객체지향 프로그래밍

Object-Oriented Programming

Chapter 02. 자바 프로그램 구조와 기초 문법

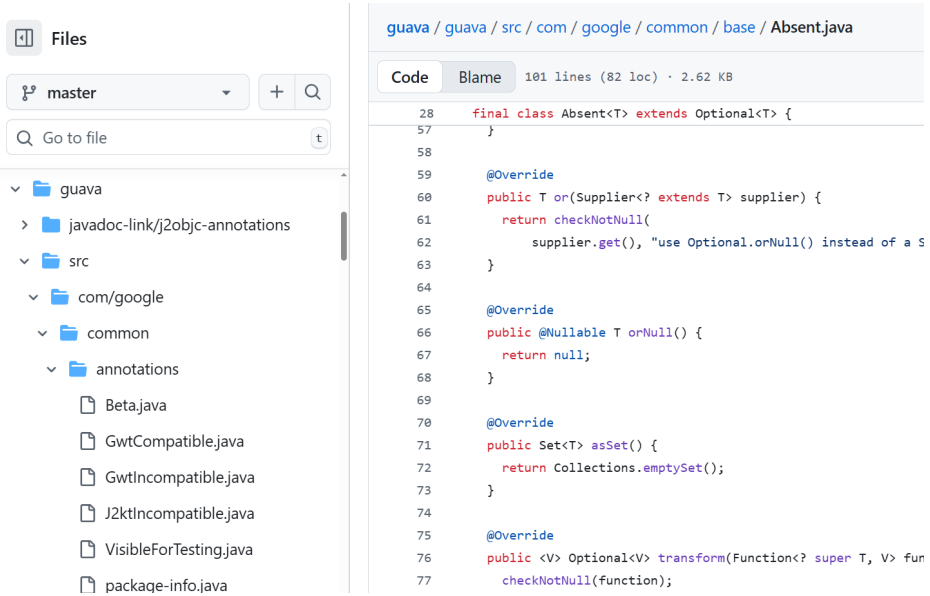
1. 자바 프로그램 기본 구조
2. 식별자
3. 변수
4. 타입 변환
5. 기본 입출력
6. 연산자

자바 프로그램 기본 구조

자바 프로그램 구조

클래스 파일 (*.java): 클래스, 메서드 등 작성

패키지 (폴더): 연관된 클래스들을 묶어 접근 제어와 namespace 관리를 제공하는 단위



자바 프로그램 기본 구조

문법 요소

하나의 클래스 파일은 다양한 문법 요소로 구성

- **클래스 (class)** : 객체 지향 언어에서 프로그램을 개발하는 단위
- **메서드 (method)** : 수행할 작업을 나열한 코드의 모임
- **실행문 (statement)** : 변수 선언, 값 저장, 메서드 호출 등의 작업을 지시하는 코드
- **주석문 (comment)** : 프로그램에 덧붙이는 설명문
- **식별자 (identifier)** : 클래스, 변수 등의 이름

```java
package sec01;

/**
 * 콘솔에 '안녕' 메시지를 출력하는 자바 프로그램
 */

public class Hello {
    public static void main(String[] args) {
        /*
         * 메인 메소드 내부
         */
        System.out.println("안녕!");  // 화면에 문자 출력

        System.out.println("안녕" + "!");

        String hello = "안녕!";
        System.out.println(hello);
    }
}
```

식별자

식별자 (identifier) : 클래스, 변수, 메서드 등을 구별 짓기 위한 이름

- 문자, 언더바(_), $로 시작해야 한다. 한글도 가능하며, 영문자는 대·소문자를 구분한다.
- +, - 등 연산자를 포함하면 안 된다.
- **자바 키워드**를 사용하면 안 된다.

