**Lecture3**

[Java 기본 문법] - 변수, 타입, 연산자

#Java 프로그램 기본 구조

1. 클래스: 객체 지향 언어에서 프로그램을 개발하는 단위
2. 메소드: 수행할 작업을 나열한 코드의 모임
3. 실행문: 작업을 지시하는 변수 선언, 값 저장, 메소드 호출 등의 코드

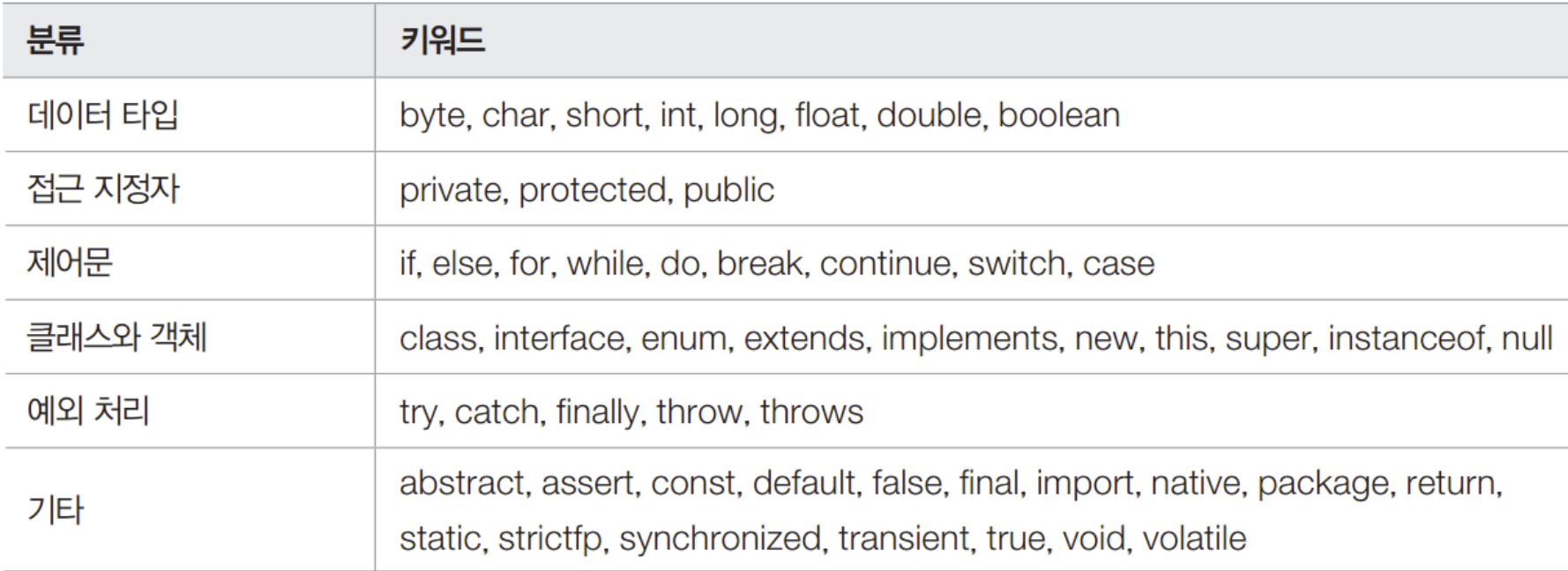
#주석문(comment): 행 주석 //, 범위 주석 /\* \*/, 문서 주석 /\*\* \*/

#식별자(Identifier)

: 보고 느낄 수 있는 모든 사물(객체)들을 각각 구별할 수 있는 것

-규칙: 문자, 언더바, $로 시작, +/- 등 연산자 포함X, \**자바 키워드* 사용X, 길이 제한X

*\*자바 키워드*



-관례 → 카멜 표기법(Camel Case)

1. 변수(Variable)

: 프로그램은 메모리에 데이터를 보관, 각 메모리를 변수로 구분 / 종류 다양→데이터 타입 사용

ex) 변수 선언-int age 21 / 타입 + 변수명 + 메모리 공간

2. 데이터 타입(Data Type)-기초 타입, 참조 타입

3. 연산자(Operator)

* 대입 연산자(=): 오른쪽 값을 왼쪽 변수에 저장하는 기호
* 산술 연산자\*(+, -, \*, /, %): 수치 계산을 위한 연산자
* 증감 연산자(++, —): 변숫값을 1만큼 증가 또는 감소(변수 앞에 있는 경우 전위 연산, 뒤에 있는 경우 후위 연산)
* 비교 연산자(==, !=, >, >= 등)
* 논리 연산자: 피연산자의 조건을 결합해서 true, false 조사, 논리 타입에만 사용  
  –쇼트 서킷(short circuit)  
  조건식1 && 조건식2 → 조건식1이 false면 결과도 false이므로 조건식2의 진릿값 조사할 필요X

