연산자

- 연산자 종류
- 산술
- 비교 / 논리
- 비트 / 쉬프트
- 대입
- 삼항

3-1

연산자(Operator)

- 피 연산자(연산 대상)에 따라 연산자의 종류는 달라진다.
  - 단항 연산자 : 증감 연산(++, --)

```
num ++
```

  - 이항 연산자 : 산출연산, 비교연산, 논리연산, 비트연산, 쉬프트 연산, 대입연산

```
num + price
```

  - 삼항 연산자 : 조건식 ?값1 : 값2

```
(num >10) ? true : false
```

3-2

연산자 우선순위

- 연산자의 우선순위를 고려하면 프로그램을 효과적으로 개발할 수 있다.

순위	연산자
1	() [] .
2	++ -- ~ ! (type)
3	* / %
4	+ -
5	>> << >>>
6	> < >= <= instanceof
7	== !=
8	& ^ |
9	&& ||
10	?:
11	= += -= *= %= /= &= ^= |= <<= >>= >>>=

3-3

연산식

- 연산자와 연산의 우선순위에 의해서 구문이 실행된다.

```
int result = (++num + 1) + (result / 10) * 10
```

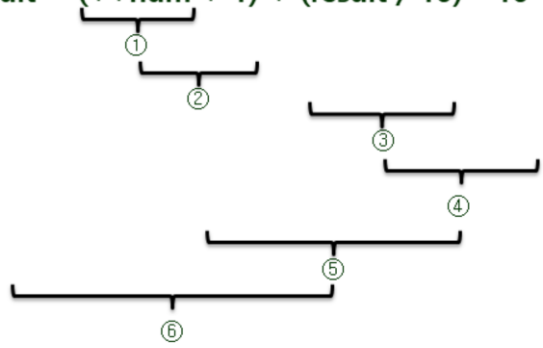
① ++num
② ① + 1
③ result / 10
④ ③ * 10
⑤ ② + ④
⑥ result = ⑤

3-4

연산식

- 연산자와 연산의 우선순위에 의해서 구문이 실행된다.

```
int result = (++num + 1) + (result / 10) * 10
```



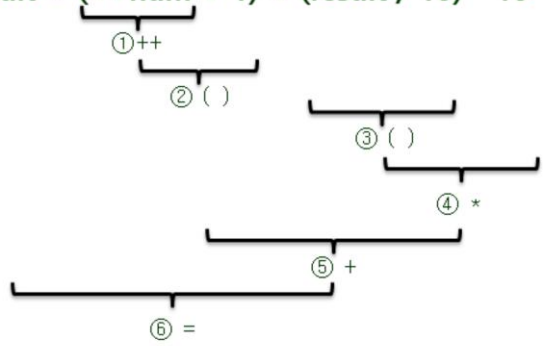
① ++num
② ① + 1
③ result / 10
④ ③ * 10
⑤ ② + ④
⑥ result = ⑤

3-5

연산식

- 연산자와 연산의 우선순위에 의해서 구문이 실행된다.

```
int result = (++num + 1) + (result / 10) * 10
```



① ++num
② ① + 1
③ result / 10
④ ③ * 10
⑤ ② + ④
⑥ result = ⑤

3-6

산술 연산자

- 연산 대상의 자료형이 서로 다를 경우 결과는 큰 자료형을 따른다.

연산자	구문	내용
++	++number number++	number에 1을 먼저 증가후 구문처리 구문 처리후 number에 1을 이후 증가
--	--number number--	number에 1을 먼저 감소후 구문처리 구문 처리후 number에 1을 이후 감소
+	num1 + num2	더하기
-	num1 - num2	배기
*	num1 * num2	곱하기
/	num1 / num2	나누기
%	num1 % num2	나머지

3-7

산술 연산자

- **실습)**

```
byte b1=10, b2=20, b3;
//b3=b1+b2;//컴파일 오류
b3=10+20;
//b3=127+1;  //컴파일 오류
float f1=10;
b3=(byte)(1+127);
long i=(long)(Integer.MAX_VALUE+1);
System.out.println(f1);
System.out.println(b3);
System.out.println(i);
System.out.println(b1++);
System.out.println(++b2);
System.out.println(b1);
System.out.println(b2);
```

3-8

산술 연산자

- **실습)**

```
public static void main(String[]  args) {
    float f1=10.0;  //컴파일 오류
    System.out.println(1+30+2.7F+1L);
    System.out.println(5/2);
    System.out.println(5/2.0);
    System.out.println(1+'A');
    System.out.println('A'+'A');
    System.out.println('A'+"A");
    System.out.println(123+"456");
}
```

