

경기과학기술대학교 인공지능학과 · 전공기초

자바 프로그래밍

2주차

프로그램을 실행하고, 입력받아 출력하는 코드를 만든다

자바 프로그램은
클래스, `main()`, 변수, 문장으로 이루어진다.

오늘의 강의

교시마다 개념 설명 후, 직접 실행하는 순서로 진행

1교시



무엇으로 쓰나

Programiz 첫 실행
구조와 데이터 타입

2교시



어떻게 받고 계산하나

Scanner 키 입력
연산자

3교시



어떻게 가르나

조건문
가위바위보 게임

1교시

첫 실행과 프로그램의 구조

설치 없이 브라우저에서 실행한다

Programiz 와 Main

클래스와 main()

타입 · 변수 · 리터럴

Programiz 온라인 컴파일러



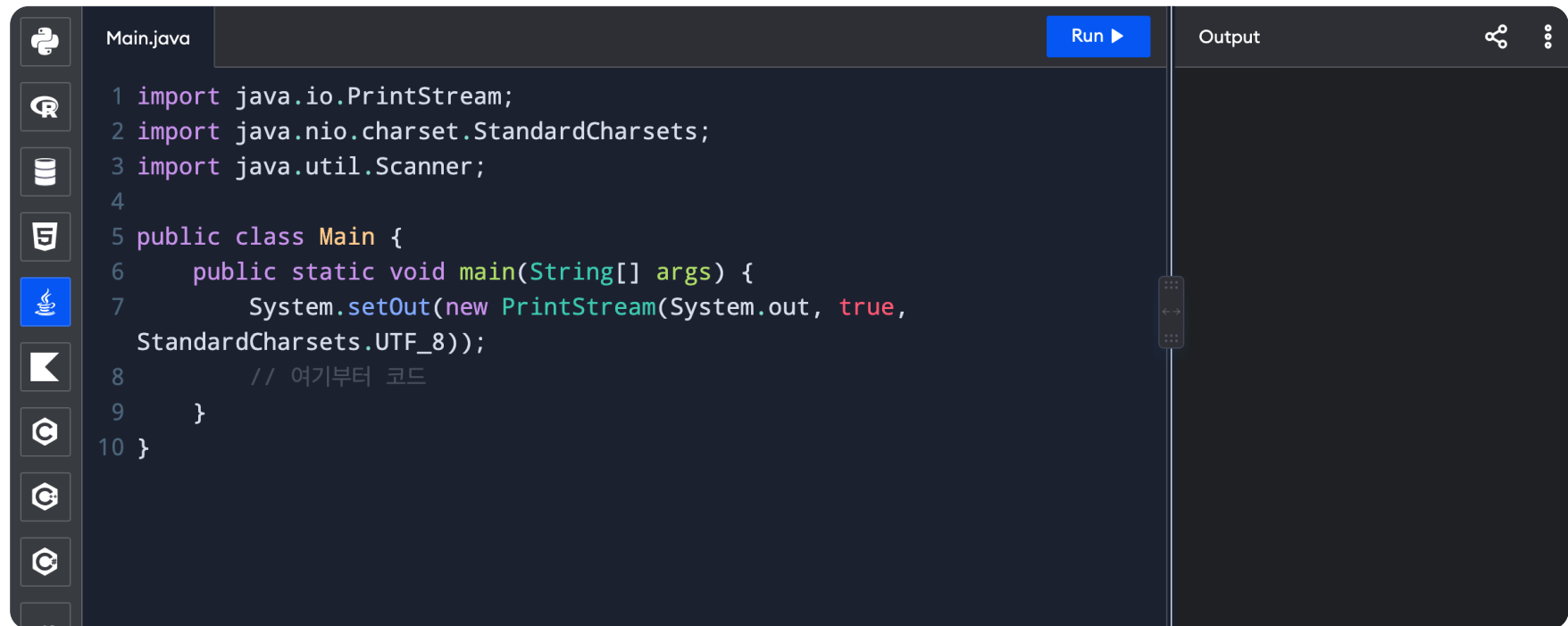
<https://www.programiz.com/java-programming/online-compiler/>

