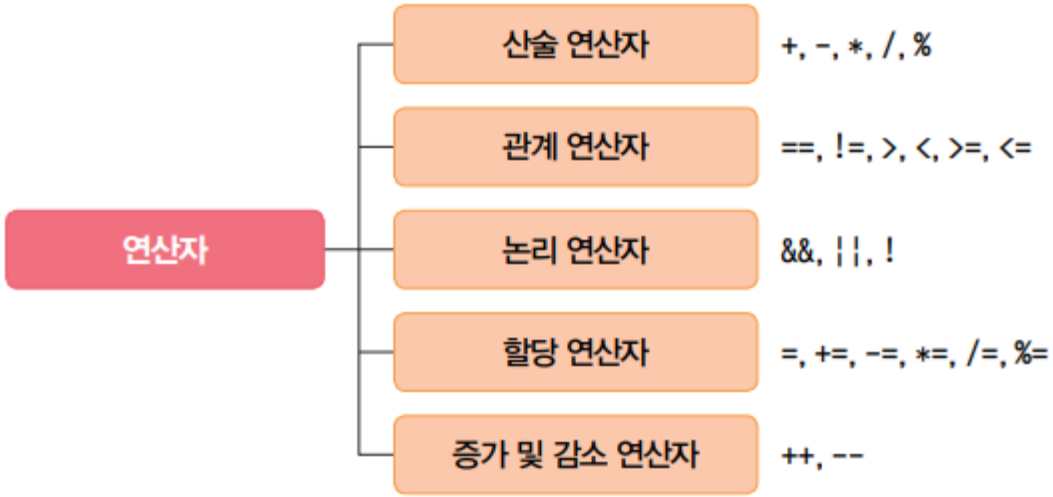
Section 01
연산자

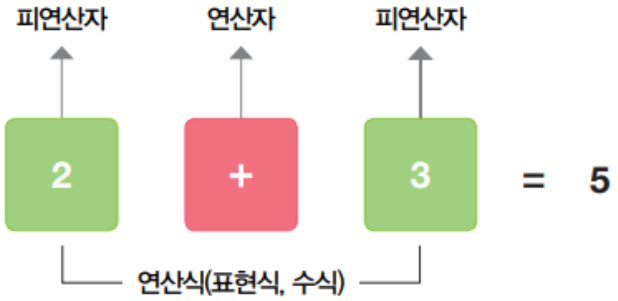
1. 연산자

■ 연산자의 개요

- 연산자는 컴파일러에 어떤 작업을 수행하도록 지시하는 기호



[그림 3-1] 연산자의 유형



[그림 3-2] 연산자와 피연산자

1. 연산자

■ 산술 연산자

- 수학 표현식의 덧셈, 뺄셈 등과 같은 산술 연산을 수행하는 데 사용함

[표 3-1] 산술 연산자의 종류

연산자	설명	예
+	왼쪽 피연산자와 오른쪽 피연산자를 더한다.	3 + 2 = 5
-	왼쪽 피연산자에서 오른쪽 피연산자를 뺀다.	3 - 2 = 1
*	왼쪽 피연산자와 오른쪽 피연산자를 곱한다.	3 * 2 = 6
/	왼쪽 피연산자를 오른쪽 피연산자로 나누고 몫을 반환한다.	3 / 2 = 1
%	왼쪽 피연산자를 오른쪽 피연산자로 나누고 나머지 값을 반환한다.	3 % 2 = 1

