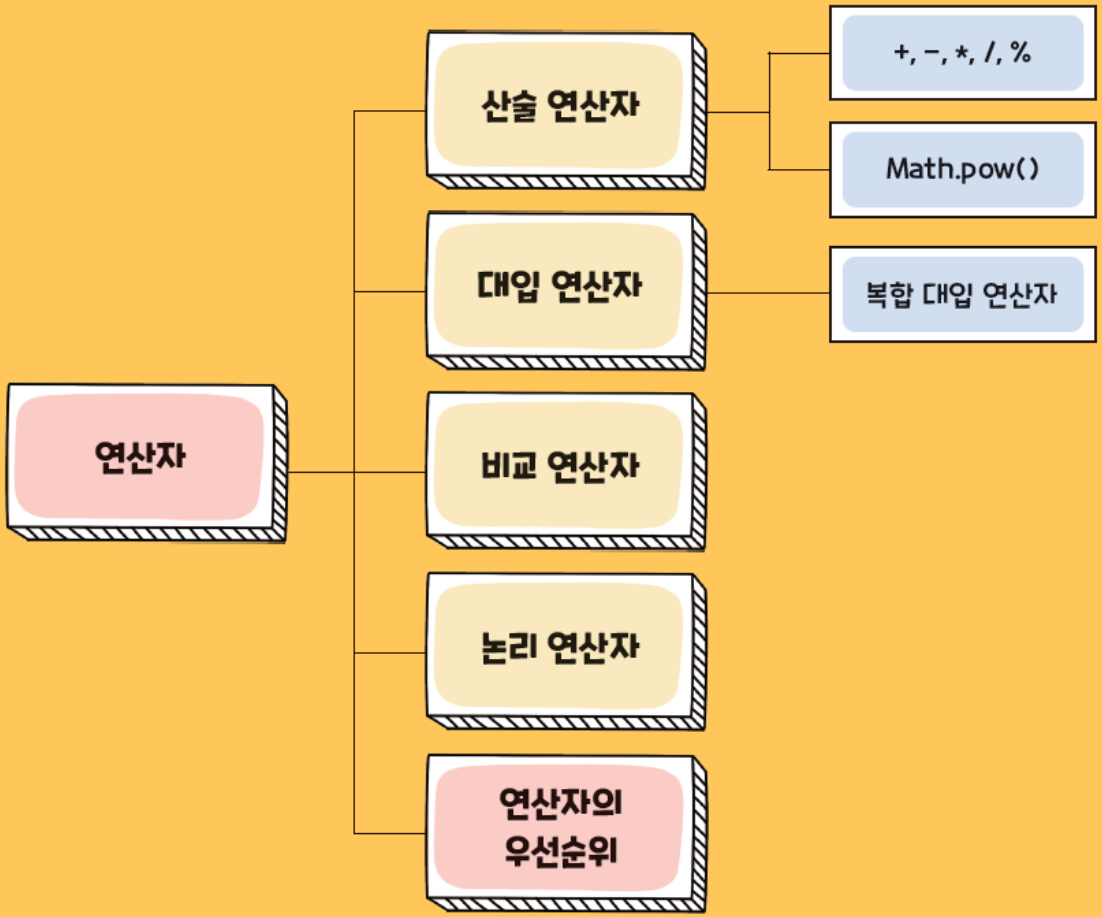
bantaehak

Chapter 03
연산자

목차

Preview



학습목표

- 연산자의 개념을 이해하고 그 종류를 파악합니다.
- 산술 연산자, 대입 연산자, 비교 연산자, 논리 연산자에 대해 알아봅니다.
- 연산자의 우선순위를 이해하고 실습합니다

Section 01
산술 연산자

1. 기본 산술 연산자

- 산술 연산자

표 3-1 기본적인 산술 연산자

산술 연산자		사용 예	설명
+	더하기	a = 5+3	5와 3을 더한 값을 a에 대입
−	빼기	a = 5−3	5에서 3을 뺀 값을 a에 대입
*	곱하기	a = 5*3	5와 3을 곱한 값을 a에 대입
/	나누기	a = 5/3	5를 3으로 나눈 값을 a에 대입
%	나머지	a = 5%3	5를 3으로 나누고 나머지를 a에 대입

1. 기본 산술 연산자

산술 연산자 예제1

코드 3-1 Code03_01.java

```
01  public class Code03_01 {
02      public static void main(String[] args) {
03          int n1, n2, res;
04          n1 = 5;
05          n2 = 3;
06          res = n1 + n2;
07          System.out.println(res);
08      }
09  }
```

```
8
```