자바

로그인 없음

설치 없음

Programiz 화면 구성



① 언어 사이드바

② 편집기 Main.java

③ Run 버튼

④ Output 패널

Run 버튼 한 번에 일어나는 일

Run 버튼 한 번

① 소스

Main.java 를 넘긴다

② 컴파일

javac 가 .class 를 만든다

③ 실행

JVM 이 .class 를 실행한다

컴파일 오류가 나면 Output 에 javac 메시지가 먼저 나온다

메인 클래스는 Main

편집기 파일은 Main.java 하나로 고정

1

메인 클래스는 Main 으로 쓴다

public 클래스 이름은 파일 이름과 같아야 한다.

2

다른 클래스에는 public 을 붙이지 않는다

public 이 없으면 파일 이름과 달라도 된다.

이름이 다르면 나는 오류

```
Main.java Run ▶  
1 public class Hello {  
2     public static void main(String[] args) {  
3         System.out.println("Hello");  
4     }  
5 }  
6  
7 class Main {  
8 }
```

Output

ERROR!
Main.java:1: error: class Hello is public, should be
declared in a file named Hello.java
public class Hello {
 ^
1 error

=== Code Exited With Errors ===

public class Hello 와 class Main 이 함께 있을 때

public 을 떼면 클래스가 하나일 때 다른 이름도 실행됩니다.

Programiz 시작 템플릿

```
1 import java.io.PrintStream;
2 import java.nio.charset.StandardCharsets;
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) {
7         System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));
8         // 여기부터 코드를 쓴다
9     }
10 }
```

1~3줄 import

7줄 한글 출력 설정, 그대로 복사

8줄부터 내 코드

모든 예제는 이 열 줄에서 시작한다.

7줄이 필요한 이유

7줄 없음

```
System.out.println("헬로2030");
```

출력

??2030

7줄 있음

```
System.out.println("헬로2030");
```

출력

헬로2030

Programiz 서버가 한글을 못 보여주기 때문에 main 첫 줄에 이 설정을 붙인다

직접 실습

헬로2030 코드

main() 안에 넣을 두 줄

```
int n = 2030;
```

```
System.out.println("헬로"+n);
```

실행 단계

Run 을 누른다

헬로2030 실행 결과

Output



헬로2030

=== Code Execution Successful ===

자기소개 프로그램 만들기

미션

- 클래스는 Main
- 이름 · 나이 · 학과 세 줄
- println 세 번

실행 결과

Kitae Hwang
20 years old
Department of Computer Engineering

자기 정보로 바꾼다. 한글도 된다

자기소개 실행 결과

```
Output

Kitae Hwang
20 years old
Department of Computer Engineering

=== Code Execution Successful ===
```

println 세 줄이 적은 순서 그대로 한 줄씩 출력된다.

예제 2-1 Hello 코드

```
1 public class Hello {  
2  
3     public static int sum(int n, int m) {  
4         return n + m;  
5     }  
6  
7     public static void main(String[] args) {  
8         int i = 20;  
9         int s;  
10        char a = '?';  
11        s = sum(i, 10);  
12        System.out.println(a);  
13        System.out.println("Hello");  
14        System.out.println(s);  
15    }  
16  
17 }
```

실행 결과

?
Hello
30

오늘 배울 것이
이 안에 거의 다 있다.

Hello 예제 읽기

```
public class Hello {  
    public static int sum(int n, int m) {  
        return n + m;  
    }  
  
    public static void main(String[] args) {  
        int i = 20;  
        int s;  
        char a = '?';  
        s = sum(i, 10);  
  
        System.out.println(a);  
        System.out.println("Hello");  
        System.out.println(s);  
    }  
}
```

클래스 선언

메소드 sum()

main(), 실행 시작

변수 선언과 문장

출력 println

클래스

- class 키워드로 이름을 선언하고 { } 안에 변수와 메소드를 쓴다
- public 은 다른 클래스에서 참조할 수 있다는 뜻

```
public class Hello { ... }
```

자바에는 전역 변수도 전역 함수도 없습니다.

