2장. 연산자 , 입력 처리

Operator

항과 연산자

- **항(operand)**
  - 연산에 사용되는 값

- **연산자(operator)**
  - 연산에 사용되는 기호

    예) 3 + 7 (3과 7은 항, '+'는 연산자)

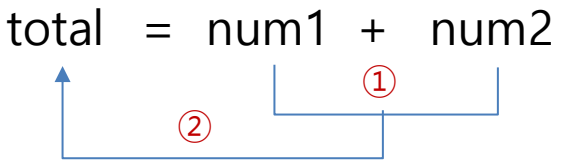
- **항의 개수에 따른 연산자 구분**

연산자	설명	연산 예
단항 연산자	항이 한 개인 연산자	++num
이항 연산자	항이 두 개인 연산자	num1 + num2
삼항 연산자	항이 세 개인 연산자	(5>3) ? 1 : 0

대입 및 부호 연산자

대입 연산자

- 변수에 값을 대입하는 연산자
- 연산의 결과를 변수에 대입
- 왼쪽 변수(lvalue)에 오른쪽 값(rvalue)를 대입



부호 연산자

- 양수/음수의 표현, 값의 부호를 변경
- 변수에 +, -를 사용한다고 해서 변수의 값이 변하는 것이 아님.
- 변수의 값을 변경하려면 대입연산자를 사용해야함

```
int num = 10;
System.out.println(num);
System.out.println(-num); //부호만 바뀜
System.out.println(num);

num = -num;  //값이 바뀜
System.out.println(num);
```

대입 연산자

- 실습 예제

아래의 실행 결과대로 변수의 값을 서로 바꾸는 프로그램을 작성 하세요

(파일 이름 : swap.java)

☞ 실행 결과

```
교환 전
x = 0, y = 1
---------------
교환 후
x = 1, y = 0
```

산술 연산자

- **산술 연산자**

연산자	기 능	연산 예
+	두 항을 더합니다.	5+3
-	앞 항에서 뒤 항을 뺍니다.	5-3
*	두 항을 곱합니다.	5*3
/	앞 항에서 뒤 항을 나누어 몫을 구합니다.	5/3
%	앞 항에서 뒤 항을 나누어 나머지를 구합니다.	5%3

증감 연산자

- **증가 감소 연산자**
  - 1만큼 더하거나 1만큼 뺄 때 사용하는 연산자

연산자	기 능	연산 예
++	항의 값에 1을 더 합니다.	val = ++num; **// num = num+1; val=num** val = num++; **//val=num; num=num+1;**
--	항의 값에서 1을 뺍니다.	val = --num **// num = num+1; val=num** val = num--; **//val=num; num=num+1;**

산술 연산자 연습문제

```java
public class OperationEX2 {
    public static void main(String[] args) {
        //증가, 감소 연산자
        int num = 10;
        int val = 0;

        //val = num++; //val = num 이후 num = num + 1;
        val = ++num;   //num = num + 1 이후 val = num
        System.out.println(val);
        System.out.println(num);

        //val = num--;  //val = num 이후 num = num - 1;
        val = --num;    //num = num - 1 이후 val = num
        System.out.println(val);
        System.out.println(num);

        //산술 연산자
        int mathScore = 90;
        int engScore = 75;

        int totalScore = mathScore + engScore;
        System.out.println(totalScore);

        double avgScore = (double)totalScore / 2;
        System.out.println(avgScore);
    }
}
```

관계(비교) 연산자

- **관계(비교) 연산자**

연산자	기 능	연산 예
>	왼쪽 항이 크면 참을, 아니면 거짓을 반환합니다.	num > 3;
<	왼쪽 항이 작으면 참, 아니면 거짓을 반환합니다.	num < 3;
>=	왼쪽 항이 크거나 같으면 참, 아니면 거짓을 반환합니다.	num >= 3;
<=	왼쪽 항이 작거나 같으면 참, 아니면 거짓을 반환합니다.	num <= 3;
==	두 개의 항 값이 같으면 참, 아니면 거짓을 반환합니다.	num == 3;
!=	두 개의 항 값이 다르면 참, 아니면 거짓을 반환합니다.	num != 3

논리 연산자

- **논리 연산자**

연산자	기 능	연산 예
&&	두 항이 모두 참인 경우에만 결과 값이 참 입니다.	boolean = (7<3) && (5>2)
||	두 항중 하나의 항만 참이면 결과 값이 참 입니다.	boolean = (7<3) || (5>2)
!	단항 연산자, 참인 경우는 거짓으로, 거짓인 경우는 참으로 바꿉니다.	boolean = !(7>3)