1. 기본 산술 연산자

- 산술 연산자 예제2

코드 3-2 Code03_02.java

```
01  public class Code03_02 {
02      public static void main(String[] args) {
03          int n1, n2, res;
04          n1 = 5;
05          n2 = 3;
06          res = n1 + n2;
07          System.out.println(res);
08          res = n1 - n2;
09          System.out.println(res);
10          res = n1 * n2;
11          System.out.println(res);
12          res = n1 / n2;
13          System.out.println(res);
14          res = n1 % n2;
15          System.out.println(res);
16      }
17  }
```

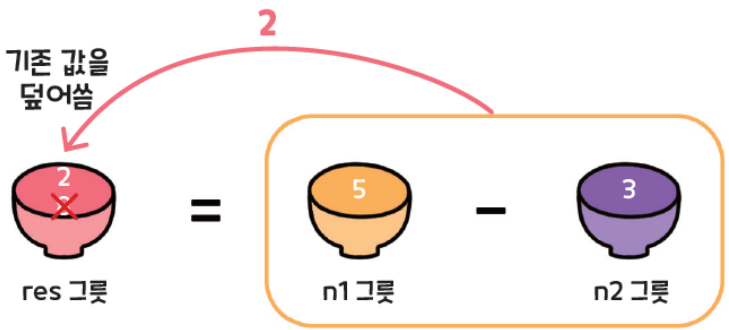


그림 3-1 연산 결과를 덮어씀

```
8
2
15
1
2
```

1. 기본 산술 연산자

■ 산순 연산자의 우선순위

- 하나의 연산에 여러 개의 연산자가 있는 경우

  → 연산자의 우선순위가 정해져 있어야 정확한 계산 결과를 산출할 수 있음

- 우선순위 예제1 – 연산자의 우선순위가 동일한 경우

코드 3-3 Code03_03.java

```
01  public class Code03_03 {
02      public static void main(String[] args) {
03          int a = 3, b = 4, c = 5;
04          System.out.println(a + b - c);
05          System.out.println(a - c + b);
06          System.out.println(-c + a + b);
07      }
08  }
```

```
2
2
2
```

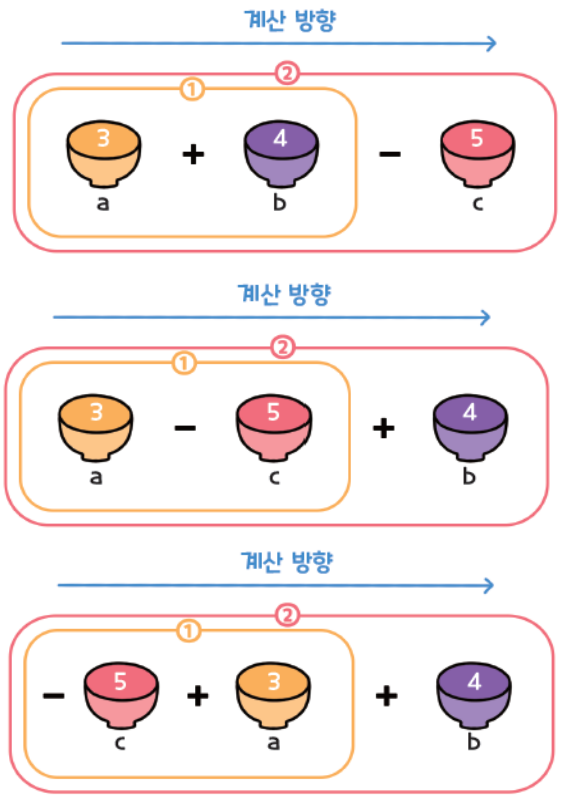


그림 3-2 더하기, 빼기가 여러 개일 때의 계산 순서

1. 기본 산술 연산자

- 산순 연산자의 우선순위
  - 우선순위 예제2 – 연산자의 우선순위가 동일한 경우

코드 3-4

Code03_04.java

```
01   public class Code03_04 {
02       public static void main(String[] args) {
03           double a = 2, b = 4, c = 6;
04           System.out.println(a / b * c);
05           System.out.println(a * c / b);
06           System.out.println(c / b * a);
07       }
08   }
```

```
3.0
3.0
3.0
```

