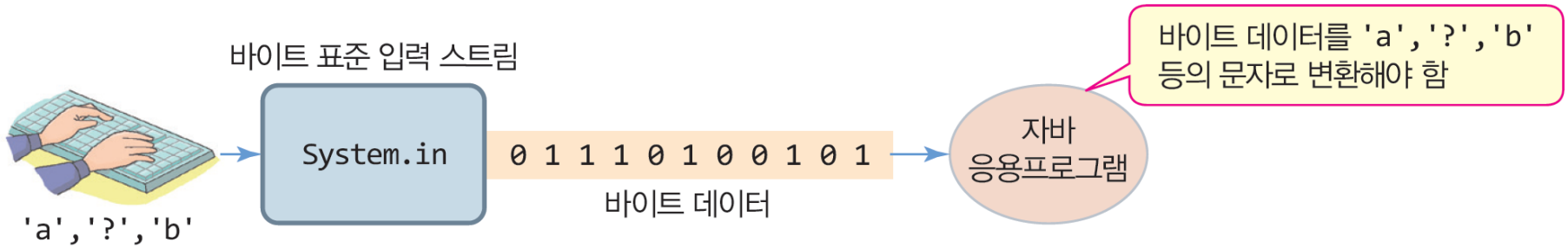
Revisit Java Programming
(Operator and Branch)

Fall, 2020

Table of Contents

Standard Input (Scanner)

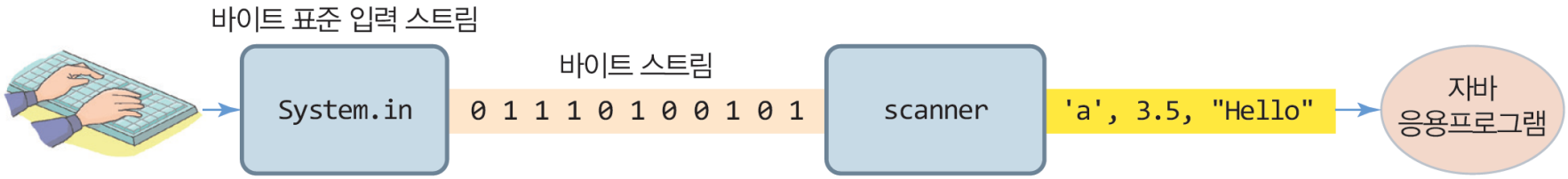
- System.in
  - Standard Input from Keyboard
  - returns key inputs as bytes



Standard Input (Scanner)

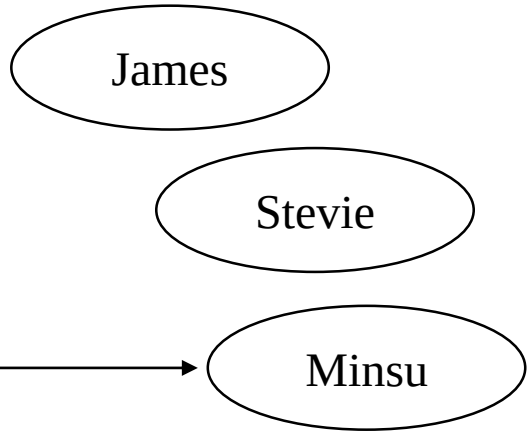
- Scanner Class
  - Let System.in read keyboard inputs
  - Return either character, integer, real number, boolean, or String from the inputs
- Instantiate the class
  - java.util.Scanner; will be used (Verbose, you can shorten it via import)

```
import java.util.Scanner; // import for concise code
...
Scanner a = new Scanner(System.in); // instantiate Scanner class
```



Standard Input (Scanner)

- Basic introduction to Class
  - java.util.Scanner is a Class
  - your kr.ac.sejong.icse.HellWorld is also another Class



Human Class	
Local Variables	float height; float weight; boolean gender; …

Methods	void setWeight(float weight);
	float getWeight();
	void shout();

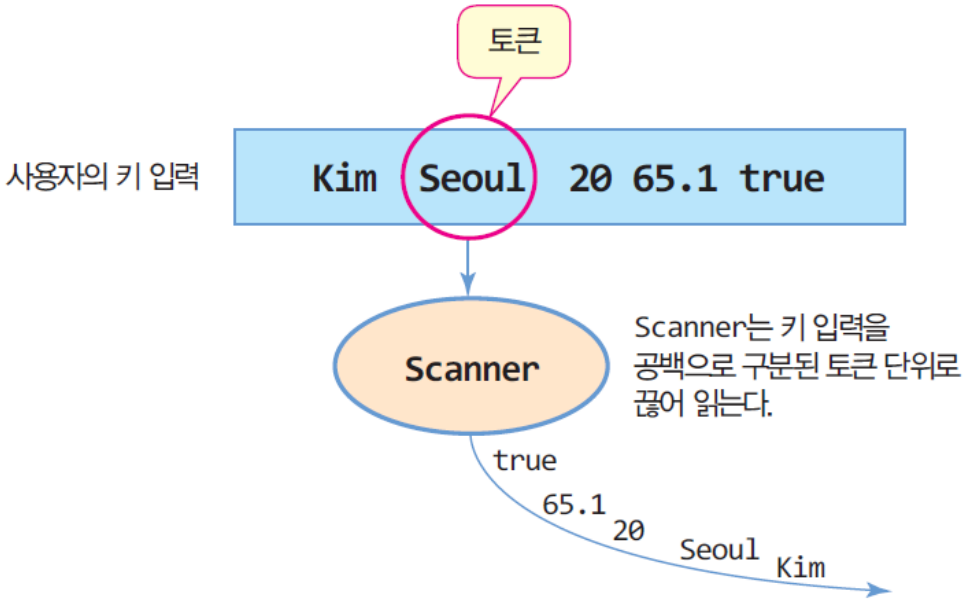
instantiate

```
Human minsu = new Human();
minsu.setWeight(70.5);
int minsuWeight = minsu.getWeight();
```

