**[ 14 ] API-III**

목표 : Scanner, Wapper, Timer

1. Scanner 및 Sysout

※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");

}

2. 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 |
| double | Double |
| boolean | Boolean |
| char | Char |

2-1 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(10);

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("같은 객체");

}**else** {

System.***out***.println("다른 객체");

}

**if**(iObj.equals(jObj)) {

System.***out***.println("같은 값");

}**else** {

System.***out***.println("다른 값");

}

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));

}

2-2 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)**

<예제> /\*가변입력값 :

메서드나 생성자를 통해서 같은 데이터 타입의 입력값의 갯수가

정해지지 않고 여러개로 입력될 수 있을 때, 가변인자로 등록할 수 있게 한다 \*/

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

**int** total = *valueSum*("10","20","40");

System.***out***.println("들어온 값의 합계 : " + total);

}

**public** **static** **int** valueSum(String ... values) {

**int** result = 0;

**for**(String value : values) {

**int** temp = Integer.*parseInt*(value);

result += temp;

}

**return** result;

}//valueSum

<연습문제 : 주민번호를 입력받아 여자인지 남자인지 출력하세요 1이나 3은 남자를 출력, 2나 4는 여자를 그 외의 숫자를 입력하면 떼끼를 출력하도록 구현하세요. >

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

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

System.***out***.print("주민등록번호는?");

String juminNum = scanner.next();

**int** gender = Integer.*parseInt*(juminNum.substring(7,8));

String result = (gender==1 || gender==3) ? "남자" :

(gender==2 || gender==4)? "여자" :"주민번호오류";

System.***out***.printf("당신은 %s입니다", result);

scanner.close();

}

3. Timer 클래스

자바에서의 타이머는 2가지입니다. 일정한 시간마다 작동되는 타이머 프로그램과 일정한 시간이 되면 한번 실행되는 타이머 프로그램입니다.

※ Timer, TimerTask클래스

Timer객체는 일정한 시간이 되면, TimerTask객체가 작동되도록 하거나 일정시간마다 TimerTask객체가 작동되도록 합니다. 단, TimerTask클래스는 추상클래스이므로, TimerTask클래스를 상속받는 클래스를 만들어서 사용해야 합니다

public class TimerTaskEx1 extends TimerTask {

@Override

public void run() {

System.out.println("TimerTaskEx 1 : 1의 작업이 실행 중 ☏");

}

}

public class TimerTaskEx2 extends TimerTask{

@Override

public void run() {

System.out.println("TimerTaskEx 2 : 2의 작업이 실행 중 ♨");

}

}

public static void main(String[] args) throws InterruptedException {

System.out.println("START");

Timer timer = new Timer(true);

TimerTask t1 = new ExTimerTask1();

TimerTask t2 = new ExTimerTask2();

timer.schedule(t1, 2000); //2초후 실행

timer.schedule(t2, 1000, 500);//1초후 0.5초마다 실행

Thread.sleep(11000);

System.out.println("Done");

}

5. Decimalformat과 Pattern

**import** java.text.DecimalFormat;

**public** **class** MainFormatClass {

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

**double** num = 1234567.8989; //1,234,567.899 등 다양한 패턴으로 출력을 원할 때

DecimalFormat df0 = **new** DecimalFormat("000000000");

System.***out***.println(df0.format(num));

DecimalFormat df1 = **new** DecimalFormat("#########");

System.***out***.println(df1.format(num));

DecimalFormat df2 = **new** DecimalFormat("0.00000");

System.***out***.println(df2.format(num));

DecimalFormat df3 = **new** DecimalFormat("0,000.00000");

System.***out***.println(df3.format(num));

DecimalFormat df4 = **new** DecimalFormat("#,###.#####");

System.***out***.println(df4.format(num));

DecimalFormat df5 = **new** DecimalFormat("#.##%");

System.***out***.println(df5.format(num));

DecimalFormat df6 = **new** DecimalFormat("#.##E0");

System.***out***.println(df6.format(num));

String[] pattern = {"00000000","#########","#,###.###"};

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

String[] pattern = {"0", "#.###",

"00000000.##", "#,###,###.###",

"#.##%", "#.##E0"};

**for**(**int** i=0 ; i<pattern.length ; i++){

DecimalFormat df = **new** DecimalFormat(pattern[i]);

System.***out***.printf("%s 패턴 타입 : %s\n",pattern[i], df.format(num));

}

}

}