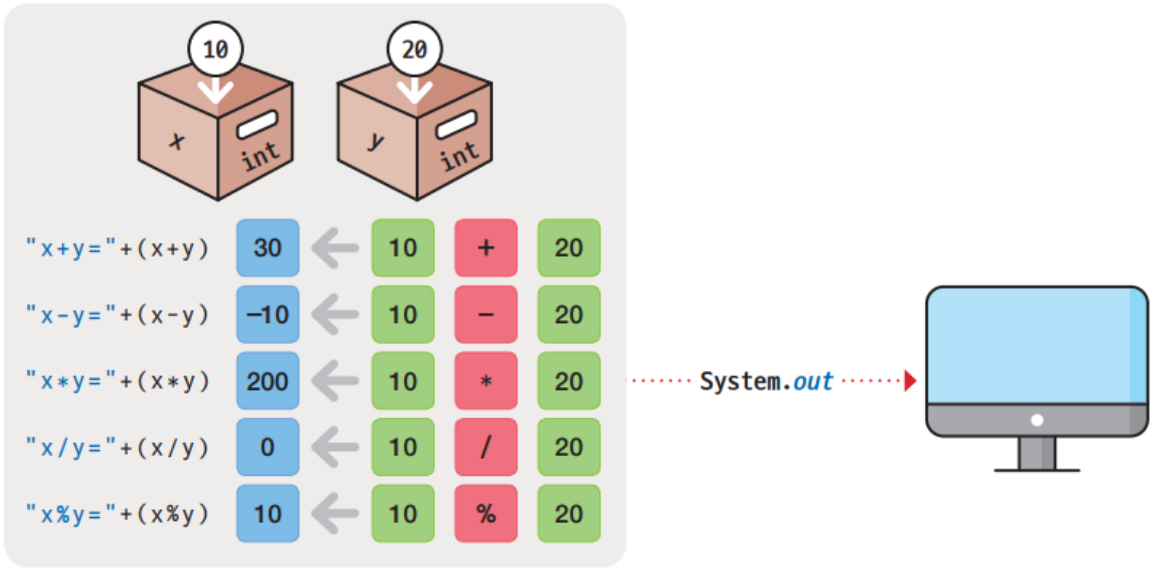
1. 연산자

산술 연산자 사용 예시

```
public class Example01 {
   public static void main(String[] args) {
      int x = 10, y = 20;
      System.out.println("x + y = " + (x + y));
      System.out.println("x - y = " + (x - y));
      System.out.println("x * y = " + (x * y));
      System.out.println("x / y = " + (x / y));
     System.out.println("x % y = " + (x % y));
   }
}
```

실행 결과

```
x + y = 30
x - y = -10
x * y = 200
x / y = 0
x % y = 10
```



[그림 3-3] 산술 연산자의 처리 과정

1. 연산자

예제 3-1 산술 연산자를 이용하여 정수와 실수 연산하기

```
01 public class Operator01 {
02    public static void main(String[] args) {
03       int a = 10;
04       double b = 2.0;
05       System.out.println("a + b = " + (a + b));
06       System.out.println("a - b = " + (a - b));
07       System.out.println("a * b = " + (a * b));
08       System.out.println("a / b = " + (a / b));
09       System.out.println("a % b = " + (a % b));
10    }
11 }
```

실행 결과

```
a + b = 12.0
a - b = 8.0
a * b = 20.0
a / b = 5.0
a % b = 0.0
```

1. 연산자

■ 관계 연산자

- 두 피연산자의 크고 작음을 비교하기 위해 사용힘
- 두 피연산자의 관계가 참이면 true를 반환하고 거짓이면 false를 반환함
- 일반적으로 관계 연산자는 if문, while문 등에서 조건식을 판별하는 데 사용함

[표 3-2] 관계 연산자의 종류

연산자	설명	예
==	왼쪽 피연산자와 오른쪽 피연산자가 같다.	3 == 2 ⟶ false
!=	왼쪽 피연산자와 오른쪽 피연산자가 같지 않다.	3 != 2 ⟶ true
>	왼쪽 피연산자가 오른쪽 피연산자보다 크다.	3 > 2 ⟶ true
<	왼쪽 피연산자가 오른쪽 피연산자보다 작다.	3 < 2 ⟶ false
>=	왼쪽 피연산자가 오른쪽 피연산자보다 크거나 같다.	3 >= 2 ⟶ true
<=	왼쪽 피연산자가 오른쪽 피연산자보다 작거나 같다.	3 <= 2 ⟶ false

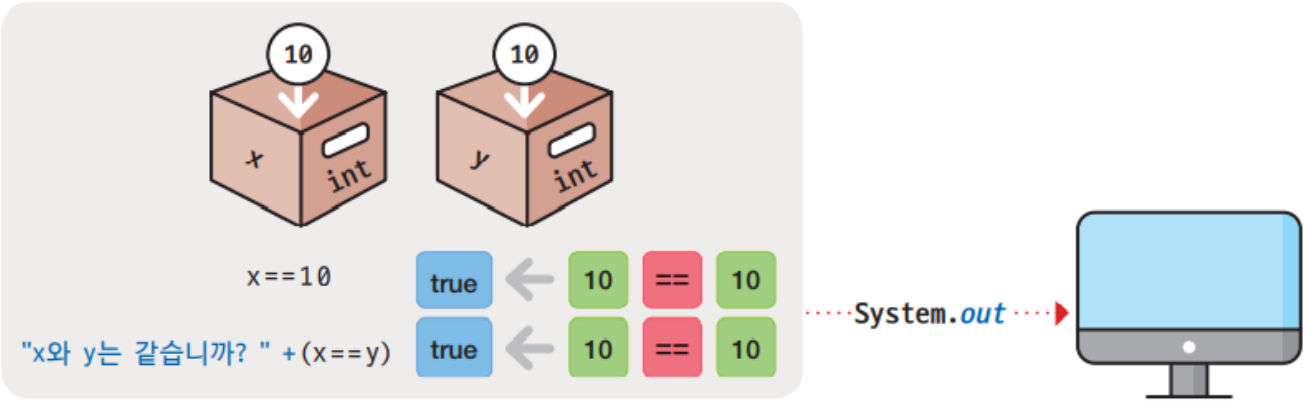
1. 연산자

관계 연산자 사용 예시

```
public class Example02 {
   public static void main(String[] args) {
      int x = 10, y = 10;
      System.out.println(x == 10);
      System.out.println("x와 y는 같습니까? " + (x == y));
   }
}
```