자바 키워드 : 특정 문법적 역할을 수행하도록 미리 지정해 놓은 **예약어**

분류	키워드
데이터 타입	byte, char, short, int, long, float, double, boolean
접근 지정자	private, protected, public
제어문	if, else, for, while, do, break, continue, switch, case
클래스와 객체	class, interface, enum, extends, implements, new, this, super, instanceof, null
예외 처리	try, catch, finally, throw, throws
기타	abstract, assert, const, default, false, final, import, native, package, return, static, strictfp, synchronized, transient, true, void, volatile

식별자

식별자 명명 관례

변수, 메서드: 모두 소문자로 표기.
단, 복합 단어일 때는 두 번째 단어부터 단어의 첫 자만 대문자로 표기

```
int thisYear;
String currentPosition;
boolean isEmpty;
public int getYear( ) { }
```

클래스: 첫 자만 대문자로 표기하고 나머지는 소문자로 표기.
단, 복합 단어일 때는 두 번째 단어부터 단어의 첫 자만 대문자로 표기

```
public class HelloDemo { }
public interface MyRunnable { }
```

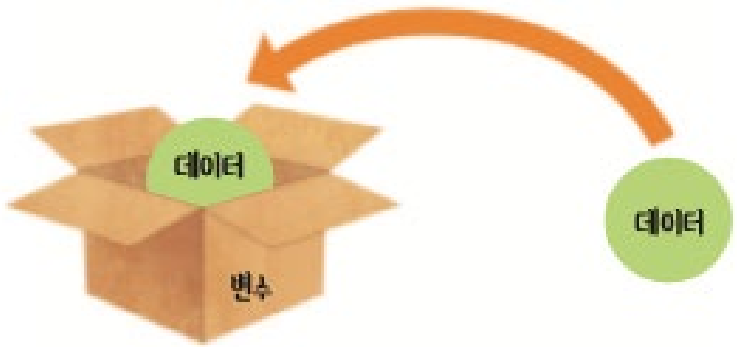
상수: 전체를 대문자로 표기.
언더바(_)로 단어 구분

```
final int NUMBER_ONE = 1;
final double PI = 3.141592;
```

변수

변수(Variable)

프로그램은 기억 공간에 **데이터(data)**를 보관하고, 각 기억 공간을 **변수(variable)**로 구분



변수

데이터 타입(data type)

- **값(value)**과 **연산(operation)**의 집합
- 데이터의 유형에 따라 **저장 방식**과 **처리 방식**이 달라지므로, 데이터 타입을 두어 이를 구분

정수 타입	실수 타입	논리 타입	문자 타입
-12, 0, 1024, 256	-1.267, 0.13, 10.23, 20000	참, 거짓	A, B, C, !, #, *

변수

데이터 타입(data type)

변수

기초 데이터 타입

분류	기초 타입	기억 공간 크기	기본 값	값의 범위
정수	byte	8비트	0	−128~127
	short	16비트	0	−32,768~32,767
	int	32비트	0	−2,147,483,648~2,147,483,647
	long	64비트	0L	−9,223,372,036,854,775,808 ~9,223,372,036,854,775,807
문자	char	16비트	null	0('\u0000')~65,535('\uFFFF')
실수	float	32비트	0.0f	약 $\pm 3.4 \times 10^{-38} \sim \pm 3.4 \times 10^{+38}$
	double	64비트	0.0d	약 $\pm 1.7 \times 10^{-308} \sim \pm 1.7 \times 10^{+308}$
논리	boolean	8비트	false	true와 false

변수

변수 선언문

- 데이터 타입, 변수 이름(식별자)로 구성

선언과 동시에 초기화

```
int age = 23;
double height = 189.52;
boolean married = false;
```

데이터 타입 변수 이름 = 초기값;

선언 이후 초기화

```
int age;
age = 23;
```

변수

변수 선언문

- 데이터 타입, 변수 이름(식별자)로 구성

한 줄에서 여러 개의 변수 선언

```
int friends, children, age;
double weight = 102.0, height = 105.5;
```

변수

변수의 사용

- 변수의 데이터를 읽기
- 변수에 데이터를 쓰기

데이터 저장

```
int count = 10000; // count 변수 선언, 10000으로 초기화
count = 15000; // count 변수에 15000를 저장
```

데이터 읽기

함수 호출 매개변수(parameter), 피연산자 등에 변수 이름을 사용

```
System.out.println(count); // count 변수에 담긴 데이터를 출력
int newCount = 0;
newCount = count; // count 변수에 담긴 데이터를 불러와 newCount 변수 초기화
```