  - Until lectures for Class, you will only use static methods

Standard Input (Scanner)

- Keyboard Inputs
  - Scanner delimits inputs with
    - Tab: '\t'
    - New line: '\n'
    - White Space: ' '
    - etc.




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

String name = scanner.next();              // "Kim"
String city = scanner.next();              // "Seoul"
int age = scanner.nextInt();               // 20
double weight = scanner.nextDouble();      // 65.1
boolean single = scanner.nextBoolean();    // true
```

Standard Input (Scanner)

메소드	설명
String next()	다음 토큰을 문자열로 리턴
byte nextByte()	다음 토큰을 byte 타입으로 리턴
short nextShort()	다음 토큰을 short 타입으로 리턴
int nextInt()	다음 토큰을 int 타입으로 리턴
long nextLong()	다음 토큰을 long 타입으로 리턴
float nextFloat()	다음 토큰을 float 타입으로 리턴
double nextDouble()	다음 토큰을 double 타입으로 리턴
boolean nextBoolean()	다음 토큰을 boolean 타입으로 리턴
String nextLine()	'\n'을 포함하는 한 라인을 읽고 '\n'을 버린 나머지 문자열 리턴
void close()	Scanner의 사용 종료
boolean hasNext()	현재 입력된 토큰이 있으면 true, 아니면 입력 때까지 무한정 대기, 새로운 입력이 들어올 때 true 리턴. crtl-z 키가 입력되면 입력 끝이므로 false 리턴

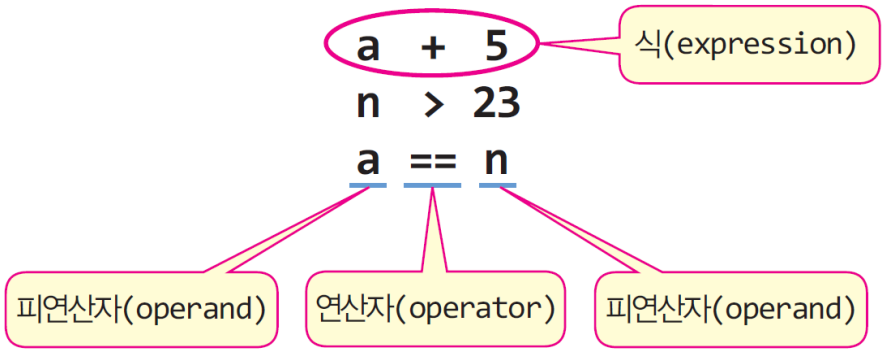
Practice 0

- Practice Scanner

```java
import java.util.Scanner;