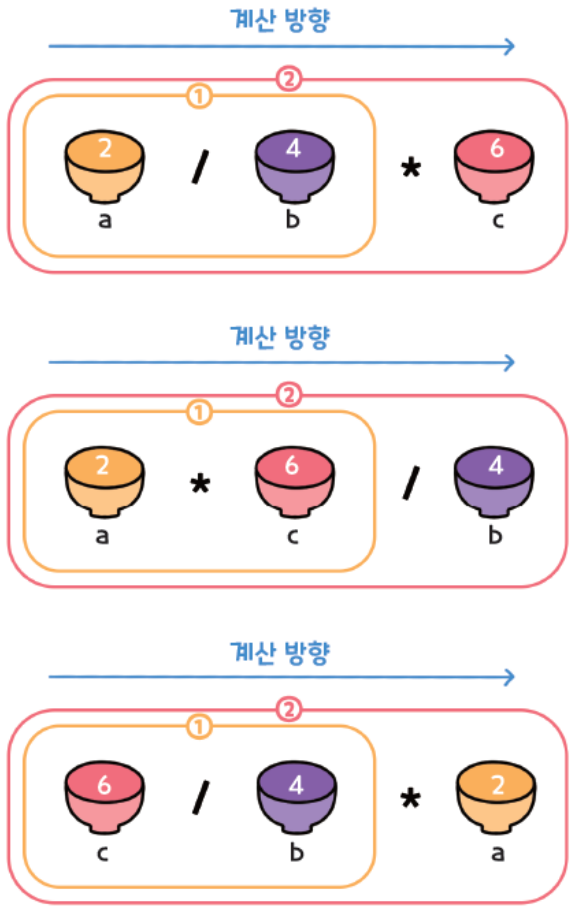


그림 3-3 곱하기, 나누기가 여러 개일 때의 계산 순서

1. 기본 산술 연산자

■ 산순 연산자의 우선순위

▪ 우선순위 예제3 – 연산자의 우선순위가 다른 경우

코드 3-5 Code03_05.java

```
01  public class Code03_05 {
02      public static void main(String[] args) {
03          int a = 3, b = 4, c = 5;
04          System.out.println(a * b + c);
05          System.out.println(c + a * b);
06      }
07  }
```

```
17
17
```

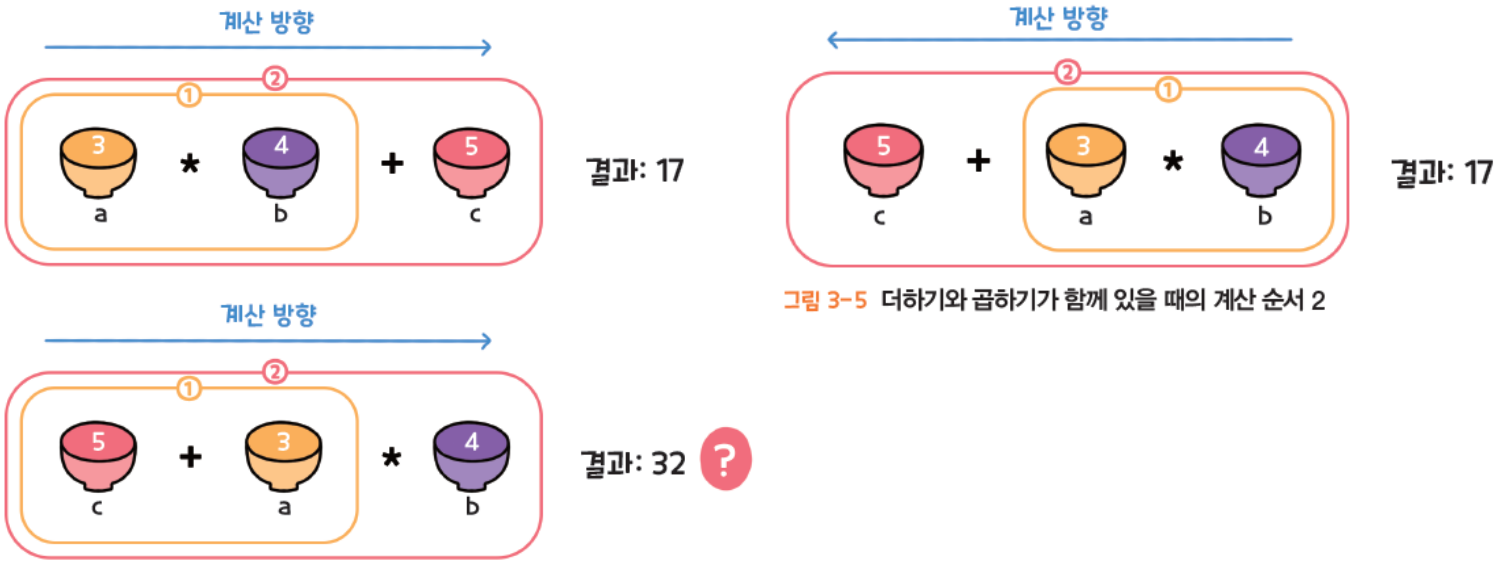


그림 3-5 더하기와 곱하기가 함께 있을 때의 계산 순서 2

그림 3-4 더하기와 곱하기가 함께 있을 때의 계산 순서 1

2. 제곱 연산

- Math.pow(밑수, 지수)
  - 제곱 연산 예제 – 3^2, 4^3

코드 3-6 Code03_06.java

```
01  public class Code03_06 {
02      public static void main(String[] args) {
03          double num;
04          num = Math.pow(3, 2);
05          System.out.println(num);
06          num = Math.pow(4, 3);
07          System.out.println(num);
08      }
09  }
```

```
9.0
64.0
```

2. 제곱 연산

[하나 더 알기] Math 클래스

자바 프로그래밍을 하다 보면 수학과 관련된 계산을 해야 하는 경우가 종종 있습니다. 간단한 사칙연산은 지금까지 배운 내용으로 쉽게 해결할 수 있지만 제곱, 제곱근, 최댓값, 최솟값, 삼각함수 등의 복잡한 계산은 코드로 풀어내기가 어렵습니다. 그래서 자바는 다음과 같이 다양한 수학 계산 메서드가 포함된 Math 클래스를 제공하고 있습니다.

