- 인스턴스변수가 참조형일 때, 참조하는 객체도 복제되게 오버라이딩해야함.

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

    public String toString() {
        return "x="+x +", y=";
    }

    public Object clone() {
        Object obj=null;
        try {
            obj = super.clone();
        } catch (CloneNotSupportedException e) {
            return obj;
        }
    }
}
```

Cloneable인터페이스를 구현한 클래스에서만 clone()을 호출할 수 있다. 이 인터페이스를 구현하지 않고 clone()을 호출하면 예외가 발생한다.

```
class CloneEx1 {
    public static void main(String[] args) {
        Point original = new Point(3, 5);
        Point copy = (Point)original.clone();
        System.out.println(original);
        System.out.println(copy);
    }
}
```

```
class Circle implements Cloneable {
    Point p; // 원점
    double r; // 반지름

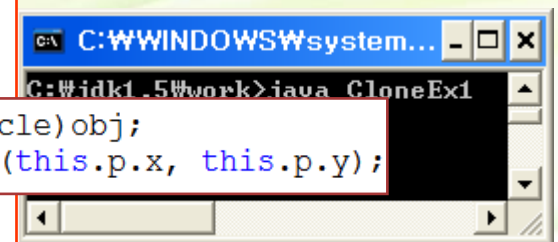
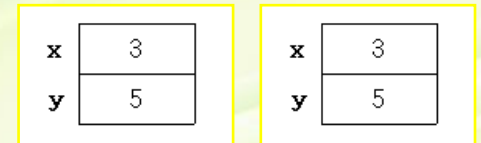
    Circle(Point p, double r) {
        this.p = p;
        this.r = r;
    }

    public Object clone() {
        Object obj = null;
        try {
            obj = super.clone();
        } catch (CloneNotSupportedException e) {}

        return obj;
    }

    public String toString() {
        return p.toString() + ", r="+r;
    }
}
```

```
Circle c1 = new Circle(new Point(10,20), 2.0);
Circle c2 = (Circle)c1.clone();
```



class Point implements Cloneable {

// Cloneable 인터페이스를 구현한 클래스에서만 clone()을 호출할 수 있다. 이 인터페이스를
// 구현하지 않고 clone()을 호출하면 예외가 발생한다.

int x; int y;

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

public String toString() { return "x="+x +", y="+y; }

public Point clone() {

Object obj=null;

try {

obj = super.clone();

} catch(CloneNotSupportedException e) {

// clone메서드에는 CloneNotSupportedException이 선언되어 있으므로

// 이 메서드를 호출할 때는 try-catch문을 사용해야한다.

return (Point)obj;

}

}

class CloneTest {

public static void main(String[] args){

Point original = new Point(3, 5);

Point copy = original.clone(); // 객체를 복제해서 새로운 객체를 만든다.

System.out.println(original);

System.out.println(copy);

}

}

getClass()

- 자신이 속한 클래스의 Class객체를 반환하는 메서드
- Class객체는 클래스의 모든 정보를 담고 있으며, 클래스당 단 1개만 존재
클래스파일(*.class)이 메모리에 로드될때 생성된다.



Card.class파일

ClassLoader

Class객체

- Class객체를 얻는 여러가지 방법

```
Card c = new Card();  
Class cObj = c.getClass();
```

```
Card c2 = new Card();  
Card c2 = (Card)cObj.newInstance();
```

```
Class cObj = Card.class;  
String className = cObj.getName();
```

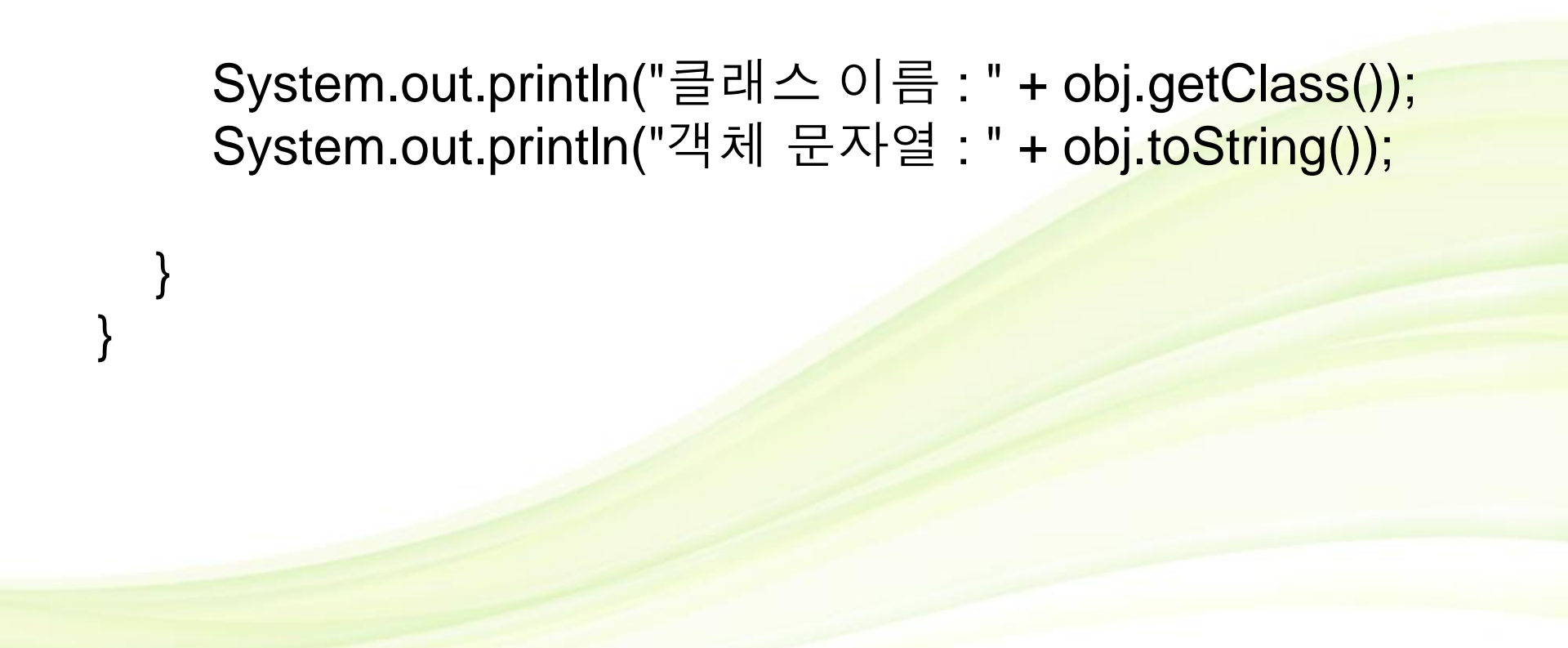
```
String className = Card.class.getName();
```

```
Class cObj = Class.forName("Card");
```



```
class Aclass {  
}
```

```
class Object1 {  
    public static void main(String[] args) {  
        Aclass obj = new Aclass();  
  