변수

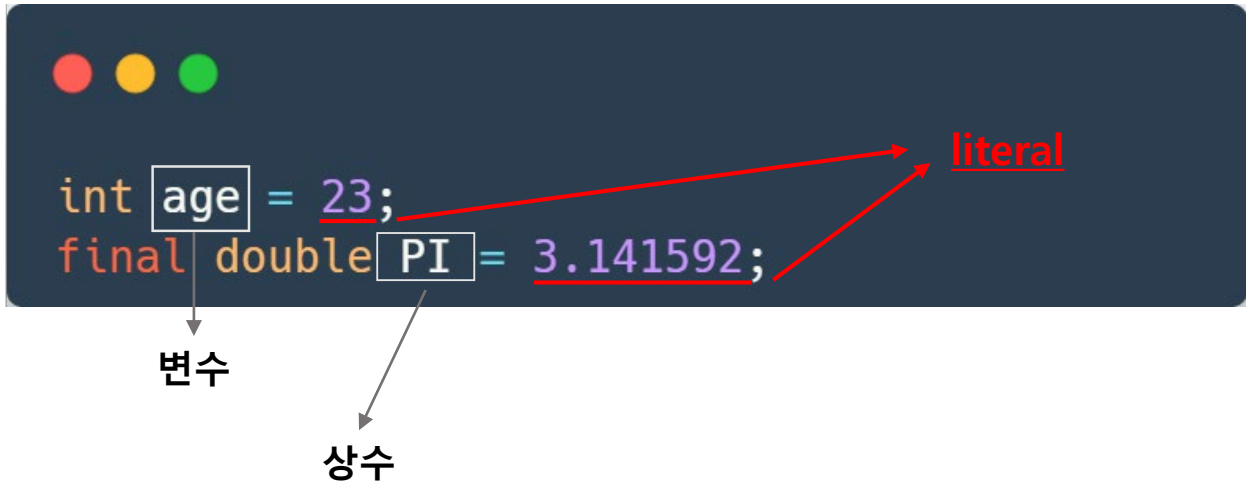
상수(constant)

- 프로그램 실행 도중 변경할 수 없는 데이터를 담는 변수
- **final** 키워드로 지정
- 한번 초기화되면, 데이터를 덮어쓸 수 없음

변수

리터럴(literal)

- 프로그램에서 미리 정의된 값
- 변하지 않는 **데이터 그 자체**



변수

리터럴 표기 방법 – 정수 타입

- 접두사(prefix)로 표기 진법을 구분 (기본 10진수)
- long 타입 명시를 위해서는 접미사로 L 사용
- 접미사 없이 표기할 경우 int 타입

```
int fifteen = 15;         // 10진수 표기
byte fifteen = 0b1111;    // 2진수 표기
short fifteen = 017;      // 8진수 표기
int fifteen = 0xF;        // 16진수 표기
long speed = 300000000L;// L 접미사로 long 타입 명시
```

접두사	진법	예시
0b	2	0b1101
0	8	059
0x	16	0xFFA1

변수

리터럴 표기 방법 – 실수 타입

- 접미사 없이 표기할 경우 double 타입
- Float 타입 명시를 위해서는 접미사로 F 사용

```
double half = 0.5;      // 일반 표기법
double half = 5E-1;     // 지수 표기법
float pi = 3.14159F;    // F 접미사로 float 타입 명시
float pi = 3.14159;     // ❌ 오류
```

변수

리터럴 표기 방법 – 문자 타입

- char는 문자 하나를 나타내는 데이터 타입
- 작은 따옴표로 감싼 단일 문자, ASCII 코드, 유니코드로 표기 가능

```
char c = 'A';    // 문자
char c = 65;     // ASCII 코드로 대입
char '\u0041';   // 유니코드 값으로 대입
char c = "A";    // ❌ 오류 - "A"는 문자가 아닌 문자열
```