- Math.abs(): 절댓값
- Math.log(): 로그값
- Math.max(): 최댓값
- Math.min(): 최솟값
- Math.sin(), Math.cos(), Math.tan(): 삼각함수

2. 제곱 연산

확인문제

다음 빈칸에 알맞은 말을 넣으시오.

- 자바의 다섯 가지 사칙연산자 기호는 ㉠ 이다.
- 더하기와 곱하기 중에서 먼저 연산되는 것은 ㉡ 이다.

정답

Click!

[LAB] 덤벨의 파운드를 킬로그램으로 환산하기

한빛 헬스장에는 미국에서 수입한 덤벨만 있어 무게가 킬로그램(kg)이 아닌 파운드(lb)로 표시되어 있습니다.

우리는 파운드보다 킬로그램에 익숙하니 덤벨의 무게를 쉽게 파악할 수 있도록 파운드와 킬로그램을 상호 변환하는 프로그램을 만들어봅시다.

[조건]

- 1파운드 = 0.453592킬로그램
- 1킬로그램 = 2.204623파운드

실행 결과

```
파운드(lb)를 입력하세요: 15 ●——— 사용자가 입력
15.0파운드(lb)는 6.8038799999999995킬로그램(kg)입니다.
킬로그램(kg)을 입력하세요: 15 ●——— 사용자가 입력
15.0킬로그램(kg)은 33.069345000000006파운드(lb)입니다.
```

[LAB] 덤벨의 파운드를 킬로그램으로 환산하기

1. Chapter03 프로젝트에 Lab03_01.java 파일을 만들고, 키보드로 값을 입력받기 위해 Scanner 클래스를 사용할 수 있도록 준비하기

```
import java.util.Scanner; // public class 밖에 입력
Scanner s = new Scanner(System.in);
                          // main() 메서드 내 첫 행에 입력
s.close();                // main() 메서드 내 마지막 행에 입력
```

2. double형 변수를 준비하고, 파운드 값을 입력받아 킬로그램 값으로 변환하기

```
double pound, kg;
System.out.print("파운드(lb)를 입력하세요: ");
pound = s.nextDouble();
kg = pound * 0.453592;
System.out.println(pound + "파운드(lb)는 " + kg + "킬로그램(kg)입니다.");
```

3. 킬로그램 값을 입력받아 파운드 값으로 변환하기

```
System.out.print("킬로그램(kg)을 입력하세요: ");
kg = s.nextDouble();
pound = kg * 2.204623;
System.out.println(kg + "킬로그램(kg)은 " + pound + "파운드(lb)입니다.");
```

Section 02
대입 연산자

1. 대입 연산자의 개념

- 대입 연산자(=)
  - 오른쪽의 값이나 계산 결과를 왼쪽의 변수에 대입함
  - =의 왼쪽에는 변수가 한 개여야 함

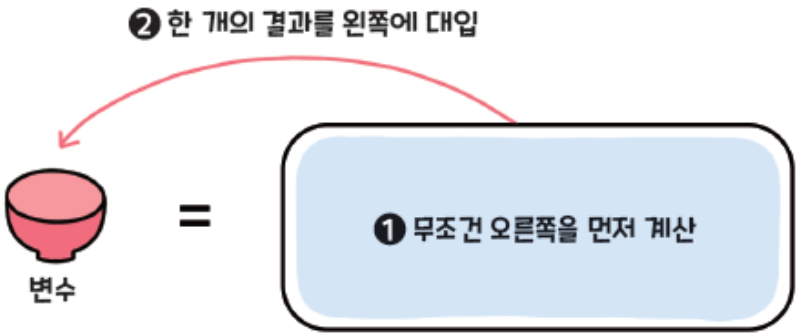


그림 3-6 한 개의 대입 연산자

  - 대입 연산자 예제

코드 3-7 Code03_07.java

```
01   public class Code03_07 {
02       public static void main(String[] args) {
03           int num;
04           num = 100;
05           num = 100 * 200;
06           num = Integer.parseInt("100") + Integer.parseInt("200");
07       }
08   }
```