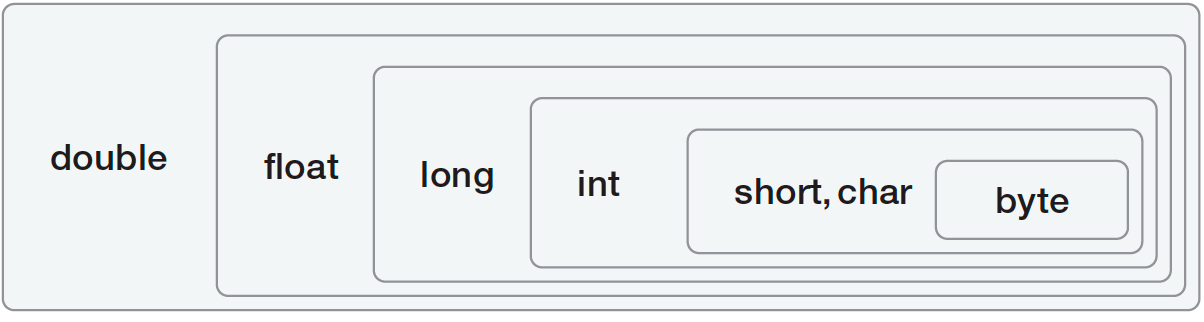
조건식1 || 조건식2 → 조건식1이 true면 결과도 true이므로 조건식2의 진릿값 조사할 필요X

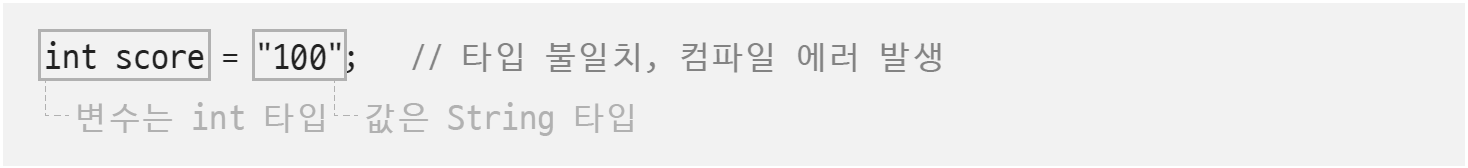
#Java 기본 문법 예제

-화면 데이터 출력: System.out.println()-행 바꿈, System.out.print(), System.out.printf()-포맷 지정

-형식 문자열 출력: %s-해당 위치에 string값 삽입, %d-int, %f-double, %d-boolean

**Lecture4**

타입의 범주는 위와 같다.



대입할 값과 변수의 타입이 서로 달라 문제가 생긴다.

**따라서 변수와 대입값의 타입을 일치시켜야 한다.**

**→타입 변환**이라 한다.

**자동 타입 변환: 암묵적으로 타입이 변환된다.**

-작은 타입의 데이터는 같은 범주의 큰 데이터 타입으로 자동 변환된다.

**강제 타입 변환**: **명시적으로 타입을 강제 변환한다.**

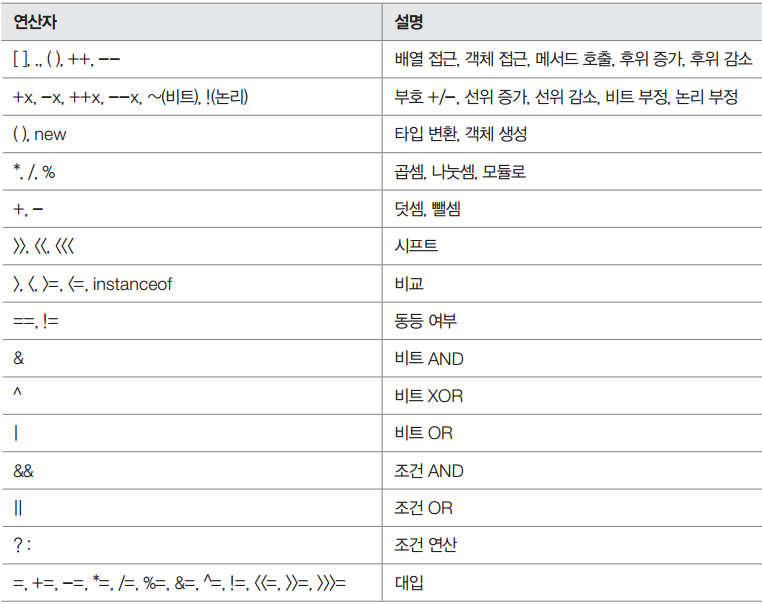
-타입을 강제로 바꾸기 위해 캐스팅을 사용한다.