리터럴 표기 방법 – 논리 타입

```
boolean condition = true;
boolean condition = false;
```

변수

상수 VS 리터럴

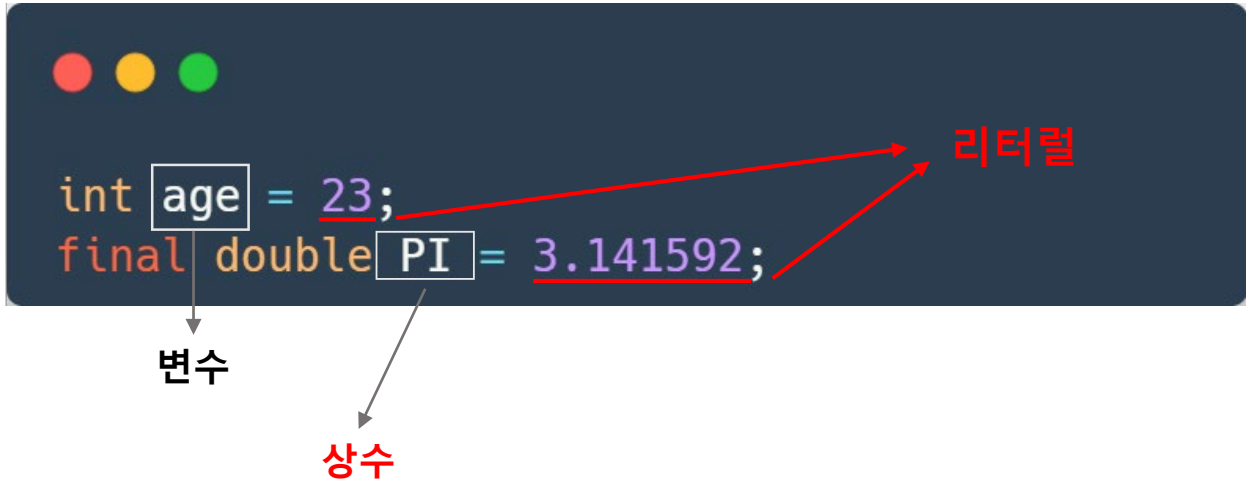
상수 (constant)

프로그램 실행 중 항상
동일한 값을 가지는 변수

리터럴 (literal)

프로그램에서
미리 정의한 값

상수는 리터럴에 의미 있는 이름을 붙여
코드의 가독성을 높이고, 쉽게 수정할 수 있게 한다.