1. 대입 연산자의 활용

복합 대입 연산자

- 변수의 값을 변경한 후 이를 자신에게 대입함

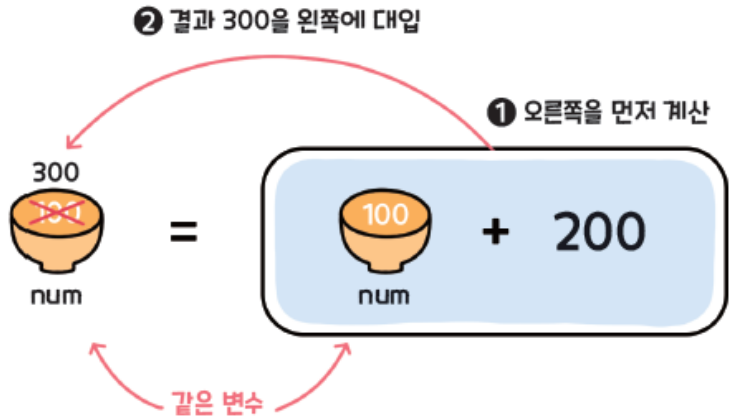


그림 3-7 자신과 연산한 후 자신에게 대입

- 복합 대입 연산자 예제

코드 3-8 Code03_08.java

```
01   public class Code03_08 {
02       public static void main(String[] args) {
03           int num;
04           num = 100;
05           num = num + 200;
06           System.out.println(num);
07       }
08   }
```

```
300
```

1. 대입 연산자의 활용

복합 대입 연산자

표 3-2 복합 대입 연산자

복합 대입 연산자	사용 예	설명
+=	num += 3	num = num + 3과 동일
-=	num -= 3	num = num - 3과 동일
*=	num *= 3	num = num * 3과 동일
/=	num /= 3	num = num / 3과 동일
%=	num %= 3	num = num % 3과 동일

증감 연산자

- ++: 변수의 값이 1씩 증가함
- --: 변수의 값이 1씩 감소함

표 3-3 증감 연산자

증감 연산자	사용 예	설명
++	num++ ++num	num = num+1, num += 1과 동일
--	num-- --num	num = num-1, num -= 1과 동일

1. 대입 연산자의 활용

복합 대입 연산자

복합 대입 연산자 예제

코드 3-9 Code03_09.java

```
01  public class Code03_09 {
02      public static void main(String[] args) {
03          int num = 20;
04          num++ ; System.out.print(num+" ");
05          num-- ; System.out.print(num+" ");
06          num += 3 ; System.out.print(num+" ");
07          num -= 3 ; System.out.print(num+" ");
08          num *= 3 ; System.out.print(num+" ");
09          num /= 3 ; System.out.print(num+" ");
10          num %= 3 ; System.out.print(num+" ");
11      }
12  }
```

```
21 20 23 20 60 20 2
```

1. 대입 연산자의 활용

확인문제

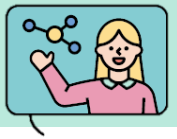
다음 빈칸에 알맞은 말을 넣으시오.

- 대입 연산자의 기호는 [㉠]이며, 이는 오른쪽의 값이나 계산 결과를 왼쪽의 변수에 대입하라는 의미이다.
- a = a + 1을 줄여서 a [㉡] 1 또는 a [㉢]로 나타낼 수 있다.

정답

Click!

[LAB] 편의점의 일일 매출 계산하기



편의점은 본사로부터 구입한 물품을 손님에게 판매하는데, 이때 본사로부터 구입한 물품의 가격에 이윤을 붙여서 판매 가격을 책정합니다.

편의점에서 취급하는 물품의 구입 가격과 판매 가격이 다음과 같을 때, 오늘의 총매출액을 계산하는 프로그램을 만들어봅시다.

실행 결과

```
오늘 총매출액은 11600원입니다.
```

구분	캔 커피	삼각 김밥	바나나맛 우유	도시락	콜라	새우깡
구입 가격	500	900	800	3500	700	1000
판매 가격	1800	1400	1800	4000	1500	2000

[구매 및 판매 내역]

- 삼각 김밥(900원) 10개 구입
- 바나나맛 우유(1800원) 2개 판매
- 도시락(3500원) 5개 구입
- 도시락(4000원) 4개 판매
- 콜라(1500원) 1개 판매
- 새우깡(2000원) 4개 판매
- 캔커피(1800원) 5개 판매

[LAB] 편의점의 일일 매출 계산하기

1. Lab03_02.java 파일을 만들고, main() 메서드 내부에서 총매출액 변수 total을 0으로 초기화하기

```
int total = 0;
```

2. 구입한 물품의 경우 구입 가격에 개수를 곱한 후 총매출액에서 빼기

```
total -= 900*10;
total -= 3500*5;
```

3. 판매한 물품의 경우 판매 가격에 개수를 곱한 후 총매출액에 더하기

```
total += 1800*2;
total += 4000*4;
total += 1500;
total += 2000*4;
total += 1800*5;
```

