**[ 13 ] API**

**목표 : Java API를 이해한다.**

**문자열 관련 API, 날짜 관련 API**, 수학관련 API, **Object**, Scanner, Wrapper

1. JDK 라이브러리(자바 표준 라이브러리), 자바 필수 API

지금까지 무심코 사용해 왔던 String클래스와 System클래스도 모두 이 라이브러리에 속한 클래스입니다. 이 라이브러리는 JDK안에 포함되어 있는데 JVM이 자바 프로그램을 실행할 때 이 라이브러리의 클래스와 인터페이스들을 자동으로 읽어 들이기 때문에 우리가 지금까지 그 존재를 인식하지 않고도 사용할 수 있었던 겁니다.

※JDK에서 많은 패키지 제공

1. **java.lang** 자바프로그램의 기본적인 기능을 제공. 명시적으로 지정하지 않아도 모든 자바 프로그램에 포함되는 패키지 java.lang.String나 java.lang.Exception은 모두 java.lang.을 생략 가능
2. **java.util** 유용한 유틸리티 클래스를 제공
3. **java.io** 입출력 기능을 제공하는 패키지

이러한 라이브러리 안의 클래스나 인터페이스를 API라 부르고 http://java.sum.com를 통해 API 규격서를 다운로드 받아 로컬디스크에 저장해 두고 수시로 봅니다.

([<https://docs.oracle.com/javase/8/docs/api/index.html>](http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html) )

(<https://docs.oracle.com/en/java/javase/17/docs/api/index.html>)

JAVA는 개발자들이 편리하게 이용할 수 있는 풍부한 클래스들이 많습니다.

이러한 클래스들을 이용해서 개발자들은 깊은 지식이 없이, 그냥 사용만 하면 되고, 이러한 기능들을 정의해 둔 클래스들을 API(Application Programming Interface)라고 합니다. 그냥 쉽게 말하자면, API란, 누군가가 만들어 놓은 기능들이라고 생각하면 됩니다.

1. 문자열에 관련된 대표적 클래스 : String

※String은 객체 자료형입니다. 우리가 문자열을 사용하면서 아래와 같이 사용하였습니다.

String str = “HelloWorld~~”;

String은 분명 대문자로 시작하므로, 기초데이터가 아닌 객체데이터입니다. 하지만 뒤에 생성자(new)를 사용하지 않고 기초데이터를 만들 때처럼 초기화 하고 있습니다.

int i = 10;

String의 경우만 특이하게 객체데이터이지만, 기초데이터처럼 사용할 수 있게 하였습니다.

원래는 아래처럼 사용했어야 했겠죠.

String string = new String(“HelloWorld~~”);

두 경우 모두 사용할 수 있습니다.

String string = “HelloWorld”;

String string = new String(“HelloWorld~~”);

그럼 위의 두 경우의 차이를 알아봅니다.

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

String str1 = "Java";

String str2 = "Java";

String str3 = **new** String("Java");

// new로 생성하지 않고 ""를 사용한 문자열 상수는 자동으로 힙영역에 String 객체가 생성됨.

// 이미 존재하는 문자열 상수를 다시 사용한다면 이미 생성된 문자열 상수를 공유한다.

System.***out***.println(str1==str2? "(str1과 str2은 같은 객체)":"(str1과 str2은 다른 객체)");

System.***out***.println(str1.equals(str2)? "(1과2은 같은 스트링)":"(1과2은 같은 스트링)");

System.***out***.println(str1==str3? "(str1과 str3은 같은 객체)":"(str1과 str3은 다른 객체)");

System.***out***.println(str1.equals(str3)? "(1과3은 같은 스트링)":"(1과3은 같은 스트링)");

}

※String의 주요 기능들(메소드)