        System.out.println("클래스 이름 : " + obj.getClass());  
        System.out.println("객체 문자열 : " + obj.toString());  
    }  
}
```



String 클래스의 생성자와 메서드

메서드 / 설명	예 제	결 과
String(String s) 주어진 문자열(s)을 갖는 String인스턴스를 생성한다.	String s = new String("Hello");	s = "Hello"
String(char[] value) 주어진 문자열(value)을 갖는 String인스턴스를 생성한다.	char[] c = {'H', 'e', 'l', 'l', 'o'} String s = new String(c);	s = "Hello"
String(StringBuffer buf) StringBuffer인스턴스가 갖고 있는 문자열과 같은 내용의 String인스턴스를 생성한다.	StringBuffer sb = new StringBuffer("Hello"); String s = new String(sb);	s = "Hello"
char charAt(int index) 지정된 위치(index)에 있는 문자를 알려준다. (index는 0부터 시작)	String s = "Hello"; String n = "0123456"; char c = s.charAt(1); char c2 = n.charAt(1);	c = 'e' c2 = '1'
String concat(String str) 문자열(str)을 뒤에 덧붙인다.	String s = "Hello"; String s2 = s.concat(" World");	s2 = "Hello World"
boolean contains(CharSequence s) 지정된 문자열(s)이 포함되었는지 검사한다.	String s = "abcedfg"; boolean b = s.contains("bc");	b = true
boolean endsWith(String suffix) 지정된 문자열(suffix)로 끝나는지 검사한다.	String file = "Hello.txt"; boolean b = file.endsWith("txt");	b = true
boolean equals(Object obj) 매개변수로 받은 문자열(obj)과 String인스턴스의 문자열을 비교한다. obj가 String이 아니거나 문자열이 다르면 false를 반환한다.	String s = "Hello"; boolean b = s.equals("Hello"); boolean b2 = s.equals("hello");	b = true b2 = false
boolean equalsIgnoreCase(String str) 문자열과 String인스턴스의 문자열을 대소문자 구분없이 비교한다.	String s = "Hello"; boolean b = s.equalsIgnoreCase("HELLO"); boolean b2 = s.equalsIgnoreCase("heLLo");	b = true b2 = true
int indexOf(int ch) 주어진 문자(ch)가 문자열에 존재하는지 확인하여 위치(index)를 알려준다. 못 찾으면 -1을 반환한다. (index는 0부터 시작)	String s = "Hello"; int idx1 = s.indexOf('o'); int idx2 = s.indexOf('k');	idx1 = 4 idx2 = -1

String클래스의 생성자와 메서드

int indexOf(String str) 주어진 문자열이 존재하는지 확인하여 그 위치(index)를 알려준다. 없으면 -1을 반환한다. (index는 0부터 시작)	String s = "ABCDEFGFG"; int idx = s.indexOf("CD");	idx = 2
String intern() 문자열을 constant pool에 등록한다. 이미 constant pool에 같은 내용의 문자열이 있을 경우 그 문자열의 주소값을 반환한다.	String s = new String("abc"); String s2 = new String("abc"); boolean b = (s==s2); boolean b2 = s.equals(s2); boolean b3 = (s.intern()==s2.intern());	b = false b2 = true b3 = true
int lastIndexOf(int ch) 지정된 문자 또는 문자코드를 문자열의 오른쪽 끝에서부터 찾아서 위치(index)를 알려준다. 못 찾으면 -1을 반환한다.	String s = "java.lang.Object"; int idx1 = s.lastIndexOf('.'); int idx2 = s.indexOf('.');	idx1 = 9 idx2 = 4
int lastIndexOf(String str) 지정된 문자열을 인스턴스의 문자열 끝에서 부터 찾아서 위치(index)를 알려준다. 못 찾으면 -1을 반환한다.	String s = "java.lang.java"; int idx1 = s.lastIndexOf("java"); int idx2 = s.indexOf("java");	idx1 = 10 idx2 = 0
int length() 문자열의 길이를 알려준다.	String s = "Hello"; int length = s.length();	length = 5
String replace(char old, char nw) 문자열 중의 문자(old)를 새로운 문자(nw)로 바꾼 문자열을 반환한다.	String s = "Hello"; String s1 = s.replace('H', 'C');	s1 = "Cello"
String replace(CharSequence old, CharSequence nw) 문자열 중의 문자열(old)을 새로운 문자열(nw)로 모두 바꾼 문자열을 반환한다.	String s = "Hellollo"; String s1 = s.replace("ll", "LL");	s1 = "HeLLoLLo"
String replaceAll(String regex, String replacement) 문자열 중에서 지정된 문자열(regex)과 일치하는 것을 새로운 문자열(replacement)로 모두 변경한다.	String ab = "AABBAABB"; String r = ab.replaceAll("BB", "bb");	r = "AAbbAAbb"
String replaceFirst(String regex, String replacement) 문자열 중에서 지정된 문자열(regex)과 일치 하는 것 중, 첫 번째 것만 새로운 문자열(replacement)로 변경한다.	String ab = "AABBAABB"; String r = ab.replaceFirst("BB", "bb");	r = "AAbbAABB"

String클래스의 생성자와 메서드

String[] split(String regex) 문자열을 지정된 분리자(regex)로 나누어 문자열 배열에 담아 반환한다.	<pre>String animals = "dog,cat,bear"; String[] arr = animals.split(",");</pre>	<pre>arr[0] = "dog" arr[1] = "cat" arr[2] = "bear"</pre>
String[] split(String regex, int limit) 문자열을 지정된 분리자(regex)로 나누어 문자열배열에 담아 반환한다. 단, 문자열 전체를 지정된 수(limit)로 자른다.	<pre>String animals = "dog,cat,bear"; String[] arr = animals.split(",", 2);</pre>	<pre>arr[0] = "dog" arr[1] = "cat,bear"</pre>
boolean startsWith(String prefix) 주어진 문자열(prefix)로 시작하는지 검사한다.	<pre>String s = "java.lang.Object"; boolean b =s.startsWith("java"); boolean b2=s.startsWith("lang");</pre>	<pre>b = true b2 = false</pre>
String substring(int begin) String substring(int begin, int end) 주어진 시작위치(begin)부터 끝 위치(end) 범위에 포함된 문자열을 얻는다. 이 때, 시작위치의 문자는 범위에 포함되지만, 끝 위치의 문자는 포함되지 않는다.	<pre>String s = "java.lang.Object"; String c = s.substring(10); String p = s.substring(5,9);</pre>	<pre>c = "Object" p = "lang"</pre>
String toLowerCase() String인스턴스에 저장되어있는 모든 문자열을 소문자로 변환하여 반환한다.	<pre>String s = "Hello"; String s1 = s.toLowerCase();</pre>	<pre>s1 = "hello"</pre>
String toString() String인스턴스에 저장되어 있는 문자열을 반환한다.	<pre>String s = "Hello"; String s1 = s.toString();</pre>	<pre>s1 = "Hello"</pre>
String toUpperCase() String인스턴스에 저장되어있는 모든 문자열을 대문자로 변환하여 반환한다.	<pre>String s = "Hello"; String s1 = s.toUpperCase();</pre>	<pre>s1 = "HELLO"</pre>
String trim() 문자열의 왼쪽 끝과 오른쪽 끝에 있는 공백을 없앤 결과를 반환한다. 이 때 문자열 중간에 있는 공백은 제거되지 않는다.	<pre>String s = " Hello World "; String s1 = s.trim();</pre>	<pre>s1 = "Hello World"</pre>
static String valueOf(boolean b) static String valueOf(char c) static String valueOf(int i) static String valueOf(long l) static String valueOf(float f) static String valueOf(double d) static String valueOf(Object o)	<pre>String b = String.valueOf(true); String c = String.valueOf('a'); String i = String.valueOf(100); String l = String.valueOf(100L); String f = String.valueOf(10f); String d = String.valueOf(10.0); java.util.Date dd =new java.util.Date(); String date = String.valueOf(dd);</pre>	<pre>b = "true" c = "a" i = "100" l = "100" f = "10.0" d = "10.0" date = "Sun Jan 27 21:26:29 KST 2008"</pre>
지정된 값을 문자열로 변환하여 반환한다. 참조변수의 경우, toString()을 호출한 결과를 반환한다.		

String

문자열 상수를 생성자에 전달해서 String 객체를 생성한 예

예 `String str = new String("java");`

String 객체에 문자열 리터럴을 저장

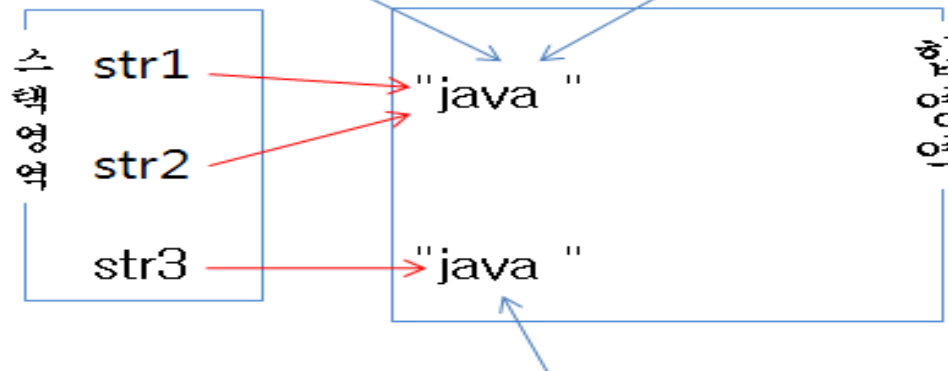
예 `String str = "java";`

`String str1 = "java ";`

문자열 상수는 컴파일하면 자동으로 힙 영역에 String 객체가 생성됨

`String str2 = "java ";`

이미 존재하는 문자열 상수로 다시 사용한다면 이미 생성된 문자열 상수를 공유하게 된다.



`String str3 = new String ("java ");`

상수풀

상수풀은 클래스와 같은 Heap의 Permanent area(고정 영역)에 생성되어 Java 프로세스의 종료까지 그 생을 함께 합니다.

String을 new로 생성하지 않고 "" 리터럴을 사용하여 생성할 경우, 내부적으로 new String() 메소드 호출 이후에 String.intern()이라는 메소드가 호출되어 고유의 인스턴스를 공유하도록 interned 됩니다. 이것은 생성한 String을 Constant pool에 등록하는 (만약 이전에 같은 char sequence의 문자열이 이미 상수풀에 있다면 문자열을 힙에서 해제하고 그 상수풀의 레퍼런스를 반환) 작업을 수행.

즉, 같은 패키지의 같은 클래스 내에서는 정수(literal) 스트링들은 동일한 String 객체를 참조한다. 이것은 메모리를 절약하는 효과가 있다. 자바는 속도보단 유지 보수가 주 목적이다
여긴 힙이랑 별개의 공간이라서 삭제되지 않고 프로그램이 끝나면 삭제된다

■ 데이터 영역(Data Area)

데이터 영역은 전역 변수와 **static** 변수가 할당되는 영역이다. 이 영역에 할당되는 변수들은 일반적으로 프로그램의 시작과 동시에 할당되고, 프로그램이 종료되어야만 메모리에서 소멸된다.

■ 스택 영역(Stack Area)

스택 영역은 함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역이다.

■ 힙 영역(Heap Area)

힙 영역은 프로그래머가 관리하는 메모리 영역이다. 즉 프로그래머의 필요에 의해서 메모리 공간이 할당 및 소멸되는 영역이다. 동적 할당으로 생성되는 메모리 영역이다



빈 문자열("", empty string)

- 내용이 없는 문자열. 크기가 0인 char형 배열을 저장하는 문자열
- 크기가 0인 배열을 생성하는 것은 어느 타입이나 가능