4. 총매출액을 출력하기

```
System.out.println("오늘 총매출액은 " + total + "원입니다.");
```

Section 03
비교 연산자와
논리 연산자

1. 비교 연산자의 개념

■ 비교 연산자

- 어떤 것이 큰지, 작은지, 같은지를 비교하는 연산자로, 관계 연산자라고도 부름
- 비교 연산자의 결과: yes 또는 true(참), no 또는 false(거짓)

(a) 100은 200보다 작다: 참(true)

(b) 100은 200보다 크다: 거짓(false)

그림 3-8 비교 연산자의 기본 개념

- 비교 연산자의 종류

표 3-4 비교 연산자

비교 연산자	의미	설명
==	같다	왼쪽 값과 오른쪽 값이 같으면 참
!=	같지 않다	왼쪽 값과 오른쪽 값이 다르면 참
>	크다	왼쪽 값이 오른쪽 값보다 크면 참
<	작다	왼쪽 값이 오른쪽 값보다 작으면 참
>=	크거나 같다	왼쪽 값이 오른쪽 값보다 크거나 같으면 참
<=	작거나 같다	왼쪽 값이 오른쪽 값보다 작거나 같으면 참

2. 비교 연산자의 활용

■ 비교 연산자 예제

- 운전면허 필기시험 점수를 입력받아 70점 이상이면 합격(true), 그렇지 않으면 불합격(false)으로 처리하는 프로그램

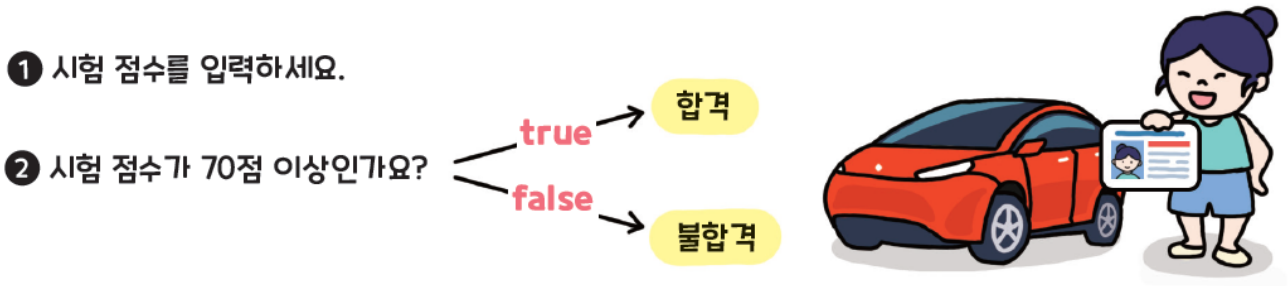


그림 3-9 비교 연산자의 기본 활용

2. 비교 연산자의 활용

비교 연산자 예제1

- 운전면허 필기시험 점수를 입력받아 70점 이상이면 합격(true), 그렇지 않으면 불합격(false)으로 처리하는 프로그램

코드 3-10 Code03_10.java

```
01  import java.util.Scanner;
02  public class Code03_10 {
03      public static void main(String[] args) {
04          Scanner s = new Scanner(System.in);
05          int score;
06
07          System.out.print("필기시험 점수를 입력하세요: ");
08          score = s.nextInt();
09          System.out.println(score >= 70);
10
11          s.close();
12      }
13  }
```

```
필기시험 점수를 입력하세요: 80 ←——— 사용자가 입력
true
```

2. 비교 연산자의 활용

■ 비교 연산자 예제2

코드 3-11 Code03_11.java

```
01  public class Code03_11 {
02      public static void main(String[] args) {
03          int n1 = 100, n2 = 200;
04
05          System.out.print(n1 == n2);    // n1이 n2와 같음
06          System.out.println(n1 != n2);  // n1이 n2와 같지 않음
07
08          System.out.print(n1>n2);
09          System.out.println(n1<n2);
10
11          System.out.print(n1>=n2);
12          System.out.println(n1<=n2);
13      }
14  }
```

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

2. 비교 연산자의 활용

[하나 더 알기] 비교 연산자(==)와 대입 연산자(=)

n1과 n2가 같은지 확인하는 비교 연산자는 ==인데, 수학의 등호(=)와 헷갈려서 =을 한 개만 입력하지 않도록 주의해야 합니다. 예를 들어 다음과 같이 작성하면 엉뚱한 결과를 얻게 됩니다.

```
int n1 = 100, n2 = 200;
System.out.print(n1 = n2);
```

```
200
```

'n1 = n2'는 n2의 값을 n1에 넣으라는 의미이므로 n2의 값인 200이 n1에 대입되어 200이 출력됩니다. 숙련된 프로그래머라도 간혹 이러한 실수를 저지르곤 하니 주의하기 바랍니다.