실행 결과

```
true
x와 y는 같습니까? true
```



[그림 3-4] 관계 연산자의 처리 과정

1. 연산자

예제 3-2 관계 연산자를 이용하여 두 정수 비교하기

```
01 public class Operator02 {
02    public static void main(String[] args) {
03       int a = 10, b = 25;
04       System.out.println("a == b = " + (a == b));
05       System.out.println("a != b = " + (a != b));
06       System.out.println("a > b = " + (a > b));
07       System.out.println("a < b = " + (a < b));
08       System.out.println("b >= a = " + (b >= a));
09       System.out.println("b <= a = " + (b <= a));
10    }
11 }
```

실행 결과

```
a == b = false
a != b = true
a > b = false
a < b = true
b >= a = true
b <= a = false
```

1. 연산자

■ 논리 연산자

- 표현식이 참인지 거짓인지 확인하는 데 사용함
- 일반적으로 논리 연산자는 if문, while문 등에서 조건식을 판별하는 데 사용함

[표 3-3] 논리 연산자의 종류

연산자	설명	예
&&	두 피연산자의 조건이 모두 참이면 true를 반환하고, 그렇지 않으면 false를 반환한다(논리적 AND).	a && b
||	두 피연산자의 조건 중 하나라도 참이면 true를 반환하고, 그렇지 않으면 false를 반환한다(논리적 OR).	a || b
!	조건을 부정한다. 즉 피연산자의 조건이 참이면 false를 반환하고, 피연산자의 조건이 거짓이면 true를 반환한다(논리적 NOT).	!a

[표 3-4] 논리 연산자의 실행값

a	b	a && b	a || b	!a
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

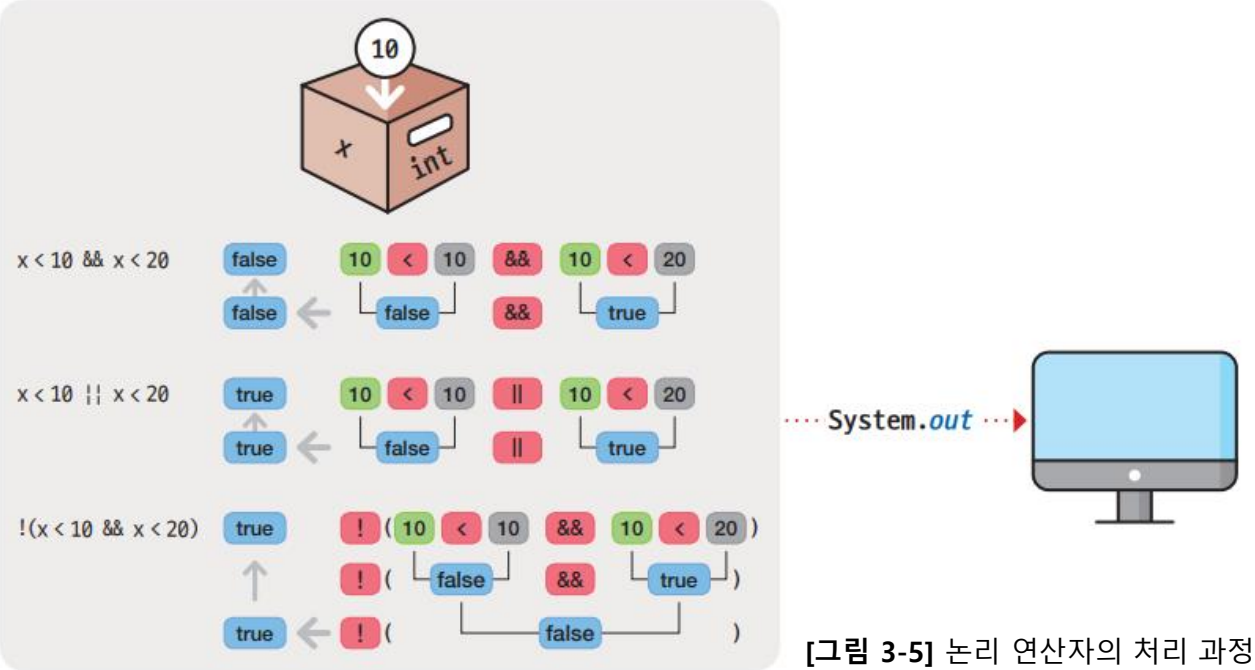
1. 연산자

논리 연산자 사용 예시

```
public class Example03 {
   public static void main(String[] args) {
      int x = 10;
      System.out.println(x < 10 && x < 20);
      System.out.println(x < 10 || x < 20);
      System.out.println(!(x < 10 && x < 20));
   }
}
```