```
char[] cArr = new char[0]; // 크기가 0인 char배열  
int[] iArr = {};          // 크기가 0인 int배열
```

- String str=""은 가능해도 char c = "";는 불가능
- String은 참조형의 기본값인 null 보다 빈 문자열로 초기화하고 char형은 기본값인 '\u0000'보다 공백으로 초기화하자.

```
String s = null;  
char c = '\u0000';
```



```
String s = ""; // 빈 문자열로 초기화  
char c = ' '; // 공백으로 초기화
```

```
String str1 = "";  
String str2 = "";  
String str3 = "";
```

```
String str4 = new String("");  
String str5 = new String("");  
String str6 = new String("");
```

```
class StrCompare {  
    public static void main(String[] args) {
```

```
        String str1 = new String("java");  
        String str2 = "java";  
        String str3 = "java";  
        if (str1.equals(str2))  
            System.out.println("equal 1,2");  
        if (str2.equals(str3))  
            System.out.println("equal 2,3");  
        if (str1 == str2)  
            System.out.println("== 1,2");  
        if (str2 == str3)  
            System.out.println("== 2,3");
```

```
    }  
}
```

```
class String1 {  
    public static void main(String [] args){  
        char[] s = {'k','o','r','e','a'};  
        String str1 = "Hello";           // 문자열 대입  
  
        String str2 = new String("java"); // 문자열 생성자  
        String str3 = new String(s);      // 문자 배열을 이용한 생성자  
        System.out.println(str1);  
        System.out.println(str2);  
        System.out.println(str3);  
        System.out.println(str1 + str2 + str3);  
    }  
}
```



```
class String2 {  
    public static void main(String[] args) {  
        int a = 100;           // 정수  
        float b = 10.0f;       // 부동소수점  
        String str = "점수 = "; // 문자열  
  
        // 문자열 + 숫자 + '문자' + 숫자  
        System.out.println(str + a + ',' + b);  
        // 숫자 + 숫자 + 숫자 + 숫자 + 문자  
        System.out.println(2+0+0+2+" 월드컵");  
        // 문자 + 숫자 + 숫자 + 숫자  
        System.out.println("월드컵 " +2+0+0+2);  
    }  
}
```

```
class String4 {  
    public static void main(String[] args) {  
        int i;  
        String str = "2002 WorldCup Korea";  
        System.out.println("정상 문자열 :"+ str );  
        System.out.print("문자열 뒤집기 :");  
        for(i = str.length()-1; i >= 0; i--)  
            System.out.print(str.charAt(i));  
        System.out.print("\n짝수 문자열 :");  
        for(i = 0; i < str.length(); i++){  
            if(i%2 != 0)    System.out.print(  
str.charAt(i));  
            else            System.out.print(" ");  
        }  
    }  
}
```

```
class String9 {  
    public static void main(String[] args) {  
        String str1 = "JAVA";  
        String str2 = "Java";  
        System.out.println("문자열 str1 = " + str1);  
        System.out.println("문자열 str2 = " + str2);  
        System.out.println("JAVA와 java는 " +  
            ((str1.equals(str2))?"동일":"틀림"));  
        System.out.println("JAVA와 java는 " +  
            ((str1.equalsIgnoreCase(str2))?"동일":"틀림"));  
    }  
}
```

문자열과 기본형간의 변환

- 기본형 값을 문자열로 바꾸는 두 가지 방법(방법2가 더 빠름)

```
int i = 100;  
String str1 = i + ""; // 100을 "100"으로 변환하는 방법1  
String str2 = String.valueOf(i); // 100을 "100"으로 변환하는 방법2
```

- 문자열을 기본형 값으로 변환하는 방법

```
int i = Integer.parseInt("100"); // "100"을 100으로 변환하는 방법1  
int i2 = Integer.valueOf("100"); // "100"을 100으로 변환하는 방법2(JDK1.5이후)  
char c = "A".charAt(0); // 문자열 "A"를 문자 'A'로 변환하는 방법
```

기본형 → 문자열	문자열 → 기본형
String valueOf(boolean b)	boolean Boolean.getBoolean(String s)
String valueOf(char c)	byte Byte.parseByte(String s)
String valueOf(int i)	short Short.parseShort(String s)
String valueOf(long l)	int Integer.parseInt(String s)
String valueOf(float f)	long Long.parseLong(String s)
String valueOf(double d)	float Float.parseFloat(String s)
	double Double.parseDouble(String s)

```

class StringEx7{
    public static void main(String[] args)    {
        int value = 100;
        String strValue = String.valueOf(value); // int를 String으로 변환한다.
        int value2 = 100;
        String strValue2 = value2 + ""; // int를 String으로 변환하는 또 다른 방법
        System.out.println(strValue);
        System.out.println(strValue2);
    }
}

class StringEx9 {
    public static void main(String[] args) {
        String fullName = "Hello.java";
        // fullName에서 '.'의 위치를 찾는다.
        int index = fullName.indexOf('.');
        // fullName의 첫번째 글자부터 '.'이 있는 곳까지 문자열을 추출한다.
        String fileName = fullName.substring(0, index);
        // '.'의 다음 문자 부터 시작해서 문자열의 끝까지 추출한다.
        // fullName.substring(index+1, fullName.length());의 결과와 같다.
        String ext = fullName.substring(index+1);
        System.out.println(fullName + "의 확장자를 제외한 이름은 " + fileName);
        System.out.println(fullName + "의 확장자는 " + ext);
    }
}

```

StringBuffer 클래스의 특징

- String처럼 문자형 배열(char[])을 내부적으로 가지고 있다.

```
public final class StringBuffer implements java.io.Serializable
{
    private char[] value;
    ...
}
```

- 그러나, String클래스와 달리 내용을 변경할 수 있다.(mutable)

```
StringBuffer sb = new StringBuffer("abc");
sb.append("123");
```

- 인스턴스를 생성할 때 버퍼(배열)의 크기를 충분히 지정해주는 것이 좋다.
(버퍼가 작으면 성능 저하 - 작업 중에 더 큰 배열의 생성이 필요)

```
public StringBuffer(int length) {
    value = new char[length];
    shared = false;
}
```

```
public StringBuffer(String str) {
    this(str.length() + 16);
    append(str);
}
```

- String클래스와 달리 equals()를 오버라이딩하지 않았다.