String concat(String str) : 저장된 문자열과 str문자열을 결합

String substring(int begin) : 시작위치부터 마지막까지의 문자열을 반환

int length() : 문자열 길이를 반환

String toUpperCase() : 대문자로 반환

String toLowerCase() : 소문자로 반환

char charAt(int index) : index 위치의 문자를 반환

int indexOf(char ch) : 첫번째 ch문자의 위치를 반환

int lastIndexOf(char ch) : 마지막 ch문자의 위치를 반환

boolean equals(String str) : 지정된문자열과 str문자열이 같은지 비교

boolean equalslgnoreCase(String str) : 대소문자구분없이 지정된문자열과 str문자열이 같은지 비교

String trim() : 문자열 앞뒤 공백제거

String replace(char old, char new) : 문자열 내의 old문자를 new문자로 반환

String repalceAll(String old, String new) : 문자열 내의 정규표현식 문자열을 new로 반환

<예제>

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

String str1 = "abcXabc";

String str2 = **new** String("abcXabc");

String str3 = " Java ";

System.***out***.println(str1.concat(str2)); //abcXabcabcXabc

System.***out***.println(str1.substring(3)); //Xabc

System.***out***.println(str1.substring(3, 5));//3번째부터5번째앞까지 Xa

System.***out***.println(str1.length() ); // 7(글자갯수)

System.***out***.println(str1.toUpperCase()); // ABCXABC (대문자로)

System.***out***.println(str1.toLowerCase()); // abcxabc (소문자로)

System.***out***.println(str2.charAt(3)); // X (3번째 문자)

System.***out***.println(str2.indexOf('b')); // 1(처음으로 나오는 'b'의 위치)

System.***out***.println(str2.lastIndexOf('b'));// 5(마지막 'b'의 위치)

System.***out***.println(str2.indexOf('@')); // '@'가 없으면 -1

String str4 = "abcXabcXabc";

System.***out***.println(str4.lastIndexOf('b'));//9

System.***out***.println(str4.indexOf('b')); // 1(0번째부터 처음 나오는 'b'의 위치)

System.***out***.println(str4.indexOf('b', 2));//5(2번째부터 처음나오는 'b'의 위치)

System.***out***.println(str1.equals("abcXabc")); //true(같은문자인지)

System.***out***.println(str1.equals("ABCXabc")); //false

System.***out***.println(str1.equalsIgnoreCase("ABCXabc"));//true(대소문자구분없이)

System.***out***.println(str3.trim()); // 앞뒤공백제거 "java"

System.***out***.println(str1.replace('a', 'A')); // a를 A로 바꿔

System.***out***.println(str1.replace("ab", "A"));

System.***out***.println(str1.replaceAll("[a-b]", "걍바꿔")); //문자열 교체

System.***out***.println();

System.***out***.println("str1="+str1);

System.***out***.println("str2="+str2);

System.***out***.println("str3="+str3);

}

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

Scanner sc = **new** Scanner(System.***in***);

String fn;

**do** {

System.***out***.print("I:input, U:update, X:exit");

fn = sc.next();

**if**(fn.equalsIgnoreCase("X")) {

**break**;

}

}**while**(**true**);

sc.close();

}

<quiz>. 반복적으로 전화번호를 입력받아 다음과 같은 형식을 출력하는 반복문으로 구현하시오. 전화번호 대신 x를 입력하면 종료합니다

String tel;

while(){

// 번호(031-234-5678)입력 하면 (sc.next())

"입력한 전화번호 : 031-234-5678

짝수번째 문자열 : 0 1 2 4 5 7

문자를 꺼꾸로 : 8765-432-130

전화번호 맨앞자리는: 031

전화번호 뒷자리는: 5678"을 sysout

// x(X)를 입력하면 프로그램 끝

}

<예제>전화번호 뒷자리로 검색하고 있으면 전화번호 전체를 출력하기, 없는 번호이면 없는 번호라고 출력하기

1. 한번 검색하기

// 검색하고자 하는 뒷번호가 중복이 없다는 전제하에.