실행 결과

```
false
true
true
```



[그림 3-5] 논리 연산자의 처리 과정

1. 연산자

예제 3-3 논리 연산자를 이용하여 정수 연산하기

```
01 public class Operator03 {
02    public static void main(String[] args) {
03       int a = 5, b = 10;
04
05       System.out.println(a > b);
06       System.out.println(!(a > b));
07       System.out.println((a == b) && (a == 5));
08       System.out.println((a != b) && (a == 5));
09       System.out.println(!((a != b) && (a == 5)));
10    }
11 }
```

실행 결과

```
false
true
false
true
false
```

1. 연산자

■ 대입 연산자

- 변수에 값을 할당하는 데 사용함
- 왼쪽 피연산자가 변수이고 오른쪽 피연산자가 값

→오른쪽에 있는 값은 왼쪽에 있는 피연산자의 자료형과 같아야 함

→그렇지 않으면 컴파일러에서 오류가 발생함

[표 3-5] 할당 연산자의 종류

연산자	설명	예
=	왼쪽 피연산자에 오른쪽 피연산자의 값을 할당한다.	`c = a + b`
+=	왼쪽 피연산자와 오른쪽 피연산자를 더한 값을 왼쪽 피연산자에 할당한다.	`b += a` `b = b + a`
-=	왼쪽 피연산자에서 오른쪽 피연산자를 뺀 값을 왼쪽 피연산자에 할당한다.	`b -= a` `b = b - a`
*=	왼쪽 피연산자와 오른쪽 피연산자를 곱한 값을 왼쪽 피연산자에 할당한다.	`b *= a` `b = b * a`
/=	왼쪽 피연산자를 오른쪽 피연산자로 나누고 몫을 왼쪽 피연산자에 할당한다.	`b /= a` `b = b / a`
%=	왼쪽 피연산자를 오른쪽 피연산자로 나누고 나머지 값을 왼쪽 피연산자에 할당한다.	`b %= a` `b = b % a`

1. 연산자

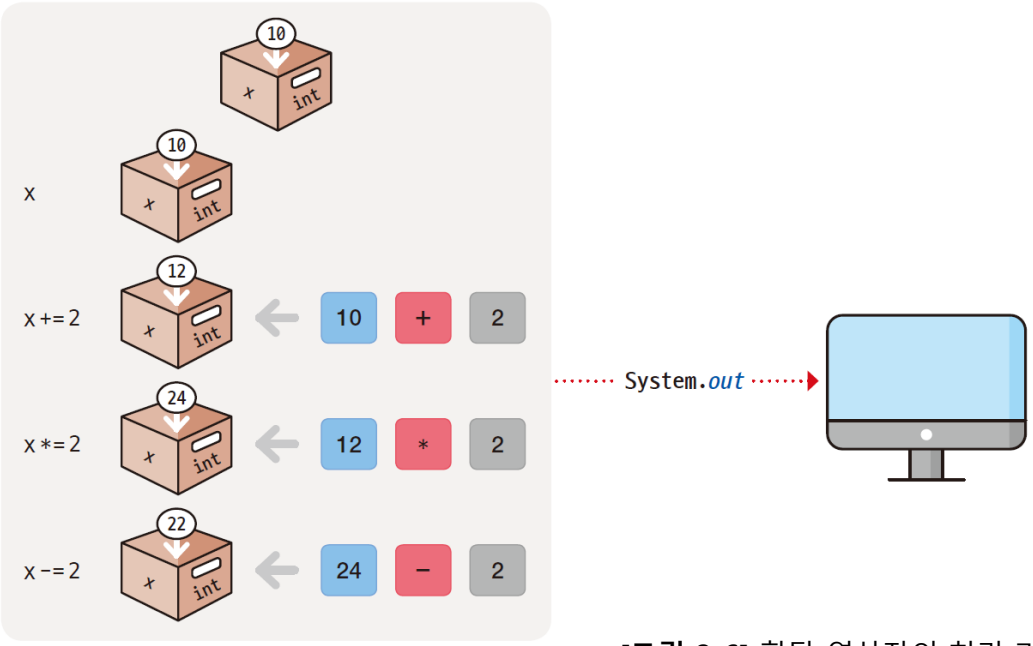
대입 연산자 사용 예시

```
public class Example04 {
   public static void main(String[] args) {
      int x = 10;
      System.out.println(x);
      x += 2;
      System.out.println(x);
      x *= 2;
      System.out.println(x);
      x -= 2;
      System.out.println(x);
   }
}
```

실행 결과

```
10
12
24
22
```

x 값의 변화



[그림 3-6] 할당 연산자의 처리 과정

1. 연산자

예제 3-4 대입 연산자를 이용하여 정수 연산하기

```
01 public class Operator04 {
02    public static void main(String[] args) {
03       int a = 10;
04       int b = 20;
05       int c;
06       System.out.println(c = a);
07       System.out.println(b += a);
08       System.out.println(b -= a);
09       System.out.println(b *= a);
10       System.out.println(b /= a);
11       System.out.println(b %= a);
12    }
13 }
```