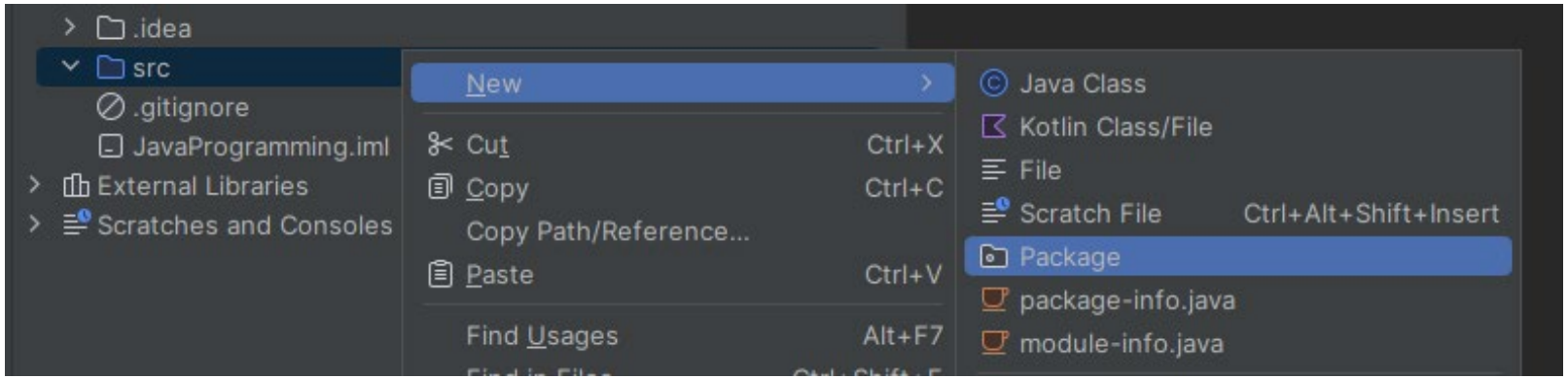


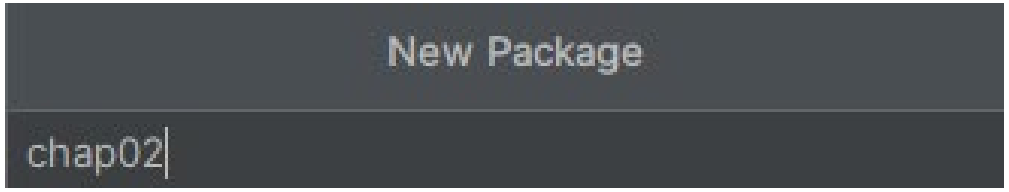
변수

Practice

1. 실습용 패키지 생성

IntelliJ IDEA – src 우클릭 - New – Package – **chap02** 입력



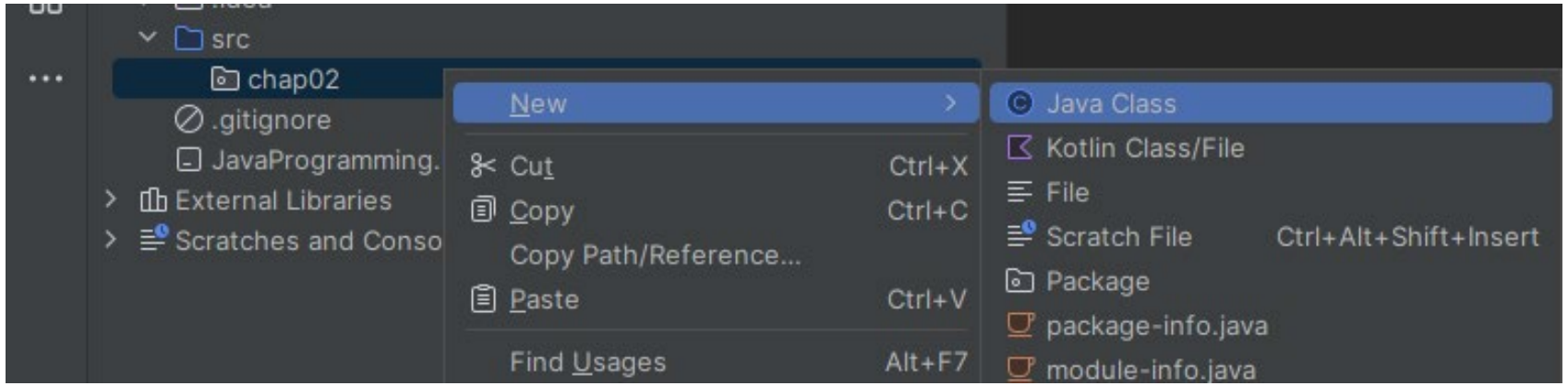


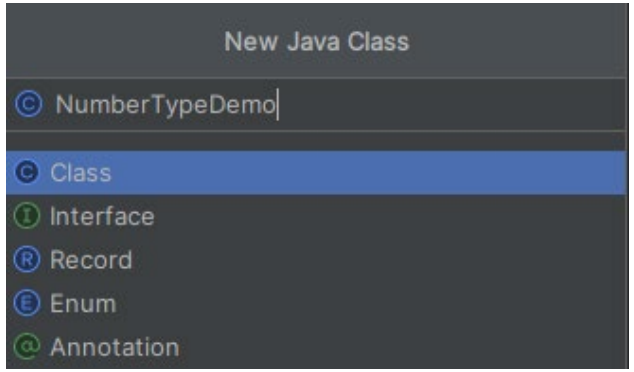
변수

Practice

2. 클래스 생성

IntelliJ IDEA – chap02 패키지 우클릭 – New – Java Class – NumberTypeDemo 입력





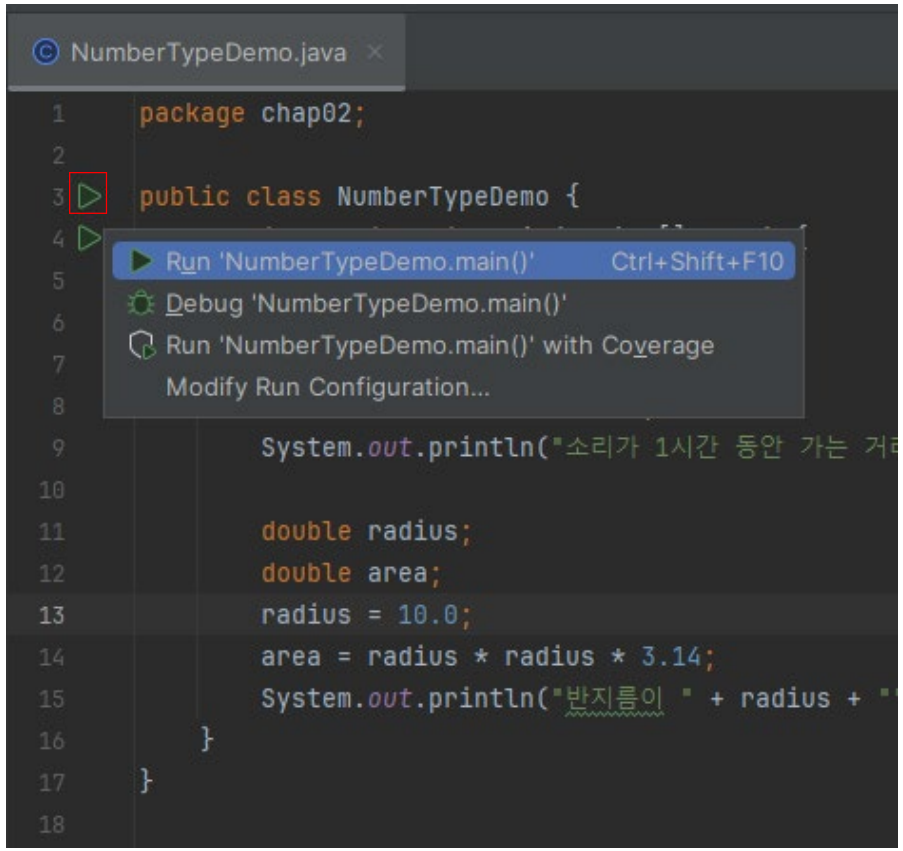
변수

Practice

3. 실행 방법

프로젝트 내 main 메서드가 여러 개 존재할 때, 특정 메서드를 실행하는 방법

- 실행할 메서드가 작성된 클래스 파일 열기
- ▶ 버튼 클릭
- Run '*.main()'



변수

Practice

- **예제 2-2**: 정수 및 실수 타입 응용
  - package: chap02
  - class: NumberTypeDemo

- **예제 2-3**: 문자 및 논리 타입 응용
  - package: chap02
  - class: CharBoolDemo

타입 변환

자동 타입 변환

- **일부 연산식에 대해 자동으로 발생하는 데이터 타입 변환**
- 자바에서는 하나의 연산식(expression)을 서로 다른 타입의 데이터들로 구성 가능
- 이때 자바 컴파일러는 가장 큰 데이터 타입으로 모든 데이터 타입을 통일

```
double d1 = 5 * 3.14; // 정수 5를 실수 5.0으로 자동 타입 변환
double d2 = 1;        // 정수 1을 실수 1.0으로 자동 타입 변환
```

타입 변환

강제 타입 변환

- **타입 변환 연산자를 사용해 데이터 타입을 강제로 변환하는 연산**

```
// double의 3.14를 float로 형 변환해 f에 3.14F 저장
float f = (float)3.14;

// int의 300을 byte로 형 변환하면 데이터 손실 발생
byte b = (byte)300;

// double의 3.14를 byte로 형 변환하면 데이터가 손실되고 3만 저장
byte x = (byte)3.14;

// float의 3.14를 double로 형 변환하면 데이터 손실 없이 저장
double d = (double)3.14f;
```