// Textbook
public class P5WrapUp {

	public static void main(String[] args) {
		// Write a program
		// Ask name, city, age, weight, isSingle and print out
		System.out.println("이름, 도시, 나이, 체중, 독신 여부를 빈칸으로 분리하여 입력하세요");
		Scanner scanner = new Scanner(System.in);
		String name = scanner.next(); // 문자열 읽기
		System.out.print("이름은 " + name + ", ");
		scanner.close();
	}
}
```

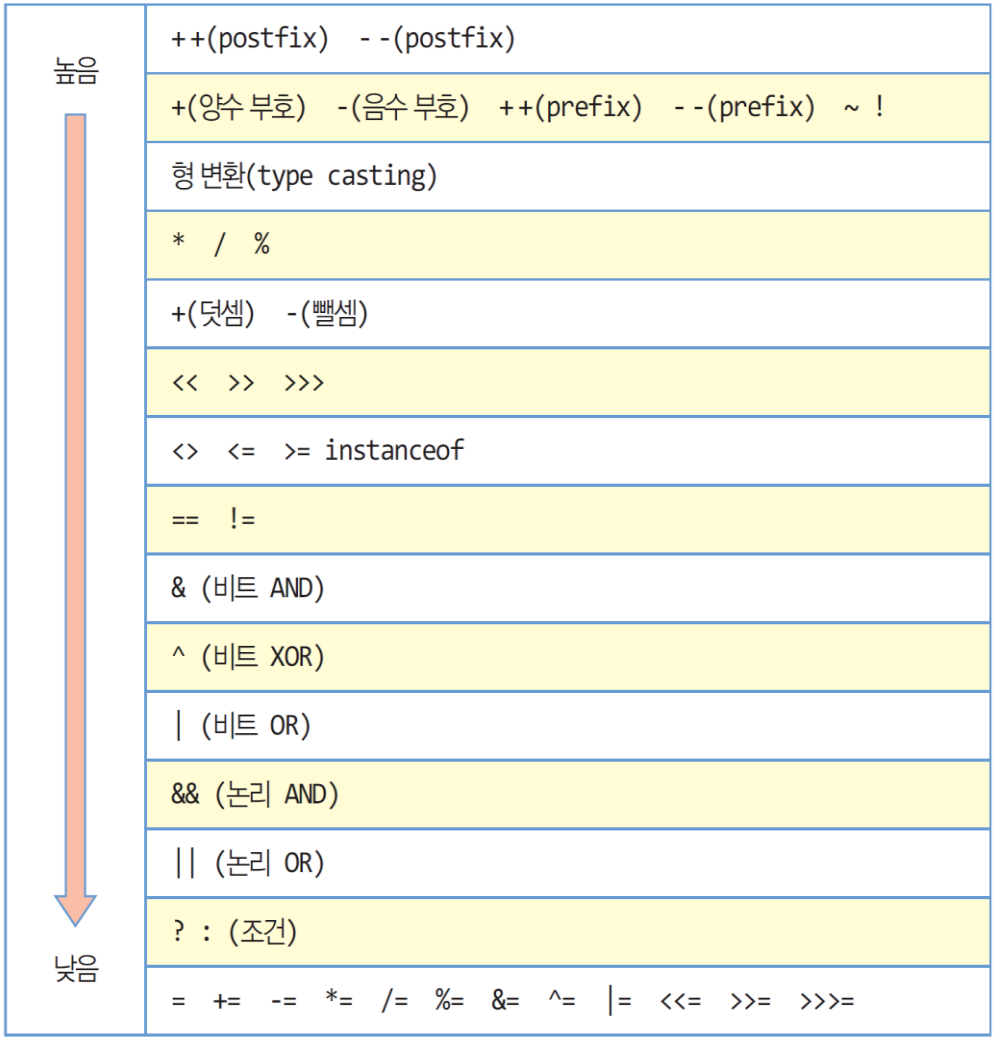
Operators in an expression

- You can get results from expression



연산의 종류	연산자	연산의 종류	연산자
증감	++ --	비트	& | ^ ~
산술	+ - * / %	논리	&& || ! ^
시프트	>> << >>>	조건	? :
비교	> < >= <= == !=	대입	= *= /= += -= &= ^= |= <<= >>= >>>=

Operator Precedence



- Some Rules
  - For same precedence
    - From left to right
  - Exception
    - 대입 연산자, --, ++, +,-(양수 음수 부호), !, 형 변환은 오른쪽에서 왼쪽으로 처리
- Top precedence
  - ()
  - Nested (), from inside to outside

Arithmetic Operator

- Add (+), Minus (-), Multiply (*), Division (/), Modular Operator (%)
  - Some Example
    - Quotient and Remainder

```
69/10 = 6          ← Quotient 6
69%10 = 9          ← Remainder 9
```

    - Check Odd / Even number

```
int r = n % 2;          // r이 1이면 n은 홀수, 0이면 짝수
```

연산자	의미	예	결과
+	더하기	25.5 + 3.6	29.1
-	빼기	3 - 5	-2
*	곱하기	2.5 * 4.0	10.0
/	나누기	5/2	2
%	나머지	5%2	1

Practice 1

- Practice Arithmetic Operator

```java
// Java provides
// Arithmetic Operators: + - * / %
// Incremental/Decremental Operator: ++ --
// Assignment Operator =   = += -= *= /= %= &= ^= |= <<= >>= >>>=
// Comparative Operator: < > <= >= == !=
// Logical Operator: ! || &&
// Conditional Operator
// Bitwise Operator
// Shift Operator
public class P1ArithmeticOperator {

	public static void main(String[] args) {
		double x = 8d;
		double y = 5d;

		System.out.println(x + y);
		System.out.println(x - y);
		System.out.println(x * y);
		System.out.println(x / y);
		System.out.println(x % y);

	}
}
```

Practice 2 (예제 2-5 : /와 % 산술 연산)

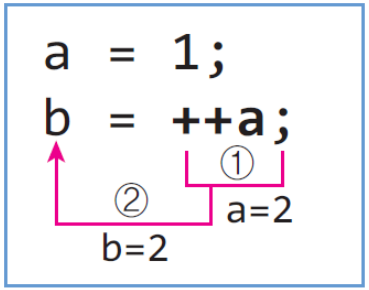
초 단위의 정수를 입력받고, 몇 시간, 몇 분, 몇 초인지 출력하는 프로그램을 작성하라.

```
import java.util.Scanner;

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

        System.out.print("정수를 입력하세요: ");
        int time = scanner.nextInt();         // 정수 입력
        int second = time % 60;               // 60으로 나눈 나머지는 초
        int minute = (time / 60) % 60;   // 60으로 나눈 몫을 다시 60으로 나눈 나머지는 분
        int hour = (time / 60) / 60;         // 60으로 나눈 몫을 다시 60으로 나눈 몫은 시간