변수도 메소드도 클래스 안에 있다

주석문

// 한 줄 주석

그 줄 끝까지가 주석이다

```
int n = 2030; // 예제 숫자
// 이 줄 전체가 주석
```

/* */ 여러 줄 주석

닫기 전까지 모두 주석이다

```
/* 두 줄 이상
   설명할 때 */
```

컴파일러는 주석을 무시한다.

main() 메소드

```
public static void main(String[] args)
```

public

누구나 부를 수
있다

static

객체 없이 실행
된다

void

돌려주는 값이
없다

main

JVM 이 찾는
이름

한 클래스에 main() 은 하나

메소드(함수)

C/C++ 의 함수를 자바에서는 메소드라고 부른다. 클래스 안에만 쓴다.

정의

```
public static int sum(int n, int m) {  
    return n + m;  
}
```

클래스 안에 적는다.



호출

```
int i = 3;  
int s = sum(i, 10);
```

3 과 10 을 넘기고 13 을 받는다.

작성법과 호출법은 함수와 같다. 놓이는 자리가 다르다.

문장과 화면 출력

문장은 ; 로 끝난다

```
int i = 20;
```

세미콜론을 빠뜨리면
컴파일 오류가 난다.

println 과 print

```
System.out.print("학점은 ");  
System.out.println("B입니다.");
```

실행 결과

학점은 B입니다.

문제:

1

자바에서 클래스 바깥에 선언될 수 있는 것은?

① 변수 ② 메소드 ③ 둘 다 ④ 없다

2

main() 메소드는 어떻게 선언하는가?

틀린 코드 찾기: 클래스와 main()

어디가 틀렸나

①

```
int count = 0;

public class Main {

    public static void
    main(String[] args) {
    }
}
```

②

```
public class Main {

    public void
    main(String[] args) {
    }
}
```

③

```
public class Main {

    public static void
    main(String[] args) { }

    public static void
    main(String[] args) { }
}
```

식별자 이름 규칙

식별자는 클래스 · 변수 · 상수 · 메소드에 붙이는 이름이다.

쓸 수 없다

- 특수문자 % & @ 와 공백
- 첫 글자 숫자
- 키워드 if while class
- true false null

쓸 수 있다

- ✓ \$ 와 밑줄 _
- ✓ 한글 이름
- ✓ 길이 제한 없음

대소문자를 구별한다. barChart 와 barchart 는 다른 이름이다.

식별자 예 8개



규칙 위반

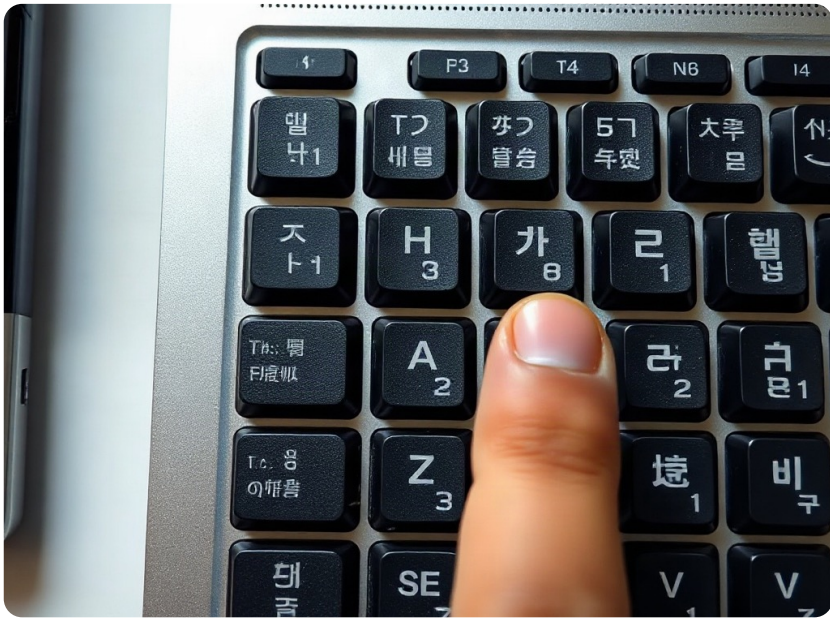
```
int 3Chapter;  
class if { }  
char false;  
class %calc { }
```