논리 연산자

```java
public class OperationEx3 {

    public static void main(String[] args) {
        //관계(비교) 연산자
        System.out.println(7 < 3);
        System.out.println(7 > 3);
        System.out.println(7 == 3);
        System.out.println(7 !=  3);

        //논리 연산자
        System.out.println((7 > 3) && (5 > 2));
        System.out.println((7 > 3) || (5 < 2));  //단락 회로
        System.out.println(!(7 > 3));
    }
}
```

논리 연산자

- **논리 연산자 – 단락 회로**
  - 앞 조건이 거짓이면 뒤 조건은 연산하지 않음(&&인 경우)
  - 앞 조건이 참이면 뒤 조건은 연산하지 않음(||인 경우)

```
public class LogicalOperator {

    public static void main(String[] args) {
        //단락회로 평가 실습
        int n = 10;
        int i = 2;
        boolean value = ((n = n + 10) < 10) && ((i = i + 2) < 10);
        System.out.println(value);  //false
        System.out.println(n);      //20
        System.out.println(i);      //2

        System.out.println("------");
        value = ((n = n + 10) > 10) || ((i = i + 2) < 10);
        System.out.println(value);  //true
        System.out.println(n);      //30
        System.out.println(i);      //2
    }
}
```

복합대입 연산자

- **복합대입 연산자**

연산자	기 능	연산 예
+=	두 항의 값을 더해서 왼쪽 항에 대입합니다.	num += 2; num=num+2와 같음
-=	왼쪽 항에서 오른쪽 항을 빼서 그 값을 왼쪽 항에 대입합니다.	num -= 2; num=num-2와 같음
*=	두 항의 값을 곱해서 왼쪽 항에 대입합니다.	num *= 2; num=num*2와 같음
/=	왼쪽 항을 오른쪽 항으로 나누어 그 몫을 왼쪽 항에 대입합니다.	num /= 2; num=num/2와 같음
%=	왼쪽 항을 오른쪽 항으로 나누어 그 나머지를 왼쪽 항에 대입합니다.	num %= 2; num=num%2와 같음

조건 연산자

- **조건 연산자**

  삼항 연산자 -> 제어문중 조건문을 간단히 표현할 때 사용할 수 있음

연산자	기 능	연산 예
조건식?결과1:결과2;	조건식이 참이면 결과1, 조건식이 거짓이면 결과2가 선택됩니다.	int num = (5>3)?10:20;

조건 연산자

```java
public class OperationEx4 {

    public static void main(String[] args) {
        //복합대입 연산자
        int num = 10;
        System.out.println(num += 1);
        System.out.println(num %=10);
        num -= 1;
        System.out.println(num);

        //조건 연산자
        System.out.println("---------");
        //부모님의 나이 비교
        boolean bool = (5 > 3) ? true : false;
        System.out.println(bool);

        int fatherAge = 45;
        int motherAge = 47;

        char ch;
        ch = (fatherAge > motherAge) ? 'T' : 'F';
        System.out.println(ch);
    }
}
```

비트 연산자

- **비트 연산자**

연산자	기 능	연산 예
~	비트의 반전(1의 보수)	a = ~a
&	비트 단위 AND	1 & 1 -> 1을 반환, 그 외는 0
|	비트 단위 OR	0 | 0 -> 0을 반환, 그 외는 1
<<	왼쪽 shift	a<< 2 변수 a를 2비트 만큼 왼쪽으로 이동
>>	오른쪽 shift	a<< 2 변수 a를 2비트 만큼 오른쪽으로 이동

비트 연산자

- **비트 논리연산자**

```
int num1 = 5;
int num2 = 10;
int result = num1 &
num2;
```

```
num1 : 0 0 0 0 0 1 0 1
& num2 : 0 0 0 0 1 0 1 0
-----------------------
result : 0 0 0 0 0 0 0 0
```

- **비트 이동 연산자**

왼쪽으로 2자리 이동

```
int num = 5;
num << 2;
```

```
num      : 0 0 0 0 0 1 0 1
num<<2 : 0 0 1 0 1 0 0 0
```

비트 연산자

```java
public class OperationEx5 {
    public static void main(String[] args) {
        //비트 논리 연산자
        int num1 = 5;     //00000101
        int num2 = 10;    //00001010
        int result = num1 & num2;  //00000000
        System.out.println(result);

        result = num1 | num2;      //00001111
        System.out.println(result);

        //비트 이동 연산자
        int val = 2;   //00000010  -> 10진수 2
        System.out.println(val << 1);  //00000100 -> 10진수 4
        System.out.println(val << 2);  //00001000 -> 10진수 8
        System.out.println(val);
        System.out.println(val >> 1);  //00000001  -> 10진수 1
    }
}
```