실행 결과

```
10
30
20
200
20
0
```

1. 연산자

■ 증가 및 감소 연산자

- ++는 피연산자에 1을 더하고 --는 피연산자에서 1을 뺌
- 증가 및 감소 연산자가 변수의 앞에 위치하느냐 뒤에 위치하느냐에 따라서 값이 달라지기 때문에 주의해야 함

→[예] a가 5일 때

✓a++, ++a : 6

✓a--, --a 값은 4

[표 3-6] 증가 및 감소 연산자의 종류

연산자	설명	예
++	피연산자를 1 증가시킨다.	a++ ++a
--	피연산자를 1 감소시킨다.	a-- --a

1. 연산자

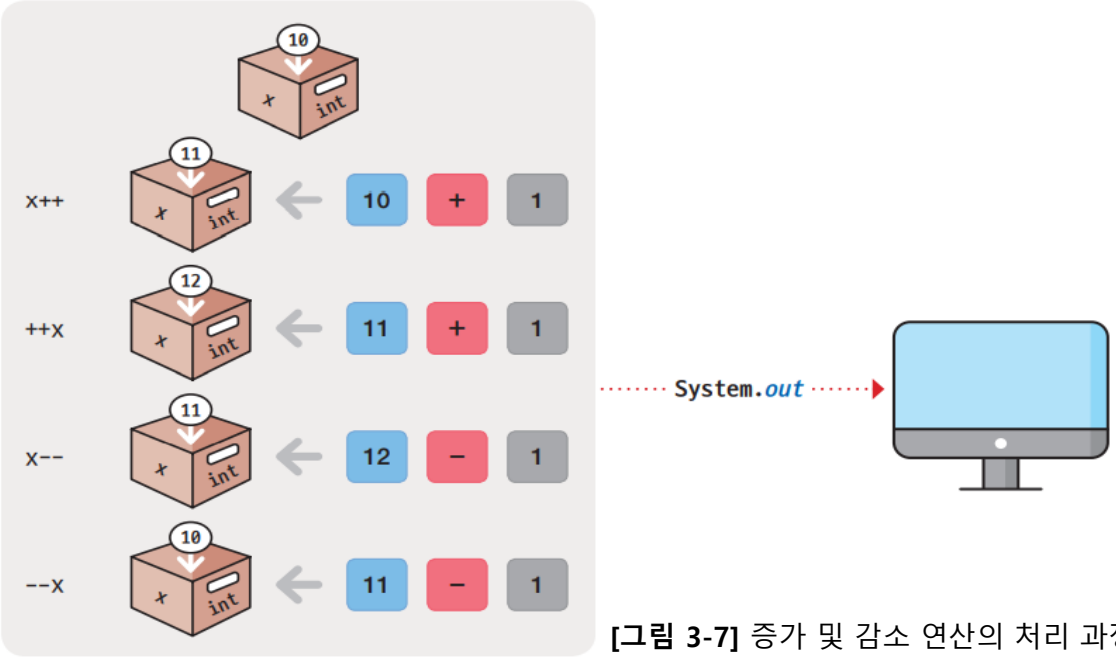
증가 및 감소 연산자 사용 예시

```
public class Example05 {
   public static void main(String[] args) {
      int x = 10;
      System.out.println(x++);
      System.out.println(++x);
      System.out.println(x--);
      System.out.println(--x);
   }
}
```

실행 결과

```
10
12
12
10
```

x 값의 변화



[그림 3-7] 증가 및 감소 연산의 처리 과정

1. 연산자

예제 3-5 증가 및 감소 연산자를 이용하여 정수 연산하기

```
01 public class Operator05 {
02    public static void main(String[] args) {
03       int a = 10, b = 10, c = 10, d = 10;
04
05       System.out.println("a++ => " + (a++));
06       System.out.println("a => " + a);
07       System.out.println("++b => " + (++b));
08       System.out.println("c-- => " + (c--));
09       System.out.println("--d => " + (--d));
10    }
11 }
```

실행 결과

```
a++ => 10
a => 11
++b => 11
c-- => 10
--d => 9
```

1. 연산자

■ 연산자의 우선순위

[표 3-6] 연산자의 우선순위

연산자	설명	우선순위
++, --, !	오른쪽에서 왼쪽으로(←)	높다
*, /, %	왼쪽에서 오른쪽으로(→)	
+, -	왼쪽에서 오른쪽으로(→)	
>, >=, <, <=	왼쪽에서 오른쪽으로(→)	
==, !=	왼쪽에서 오른쪽으로(→)	
&&	왼쪽에서 오른쪽으로(→)	
||	왼쪽에서 오른쪽으로(→)	
=, +=, -=, *=, /=, %=	오른쪽에서 왼쪽으로(←)	낮다