        System.out.print(time + "초는 ");
        System.out.print(hour + "시간, ");
        System.out.print(minute + "분, ");
        System.out.println(second + "초입니다.");

        scanner.close();
    }
}
```

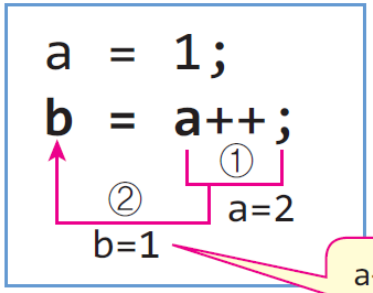
```
정수를 입력하세요:5000
5000초는 1시간, 23분, 20초입니다.
```

Incremental Operator

- Expressed by Arithmetic Operator
  - ++ : add 1 to the value
    - a ++; // Identical to a = a + 1;
  - -- : minus 1 to the value
    - b --; // Identical to b = b – 1;



(a) 전위 연산자



(b) 후위 연산자

a++의 연산은 증가 전의 값 1을 반환한다.

연산자	내용	연산자	내용
a++	a를 1 증가하고 증가 전의 값 반환	++a	a를 1 증가하고 증가된 값 반환
a--	a를 1 감소하고 감소 전의 값 반환	--a	a를 1 감소하고 감소된 값 반환

Practice 3

- Practice Incremental Operator

```java
// Java provides
// Arithmetic Operators: + - * / %
// Incremental/Decremental Operator: ++ --
// Assignment Operator =   = += -= *= /= %= &= ^= |= <<= >>= >>>=
// Comparative Operator: < > <= >= == !=
// Logical Operator: ! || &&
// Conditional Operator
// Bitwise Operator
// Shift Operator
public class P2IncrementalOperator {

        public static void main(String[] args) {
                // ++ -- is just for convenience
                // represented by arithmetic operators

                int x = 2;
                // ++ prefix -> increase first
                // ++ postfix -> increase next

                // equivalent to
                // x = x+1; System.out.println(x);
                System.out.println(++x);
                // equivalent to
                // System.out.println(x);
                // x = x+1;
                System.out.println(x++);
                System.out.println(x);

                // Practice with --
        }
}
```

Assignment Operator

- Assign the right value to the left variable

```
int a = 1, b = 3;
a = b;          // b 값을 a에 대입하여 a=3
a += b;         // a = a + b의 연산이 이루어져, a=6. b는 3 그대로
```

대입 연산자	내용	대입 연산자	내용
a = b	b의 값을 a에 대입	a &= b	a = a & b와 동일
a += b	a = a + b와 동일	a ^= b	a = a ^ b와 동일
a -= b	a = a - b와 동일	a |= b	a = a | b와 동일
a *= b	a = a * b와 동일	a <<= b	a = a << b와 동일
a /= b	a = a / b와 동일	a >>= b	a = a >> b와 동일
a %= b	a = a % b와 동일	a >>>= b	a = a >>> b와 동일

Practice 4

- Practice Assignment Operator

```java
// Java provides
// Arithmetic Operators: + - * / %
// Incremental/Decremental Operator: ++ --
// Assignment Operator =   = += -= *= /= %= &= ^= |= <<= >>= >>>=
// Comparative Operator: < > <= >= == !=
// Logical Operator: ! || &&
// Conditional Operator
// Bitwise Operator
// Shift Operator
public class P3AssignmentOperator {

        public static void main(String[] args) {
                // You've already used Assignment Operator =
                // it allows you to assign a value to a variable
                // the following makes a variable x have a value 2
                int x = 2;

                // += -= *= /= % = is just for convenience
                // x += 2 is equivalent to x = x + 2;
                x += 8;
                System.out.println(x);
                // Practice the convenient method

        }
}
```

Practice 5

다음 코드의 실행 결과는 무엇인가?

```
public class AssignmentIncDecOperator {
    public static void main(String[] args) {
        int a=3, b=3, c=3;

        // 대입 연산자 사례
        a += 3;         // a=a+3 = 6
        b *= 3;         // b=b*3 = 9
        c %= 2;         // c=c%2 = 1
        System.out.println("a=" + a + ", b=" + b + ", c=" + c);

        int d=3;
        // 증감 연산자 사례
        a = d++;  // a=3, d=4
        System.out.println("a=" + a + ", d=" + d);
        a = ++d;  // d=5, a=5
        System.out.println("a=" + a + ", d=" + d);
        a = d--;        // a=5, d=4
        System.out.println("a=" + a + ", d=" + d);
        a = --d;        // d=3, a=3
        System.out.println("a=" + a + ", d=" + d);
    }
}
```

```
a=6, b=9, c=1
a=3, d=4
a=5, d=5
a=5, d=4
a=3, d=3
```

Comparative Operator

- Return true or false

연산자	내용	예제	결과
a < b	a가 b보다 작으면 true	3<5	true
a > b	a가 b보다 크면 true	3>5	false
a <= b	a가 b보다 작거나 같으면 true	1<=0	false
a >= b	a가 b보다 크거나 같으면 true	10>=10	true
a == b	a가 b와 같으면 true	1==3	false
a != b	a가 b와 같지 않으면 true	1!=3	true

Practice 6

- Practice Comparative Operator

```java
// Java provides
// Arithmetic Operators: + - * / %
// Incremental/Decremental Operator: ++ --
// Assignment Operator =   = += -= *= /= %= &= ^= |= <<= >>= >>>=
// Comparative Operator: < > <= >= == !=
// Logical Operator: ! || &&
// Conditional Operator
// Bitwise Operator
// Shift Operator
public class P4ComparativeOperator {

	public static void main(String[] args) {
		// Comparative Operators < > <= >= == !=
		// returns boolean value

		int x = 35;
		int y = 49;

		System.out.println("Is x larger than y? " + (x > y));