연산자 우선 순위

- **연산자 우선 순위**

  위쪽과 왼쪽이 우선 순위가 높음

우선순위	형태	연산자
1	일차식	( ) [ ]
2	단항	++ -- !
3	산술	% * / + -
4	비트이동	<< >>
5	관계	< > == !=
6	비트 논리	& | ~
7	논리	&& || !
8	조건	? :
9	대입	= += -= *=

입력 처리 – Scanner 클래스

화면에서 입력하기 - Scanner 클래스

- 문자, 숫자등의 자료를 표준 입력으로부터 읽어 들일 수 있다.
- Java.util 패키지에 속해있어서 import해야 한다.
- 종료시에 close() 메서드를 사용한다. -> scanner.close()

```
Scanner scanner = new Scanner(System.in)
```

메서드	설명
int nextInt()	int 자료형을 읽습니다.
double nextDouble()	double 자료형을 읽습니다.
String next()	문자열 String을 읽습니다.(₩n 버퍼에 남음)
String nextLine()	문자열 String을 읽습니다.(₩n을 포함)

Java API

- **Java Document -> Scanner 클래스 찾기**

  프로그램에서 데이터를 주고받기 위한 방법 등을 기술한 문서

Module java.base
Package java.util

Class Scanner

java.lang.Object
    java.util.Scanner

All Implemented Interfaces:

`Closeable, AutoCloseable, Iterator<String>`

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches wh
methods.

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

입력 처리 – Scanner 클래스

- Scanner 클래스 사용하기

```
import java.util.Scanner;

public class ScannerEx {

    public static void main(String[] args) {
        //Scanner 클래스 사용
        Scanner scanner = new Scanner(System.in);

        System.out.print("당신의 이름은 무엇입니까? ");
        String name = scanner.nextLine();
        System.out.println("당신의 이름은 " + name + "이군요.");

        System.out.print("당신의 나이는 몇 살입니까? ");
        int age = scanner.nextInt();
        System.out.println("당신의 나이는 " + age + "세 이군요.");

        scanner.close();
    }
}
```

Scanner 객체 생성

nextLine()로 이름입력

nextInt()로 나이입력

close()로 종료한다.

구매 포인트 계산 프로그램

- 고객의 구매 포인트 계산 프로그램

```
public class BonusPoint {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.print("고객의 이름을 입력하세요 : ");
        String name = scan.nextLine(); //고객 이름

        System.out.print("구매 금액을 입력하세요 : ");
        int price = scan.nextInt();  //가격
        int bonusPoint = 0;          //보너스 포인트
        double bonusRatio = 0.05;    //보너스적립율 : 5%

        //보너스포인트 = 가격 x 보너스적립율
        bonusPoint = (int)(price * bonusRatio);
        System.out.println(name + " 님의 보너스 포인트는 " + bonusPoint + "점입니다." );

        scan.close();
    }
}
```

```
고객의 이름을 입력하세요 : 김검소
구매 금액을 입력하세요 : 10000
김검소님의 구매 포인트는 500점입니다.
```

속도 KM를 마일로 변환하는 프로그램

메이저리그는 점점 더 빠른 구속을 추구하고 있다. 올 시즌 메이저리그 포심 패스트볼 평균 구속은 시속 93.2마일(150.0km)에 달한다. 이제는 100마일(160.9km)이 넘는 공도 어렵지 않게 볼 수 있게 됐다.

투수에게 있어 구속이 가장 중요한 요소는 아니다. 구종, 제구, 구위 등 수 많은 요소들이 어우러져야 비로소 뛰어난 투구를 할 수 있다. 하지만 같은 조건이라면 당연히 구속이 빠를수록 유리하다. 구속이 빠를수록 타자들이 공에 대처할 수 있는 물리적인 시간이 줄어들기 때문이다.

속도 KM를 마일로 변환하는 프로그램

◈ 속도 KM를 마일로 바꾸는 프로그램

```
public class KmToMile {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        final double RATE_KPH_MPH = 1.609344;  //변환 상수
        double kph = 0.0;         //km/h
        double mph = 0.0;  //mile/h

        System.out.print("당신의 구속을 입력하세요(km/h) : ");
        kph = scan.nextDouble();

        //mile = km / 변환 상수
        mph = kph / RATE_KPH_MPH;

        //printf("문자열 포맷", 객체)
        System.out.printf("공의 속도는 %.2f[MPH]입니다.", mph);
        scan.close();
    }
}
```

```
당신의 구속을 입력하세요(km/h) : 140.5
공의 속도는 87.30[MPH]입니다.
```

입력 처리 – 연습 예제

- 실습 예제

숫자를 입력받아 홀수/짝수를 판별하는 프로그램을 작성하세요

(조건 연산자 활용 – 예)

☞ 실행 결과

```
숫자 입력 : 11
홀수
```