```
StringBuffer sb = new StringBuffer("abc");
StringBuffer sb2 = new StringBuffer("abc");
System.out.println(sb==sb2);           // false
System.out.println(sb.equals(sb2)); // false
```

```
String s  = sb.toString();
String s2 = sb2.toString();
System.out.println(s.equals(s2)); // true
```

StringBuffer 클래스의 생성자와 메서드

메서드 / 설명	예제 / 결과
StringBuffer()	<code>StringBuffer sb = new StringBuffer();</code>
16문자를 담을 수 있는 버퍼를 가진 StringBuffer 인스턴스를 생성한다.	<code>sb = ""</code>
StringBuffer(int length)	<code>StringBuffer sb = new StringBuffer(10);</code>
지정된 개수의 문자를 담을 수 있는 버퍼를 가진 StringBuffer 인스턴스를 생성한다.	<code>sb = ""</code>
StringBuffer(String str)	<code>StringBuffer sb = new StringBuffer("Hi");</code>
지정된 문자열 값(str)을 갖는 StringBuffer 인스턴스를 생성한다.	<code>sb = "Hi"</code>
StringBuffer append(boolean b) StringBuffer append(char[] str) StringBuffer append(float f) StringBuffer append(long l) StringBuffer append(String str)	StringBuffer append(char c) StringBuffer append(double d) StringBuffer append(int i) StringBuffer append(Object obj)
매개변수로 입력된 값을 문자열로 변환하여 StringBuffer 인스턴스가 저장하고 있는 문자열의 뒤에 덧붙인다.	<code>StringBuffer sb = new StringBuffer("abc");</code> <code>StringBuffer sb2 = sb.append(true);</code> <code>sb.append('d').append(10.0f);</code> <code>StringBuffer sb3 =</code> <code>sb.append("ABC").append(123);</code>
int capacity()	<code>sb = "abctrue10.0ABC123"</code> <code>sb2 = "abctrue10.0ABC123"</code> <code>sb3 = "abctrue10.0ABC123"</code>
	<code>StringBuffer sb = new StringBuffer(100);</code> <code>sb.append("abcd");</code> <code>int bufferSize = sb.capacity();</code> <code>int stringSize = sb.length();</code>
StringBuffer 인스턴스의 버퍼 크기를 알려준다. length()는 버퍼에 담긴 문자열의 크기를 알려준다.	<code>bufferSize = 100</code> <code>stringSize = 4 (sb에 담긴 문자열이 "abcd"이므로)</code>
char charAt(int index)	<code>StringBuffer sb = new StringBuffer("abc");</code> <code>char c = sb.charAt(2);</code>
지정된 위치(index)에 있는 문자를 반환한다.	<code>c = 'c'</code>
StringBuffer delete(int start, int end)	<code>StringBuffer sb = new StringBuffer("0123456");</code> <code>StringBuffer sb2 = sb.delete(3, 6);</code>
시작위치(start)부터 끝 위치(end) 사이에 있는 문자를 제거한다. 단, 끝 위치의 문자는 제외.	<code>sb = "0126"</code> <code>sb2 = "0126"</code>
StringBuffer deleteCharAt(int index)	<code>StringBuffer sb = new StringBuffer("0123456");</code> <code>sb.deleteCharAt(3);</code>
지정된 위치(index)의 문자를 제거한다.	<code>sb = "012456"</code>

StringBuffer클래스의 생성자와 메서드

메서드 / 설명	예 제 / 결 과
	<code>StringBuffer sb = new StringBuffer("0123456");</code>
StringBuffer insert(int pos, boolean b) StringBuffer insert(int pos, char[] str) StringBuffer insert(int pos, float f) StringBuffer insert(int pos, double d) StringBuffer insert(int pos, Object obj) StringBuffer insert(int pos, char c) StringBuffer insert(int pos, int i) StringBuffer insert(int pos, long l) StringBuffer insert(int pos, String str)	<code>sb.insert(4, '.');</code>
두 번째 매개변수로 받은 값을 문자열로 변환하여 지정된 위치(pos)에 추가한다. pos는 0부터 시작	<code>sb = "0123.456"</code>
int length()	<code>int length = sb.length();</code>
StringBuffer인스턴스에 저장되어 있는 문자열의 길이를 반환한다.	<code>length = 7</code>
StringBuffer replace(int start, int end, String str)	<code>sb.replace(3, 6, "AB");</code>
지정된 범위(start~end)의 문자들을 주어진 문자열로 바꾼다. end위치의 문자는 범위에 포함안됨.	<code>sb = "012AB6"</code> "345"를 "AB"로 바꿨다.
StringBuffer reverse()	<code>sb.reverse();</code>
StringBuffer인스턴스에 저장되어 있는 문자열의 순서를 거꾸로 나열한다.	<code>sb = "6543210"</code>
void setCharAt(int index, char ch)	<code>sb.setCharAt(5, 'o');</code>
지정된 위치의 문자를 주어진 문자(ch)로 바꾼다.	<code>sb = "01234o6"</code>
void setLength(int newLength)	<code>sb.setLength(5);</code> <code>StringBuffer sb2=new StringBuffer("0123456");</code> <code>sb2.setLength(10);</code> <code>String str = sb2.toString().trim();</code>
지정된 크기로 문자열의 길이를 변경한다. 크기를 늘리는 경우에 나머지 빈 공간을 널문자 '\u0000'로 채운다.	<code>sb = "01234"</code> <code>sb2 = "0123456 "</code> <code>str = "0123456"</code>
String toString()	<code>String str = sb.toString();</code>
StringBuffer인스턴스의 문자열을 String으로 반환한다.	<code>str = "0123456"</code>
String substring(int start) String substring(int start, int end)	<code>String str = sb.substring(3);</code> <code>String str2 = sb.substring(3, 5);</code>
지정된 범위 내의 문자열을 String으로 뽑아서 반환 한다. 시작위치(start)만 지정하면 시작위치부터 문자열 끝까지 뽑아서 반환한다.	<code>str = "3456"</code> <code>str2 = "34"</code>

SpringBuffer

```
public class Ex11 {  
    public static void main(String[] args) {  
        StringBuffer str = new StringBuffer();  
        int len=str.length();  
        int size=str.capacity();  
  
        System.out.println(str + " / " + len + " / " + size);  
  
        str.append("누구든지 사랑하기 위해선 "); //문자열 추가  
        len=str.length();  
        size=str.capacity();  
        System.out.println(str + " / " + len + " / " + size);  
  
        str.append("한 번쯤 증오의 가슴이어야 했다.");  
        len=str.length();  
        size=str.capacity();  
        System.out.println(str + " / " + len + " / " + size);  
    }  
}
```

SpringTokenizer

```
import java.util.StringTokenizer;
public class Ex13 {
    public static void main(String[] args) {
        StringTokenizer stok01=
            new StringTokenizer("사과,바나나,귤,오렌지,키위", ",");
        while(stok01.hasMoreTokens())
            //토큰이 있으면
            System.out.println(stok01.nextToken());
            //차례대로 파싱된 문자열을 얻어온다.
        }
    }
```

Wrapper 클래스

Wrapper 클래스는 기본 자료 형들을 객체처럼 사용할 수 있도록 기본 자료 형에 대응되는 클래스들을 말한다.

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

Wrapper 클래스

중요 메소드

<code>public byte byteValue()</code>	객체의 값을 byte 형으로 반환
<code>public double doubleValue()</code>	객체의 값을 double 형으로 반환
<code>public float floatValue()</code>	객체의 값을 float 형으로 반환
<code>public int intValue()</code>	객체의 값을 int 형으로 반환
<code>public long longValue()</code>	객체의 값을 long 형으로 반환
<code>public short shortValue()</code>	객체의 값을 short 형으로 반환

Integer 클래스

int 형에 대한 래퍼 클래스인 Integer 클래스가 제공하는 다양한 메소드

생성자	
Integer (int value)	int 값을 매개변수로 갖는 생성자
Integer (String str)	문자열을 매개변수로 갖는 생성자
중요 메소드	
public static Integer decode(String str)	문자열을 Integer 객체로 변환
public static int parseInt(String str)	문자열을 int 값으로 변환
public static int parseInt(String str, int radix)	문자열에 특정 지정된 진법의 int 값으로 변환
public static Integer valueOf(int i)	int 형을 Integer 객체로 변환
public static Integer valueOf(String str)	문자열을 Integer 객체로 변환
public static String toBinaryString(int num)	int 형을 2진수로 표현된 문자열 반환
public static String toHexString(int num)	int 형을 16진수로 표현된 문자열 반환
public static String toOctalString(int num)	int 형을 8진수로 표현된 문자열 반환

Double 클래스

실수 값을 대표해서 **Double** 클래스만 살펴보기로 하자. 다음은 생성자와 주요 메소드를 정리한 표이다.

생성자	
Double (double value)	double 형을 매개변수로 갖는 생성자
Double (String str)	문자열을 매개변수로 갖는 생성자
중요 메소드	
public boolean isNaN()	객체가 갖고 있는 값이 숫자가 아니면 (NaN:Not a Number)true, 숫자이면 false 반환
public boolean isInfinite()	객체의 값이 무한히 크거나 작다면 (NEGATIVE_INFINITY 또는 POSITIVE_INFINITY이면) true, 아니면 false 값 반환
public static Double valueOf(String str)	str로 지정된 문자열에 해당되는 Double 객체를 반환

Character 클래스

생성자

Character(char value)

중요 메소드

public static boolean isDefined(char ch)	ch가 유니코드이면 true, 아니면 false
public static boolean isDigit(char ch)	ch가 숫자면 true, 아니면 false
public static boolean isLetter(char ch)	ch가 문자면 true, 아니면 false
public static boolean isLetterOrDigit(char ch)	ch가 문자.숫자면 true, 아니면 false
public static boolean isLowerCase(char ch)	ch가 소문자면 true, 아니면 false
public static boolean isSpace(char ch)	ch가 공백이면 true, 아니면 false
public static boolean isUpperCase(char ch)	ch가 대문자면 true, 아니면 false
public static char toLowerCase(char ch)	ch를 소문자로 변형
public static char toUpperCase(char ch)	ch를 대문자로 변형

Wrapper 클래스