		// practice > <= >= == !=
	}
}
```

Logical Operator

- Manipulate logical values

연산자	내용	예제	결과
! a	a가 true이면 false, false이면 true	!(3<5)	false
a || b	a와 b의 OR 연산. a와 b 모두 false인 경우에만 false	(3>5)||(1==1)	true
a && b	a와 b의 AND 연산. a와 b 모두 true인 경우에만 true	(3<5)&&(1==1)	true

Practice 7

- Practice Logical Operator

```java
// Java provides
// Arithmetic Operators: + - * / %
// Incremental/Decremental Operator: ++ --
// Assignment Operator =   = += -= *= /= %= &= ^= |= <<= >>= >>>=
// Comparative Operator: < > <= >= == !=
// Logical Operator: ! || &&
// Conditional Operator
// Bitwise Operator
// Shift Operator
public class P5LogicalOperator {

	public static void main(String[] args) {
		// Logical operators allow you to concatenate two or more comparative
		// expressions

		int x = 35;

		boolean result = (x > 30) && (x < 40);
		System.out.println("Is x larger than 30 and less than 40? " + result);

		boolean isMale = true;
		int age = 50;

		boolean result2 = (!isMale) && (age >= 50);
		System.out.println("Are you female and your age is larger and equal to 50? " + result2);

		// practice ||
	}
}
```

Practice 8 (예제 2-7 : 비교 연산자와 논리 연산자 사용하기)

다음 소스의 실행 결과는 무엇인가?

```
public class LogicalOperator {
   public static void main (String[] args) {
      // 비교 연산
      System.out.println('a' > 'b');
      System.out.println(3 >= 2);
      System.out.println(-1 < 0);
      System.out.println(3.45 <= 2);
      System.out.println(3 == 2);
      System.out.println(3 != 2);
      System.out.println(!(3 != 2));

      // 비교 연산과 논리 연산 복합
      System.out.println((3 > 2) && (3 > 4));
      System.out.println((3 != 2) || (-1 > 0));
   }
}
```

```
false
true
true
false
false
true
false
false
true
```

Conditional Operator

- condition ? opr2 : opr3
  - Ternary Operator
    - If condition true
      - The result will be opr2
    - Else
      - The result will be opr3

```
int x = 5;
int y = 3;

int s;
if(x>y)
   s = 1;
else
   s = -1;
```

```
int s = (x>y)?1:-1;
```

Practice 9 (예제 2-8 : 조건 연산)

다음은 조건 연산자의 사례이다. 실행 결과는 무엇인가?

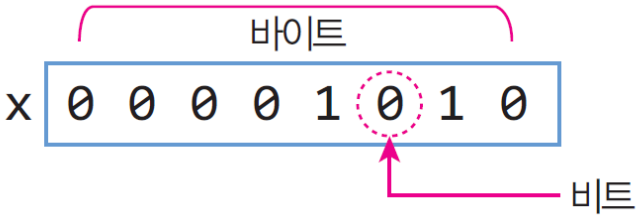
```
public class TernaryOperator {
    public static void main (String[] args) {
        int a = 3, b = 5;