Section 02

형 변환

2. 형 변환

■ 형 변환(type conversion)

- 어떤 자료형에서 다른 자료형으로 변수를 변환하는 것

[그림 3-8] 형 변환의 유형

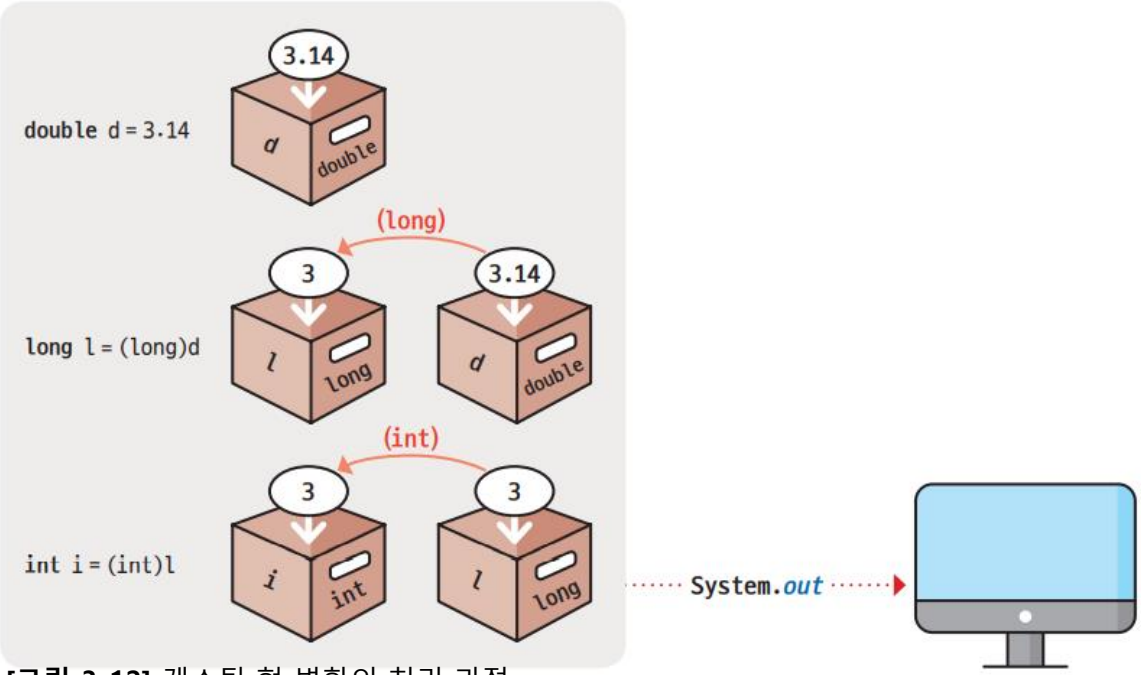
2. 형 변환

캐스팅 형 변환 사용 예시

```
public class Example07 {
   public static void main(String[] args) {
       double d = 3.14;
       long l = (long)d;
       int i = (int)l;
       System.out.println(d);
       System.out.println(l);
      System.out.println(i);
    }
}
```

실행 결과

```
3.14
3
3
```



[그림 3-12] 캐스팅 형 변환의 처리 과정

2. 형 변환

예제 3-7 캐스팅 형 변환

```
01 public class TypeConversion02 {
02    public static void main(String[] args) {
03       int x = 10;
04       double y = 3.0;
05
06       System.out.println(x / y);
07       System.out.println(x / (int) y);
08       System.out.println((int) (x / y));
09    }
10 }
```