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

String[] tels = {"010-1111-9999", "010-2222-8888", "02-555-9999"};

Scanner sc = **new** Scanner(System.***in***);

**int** idx;

System.***out***.print("검색하고자 하는 회원의 전화번호 뒷자리는 ?");

String searchTel = sc.next(); // ex. 8888

**for** (idx=0 ; idx<tels.length ; idx++) {

// 전화번호 뒷자리 빼내기

String post = tels[idx].substring(tels[idx].lastIndexOf('-')+1);

**if**(post.equals(searchTel)) {

System.***out***.println("검색하신 전화번호 전체는 "+tels[idx]);

**break**;

}

} // for - 0번째부터 끝까지 searchTel 검색하다 찾으면 break

**if**(idx==tels.length) { // searchTel이 배열 전화에 없어서 끝까지 간 경우

System.***out***.println("검색하신 전화번호는 없습니다.");

}

}

// 검색하고자 하는 뒷번호가 중복이 있을 수도 있고 없을수도 있다는 전제하에.

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

Scanner sc = **new** Scanner(System.***in***);

String[] tel = {"010-9999-9999", "010-8888-9999","010-7777-8888"};

System.***out***.print("검색하고자 하는 전번 뒷자리 ?");

String searchTel = sc.next();

**boolean** noSearch = **true**;

**for**(int idx=0 ; idx<tel.length ; idx++) {

String postTel = tel[idx].substring(tel[idx].lastIndexOf("-")+1);

**if**(searchTel.equals(postTel)) {

System.***out***.println(tel[idx]);

noSearch = false;

}

}

**if**(noSearch) {

System.***out***.println("없는 핸드폰 번호입니다");

}

}

1. X를 입력할 때까지 계속 검색하기

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

Scanner sc = **new** Scanner(System.***in***);

String[] tel = {"010-9999-9999", "010-8888-9999","010-7777-8888"};

**int** idx;

**do** {

System.***out***.print("검색하고자 하는 전번 뒷자리(종료를 원하시면 x) ?");

String searchTel = sc.next();

**if**(searchTel.equalsIgnoreCase("x"))

**break**;

**boolean** noSsearch = **true**;

**for**(idx=0 ; idx<tel.length ; idx++) {

String postTel = tel[idx].substring(tel[idx].lastIndexOf("-")+1);

**if**(searchTel.equals(postTel)) {

System.***out***.println(tel[idx]);

noSsearch = **false**;

}

}

**if**(noSsearch) {

System.***out***.println("없는 핸드폰 번호입니다");

}

}**while**(**true**);

}

1. 날짜API(Date, Calendar, GregorianCalendar)

<예제>

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

Date nowDate = **new** Date();

Calendar nowCal = Calendar.*getInstance*();

GregorianCalendar nowGc = **new** GregorianCalendar(2022,1,1);

/\* yyyy(년도 4자리) yy(년도 2자리) MM(03) M(3) dd(08) d(8)

\* E(요일) a(오전/오후) H(24시) h(12시) m(분) s(초)

\* w(이번년도에 몇번째 주인지) W(이번 월에 몇번째 주인지) D(올해의 몇번째 날) \*/

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy년 MM월 dd일(E) a hh시 mm분 ss초");

String dateStr = sdf.format(nowDate);

String calStr = sdf.format(nowCal.getTime()); // getTime : Calendar나 gregorian을 Date형으로 변환

String gcStr = sdf.format(nowGc.getTime());

System.***out***.println(dateStr);

System.***out***.println(calStr);

System.***out***.println(gcStr);

}

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

Friend[] friends = {**new** Friend("홍길동","010-9999-9999","12-08"),

**new** Friend("김길동","010-8888-8888","11-01"),

**new** Friend("이선동","010-6666-6666","09-09")};

**int** idx;

Calendar calendar = Calendar.*getInstance*();

SimpleDateFormat sdf = **new** SimpleDateFormat("MM-dd");

String today = sdf.format(calendar.getTime());

**boolean** output = **false**;

**for**(idx=0 ; idx<friends.length ; idx++) {

String birth = friends[idx].getBirth();

**if**(birth.equals(today)){

System.***out***.println(friends[idx].infoString());

output = **true**;

}//if

}//for

**if**(!output) {

System.***out***.println("오늘이 생일인 사람이 없습니다.");

}

}

1. 수학적 계산에 사용되는 클래스 Math

※ Math 클래스가 제공하는 정적 메소드

**public** **class** MathEx2 {

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

System.***out***.println(" 소숫점에서 반올림, 올림, 버림");