        System.out.println("두 수의 차는 " + ((a>b)?(a-b):(b-a)));
    }
}
```

두 수의 차는 2

Bitwise Operator

- Bit

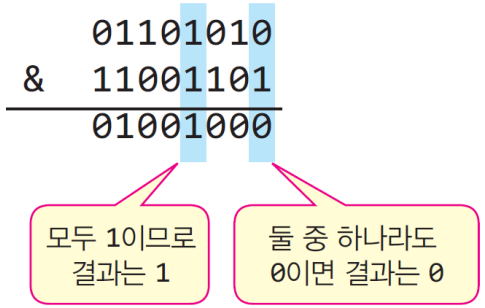
```
byte x = 10;
```

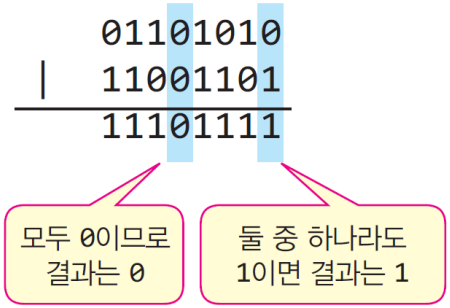


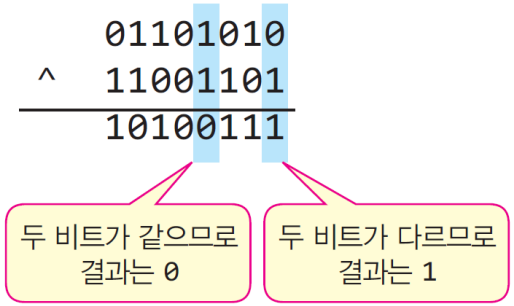
- Bitwise Operator
  - AND, OR, XOR, NOT
  - SHIFT
    - MOVE TO LEFT
    - MOVE TO RIGHT

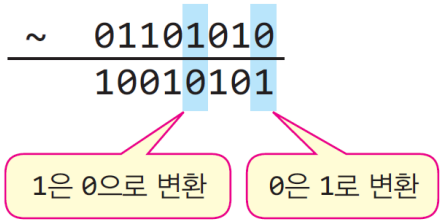
- Very useful in resource constraint environment

Bitwise Logical Operator





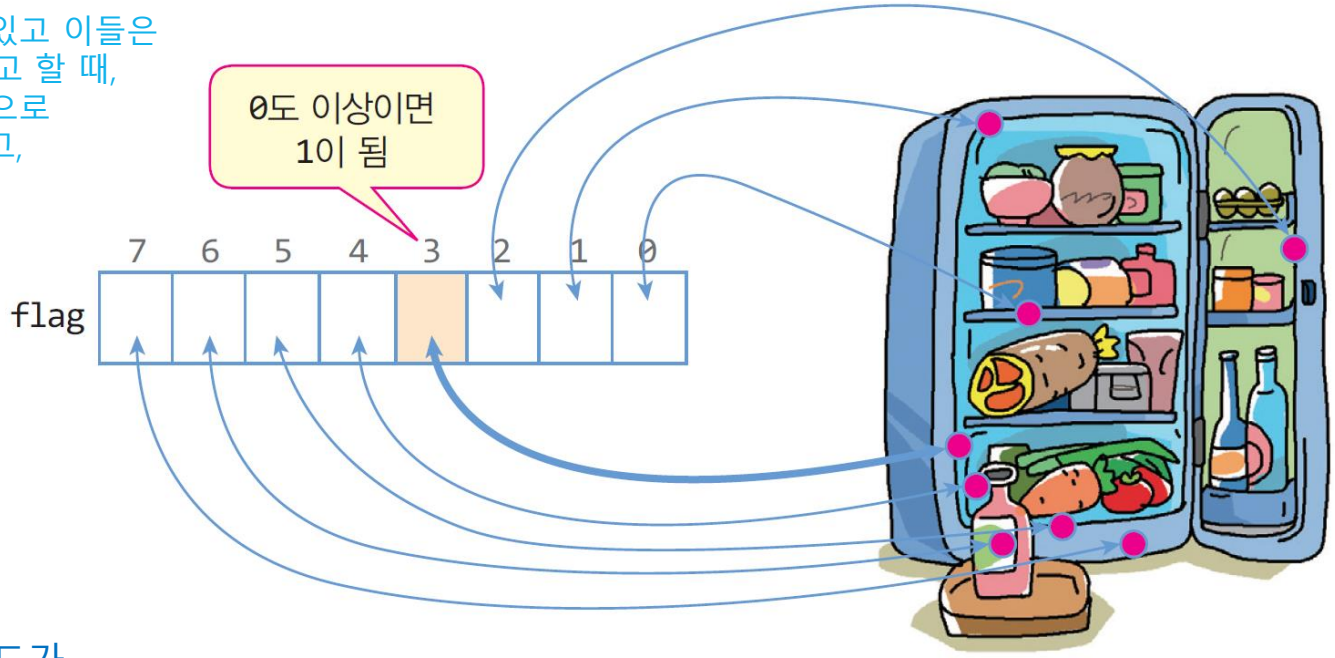




연산자	별칭	내용
a & b	AND 연산	두 비트 모두 1이면 1, 그렇지 않으면 0
a | b	OR 연산	두 비트 모두 0이면 0, 그렇지 않으면 1
a ^ b	XOR 연산	두 비트가 다르면 1, 같으면 0
~ a	NOT 연산	1을 0으로, 0을 1로 변환

Bitwise Logical Operator (Example)

냉장고에는 8개의 센서가 있고 이들은 flag 변수와 연결되어 있다고 할 때, 냉장고의 온도가 0도 이상으로 올라가면 비트 3이 1이 되고, 0도 이하이면 비트 3이 0을 유지한다.



문제) 현재 냉장고의 온도가
0도 이상인지 판단하는 코드를 작성하라.

```
byte flag = 0b00001010; // 각 비트는 8개의 센서 값을 가리킴
if(flag & 0b00001000 == 0)
    System.out.print("온도는 0도 이하");
else
    System.out.print("온도는 0도 이상");
```

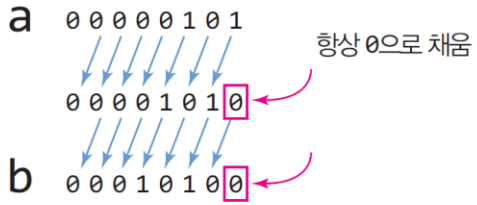
온도는 0도 이상

	x	x	x	x	Y	x	x	x
&	0	0	0	0	1	0	0	0
	0	0	0	0	Y	0	0	0

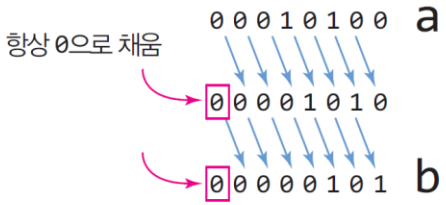
Y비트가 0 이면
& 결과는 0

Bitwise Shift Operator (Example)

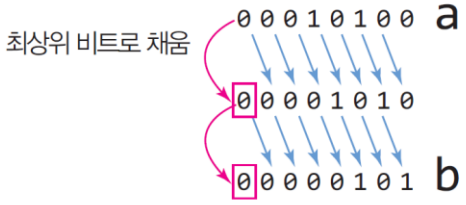
```
byte a = 5; // 5
byte b = (byte)(a << 2); // 20
```



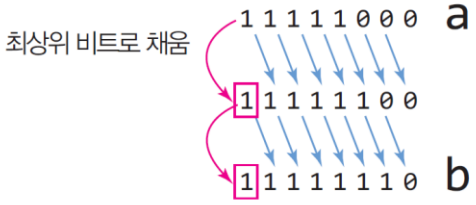