```
class Wrapper1 {  
    public static void main(String [] args){  
  
        char obj[] = {'1','a',' ','#','B'};  
        for(int i = 0 ; i< obj.length ; i++){  
            System.out.println( "문자 ["+ obj[i] +"] :");  
            if(Character.isDefined(obj[i]))  
                System.out.println(" 유니 코드입니다.");  
            if(Character.isDigit(obj[i]))  
                System.out.println(" 숫자입니다.");  
            if(Character.isLetter(obj[i]))  
                System.out.println(" 문자입니다.");  
            if(Character.isLowerCase(obj[i]))  
                System.out.println(" 소문자 입니다.");  
            if(Character.isWhitespace(obj[i]))  
                System.out.println(" 공백 입니다.");  
            if(Character.isUpperCase(obj[i]))  
                System.out.println(" 대문자 입니다.");  
            System.out.println("=====");  
        }  
    }  
}
```


Wrapper 클래스

```
class Wrapper2 {  
    public static void main(String [] args){  
  
        char obj[] = new char[args[0].length()];  
        args[0].getChars( 0, args[0].length(), obj, 0 );  
  
        for(int i = 0 ; i< obj.length ; i++){  
            System.out.println( "입력된 문자 ["+ obj[i] +"] :");  
            if(Character.isDefined(obj[i]))  
                System.out.println(" 유니 코드입니다.");  
            if(Character.isDigit(obj[i]))  
                System.out.println(" 숫자입니다.");  
            if(Character.isLetter(obj[i]))  
                System.out.println(" 문자입니다.");  
            System.out.println("=====");  
        }  
    }  
}
```

Wrapper 클래스

```
class Wrapper3 {  
    public static void main(String[] args){  
        Boolean obj1 = new Boolean(true) ;  
        Boolean obj2 = new Boolean("true");  
        if( obj1.booleanValue() )  
            System.out.println("obj1 객체는 true 입니다.");  
        if( obj2.booleanValue() )  
            System.out.println("obj2 객체는 true 입니다.");  
    }  
}
```

오토언박싱(autoUnBoxing)

- Boxing : 기본 타입의 값을 갖는 객체를 생성
- UnBoxing : 포장 객체에서 기본 타입의 값을 얻는 과정
- Jdk1.4버전까지 객체 타입을 기본형에 넣으려면

```
Integer intObj = new Integer("1");  
int intValue = intObj.intValue();
```

- 1.5버전부터 오토언박싱 기능 제공

```
Integer intObj = new Integer("1");  
int intValue = intObj;
```

```
public class AutoBoxingTest {  
    public static void main(String[] args) {  
        //1.4 이하  
        int var_int1 = 3;   Integer intObj1 = new Integer(var_int1);  
        //1.5 이상  
        intObj1 = var_int1;      System.out.println("intObj1 = " + intObj1);  
        //1.4 이하  
        Integer intObj2 = new Integer("4");   int var_int2 = intObj2.intValue();  
        //1.5 이상  
        var_int2 = intObj2;      System.out.println("var_int2 = " + var_int2);  
    }  
}
```

```
class AutoBoxingUnboxing {  
    public static void main(String[] args) {  
        Integer iValue = 10;           // autoboxing  
        Double dValue = 3.14;  
  
        System.out.println(iValue);  
        System.out.println(dValue);  
  
        int num1 = iValue;              // autounboxing  
        double num2 = dValue;  
        System.out.println(num1);  
        System.out.println(num2);  
    }  
}
```

날짜와 시간 관련 클래스

Calendar 클래스는 생성자가 protected이므로 생성자를 이용해서 직접 객체를 생성해서 사용하는 것이 아니라, static 메소드인 `getInstance()` 메소드를 사용하여 객체를 얻어와서 사용한다.

예

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



시스템으로부터 현재 시간 정보를 갖는 객체를 생성

날짜와 시간 관련 클래스

Date 클래스

날짜를 표현하는 클래스

날짜 정보를 객체간에 주고 받을 때 주로 사용

```
public class DateExample {  
    public static void main(String[] args) {  
        Date now = new Date();  
        String strNow1 = now.toString();  
        System.out.println(strNow1);  
    }  
}
```

날짜와 시간 관련 클래스

Calendar 객체안의 정보는 get() 메서드를 사용해야 하는데, 다음 표의 상수들을 이용하여 날짜와 시간에 관한 다양한 정보를 얻을 수 있다.

상수	의미	상수	의미
SECOND	초	DAY_OF_YEAR	그 해에서의 날짜
MINUTE	분	WEEK_OF_MONTH	그 달에서의 몇 째 주인지
HOUR	시간, HOUR_OF_DAY의	DATE	해당하는 달에서의 날짜
HOUR_OF_DAY	경우에는 24시간 기준	DAY_OF_MONTH	
AM_PM	오전(AM), 오후(PM)	WEEK_OF_YEAR	그 해에서의 몇 째 주인지
DAY_OF_WEEK	월, 화, 수..	MONTH	달의 이름

```
import java.util.Calendar;
```

```
class Exam_02 {
```

```
    public static void main(String[] args) {
```

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

```
        System.out.println("Calendar 클래스를 이용한 시간과  
날짜 출력");
```

```
        System.out.print(cal.get(Calendar.YEAR) + "년 ");
```

```
        System.out.print((cal.get(Calendar.MONTH)+1) + "월 ");
```

```
        System.out.print(cal.get(Calendar.DATE) + "일 ");
```

```
        System.out.print(cal.get(Calendar.HOUR) + "시 ");
```

```
        System.out.print(cal.get(Calendar.MINUTE) + "분 ");
```

```
        System.out.print(cal.get(Calendar.SECOND) + "초 ");
```

```
    }
```

```
}
```



```
import java.util.*;
class Calendar3 {
    public static void main(String[] args) {
        Calendar cal = Calendar.getInstance();
        Date date = cal.getTime();
        System.out.println("오늘은 :" + date);
        System.out.println("오늘은 올해의 : " +
            cal.get(Calendar.DAY_OF_YEAR)+ "날입니다.");
        System.out.println("오늘은 이번주 : " +
            cal.get(Calendar.DAY_OF_WEEK)+ "요일입니다.");
        System.out.println("오늘은 올해의 : " +
            cal.get(Calendar.WEEK_OF_YEAR)+ "주입니다.");
    }
}
```

```
import java.util.*;
class Date1 {
    public static void main(String[] args) {
        Date date = new Date();
        int h = date.getHours();
        int m = date.getMinutes();
        int s = date.getSeconds();
        System.out.println("현재 시간은 " + h + "시 " + m + "분 " + s + "초");
        if( h > 12 )
            System.out.println("현재 시간은 오후 " + (h-12) + "시 " + m + "분 " + s + "초");
        else
            System.out.println("현재 시간은 오전 " + h + "시 " + m + "분 " + s + "초");
    }
}
```

```
package ch09;
import java.util.Calendar;    import java.util.Date;
public class Date03 {
    public static void main(String[] args) {
        Calendar cal = Calendar.getInstance();
        Date date = cal.getTime();
        System.out.println("오늘은 :" + date);
        System.out.println("오늘은 올해의 : " +
            cal.get(Calendar.DAY_OF_YEAR)+ "날입니다.");
        System.out.println("오늘은 이번주 : " +
            cal.get(Calendar.DAY_OF_WEEK)+ "요일입니다.");
        System.out.println("오늘은 올해의 : " +
            cal.get(Calendar.WEEK_OF_YEAR)+ "주입니다.");
        switch(cal.get(Calendar.DAY_OF_WEEK)) {
            case 1 : System.out.println("일요일입니다"); break;
            case 2 : System.out.println("월요일입니다"); break;
            case 3 : System.out.println("화요일입니다"); break;
            case 4 : System.out.println("수요일입니다"); break;
            case 5 : System.out.println("목요일입니다"); break;
            case 6 : System.out.println("금요일입니다"); break;
            case 7 : System.out.println("토요일입니다"); break;
        }
    }
}
```

GregorianCalendar, SimpleDateFormat 클래스

GregorianCalendar 클래스로 시스템에서 현재 시각을 읽어오는 방법
간단하게 매개변수 없는 생성자를 사용해서 객체를 생성하면 된다.