캐스팅의 종류

-다운 캐스팅: 넓은 범위에서 좁은 범위로 해석

-업 캐스팅: 좁은 범위에서 넓은 범위로 해석

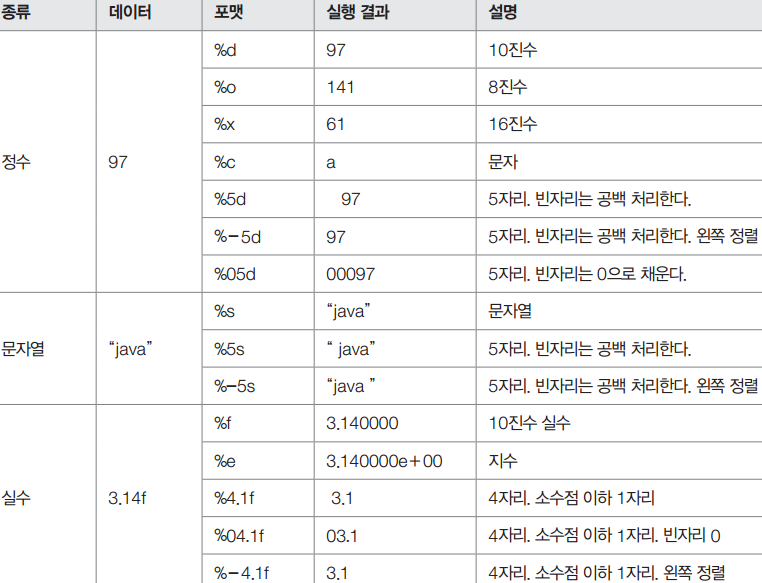
**연산자**



기본 입출력

-키보드로 데이터를 입력하고 화면에 데이터를 출력한다.

printf()의 포맷 종류와 실행 결과

****

**키보드로 데이터 입력**

-프로그램의 첫 행에 다음을 추가해 Scanner 클래스의 경로 이름을 컴파일러에 알림

**>>import java.util.Scanner;**

-키보드로 데이터를 입력 받기 위해 System.in 객체와 연결된 Scanner 객체를 생성

**>>Scanner in = new Scanner(System.in);**

-Scanner 클래스가 제공하는 다양한 메서드를 이용해 키보드로 데이터를 입력 받음

**>>int x= in.nextInt(); //정수를 읽어 변수 x에 대입**

**메소드의 개념**

-메소드란 입력값에 따른 결과를 반환하는 일련의 코드 묶음

-간단한 호출만으로도 여러 줄의 코드를 수행할 수 있음

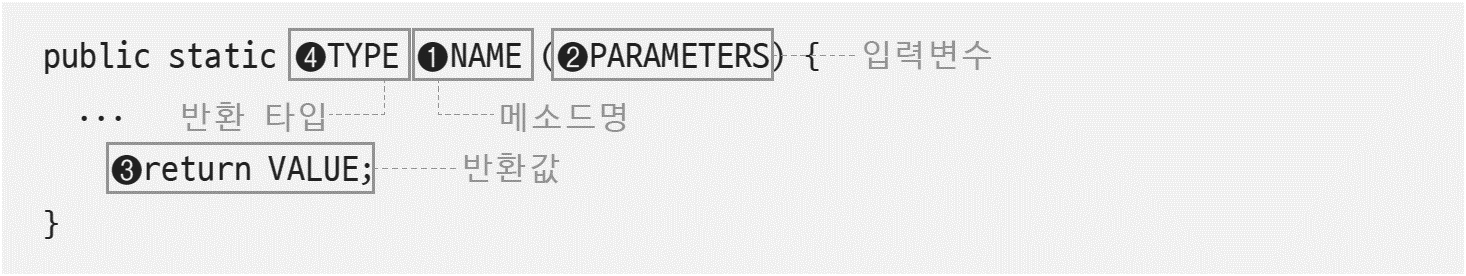
**동작 원리**

**-호출부:** 메소드를 사용하는 영역, 입력 값을 전달하여 그 결과를 돌려받는 코드

**-정의부:** 입력값을 받아 결과를 만드는 일련의 코드 묶음

메소드를 만드는 것을 정의라 하고, 이를 사용하는 것을 호출이라 부른다.