3-9

산술 연산자

- 문자열 연산

3-10

비교 연산자

- 연산 대상의 값을 비교 후 결과를 boolean(true/false)로 반환
- 조건문이나 반복문 구현할 때 사용

연산자	구문	내용
>	num1 > num2	num1이 num2보다 클때 true
<	num1 < num2	num1이 num2보다 작을때 true
>=	num1 >= num2	num1이 num2보다 크거나 같을때 true
<=	num1 <= num2	num1이 num2보다 작거나 같을때 true
==	num1 == num2	num1과 num2가 같을때 true
!=	num1 != num2	num1과 num2가 같지 않을때 true
instanceof	ref1 instanceof ref2	ref1와 ref2의 객체형이 같을때 true

3-11

비교 연산자

- **실습)**

```
public static void main(String[]  args) {
  int num1=10, num2=2, num3=10;

  System.out.println(num1++  == num3);
  System.out.println(num1  == num3);
  System.out.println(num1  >= num3);
  System.out.println(num1  > num2);
  System.out.println(num3  != num2);
}
```

3-12

논리 연산자

- 연산 대상이 boolean 형으로 주로 다중 조건 체크할때 사용
- 연산 결과는 boolean 형으로 리턴

연산자	구문	내용	공통
&	num1 & num2	모두 true 일때만 true	num1, num2 모두 boolean 값
|	num1 | num2	하나라도 true이면 true	
&&	num1 && num2	num1이 false이면 num2 처리 안됨→무조건 false	num1, num2 모두 연산식일 수도 있고 boolean 값일수도 있다.
||	num1 || num2	num1이 true이면 num2 처리 안됨→무조건 true	
!	!num1	num1이 true→false, false→true로 값이 변경	

3-13

논리 연산자

- **실습)**

```
public static void main(String[]  args) {
  int num1=10, num2=2, num3=10;

  System.out.println(++num1==10  && num3++>=10 );
  System.out.println(num1);
  System.out.println(num3);
  System.out.println(++num1==10  || num3++>=10 );
  System.out.println(num1);
  System.out.println(num3);

  System.out.println(true  & false);
  System.out.println(!(num2==2));
}
```

3-14

비트 연산자

- 연산 대상의 값을 비트로 풀여서 연산한다.
- 자바에서는 자주 사용되지 않는 연산자이다.

연산자	구문	내용
&	num1 & num2	비트 단위 논리곱(and)
|	num1 | num2	비트 단위 논라합(or)
^	num1 ^ num2	비트 단위 배타 논리합(Exclusive or)
~	~num1	비트 단위 보수

3-15

비트 연산자

- **실습)**

```
public static void main(String[]  args) {
  byte num1=10, num2=2;

  System.out.println(num1  & num2);
  System.out.println(num1  | num2);
  System.out.println(num1  ^ num2);
  System.out.println(~num1  );
}
```

3-16

쉬프트 연산자

- 연산 대상의 값을 비트로 풀여서 좌, 우로 이동하여 연산한다.
- 자바에서는 자주 사용되지 않는 연산자이다.

연산자	구문	내용
>>	num1 >> num2	num1을 비트로 풀고 num2만큼 오른쪽으로 이동 왼쪽에 공간이 생기면 부호 비트가 채워진다.
<<	num1 << num2	num1을 비트로 풀고 num2만큼 왼쪽으로 이동 오른쪽에 공간이 생기면 0으로 비트가 채워진다.
>>>	num1 >>> num2	num1을 비트로 풀고 num2만큼 오른쪽으로 이동 왼쪽에 공간이 생기면 무조건 0으로 채워진다.

3-17

쉬프트 연산자

- **실습)**

```
public static void main(String[]  args) {
  byte num1=10, num2=2;

  System.out.println(num1  >> num2);
  System.out.println(num1  << num2);
  System.out.println(num1  >>> num2);
  System.out.println(-127  >>> 2 );
  System.out.println(-127  >> 2 );
}
```

3-18

대입 연산자

- =를 이용해서 연산후 왼쪽 변수에 값을 다시 저장하는 연산자
- 수식과 대입 연산을 같이 사용하여 수식을 줄일 수 있다.

연산자	구문	내용
=	num1 = num2	대입 연산자
+=	num1 += num2	num1 = num1 + num2
-=	num1 -= num2	num1 = num1 - num2
/=	num1 /= num2	num1 = num1 / num2
*=	num1 *= num2	num1 = num1 * num2
%=	num1 %= num2	num1 = num1 % num2
&=	num1 += num2	num1 = num1 & num2

- |=, ^=, <<=, <<<=, >>=

3-19

삼항 연산자

- **? : 형태로 제공되는 삼항 연산자는 if/else 구분을 줄여서 프로그램 할 수 있다.**

조건식 ? 값1 : 값2

```
int result = num1 >=0 ? 0 : 1 ;
```

```
int result;
if(num1>=0)
   result=0;
else
   result=1;
```

3-20

LAB

3-21