System.***out***.println("9.12의 올림 : "+Math.*ceil*(9.12));

System.***out***.println("9.69의 반올림 : "+Math.*round*(9.69));

System.***out***.println("9.69 버림 : "+Math.*floor*(9.69));

System.***out***.println("소숫점 한자리에서 반올림, 올림, 버림");

System.***out***.println("9.12의 올림 : "+Math.*ceil*(9.12\*10)/10);

System.***out***.println("9.69의 반올림 : "+Math.*round*(9.69\*10)/10.0);

System.***out***.println("9.69 버림 : "+Math.*floor*(9.69\*10)/10);

System.***out***.println("십의 자리에서 반올림, 올림, 버림");

System.***out***.println("11의 올림 : "+Math.*ceil*(11/10.0)\*10);

System.***out***.println("19의 반올림 : "+Math.*round*(19/10.0)\*10);

System.***out***.println("19 버림 : "+Math.*floor*(19/10.0)\*10);

}

}

1. 난수 발생이 쉬운 Random 클래스 : 다양한 랜덤 숫자를 구하는 작업을 제공합니다.

※Math.random()

Random() 메소드는 double타입의 임의의 수를 발생합니다.

API documents(<http://docs.oracle.com/javase/8/docs/api>)을 보면 static이고, 반환 타입이 double임을 확인 할 수 있습니다. Math클래스의 random()메소드가 static이기 때문에 객체 생성하지 않고 Math.random() 형으로 바로 실행할 수 있다

public class RandomEx {

public static void main(String[] args) {

double d = Math.random(); //random메소드는 static

System.out.println(d); //0이상 1미만의 double난수

int di = (int)(d\*10);

System.out.println(di);

Random random = new Random();

int i = random.nextInt();

System.out.println(i)

System.out.println((int)(d\*10)+1);//1~10까지의 난수

}

}

※Random 클래스

사용법이 아주 쉬워 요즘 많이 쓰이고 있는 클래스. 일반메소드는 객체를 생성 후 실행가능

public static void main(String[] args) {

Random random = new Random();

int iran = random.nextInt(10); //0~9까지의 난수

double doubleran = random.nextDouble(); //Math.random()과 같음

}

<예제>

**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());

}

}

1. 자바 클래스의 상속 계층

여러 클래스의 공통된 특성(공통데이터와 메소드)은 슈퍼클래스로 선언하는 것이 좋다는 사실을 상속에서 접했습니다. JDK 라이브러리 개발자들도 그런 식으로 상속관계를 갖도록 클래스를 설계했기 때문에 비슷한 기능의 클래스들이 같은 슈퍼 클래스를 갖게 되는 경우를 종종 볼 수 있습니다.

그런데 클래스의 어떤 특성은 특정 부류의 클래스 뿐 아니라 자바의 모든 클래스들이 공통으로 갖습니다. 그래서 자바에서는 그런 공통 특성을 추출하여 Object라는 클래스로 만들어 두었습니다. 그리고 다른 모든 클래스들은 Object 클래스를 상속 받도록 만들어 두었습니다.

extends Object라는 단어가 없어도 컴파일 과정에서 자동적으로 Object 의 서브 클래스가 됩니다.

※ Object클래스의 메소드들

* public String toString() 객체의 문자열을 반환한다.
* public boolean equals(Object obj)
* public int hashCode() 자바에서 객체를 식별하는 정수 값인 해시 코드를 반환(다량의 데이터를 저장&검색하는 해싱 기법에 사용된다.)
* public Class getClass() 객체의 클래스 이름을 Class형으로 반환한다

1. toString() ; 객체의 정보를 문자열로 제공할 목적으로 정의된 메서드.

자바는 사용자가 모르게 내부적으로 동작하는 메소드 들이 많은데 그 중 하나가 toString().