**메소드의 구조**

****

1. **메소드명**: 메소드의 이름으로, 호출 시 사용
2. **매개변수**: 메소드 실행 시 입력값을 저장하는 변수, 입력변수 혹은 파라미터라고도 부름
3. **반환값**: 메소드의 최종 결과로, 반환될 값을 의미
4. **반환 타입**: 반환값의 타입을 의미(void는 반환값이 존재하지 않는 메소드이다.)

**Lecture5**

[제어문]

: 프로그램의 흐름에 영향을 주고 때에 따라 제어가 가능하도록 하는 것 → 실행문의 수행 순서 변경

1. 조건문

: 상황별 실행 흐름을 만드는 문법

1. if 문  
   -if-else 문: 두 조건식 중 하나를 선택할 때 사용 / 조건식 참→if문 코드, 거짓→else문 코드 수행

-다중 if 문: 추가 조건을 붙여야 할 때 사용ㅇ]

-중첩 if 문: if 문에 다른 if 문이 포함되는 것

1. switch 문: 상황(case)에 따라 수행 코드를 나누는 문법 → 여러 경로 중 하나를 선택할 때 사용

ex) 소괄호 사이에 기준값 grade를 두고, 그 상태를 “A”, “B, “C”, default(그 외)의 경우로 분리

-기존: 콜론 레이블 사용, break문 사용

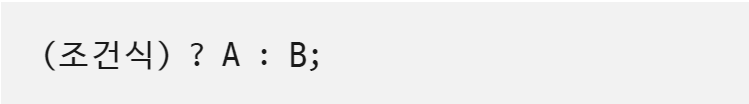
-Java 14 이상: 화살표 case 레이블, switch 연산식, 다중 case 레이블, yield 예약어

-주의 사항: 가능한 모든 값에 대하여 일치하는 case 레이블이 없으면 오류 발생

**Lecture 6**

1. **삼항 연산자**

삼항 연산자는 조건식에 따른 반환값을 지정하는 문법이다.



조건식이 참이면 A를 거짓이라면 B를 반환한다.

**2. 반복문**

:반복문은 규칙적 반복 코드를 단순화하는 문법으로 조건에 따라 같은 처리를 반복하게 하는 문법이다.

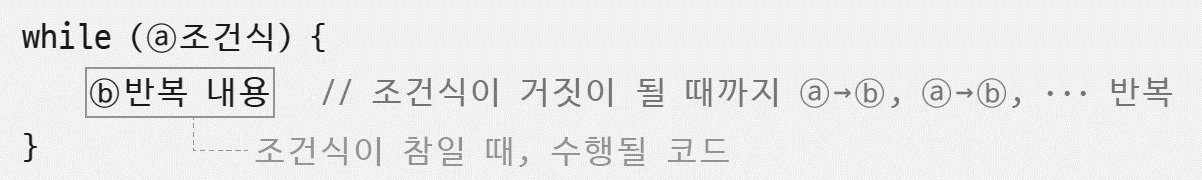
크게 while문과 for문으로 구분한다.

**-while문**: 반복 횟수가 명확하지 않을 때 사용하기 적합하다.

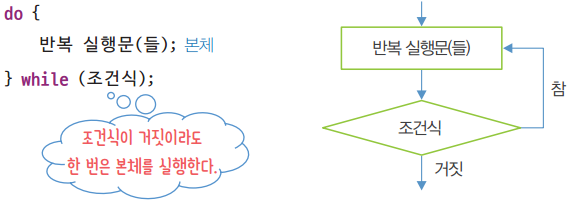
**-for문**: 반복 횟수가 명확할 때 사용하기 적합하다.

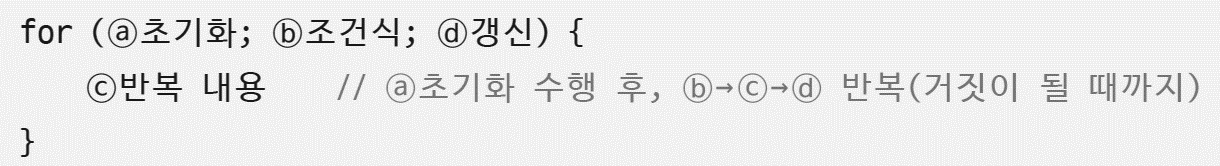
cf) 반복문 사용시 무한 루프에 빠지지 않도록 주의해야한다.

**while문**



**do-while문**



**for문**

**3. 분기문**

**break문**

:break문은 반복문을 탈출하는 문법이다.

**continue문**

:continue문은 실행 흐름을 다음 반복으로 넘기는 문법이다.  
 반복문의 모든 코드를 수행하지 않은 채 다음 반복으로 이동한다.