```
byte a = 20; // 20
byte b = (byte)(a >>> 2); // 5
```




```
byte a = 20; // 20
byte b = (byte)(a >> 2); // 5
```




```
byte a = (byte)0xf8; // -8
byte b = (byte)(a >> 2); // -2
```



시프트 연산자	내용
a >> b	a의 각 비트를 오른쪽으로 b번 시프트한다. 최상위 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다. 산술적 오른쪽 시프트라고 한다.
a >>> b	a의 각 비트를 오른쪽으로 b번 시프트한다. 최상위 비트의 빈자리는 항상 0으로 채운다. 논리적 오른쪽 시프트라고 한다.
a << b	a의 각 비트를 왼쪽으로 b번 시프트한다. 최하위 비트의 빈자리는 항상 0으로 채운다. 산술적 왼쪽 시프트라고 한다.

Practice 10 (예제 2-9 : 비트 논리 연산과 비트 시프트 연산)

다음 소스의 실행 결과는 무엇인가?

```
public class BitOperator {
    public static void main(String[] args) {
        short a = (short)0x55ff;
        short b = (short)0x00ff;

        // 비트 논리 연산
        System.out.println("[비트 연산 결과]");
        System.out.printf("%04x\n", (short)(a & b)); // 비트 AND
        System.out.printf("%04x\n", (short)(a | b)); // 비트 OR
        System.out.printf("%04x\n", (short)(a ^ b)); // 비트 XOR
        System.out.printf("%04x\n", (short)(~a)); // 비트 NOT

        byte c = 20; // 0x14
        byte d = -8; // 0xf8

        // 비트 시프트 연산
        System.out.println("[시프트 연산 결과]");
        System.out.println(c <<2); // c를 2비트 왼쪽 시프트
        System.out.println(c >>2); // c를 2비트 오른쪽 시프트. 0 삽입
        System.out.println(d >>2); // d를 2비트 오른쪽 시프트. 1 삽입
        System.out.printf("%x\n", (d >>>2)); // d를 2비트 오른쪽 시프트. 0 삽입
    }
}
```

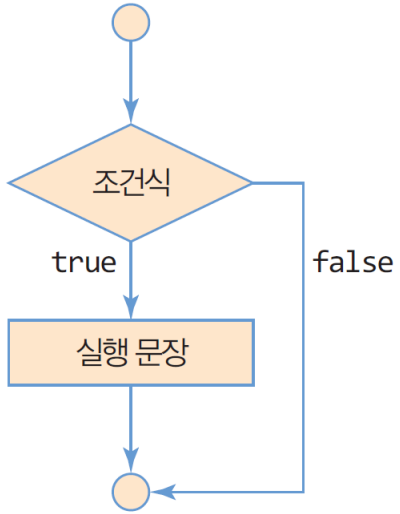
printf("%x\n", ...)는 결과 값을 16진수 형식으로 출력

```
[비트 연산 결과]
00ff
55ff
5500
aa00
[시프트 연산 결과]
80
5
-2
3ffffffe
```

Branch: if

- Syntax
  - if의 괄호 안에 조건식(논리형 변수나 논리 연산)
    - 실행문장이 단일 문장인 경우 둘러싸는 {, } 생략 가능

```
if (조건식) {
   ...실행 문장...  // 조건식이 참인 경우
}
```

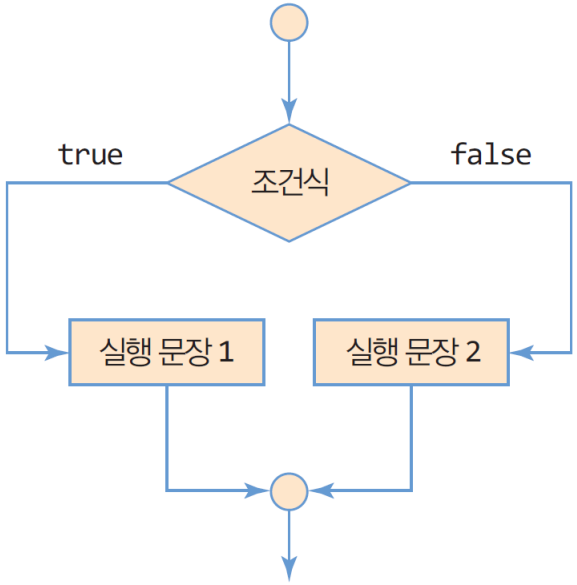