3. 논리 연산자의 개념

논리 연산자

- 비교 연산자가 여러 번 필요할 때 사용함

표 3-5 논리 연산자

논리 연산자	의미	설명	사용 예
&&	그리고(and)	둘 다 참이어야 참	(num>10) && (num<20)
||	또는(or)	둘 중 하나만 참이어도 참	(num==10) || (num==20)
!	부정(not)	참이면 거짓/거짓이면 참	!(num<100)

&&는 and로, ||는 or로, !는 not으로 읽으며, ||는 Enter 바로 위에 있는 키를 Shift를 누르고 입력하면 됩니다. 참고로 배타적 논리합(exclusive or)인 ^도 논리 연산자이지만 잘 사용되지 않으므로 이 책에서는 다루지 않았습니다.

3. 논리 연산자의 개념

논리 연산자

- num의 값이 10과 20 사이인 경우
- 조건1) num은 10보다 커야 함
- 조건2) 그리고 num은 20보다 작아야 함

```
(num > 10) && (num < 20)
```

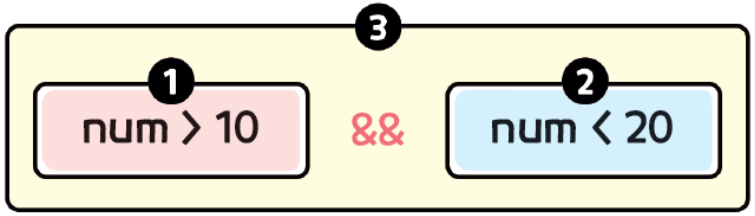


그림 3-10 && 연산

앞의 코드에서 괄호를 빼도 연산자의 우선순위에 의해 'num > 10'과 'num < 20'이 먼저 처리되고 && 가 마지막에 처리됩니다. 하지만 처리 순서가 더 명확하게 드러나도록 괄호를 사용하는 것이 좋습니다.

3. 논리 연산자의 개념

논리 연산자 예제

코드 3-12 Code03_12.java

```
01   public class Code03_12 {
02       public static void main(String[] args) {
03           int num = 99;
04
05           System.out.println( (num > 100) && (num < 200) );
06           System.out.println( (num == 99) || (num == 100) );
07           System.out.println( !(num == 100) );
08       }
09   }
```

```
false
true
true
```

3. 논리 연산자의 개념

확인문제

다음 빈칸에 알맞은 말을 넣으시오.

- '같다'를 나타내는 비교 연산자는 [㉠]이고, '같지 않다'를 나타내는 비교 연산자는 [㉡]이다.
- 둘 다 참이어야 참이 되는 논리 연산자는 [㉢]이고, 둘 중 하나만 참이어도 참이 되는 논리 연산자는 [㉣]이다.

정답

Click!

Section 04
연산자의 우선순위

1. 연산자의 우선순위

■ 연산자 우선순위

- 한 줄에 여러 연산자가 동시에 들어가는 경우 어떤 연산자를 먼저 계산할지 정함

표 3-6 연산자의 우선순위

우선순위	연산자	설명	순위가 같을 때의 진행 방향
1	() [] .	1차 연산자	→
2	+ − ++ −− ~ ! (type)	단항 연산자(변수 또는 상수 앞에 붙음)	←
3	* / %	산술 연산자	→
4	+ −	산술 연산자	→
5	<< >> >>>	비트 시프트 연산자	→
6	< <= > >= instanceof	비교 연산자	→
7	== !=	비교 연산자	→
8	&	비트 연산자	→
9	^	비트 연산자	→
10	|	비트 연산자	→
11	&&	논리 연산자	→
12	||	논리 연산자	→
13	?:	조건 삼항 연산자	→
14	= += −= *= /= %= &= ^= |= <<= >>=	대입 연산자	←

1. 연산자의 우선순위

확인문제

다음 연산자 중에서 우선순위가 가장 높은 것과 가장 낮을 것을 고르시오.

① + ② * ③ && ④ = ⑤ ( )

정답

Click!

[LAB] 기말 평균 학점 구하기

한빛대학교 1학년인 난생이는 다음과 같은 기말고사 성적표를 받았습니다.
연산자의 우선순위에 주의하면서 평균 학점을 구하는 프로그램을 작성해봅시다.