```
예 GregorianCalendar cal = new GregorianCalendar();
```

주어진 날짜와 시간을 가진 객체를 생성하려면 생성자의 전달인자로 년, 월, 일, 시, 분, 초에 대한 정보를 넘겨주어야 한다.

```
예 GregorianCalendar birthday = new GregorianCalendar(1998, 4, 14);
```

주의할 점은 월에 해당되는 매개변수에 0을 넘겨주면 1월이 되고 1을 넘겨주면 2월이 된다. 위 예처럼 4를 넘겨주면 5월을 표현한 것이 된다.

```
package ch09;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;
public class Date04 {
    public static void main(String[] args) {
        GregorianCalendar gc = new GregorianCalendar();
        int year = gc.get(GregorianCalendar.YEAR);
        System.out.println("년도 : " + year);
        Date date = gc.getTime();
        System.out.println(date);
        SimpleDateFormat sdf =
            new SimpleDateFormat("yy/MM/dd hh:mm:ss (E)");
        System.out.println(sdf.format(date));
    }
}
```

SimpleDateFormat 클래스

SimpleDateFormat

날짜와 시간을 포맷하는 클래스

예

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-mm-dd");
```

상수	의미	상수	의미
y	년	H	시(0~23)
M	월	h	시(1~12)
d	일	m	분
D	월 구분 없는 일(1~365)	s	초
E	요일	S	밀리세컨드(1/1000)
a	오전/오후	w	년의 몇 번째 주
W	월의 몇 번째 주		

```
import java.util.*;    import java.text.*;
class DateFormatEx1{
    public static void main(String[] args) {
        Date today = new Date();
        SimpleDateFormat sdf1, sdf2, sdf3, sdf4;
        SimpleDateFormat sdf5, sdf6, sdf7, sdf8, sdf9;
        sdf1 = new SimpleDateFormat("yyyy-MM-dd");
        sdf2 = new SimpleDateFormat("'yy년 MMM dd일 E요일");
        sdf3 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
        sdf4 = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss a");
        sdf5 = new SimpleDateFormat("오늘은 올 해의 D번째 날입니다.");
        sdf6 = new SimpleDateFormat("오늘은 이 달의 d번째 날입니다.");
        sdf7 = new SimpleDateFormat("오늘은 올 해의 w번째 주입니다.");
        sdf8 = new SimpleDateFormat("오늘은 이 달의 W번째 주입니다.");
        sdf9 = new SimpleDateFormat("오늘은 이 달의 F번째 E요일입니다.");
        System.out.println(sdf1.format(today)); // format(Date d)
        System.out.println(sdf2.format(today)); System.out.println(sdf3.format(today));
        System.out.println(sdf4.format(today)); System.out.println();
        System.out.println(sdf5.format(today)); System.out.println(sdf6.format(today));
        System.out.println(sdf7.format(today));
        System.out.println(sdf8.format(today));
        System.out.println(sdf9.format(today));
    }
}
```

```
import java.util.*;
import java.text.*;
// SimpleDateFormat 클래스가 속하는 패키지
```

```
class DateFormatExample1 {
    public static void main(String args[]) {
        GregorianCalendar calendar = new GregorianCalendar();
        SimpleDateFormat dateFormat =
            new SimpleDateFormat("yyyy년 MM월 dd일 aa hh시 mm분 ss초");
        String str = dateFormat.format(calendar.getTime());
        System.out.println(str);
    }
}
```


Math클래스

- 수학기산에 유용한 메서드로 구성되어 있다.(모두 static메서드)

메서드 / 설명	예 제	결 과
static int abs(int f) static float abs(float f) static long abs(long f) static double abs(double a) 주어진 값의 절대값을 반환한다.	<pre>int i = Math.abs(-10); double d = Math.abs(-10.0);</pre>	i=10 d=10.0
static double ceil(double a) 주어진 값을 올림하여 반환한다.	<pre>double d = Math.ceil(10.1); double d2 = Math.ceil(-10.1); double d3 = Math.ceil(10.0000015);</pre>	d = 11.0 d2 = -10.0 d3 = 11.0
static double floor(double a) 주어진 값을 버림하여 반환한다.	<pre>double d = Math.floor(10.8); double d2 = Math.floor(-10.8);</pre>	d = 10.0 d2=-11.0
static int max(int a, int b) static float max(float a, float b) static long max(long a, long b) static double max(double a, double b) 주어진 두 값을 비교하여 큰 쪽을 반환한다.	<pre>double d = Math.max(9.5, 9.50001); int i = Math.max(0, -1);</pre>	d = 9.50001 i = 0
static int min(int a, int b) static float min(float a, float b) static long min(long a, long b) static double min(double a, double b) 주어진 두 값을 비교하여 작은 쪽을 반환한다.	<pre>double d = Math.min(9.5, 9.50001); int i = Math.min(0, -1);</pre>	d = 9.5 i = -1
static double random() 0.0~1.0범위의 임의의 double값을 반환한다. 0.0은 범위에 포함되지만 1.0은 포함안됨	<pre>double d = Math.random(); int i = (int) (Math.random()*10)+1</pre>	d = 0.0~1.0의 실수 i = 1~10의 정수
static double rint(double a) 주어진 double값과 가장 가까운 정수값을 double형으로 반환한다.	<pre>double d = Math.rint(5.55); double d2 = Math.rint(5.11); double d3 = Math.rint(-5.55); double d4 = Math.rint(-5.11);</pre>	d = 6.0 d2 = 5.0 d3 = -6.0 d4 = -5.0
static long round(double a) static long round(float a) 소수점 첫째자리에서 반올림한 정수값(long) 을 반환한다.	<pre>long l1 = Math.round(5.55); long l2 = Math.round(5.11); long l3 = Math.round(-5.55); long l4 = Math.round(-5.11); double d = 90.7552; double d2 = Math.round(d*100)/100.0;</pre>	l1 = 6 l2 = 5 l3 = -6 l4 = -5 d = 90.7552 d2 = 90.76

Math

```
public class MathClass {
    public static void main(String[] args) {
        double d1=Math.random(); double d2=Math.random(); double d3=Math.random();
        double d4=Math.random(); double d5=Math.random(); double d6=Math.random();
        double d7=Math.random(); System.out.println(d1);      System.out.println(d2);
        System.out.println(d3);      System.out.println(d4);      System.out.println(d5);
        System.out.println(d6);      System.out.println(d7);

        /** 1~10 까지의 정수*****/
        int r1=(int)(Math.random()*10)+1;
        int r3=(int)(Math.random()*10)+1;
        System.out.println(r2);
        int r4=(int)(Math.random()*45)+1;

        /** 알파벳 무작위 추출*****/
        System.out.println("*****");
        int a = (int)(Math.random()*26)+1;
        int aa = a+64;
        System.out.println((char)aa);
    }
}
```

문제 로토 : // 1 ~ 45 숫자 중에서 중복되지 않는 6개

Random

```
import java.util.Random;

public class RandomTest1 {
    public static void main(String[] args) {
        Random random = new Random();
        System.out.println("0부터 100까지의 난수 : " + random.nextInt(101));
        System.out.println("0부터 50까지의 난수 발생 : " + random.nextInt(51));
        System.out.println("0부터 20까지의 난수 발생 : " + random.nextInt(21));
        System.out.println("int형 전체 범위의 난수 발생 : " + random.nextInt());
        System.out.println("float 타입의 난수 발생 : " + random.nextFloat());
        System.out.println("double 타입의 난수 발생 : " + random.nextDouble());
        System.out.println("long 타입의 난수 발생 : " + random.nextLong());
        System.out.println("boolean 타입의 난수 발생 : " + random.nextBoolean());
    }
}
```

Random

```
public class Random2Ex {  
    public static void main(String[] args) {  
        int[] number=new int[100]; int[] count=new int[10];  
        Random ran = new Random();  
        for (int i = 0; i < number.length;i++) {  
            number[i]=ran.nextInt(10); // number[i] 0~9사이 정수값  
            System.out.print(number[i]+" ");  
            if (i%10==9) System.out.println();  
        }  
        for(int i=0;i< number.length;i++) {  
            // number[i]가 들어있는 count인덱스에 1증가  
            count[number[i]]++;  
        }  
        for (int i=0; i < count.length ; i++ ) {  
            System.out.println( i +"의 개수 :"+  
                printGraph('#',count[i]) + " " + count[i]);  
        }  
    }  
    public static String printGraph(char ch, int value) {  
        char[] bar = new char[value];  
        for(int i=0; i < bar.length; i++) { bar[i] = ch; }  
        return new String(bar);  
    }  
}
```

열거 타입

[핵심 키워드] : 열거 타입, 열거 타입 선언, 열거 상수, 열거 타입 변수

[핵심 포인트]

데이터 중에는 몇 가지로 한정된 값을 갖는 경우가 있다.
이러한 한정된 값을 갖는 타입을 열거 타입이라고 한다.