```
if(n%2 == 0) {
   System.out.println(n + "은 짝수입니다.");
}
```

Branch : if-else

- if-else 문
  - 조건식이 true면 실행문장1 실행 후 if-else문을 벗어남
  - false인 경우에 실행문장2 실행후, if-else문을 벗어남

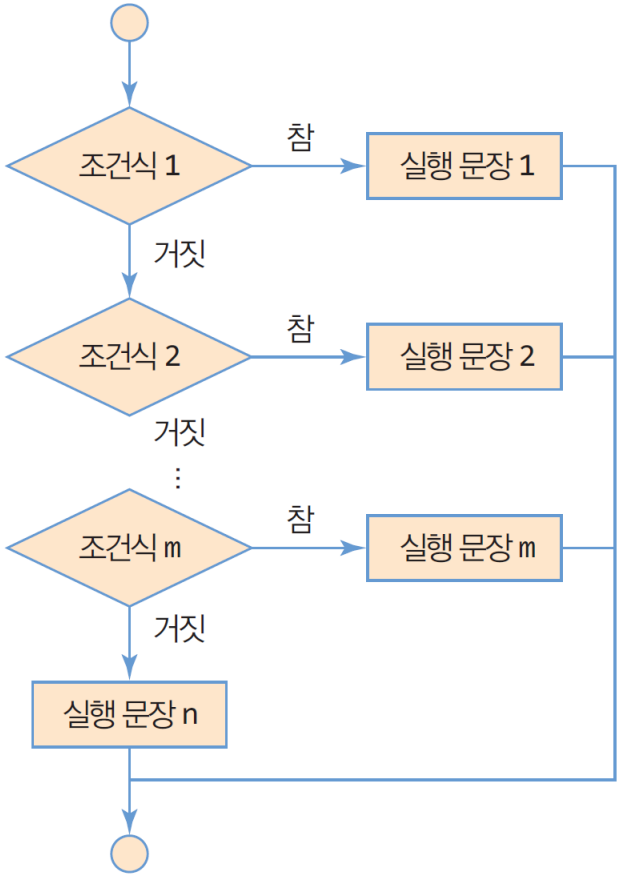
```
if (조건식) {
   ...실행 문장 1...
}
else {
   ...실행 문장 2...
}
```



Branch : if-else if-…-else

- 다중 if-else 문
  - if-else가 연속되는 모양
    - 조건문이 너무 많은 경우, switch 문 사용 권장

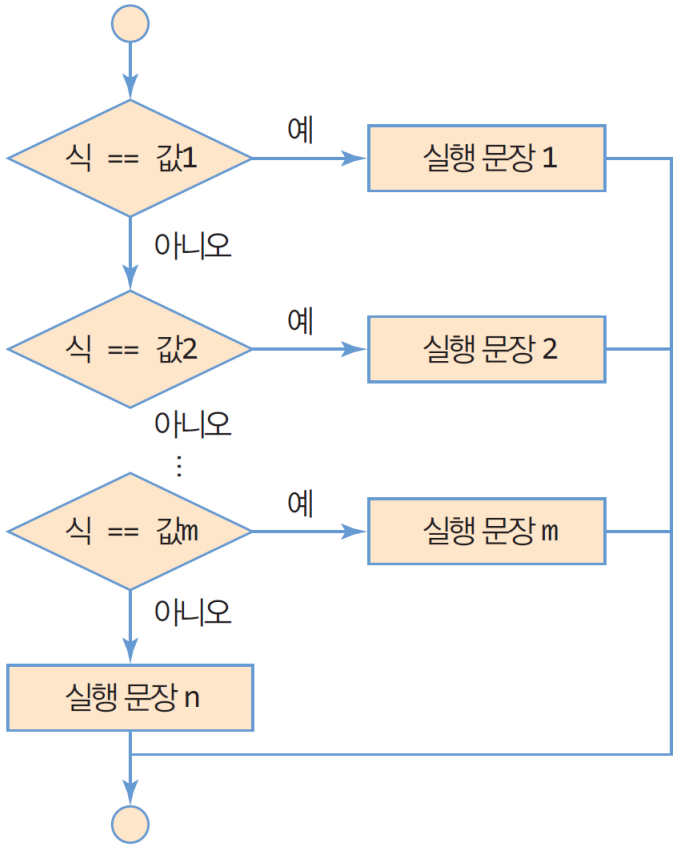
```
if (조건식 1) {
   실행 문장 1; // 조건식 1이 참인 경우
}
else if (조건식 2) {
   실행 문장 2; // 조건식 2가 참인 경우
}
else if (조건식 m) {
   ..... // 조건식 m이 참인 경우
}
else {
   실행 문장 n; // 앞의 모든 조건이 거짓인 경우
}
```



Branch: switch

- switch문은 식과 case 문의 값과 비교
  - case의 비교 값과 일치하면 해당 case의 실행문장 수행
    - break를 만나면 switch문을 벗어남
  - case의 비교 값과 일치하는 것이 없으면 default 문 실행
- default문은 생략 가능

```
switch (식) {
   case 값1:
      실행 문장 1;
      break;
   case 값2:
      실행 문장 2;
      break;
   ...
   case 값m:
      실행 문장 m;
      break;
   default:
      실행 문장 n;
}
```



Practice 11

- Practice if branch

```
3  // The lecture covers if-then-else statement
4  public class P1Branch {
5          public static void main(String[] args) {
6                  // Assuming your grade will be made absolutely as follows
7                  // >= 90 -> A
8                  // < 90 && >= 80 -> B
9                  // < 80 && >= 70 -> C
10                 // < 70 && >= 60 -> D
11                 // < 60 -> F
12
13                 int point = 85;
14                 char grade = 'F';
15
16                 if (point >= 90) {
17                         grade = 'A';
18                 } else if (point >= 80 && point < 90) {
19                         grade = 'B';
20                 } else {
21                         grade = 'C';
22                 }
23
24                 System.out.println(grade);
25
26                 // Make this program complete
27         }
28 }
```

Summary

- Operator
  - Arithmetic
  - Incremental
  - Assignment
  - Logical
  - Bitwise
  - Branch