과목(이수 학점)	성적
자바(3학점)	B(3.5)
모바일(2학점)	A0(4.0)
엑셀(1학점)	A(4.5)

실행 결과

```
평균 학점 : 3.83
```

[LAB] 기말 평균 학점 구하기

1. Lab03_03.java 파일을 만들고, 자바, 모바일, 엑셀 변수에 각각의 이수 학점을 넣어 선언하기

```
int java = 3, mobile = 2, excel = 1;
```

2. A, A0, B 학점에 대응하는 점수를 넣어 선언하기

```
double A = 4.5, A0 = 4.0, B = 3.5;
```

3. 각 과목과 해당 점수를 곱하여 더한 다음 총학점으로 나누기
   - 이때 연산자의 우선순위를 고려하여 괄호를 넣기

```
double avg;
avg = ((java*B) + (mobile*A0) + (excel*A))
       / (java + mobile + excel);
```

[LAB] 기말 평균 학점 구하기

4. 평균 학점(avg)을 그대로 출력하면 소수점 아래 숫자가 너무 길어지기 때문에 소수점 아래 두자리까지 출력하기
   - 평균 학점에 100.0을 곱하여 Math.round()로 처리하고 100.0으로 나누기

```
avg = Math.round(avg*100.0) / 100.0;
```

5. 평균 학점을 출력하기

```
System.out.println("평균 학점 : " + avg);
```

[실전예제]
거북이의 펜 바꾸기

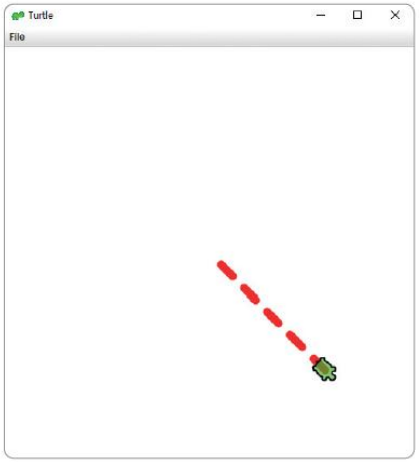
[실전 예제] 거북이의 펜 바꾸기

[문제]

거북이가 선을 그리지 않으면서 이동하게 해봅시다.

turtle.up() 메서드는 거북이가 펜을 드는 메서드로, 선을 그리지 않겠다는 의미입니다.

또한 다시 선을 그리고 싶을 때는 turtle.down() 메서드를 사용하면 됩니다.



[실전 예제] 거북이의 펜 바꾸기

[해결]

**자바에서 터틀 그래픽을 사용하려면 먼저 Turtle.java 파일을 복사하여 프로젝트에 넣어야 합니다.

Ex03_01.java

```
01  public class Ex03_01 {
02      public static void main(String[] args) {
03          Turtle turtle = new Turtle
04
05          turtle.width(5);
06          turtle.penColor("red");
07
08          turtle.right(45);
09
10          turtle.up();
11          turtle.forward(20);
12          turtle.down();
13          turtle.forward(20);
14
15          // 10~13행을 4회 반복
16      }
17  }
```

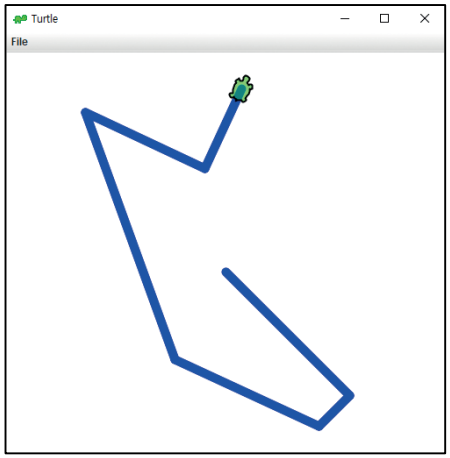
[실전예제]
입력한 값만큼
거북이 움직이기

[실전 예제] 입력한 값만큼 거북이 움직이기

[문제]

사용자가 입력한 거리와 각도만큼 거북이가 이동하되, 이동하는 횟수가 무한 반복되는 프로그램을 작성해봅시다.

```
거북이의 회전 각도: 45
거북이의 이동 거리: 200
~~~생략~~~
거북이의 회전 각도: -90
거북이의 이동 거리: 100
```



[해결]

- 다음은 무한 반복을 위한 코드로, while(true)는 그 아래 행이 무한 반복되게 함

```
while(true) {
    // 이 부분에 무한 반복할 코드 작성
}
```

[실전 예제] 입력한 값만큼 거북이 움직이기

[해결]

Ex03_02.java

```
01  import java.util.Scanner;
02  public class Ex03_02 {
03      public static void main(String[] args) {
04          Scanner s = new Scanner(System.in);
05          Turtle turtle = new Turtle();
06          int angle, distance;
07
08          turtle.width(10);
09          turtle.penColor("blue");
10
11          while(true) {
12              System.out.print("거북이의 회전 각도 : ");
13              angle = s.nextInt();       // 거북이의 회전 각도 입력
14              System.out.print("거북이의 이동 거리 : ");
15              distance = s.nextInt();    // 거북이의 선 길이 입력
16
17              turtle.right(angle);       // 거북이를 오른쪽으로 회전시킴
18              turtle.forward(distance);  // 거북이를 거리만큼 이동함
19          }
20      }
21  }
```

Thank you!