올바른 이름

```
int student_ID;  
void $func() { }  
class Monster3 { }  
int 가격;
```

int 가격; 은 진짜로 컴파일된다



명품 자바 에센셜 p.57~58

```
int 가격 = 3000;
```

- 자바는 문자를 유니코드로 다룬다
- char 는 2바이트, C 언어는 1바이트다
- 실무에서 한글 변수명은 쓰지 않는다

자바 안에서는 한글이 멀쩡하다.
바꾼 것은 화면으로 나가는 통로였다.

자바의 데이터 타입

데이터 타입은 자바에서 다룰 수 있는 데이터의 종류

기본 타입 8개

boolean

char

byte

short

int

long

float

double



값의 크기에 맞는 그릇을 고른다

기본 타입 8개

분류	타입	크기	저장하는 값
논리	<code>boolean</code>	1바이트	true, false
문자	<code>char</code>	2바이트	유니코드 문자 하나
정수	<code>Byte , short , int , long</code>	1·2·4·8바이트	소수점 없는 수
실수	<code>float , double</code>	4·8바이트	소수점 있는 수

레퍼런스 타입

[]

배열

int[] 로 만든
배열을 가리킨다



클래스

new 로 만든
객체를 가리킨다



인터페이스

구현한 객체를
가리킨다

C/C++ 포인터와 비슷하지만 실제 주소 값은 아니다.

지금은 주소 값 정도로 두고 배열과 클래스에서 다시 본다.

정수형 4종

타입	크기	범위	어디에 쓰나
byte	1바이트	-128~127	나이 · 센서 값
short	2바이트	-32,768~32,767	잘 안 쓴다
int	4바이트	약 -21억~21억	학번 · 점수 · 개수
long	8바이트	약 ± 922 경	세계 인구 · 밀리초 시각

```
byte age = 20;  
int studentId = 20261234;  
long population = 8100000000L;
```

정수는 보통 int 로 시작한다

실수형 2종

소수점 아래 자릿수가 필요하면 double 을 씁니다.

float

4바이트 · 유효 숫자 약 7자리

`float f = 3.14159265f;`

→ 3.1415927

`1.0f / 3`

→ 0.33333334

double

실수의 기본

8바이트 · 유효 숫자 약 15자리

`double d = 3.14159265;`

→ 3.14159265

`1.0 / 3`

→ 0.3333333333333333

char 와 boolean

char

문자 하나를 담는다
2바이트 유니코드

```
char grade = 'A';
```

boolean

true 와 false
값이 둘뿐이다

```
boolean present = true;
```

String

여러 글자를 담는다
기본 타입이 아니다

```
String name = "김철수";
```

'A' 는 char

"A" 는 String

타입 고르기 연습

어떤 타입으로 선언하나

1 나이

20

2 학번

20261234

3 세계 인구

8100000000

4 키

175.5

5 학점 한 글자

A

6 출석 여부

참

7 이름

김철수

타입 선언 모음 코드

변수 선언

```
1 byte age = 20;
2 int studentId = 20261234;
3 long population = 81000000000L;
4 double height = 175.5;
5 char grade = 'A';
6 boolean present = true;
7 String name = "김철수";
```

화면 출력

```
8 System.out.println("나이: " + age);
9 System.out.println("학번: " + studentId);
10 System.out.println("세계 인구: " + population);
11 System.out.println("키: " + height);
12 System.out.println("학점: " + grade);
13 System.out.println("출석: " + present);
14 System.out.println("이름: " + name);
```