타입 변환

타입 변환으로 인한 데이터 손실

- **기억 공간 크기** 차이, **정밀도** 차이로 인해 타입 변환 시 일부 조건에서 데이터의 손실 발생

```
int i = (int) 22E8;
// i == 2147483647
```

double → int

```
double d = 2000000.000000001;
float f = (float) d;
// d == 2000000.000000001
// f == 2000000.0
```

double → float

타입 변환

타입 변환으로 인한 데이터 손실

- **기억 공간 크기** 차이, **정밀도** 차이로 인해 타입 변환 시 일부 조건에서 데이터의 손실 발생

```
int i = (int) 22E8;
// i == 2147483647
```

double → int

```
double d = 2000000.000000001;
float f = (float) d;
// d == 2000000.000000001
// f == 2000000.0
```

double → float

타입 변환

Practice

- **예제 2-5**: 타입 변환 응용
  - package: chap02
  - class: CastDemo

기본 입출력

표준 입출력 객체

- System.in (표준 입력 스트림): 키보드로부터의 입력을 불러올 수 있는 객체
- System.out (표준 출력 스트림): 콘솔 화면과 연결된 객체

기본 입출력

화면에 데이터 출력

System.out 객체와 연결해 다음과 같은 메서드를 호출함으로써 콘솔 출력 수행

println() : 내용을 출력한 후 행을 바꾼다.

print() : 내용을 출력만 하고 행은 바꾸지 않는다.

printf() : 포맷을 지정해서 출력한다.

기본 입출력

printf

- **포맷 명시자**를 통해 데이터가 출력될 형식을 지정한다

```
System.out.printf("포맷 명시자", 데이터, 데이터, …);
```

```
int x = 5;
double pi = 3.14;

System.out.printf("x = %d and pi = %f\n",  x,  pi);
```

x 변수를 십진수 정수 포맷과 대응시킨다.

데이터 항목들

포맷 명시자

변수 pi를 십진수 실수 포맷과 대응시킨다.

기본 입출력

printf 포맷과 실행 결과

종류	데이터	포맷	실행 결과	설명
정수	97	%d	97	10진수
		%o	141	8진수
		%x	61	16진수
		%c	a	문자
		%5d	97	5자리. 빈자리는 공백 처리한다.
		%-5d	97	5자리. 빈자리는 공백 처리한다. 왼쪽 정렬
		%05d	00097	5자리. 빈자리는 0으로 채운다.
문자열	"java"	%s	"java"	문자열
		%5s	" java"	5자리. 빈자리는 공백 처리한다.
		%-5s	"java "	5자리. 빈자리는 공백 처리한다. 왼쪽 정렬
실수	3.14f	%f	3.140000	10진수 실수
		%e	3.140000e+00	지수
		%4.1f	3.1	4자리. 소수점 이하 1자리
		%04.1f	03.1	4자리. 소수점 이하 1자리. 빈자리 0
		%-4.1f	3.1	4자리. 소수점 이하 1자리. 왼쪽 정렬

기본 입출력

Practice

- **예제 2-6**: 포맷을 이용한 화면 출력
  - package: chap02
  - class: PrintfDemo

기본 입출력

키보드로 데이터 입력 받기

- System.in 객체는 InputStream의 데이터 타입
- InputStream은 이진(binary) 형태의 데이터 스트림만 제공
- 지정된 형태(문자열, 정수, 실수)로 데이터 입력 받기 위해 **Scanner**와 연결해 사용

기본 입출력

Scanner 타입의 메서드

- 키보드로 입력 받은 데이터를 특정 타입으로 변환하여 리턴하는 메서드들을 제공한다

메서드	반환 타입
next( )	String
nextByte( )	byte
nextShort( )	short
nextInt( )	int
nextLong( )	long
nextFloat( )	float
nextDouble( )	double
nextLine( )	String

기본 입출력

Practice

- **예제 2-7**: 키보드로 데이터 입력
  - package: chap02
  - class: PrintfDemo

연산자

연산(operation)