❖ 열거 타입

- 열거 상수(한정된 값) 를 저장하는 타입

```
Week today;
```

```
today = Week.FRIDAY;
```



열거 타입 선언

❖ 열거 타입 선언

- 소스파일(.java) 생성
- 열거타입 선언

```
public enum 열거타입이름 { ... }
```

■ 열거 상수 선언

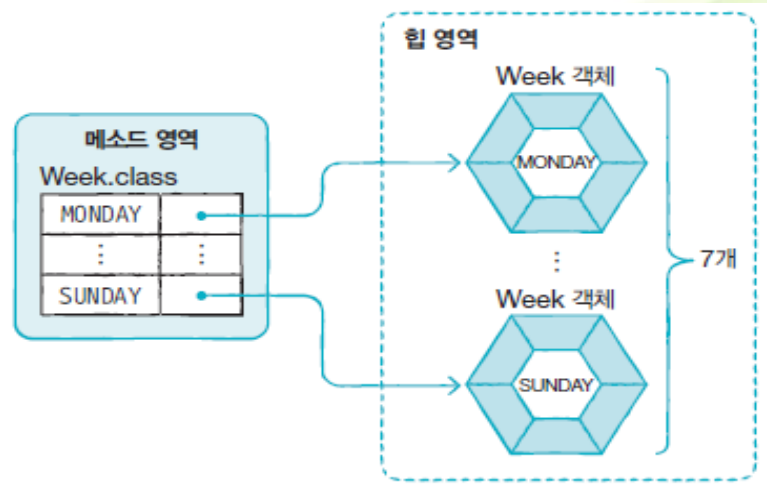
- Week.java

```
public enum Week { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY }
```

↑
열거 타입 이름

↑
열거 상수

- 열거 상수는 열거 객체.



열거 타입 변수

❖ 열거 타입 변수 선언

열거타입 변수;

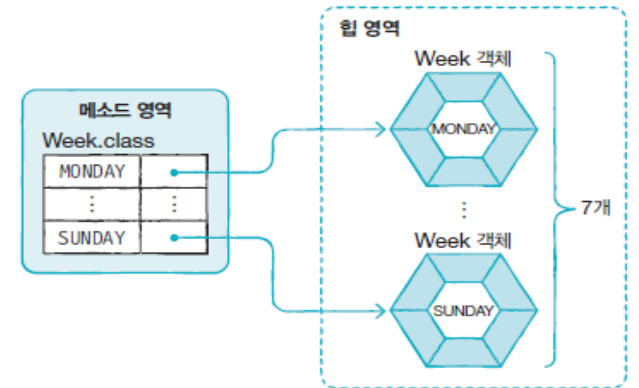
```
Week today;  
Week reservationDay;
```

Week.java

```
public enum Week {  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY,  
    SUNDAY  
}
```

열거 타입 이름

열거 상수



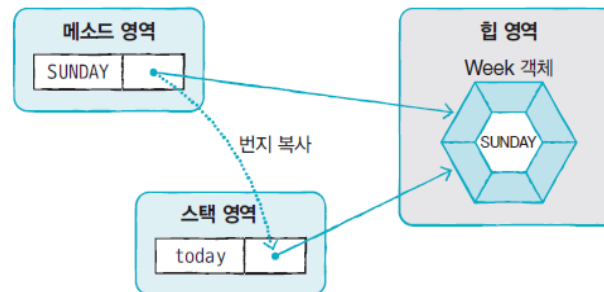
❖ 열거 상수 저장

열거타입 변수 = 열거타입.열거상수;

```
Week today = Week.SUNDAY;
```

```
today == Week.SUNDAY; //true
```

```
Week birthday = null;
```



Enum(열거 타입)의 사용

```
{접근 제한자} enum enumName {  
    값1, 값2, ...  
}
```

관련성이 있는 데이터를 상수처럼 정의해서 사용할 수 있는 단위

public static 변수를 지정해서 사용하던 것을 jdk1.5부터 제공

ordinal() : enum안에 정의 되어있는 각 값들의 인덱스번호 반환

values() : 해당 enum에 저장되어 있는 값들을 해당 enum 타입의 배열로 반환


```

class Student{
    int school;
    public Student(int school) {  this.school = school;  }
    public static int ELEMENTARY = 1;      public static int MIDDLE = 2;
    public static int HIGH = 3;              public static int UNIVERSITY = 4;
}

public class EnumTest1 {
    public static void main(String[] args) {
        Student st1 = new Student(Student.ELEMENTARY);
        Student st2 = new Student(2);
        System.out.println("상수 값을 출력한 경우");
        System.out.println("st1.school = " + st1.school);
        System.out.println("st2.school = " + st2.school);
        if (st1.school == Student.ELEMENTARY){
            System.out.println("st1.school == Student.ELEMENTARY 로 비교했을 때");
            System.out.println("당신은 초등학생입니다.");
        }
        if (st1.school == 1){    System.out.println("st1.school == 1 로 비교했을 때");
            System.out.println("당신은 초등학생입니다.");
        }
    }
}

```

```
class Student1{
```

```
    SchoolType school;
```

```
    public Student1(SchoolType school) {  
        this.school = school;
```

```
    }
```

```
}
```

```
enum SchoolType{
```

```
    ELEMENTARY,MIDDLE,HIGH,UNIVERSITY
```

```
}
```

```
public class EnumTest2 {
```

```
    public static void main(String[] args){
```

```
        Student1 st1 = new Student1(SchoolType.ELEMENTARY);
```

```
        System.out.println("상수 값을 출력한 경우");
```

```
        System.out.println("st1.school = " + st1.school);
```

```
        if (st1.school == SchoolType.ELEMENTARY){
```


```
            System.out.println("st1.school == Student.ELEMENTARY 로 비교했을 때");
```

```
            System.out.println("당신은 초등학생입니다.");
```

```
        }
```

```
    }
```

```
}
```

