**[ 20 ] 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("정수입력해");

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

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

System.***out***.print("문자열을 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.

<오늘의 예제> 컴퓨터와 나는 가위바위보 게임을 작성하시오.

단, 당신이 이길 때까지 가위바위보가 계속 진행됩니다. 컴퓨터는 임의의 가위바위보를 선택합니다. 당신은 Scanner를 이용해서 가위바위보 중 하나를 냅니다

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** WhileEx04 {  **public** **static** **void** main(String[] args) {  Scanner sc = **new** Scanner(System.***in***);  **while**(**true**) { // 무한반복  **int** computer = (**int**)(Math.*random*()\*3);  System.***out***.print("가위(0), 바위(1), 보(2) 중 하나 ? (종료시 -1)");  **int** me = sc.nextInt();  **if**(me==-1) {  **break**;  }**else** **if**((me<-1) || (me>2)) {  System.***out***.println("잘못 냈어");  }**else** **if**((me+2)%3==computer){  *printResult*(me, computer);  System.***out***.println("당신이 이겼어요");  //break; //내가 이기면 끝  }**else** **if**(me==computer){  *printResult*(me, computer);  System.***out***.println("비겼어요");  }**else**{  *printResult*(me, computer);  System.***out***.println("당신이 졌어요");  }//if  }//while  } // main  **private** **static** **void** printResult(**int** me, **int** computer) {  System.***out***.println("당신은 "+((me==0)?"가위":((me==1)?"바위":"보")));  System.***out***.println("컴퓨터는 "+((computer==0)?"가위":((computer==1)?"바위":"보")));  } // printResult  } // class |

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** Game {  **public** **static** **void** main(String[] args) {  Scanner sc = **new** Scanner(System.***in***);  **int** computer, me=0;  **while**(**true**) { // 무한반복  computer = (**int**)(Math.*random*()\*3);  System.***out***.print("가위, 바위, 보 중 하나 ? (종료시 x)");  String temp = sc.next();  **if**(temp.equalsIgnoreCase("x")) {  **break**;  }  **if**(temp.equals("가위")) {  me=0;  }**else** **if**(temp.equals("바위")) {  me=1;  }**else** **if**(temp.equals("보")){  me=2;  }**else** {  **continue**;  }    **if**((me+2)%3==computer){  *printResult*(me, computer);  System.***out***.println("당신이 이겼어요");  //break; //내가 이기면 끝  }**else** **if**(me==computer){  *printResult*(me, computer);  System.***out***.println("비겼어요");  }**else**{  *printResult*(me, computer);  System.***out***.println("당신이 졌어요");  }//if  }//while  } // main  **private** **static** **void** printResult(**int** me, **int** computer) {  System.***out***.println("당신은 "+((me==0)?"가위":((me==1)?"바위":"보")));  System.***out***.println("컴퓨터는 "+((computer==0)?"가위":((computer==1)?"바위":"보")));  } // printResult  } |

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 |
| boolean | Boolean |
| char | Char |

2-1 Wrapper클래스의 기본적인 사용방법

class WrapperExample1 {

public static void main(String args[]) {

Integer obj1 = new Integer(12);

Integer obj2 = new Integer(7);

int sum = obj1.intValue() + obj2.intValue();

System.out.println(sum);

}

}

<예제> 명령형 파라미터로 받은 문자열을 가지고 Integer객체를 생성하고 total값을 출력합니다.

class WrapperExample2 {

public static void main(String args[]) {

int total = 0;

for (int cnt = 0; cnt < args.length; cnt++) {

Integer obj = new Integer(args[cnt]);

total += obj.intValue();

}

System.out.println(total);

}

}

2-2 Wrapper 클래스의 static 메소드와 상수

<예제>

public static void main(String[] args) {

Integer integer = new Integer(10);

int i = integer.intValue();

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

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

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”

3. Timer 클래스

일상 생활에서 일정한 시간 간격을 두고 어떤 일이 실행되는 경우가 빈번 합니다. 예를 들어 보일러 컨트롤러는 1hr로 설정하면 1시간마다 보일러가 작동하고, 선풍기 타임머를 30분으로 설정 하면 30분 뒤에 선풍기가 off됩니다.

프로그램에서도 이러한 시간을 컨트롤하는 기능이 빈번히 사용 됩니다.

※ Timer, TimerTask클래스

Timer객체는 일정한 시간이 되면, TimerTask객체가 작동 됩니다. 단, TimerTask클래스는 추상클래스 입니다. 따라서 TimerTask클래스를 상속받는 클래스를 만들어서 사용 해야 합니다.

public class ExTimerTask1 extends TimerTask {

@Override

public void run() {

System.out.println("ExTimerTask1 : 1의 작업이 실행 중 ☏");

}

}

public class ExTimerTask2 extends TimerTask{

@Override

public void run() {

System.out.println("ExTimerTask2 : 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;

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

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

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

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

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

System.***out***.printf("%s\n",df.format(num));

}

}

}