프로그램에서 주어진 데이터를 계산해 결과를 얻어 내는 과정

연산자

연산자(operator)

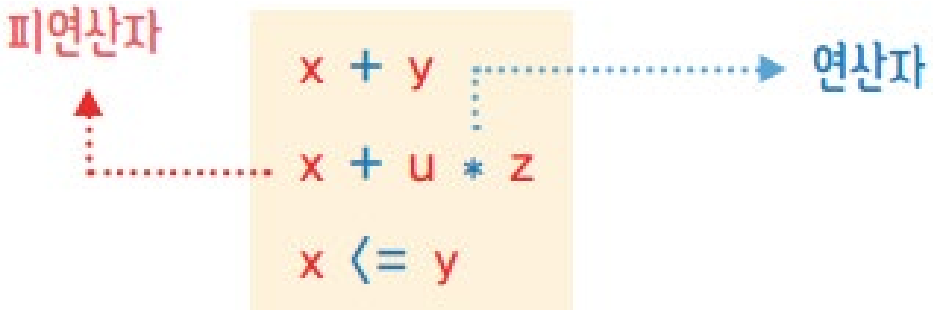
- **연산에 사용하는 표시나 기호**

피연산자(operand)

- **연산되는 데이터**
- 변수, 상수, 리터럴, 메서드 리턴 값 등

연산식(expression)

- **피연산자와 연산자의 조합**
- 하나의 타입을 갖는 연산 결과를 나타낸다



연산자

연산자의 종류

종류	연산자	설명	비고
증감	++, −−	1만큼 증가 또는 감소한다.	단항
산술	+, −, *, /, %	사칙 연산과 모듈로 연산한다.	이항
시프트	〉〉, 〈〈, 〉〉〉	비트를 좌우로 이동한다.	이항
부호	+, −	부호를 변환한다.	단항
비교	〉, 〈, 〉=, 〈=, ==, !=, instanceof	데이터 값을 비교하거나 데이터 타입을 비교한다.	이항
비트	&, |, ~, ^	비트 단위의 AND, OR, NOT, XOR	단항, 이항
논리	&&, ||, !, ^	논리적 AND, OR, NOT, XOR	단항, 이항
조건	(expr) ? x : y	expr에 따라 x 또는 y로 값을 결정한다.	삼항
대입	=, +=, −=, *=, /=, &=, |=, ^=, 〉〉=, 〈〈=, 〉〉〉=	오른쪽 값을 연산해 왼쪽에 대입한다.	이항

연산자

산술 연산자

- **+ - / *** : 사칙연산 (덧셈, 뺄셈, 나눗셈, 곱셈)
- **%**: 나머지 연산 (정수형 타입)

- 피연산자의 타입이 서로 다를 경우, 큰 범위의 타입으로 일치시킨 후 연산 수행 (자동 타입 변환)

```
// 짝수와 홀수 여부 판단. a가 1이면 n은 홀수, 0이면 짝수
int a = n % 2;

// 3의 배수인지 확인, b가 0이면 n은 3의 배수
int b = n % 3;
```

연산자

Practice

- **예제 2-8**: 산술 연산자 응용
  - package: chap02
  - class: ArithmeticDemo

연산자

Challenge

- **도전과제 1**: 직사각형 넓이 계산 (p. 75)
  - package: chap02
  - class: Challenge01

직사각형의 가로와 세로를 키보드로 입력받아
넓이를 출력하는 프로그램을 작성해 보자.

연산자

Challenge

- **도전과제 1**: 직사각형 넓이 계산 (p. 75)
  - package: chap02
  - class: Challenge01

직사각형의 가로와 세로를 키보드로 입력받아 넓이를 출력하는 프로그램을 작성해 보자.

```java
package chap02;

import java.util.Scanner;

public class Challenge01 {
    public static void main(String[] args) {
        double w, h, area;
        Scanner in = new Scanner(System.in);
        System.out.print("직사각형의 가로 길이를 입력하세요 : ");
        w = in.nextDouble();

        // 나머지 코드 작성
    }
}
```