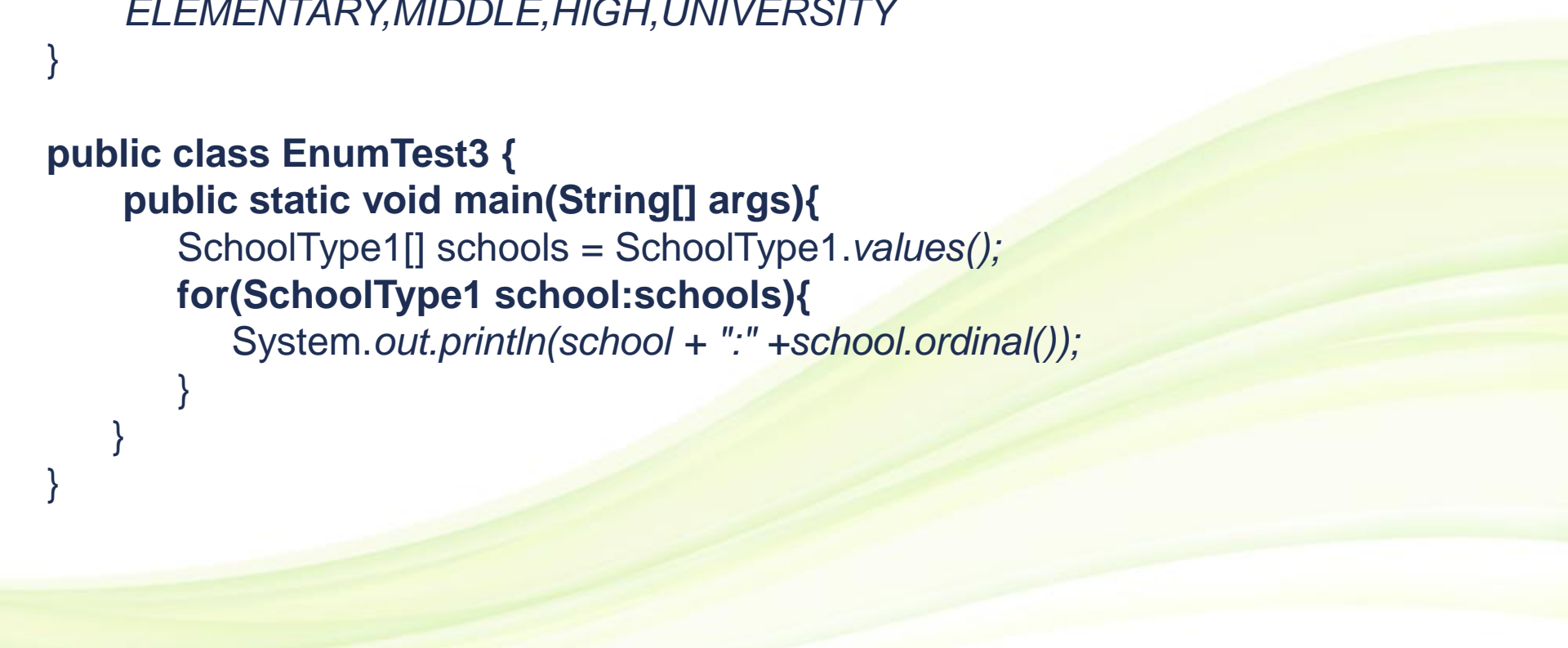


```
class Student2{  
    SchoolType1 school;  
    public Student2(SchoolType1 school) {  
        this.school = school;  
    }  
}
```



```
enum SchoolType1{  
    ELEMENTARY,MIDDLE,HIGH,UNIVERSITY  
}
```

```
public class EnumTest3 {  
    public static void main(String[] args){  
        SchoolType1[] schools = SchoolType1.values();  
        for(SchoolType1 school:schools){  
            System.out.println(school + ":" +school.ordinal());  
        }  
    }  
}
```



```
import java.util.regex.*; // Pattern과 Matcher가 속한 패키지
class RegularEx1{
    public static void main(String[] args) {
        String[] data = {"bat", "baby", "bonus", "cA", "ca", "co", "c.", "c0",
            "car", "combat", "count", "date", "disc"};
        Pattern p = Pattern.compile("c[a-z]*");
        // c로 시작하는 소문자영단어

        for(int i=0; i < data.length; i++) {
            Matcher m = p.matcher(data[i]);
            if(m.matches()) {
                System.out.print(data[i] + ",");
            }
        }
    }
}
```

```

import java.text.*;
class DecimalFormatEx1 {
    public static void main(String[] args) throws Exception {
        double number = 1234567.89;
        String[] pattern = { "0", "#", "0.0", "#.#", "0000000000.0000",
            "#####.####", "#.-", "-.#", "#,###.##", "#,###.##", "#E0",
            "0E0", "##E0", "00E0", "####E0", "0000E0", "#.#E0", "0.0E0",
            "0.0000000000E0", "00.0000000000E0", "000.0000000000E0",
            "#.#####E0", "##.#####E0", "###.#####E0",
            "#,###.##+;#,###.##-", "#.#%", "#.#\u2030",
            "\u00A4 #,###", "'#',###", "'#',###", };

        for(int i=0; i < pattern.length; i++) {
            DecimalFormat df = new DecimalFormat(pattern[i]);
            System.out.printf("%19s : %s\n", pattern[i], df.format(number));
        }

    } // main
}

```

System

모든 필드와 메소드들은 모두 정적필드와 정적메소드로 구성

키보드로부터 입력 받고 모니터로 출력하는 것을 표준입력/표준출력
이라고 부르는데 System클래스의 in과 out사용

환경변수읽기

```
String str = System.getenv("path");
```

```
public class EX12 {  
    public static void main(String[] args) {  
        String path = System.getenv("path");  
        String java_home = System.getenv("java_home");  
        System.out.println(java_home);  
        System.out.println(path);  
    }  
}
```

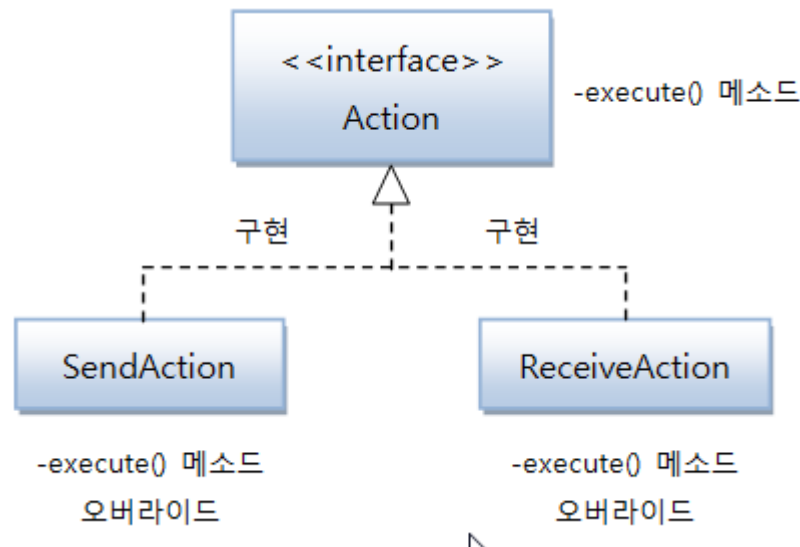
현재시간 읽기

```
class SystemExample11 {
    public static void main(String args[]) {
        long time1 = System.currentTimeMillis();
        double total = 0.0;
        for (int cnt = 1; cnt < 10000000000; cnt += 2)
            if (cnt / 2 % 2 == 0)
                total += 1.0 / cnt;
            else
                total -= 1.0 / cnt;
        double pi = total * 4;
        long time2 = System.currentTimeMillis();
        System.out.println("result = " + pi);
        System.out.printf("계산에 %d ms가 소요되었습니다.",
            time2 - time1);
    }
}
```

Class 클래스

- ❖ 동적 객체 생성(newInstance()) 실행 도중 클래스 이름이 결정될 경우 동적 객체 생성 가능

```
Class clazz = Class.forName("SendAction" 또는 "ReceiveAction");  
Action action = (Action) clazz.newInstance();  
action.execute();
```



`action.execute();`

- SendAction 의 execute() 호출
- ReceiveAction 의 execute() 호출

동적 객체 생성

```
public interface Action {  
    public void execute();  
}
```

```
public class ReceiveAction implements Action {  
    @Override  
    public void execute() {  
        System.out.println("데이터를 받습니다.");  
    }  
}
```

```
public class SendAction implements Action {  
    @Override  
    public void execute() {  
        System.out.println("데이터를 보냅니다.");  
    }  
}
```

동적 객체 생성

```
package ch09;
public class NewInstanceExample {
    public static void main(String[] args) {
        try {
            Class clazz = Class.forName("ch09.SendAction");
            // Class clazz = Class.forName("ch09.ReceiveAction");
            Action action = (Action) clazz.newInstance();
            action.execute();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Annotations

- ❖ 1.5에서 부터 지원되는 문법으로 @를 이용해서 주석을 만들거나 자바코드에 특별한 의미를 부여하는 것을 Annotation이라고 하는데 메타데이터(실제데이터가 아닌 Data를 위한 데이터)기능을 자바로 가지고 와서 사용하거나 컴파일러가 특정 오류를 억제하도록 지시하는 것과 같이 프로그램 코드의 일부가 아닌 프로그램에 관한 데이터를 제공해서 코드에 정보를 추가하는 정형화된 방법
- ❖ Annotation은 @로 시작하고 그 뒤에 이름을 붙여서 사용한다.
- ❖ JDK 제공 Annotation
 - ❖ @Override : 메소드를 재정의(Overriding)할 것임을 컴파일러에게 알려주는 역할
 - ❖ @Deprecated : 클래스, 메소드, 필드 등에 선언하며 지정한 요소를 더 이상 사용하지 않는 것을 권장(가급적 사용을 자제해달라는 의미)
 - ❖ @SuppressWarnings : 클래스, 메소드, 필드의 선언, 컴파일러의 경고를 제거(이 부분에 대해서 경고문을 출력하지 말라는 의미)

연습문제

다음과 같은 실행결과를 얻도록 Point3D 클래스의 equals()를 멤버변수인 x, y, z의 값을 비교하도록 오버라이딩하고, toString()은 실행결과를 참고해서 적절히 오버라이딩하시오.

```
class Ex01 {  
    public static void main(String[] args) {  
        Point3D p1 = new Point3D(1,2,3); Point3D p2 = new Point3D(1,2,3);  
        System.out.println(p1);           System.out.println(p2);  
        System.out.println("p1==p2?"+(p1==p2));  
        System.out.println("p1.equals(p2)" + (p1.equals(p2)));  
    }  
}
```

```
class Point3D {  
    int x,y,z;  
    Point3D(int x, int y, int z) { this.x=x; this.y=y; this.z=z; }  
    Point3D() { this(0,0,0); }  
    public boolean equals(Object obj) {  
  
    }  
    public String toString() { return "["+x+", "+y+", "+z+"]"; }  
}
```