실행 결과

```
3.3333333333333335
3
3
```

Section 03

입력 처리

3. 입력처리

Scanner 클래스

- 키보드로 데이터를 입력을 받을 때 사용함
- Scanner 클래스의 객체를 사용하는 방법

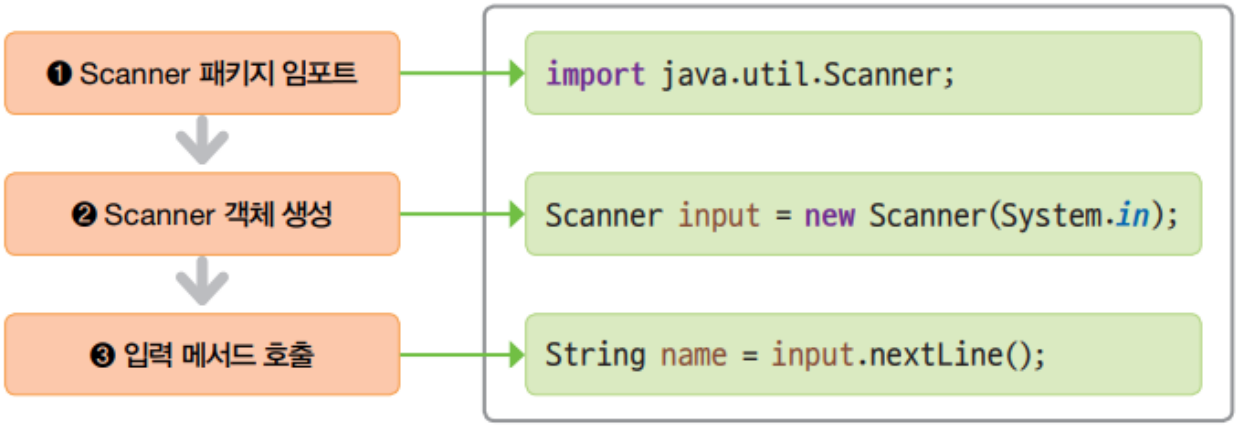
→java.util.Scanner 클래스를 임포트(import)해야 함

[표 3-8] Scanner 클래스가 제공하는 메서드

메서드	입력 자료형	메서드	입력 자료형
nextInt()	integer	nextShort()	short
nextFloat()	float	next()	공백문자 불포함 문자열
nextDouble()	double	nextLine()	한 행의 문자열(공백문자 포함)
nextLong()	long	nextBoolean()	boolean

3. 입력처리

■ 입력처리과정



[그림 3-13] Scanner 클래스 사용 과정

3. 입력처리

■ 입력처리과정

① 프로그램의 첫 행에서 Scanner 클래스의 패키지를 임포트

```
import java.util.Scanner;    // Scanner 클래스만 가져옴
import java.util.*;          // java.util 패키지 전체를 가져옴
```

② 키보드로 데이터를 입력받기 위해 System.in 객체와 연결된 Scanner 클래스의 객체를 생성함

✓System.in : 키보드 입력 스트림 객체

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

③ Scanner 클래스가 제공하는 다양한 메서드를 이용하여 키보드로 데이터를 입력받음

3. 입력처리

Scanner 클래스 사용 예시

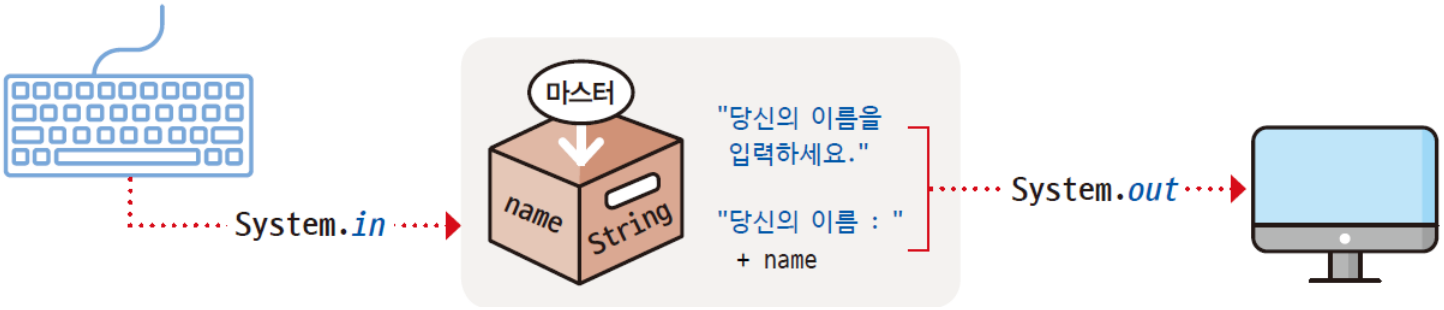
```
import java.util.Scanner;

public class Example08 {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      System.out.println("당신의 이름을 입력하세요.");
      String name = input.nextLine();
      System.out.println("당신의 이름 : " + name);
   }
}
```

실행 결과

```
당신의 이름을 입력하세요.
마스터
당신의 이름 : 마스터
```



[그림 3-14] 키보드 입력의 처리 과정

3. 입력처리

예제 3-8 Scanner 클래스를 이용하여 정수와 실수 입력하기

```
01 import java.util.Scanner;
02
03 public class Input01 {
04    public static void main(String[] args) {
05
06        Scanner s1 = new Scanner(System.in);
07        System.out.println("정숫값 입력하기");
08        int n = s1.nextInt();
09
10        System.out.println("실숫값 입력하기");
11        double db = s1.nextDouble();
13        System.out.println("정숫값 : " + n);
14        System.out.println("실숫값 : " + db);
15    }
16 }
```