<예제>

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** Sawoon {

**private** String num; // 사번

**private** String name; // 이름

**private** String part; // 부서

**private** Date enterDate; // 입사일

**public** Sawoon(){}

**public** Sawoon(String num, String name, String part){

**this**.num = num;

**this**.name = name;

**this**.part = part;

enterDate = **new** Date();

}

/\*public void print(){

System.out.print("[사번]"+num+"\t[이름]"+name);

System.out.print("\t[부서]"+part+"\t[입사일]");

SimpleDateFormat sdf =

new SimpleDateFormat("yy년MM월dd일(E)");

System.out.println(sdf.format(enterDate));

}\*/

@Override

**public** String toString() {

String temp = "[사번]"+num+"\t[이름]"+name;

temp += "\t[부서]"+part+"\t[입사일]";

SimpleDateFormat sdf =

**new** SimpleDateFormat("yy-MM-dd");

temp += sdf.format(enterDate);

**return** temp;

}

}

**package** com.ch.ex3tostring;

**public** **class** SawoonMain {

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

Sawoon kang = **new** Sawoon("23A01","강동원","전산실");

Sawoon yu = **new** Sawoon("22A10","유아인","비서실");

/\*kang.print();

yu.print();\*/

System.***out***.println(kang);

System.***out***.println(yu);

}

}

1. equals()를 오버라이딩 할 수도 이다.

**package** com.ch.ex4equals;

**public** **class** Person {

**private** **long** juminNo;

**public** Person(){}

**public** Person(**long** juminNo){

**this**.juminNo = juminNo;

}

@Override //p1.equals(p2)

**public** **boolean** equals(Object obj) {

// juminNo == obj.juminNo 이면 true리턴

// 아니면 false 리턴

**if**(obj!=**null** && obj **instanceof** Person){

**return** juminNo==((Person)obj).juminNo;

}**else**{

**return** **false**;

}

}

}

**package** com.ch.ex4equals;

**public** **class** PersonMain {

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

Person p1 = **new** Person(9101012085215L);

Person p2 = **new** Person(9101012085215L);

Person p3 = **new** Person(9805241052009L);

**if**(p1.equals(p2))

System.***out***.println("두 객체는 같은 사람이구만");

**else**

System.***out***.println("두 객체는 다른 사람이구만");

**if**(p1 == p2)

System.***out***.println("두 객체는 같은 객체네");

**else**

System.***out***.println("두 객체는 따로 네");

}

}

<예제> Card 데이터 :카드종류(♥ ◆ ♠ ♣), 번호(1~13)

생성자 new Card('♥',2);

toString() 오버라이드

equals() 오버라이드

**public** **class** Card {

**private** **char** kind; // ♥ ◆ ♠ ♣

**private** **int** num; // A(1), 2~10, 11, 12, 13

// Card c1 = new Card('♥',2);

**public** Card(**char** kind, **int** num) {

**this**.kind = kind;

**this**.num = num;

}

@Override

**public** String toString() {**return** "카드 모양은 "+kind+" "+num;}

@Override

**public** **boolean** equals(Object obj) {

// c1.equals(c2) this.kind와 c2.kind가 같고 this.num과 c2.num이 같으면 true

**if**(obj!=**null** && obj **instanceof** Card) {

**boolean** kindChk = **this**.kind == ((Card)obj).kind;

**boolean** numChk = **this**.num == ((Card)obj).num;

**return** kindChk && numChk;

}**else** {

**return** **false**;

}//if

}//equals

}

**public** **class** Ex04\_CardMain {

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

Card[] cards = {**new** Card('♥',2),

**new** Card('◆', 7),

**new** Card('♣', 8)};

Card yours = **new** Card('♣', 8);

**char**[] kinds = {'♥','◆','♠','♣'};

Random ran = **new** Random();

Card yours = **new** Card(kinds[ran.nextInt(4)], (**int**)(Math.*random*()\*13)+1);

System.***out***.println("당신카드는 "+yours);

**for**(**int** idx=0 ; idx<cards.length ; idx++) {

System.***out***.print(cards[idx]);

**if**(yours.equals(cards[idx])) {

System.***out***.println(" - 당신 카드와 일치합니다");

}**else** {

System.***out***.println(" - 당신 카드와 일치하지 않습니다");