실행 결과

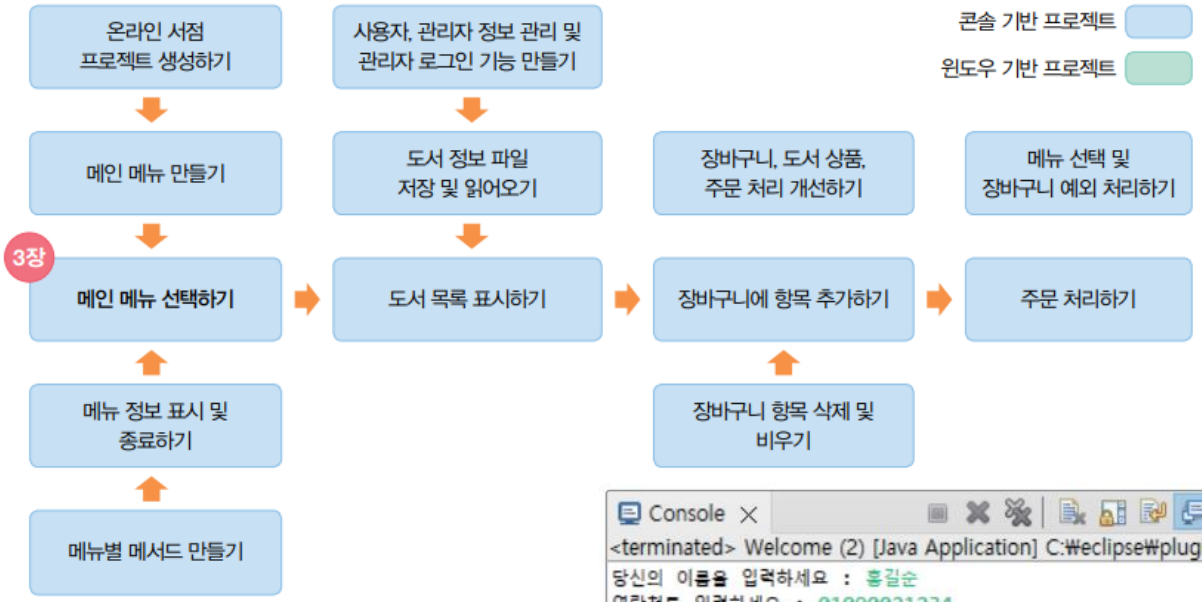
```
정숫값 입력하기
10
실숫값 입력하기
15.2
정숫값 : 10
실숫값 : 15.2
```

[프로젝트]

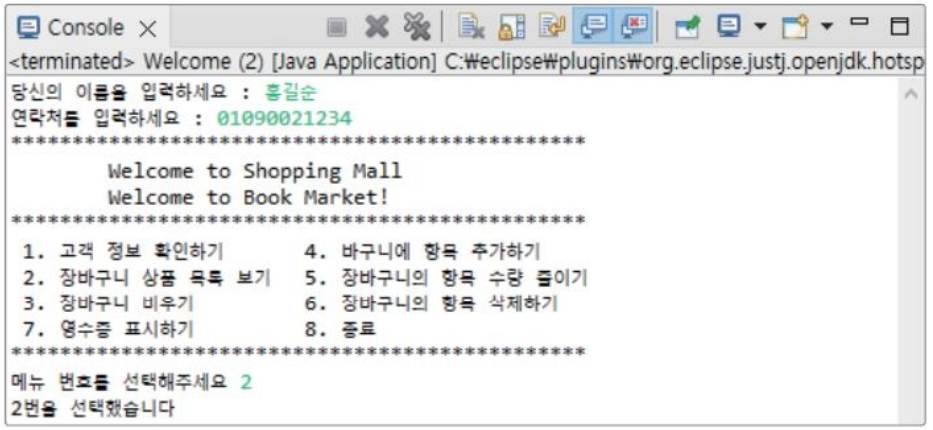
메인 메뉴 선택하기

메인 메뉴 선택하기

- 사용자의 정보(이름과 연락처)를 입력 받아 저장하고, 온라인 서점의 메인 메뉴를 만들어 사용자 입력 처리를 통해 메뉴를 선택할 수 있게 합니다.



[그림 3-15] 메인 메뉴 선택하기



[그림 3-16] 메인 메뉴 선택하기 실행 결과

메인 메뉴 선택하기

사용자 정보 입력 받기

- **01 Scanner 클래스 패키지 임포트 하기**

  →2장의 [프로젝트 2-2]에서 완성한 Welcome.java 파일을 열고 첫 행에서 java.util.Scanner 패키지를 임포트 함

- **02 Scanner 클래스의 객체 생성하기**

  →[프로젝트 3-1]에 Scanner 클래스의 객체를 생성 하는 코드를 추가함

- **03 Scanner 메서드를 이용하여 이름과 연락처 입력 받기**

  →[프로젝트 3-2] 에 이름은 문자 열로, 연락처는 정수로 입력 받는 코드를 추가함

메인 메뉴 선택하기

메뉴 번호 입력 받기

- **01 메뉴 번호 입력 받아 출력하기**

→메뉴가 끝나는 부분([프로젝트 3-3])에 메뉴 번호를 선택하여 입력하면 그 번호를 출력하는 코드를 추가함