}//if

}//for

}//main

}

실행결과 : 당신 카드 : 카드 모양은 ♣ 8

카드 모양은 ♥ 2 - 당신 카드와 일치하지 않습니다

카드 모양은 ◆ 7 - 당신 카드와 일치하지 않습니다

카드 모양은 ♣ 8 - 당신 카드와 일치합니다

1. hashCode() 해쉬코드를 반환
2. getClass메소드 : 객체가 속하는 클래스의 정보(이름)를 리턴
3. Scanner : 키보드에서 타이핑하는 문자열 또는 입출력 프로그래밍에서 값을 읽어올 때, 무엇인가를 얻어 올 때 사용

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

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("나이를입력해");

**int** age = scanner.nextInt();

System.***out***.println("입력하신 나이는 "+age);

System.***out***.print("이름을 입력해");

String str1 = scanner.next();

System.***out***.println("str1="+str1);

System.***out***.print("주소를 입력해");

scanner.nextLine(); // '\n'이전까지만 인식하고 그 뒤는 버림

// nextLine()는 잘못사용하면 데이터버려짐

String str2 = scanner.nextLine();

System.***out***.println("str2="+str2);

System.***out***.println("끝");

scanner.close();

}

위의 예제와 아래의 예제 비교

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

Scanner scanner = **new** Scanner(System.***in***);

String str2 = scanner.nextLine();

System.***out***.println("str2 = " + str2);

**int** i = scanner.nextInt();

System.***out***.println("i = "+i);

String str1 = scanner.next();

System.***out***.println("str1 = " + str1);

str2 = scanner.nextLine();

System.***out***.println("str2 = " + str2);

System.***out***.println("Done");

}

※System.out.println(sysout)

실제 개발 산출물에는 나타나지 않고, console창에만 표시 납니다. 따라서 테스트 용으로 많이 사용 됩니다.

println()과 print()과 printf()의 차이점을 알아둡니다.

1. Wrapper 클래스의 이해

※기초데이터를 객체데이타로 변환 합니다.

Wrapper클래스는 기초데이타를 객체데이터로 변화시키는 클래스 입니다. 문법은 쉽습니다

기초데이터에 상응하는 객체 데이터 클래스

|  |  |
| --- | --- |
| 기초 데이터 | 객체 데이터 |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| boolean | Boolean |
| char | Character |

1. Wrapper클래스의 기본적인 사용방법

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

**int** i = 10;

**int** j = 10;

**if**(i==j) {

System.***out***.println("i와 j는 같다");

}

Integer iObj = **new** Integer(10);

Integer jObj = **new** Integer(20);

System.***out***.println("iObj의 int값은 "+iObj.intValue());

System.***out***.println("jObj의 int값은 "+jObj.intValue());

**int** sum = iObj.intValue() + jObj.intValue();

System.***out***.println("합은 "+sum);

**if**(iObj!=jObj) {

System.***out***.println("다르다");

}

Integer obj1 = **new** Integer(10);

Integer obj2 = 20;

**if**(obj1!=obj2) {

System.***out***.println("다르다");

}

sum = obj1+obj2;

System.***out***.println("합은 "+sum);

}

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

Integer obj1 = 10; //Integer obj1 = new Integer(10);

Integer obj2 = 10; //Integer obj2 = new Integer(10);

**int** total = obj1+obj2; //int total = obj1.intValue()+obj2.intValue();

Card c = **new** Card('♥', 10);

System.***out***.println(c.equals(obj1));

**int** i =10;

**boolean** b = **true**;

Boolean bObj = **true**;

System.***out***.println(obj1.equals(i));

}

1. Wrapper 클래스의 static 메소드와 상수

<예제>

public static void main(String[] args) {

String str = "123";

i = **Integer.parseInt**(str);

System.out.println("i :"+i);

}

※String을 기초데이터로 반환하는 메소드들

Byte.parseByte("1");

Short.parseShort("23");

**Integer.parseInt("123456");**

Long.parseLong("123456");

Float.parseFloat("1.5");

Double.parseDouble("1.00005");

Boolean.parseBoolean("true")

**(反) String.valueOf(1) => “1” (cf. “”+1)**