자기 나이.학번.이름으로 바꾼다

타입 선언 모음 실행 결과

```
Output
나이: 20
학번: 20261234
세계 인구: 8100000000
키: 175.5
학점: A
출석: true
이름: 김철수

=== Code Execution Successful ===
```

8100000000 뒤의 L 을 빼면 컴파일 오류

문자열과 + 연결

String 은 기본 타입이 아니라 클래스다

```
String toolName = "JDK";
```

```
toolName + 1.8 → "JDK1.8"
```

```
(" + 3 + "," + 5 + ") → "(3,5)"
```

문자열 연결 결과 맞추기

출력을 먼저 종이에 적고 Run 을 누른다.

```
1 System.out.println("(" + 3 + "," + 5 + ");");  
2 System.out.println(1 + 2 + "3");  
3 System.out.println("1" + 2 + 3);  
4 System.out.println("헬로" + 2030);
```

문자열 연결 실행 결과

```
Output
```

```
(3,5)  
33  
123  
헬로2030  
  
=== Code Execution Successful ===
```

+ 는 왼쪽부터 계산한다

변수 선언과 변수 공간

```
int price;
```

price

int 는 4바이트

3.5 와 같은 실수는
저장할 수 없다

선언하면 타입 크기만큼 공간이 잡힌다.

선언 예

```
int radius;
```

```
double weight = 75.56;
```

```
char c1, c2, c3 = 'a';
```

초기화된 변수는 c3 하나다

정수 리터럴

리터럴은 프로그램에 직접 표현한 값이다.

15

10진수 표기다.

015

0 으로 시작하면 8진수 표기다.

0x15

0x 로 시작하면 16진수 표기다.

0b0101

0b 로 시작하면 2진수 표기다.

정수 리터럴은 int 로 컴파일된다. long 값은 24L 처럼 뒤에 L 을 붙인다.

실수 리터럴과 문자 리터럴

실수 리터럴

12. 12.0 .1234
0.1234 1234E-4

소수점이 있으면 double 이다

float 은 0.1234f 로 쓴다

f 를 빼면 컴파일 오류가 난다

문자 리터럴

'a' 'W' '가' '3' '글'

작은따옴표 안에 한 글자만 넣는다

char c = '₩uae00'; 도 같은 글자

'글' 처럼 한글도 유니코드다

특수문자 리터럴

역슬래시로 시작하는 문자는 화면에 보이지 않는 동작을 나타낸다.

문자	의미
'\b'	백스페이스
'\t'	탭
'\n'	라인피드
'\f'	폼피드

문자	의미
'\r'	캐리지 리턴
'\"'	이중 인용부호
'\''	단일 인용부호
'\\'	백슬래시

논리 리터럴과 null, 문자열



논리 리터럴

true 와 false 둘뿐

```
boolean b = 10 > 0;  
true
```

```
boolean c = 1; 오류
```



null 리터럴

레퍼런스에만 쓴다

```
String str = null;
```

기본 타입에는
쓸 수 없다



문자열 리터럴

큰따옴표로 묶는다

```
"Good" "자바" "3.19"
```

"3.19" 는 숫자가
아니라 문자열

진법 리터럴 출력해 보기

```
int a = 15;    // 10진수
int b = 015;   // 8진수
int c = 0x15;  // 16진수
int d = 0b0101; // 2진수
System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
```

먼저 할 일

- 코드 작성
- 8진수부터 계산
- 계산 끝나면 Run

진법 리터럴 실행 결과

```
Output
```

```
15
13
21
5

=== Code Execution Successful ===
```

15, 13, 21, 5. 표기가 달라도 출력은 모두 10진수다.

틀린 코드 찾기: 리터럴

어디가 틀렸나

- ① `boolean done = 1;`
- ② `float f = 0.1234;`
- ③ `int m = 019;`
- ④ `char c = "a";`

상수

```
final double PI = 3.141592;
```

|
final 키워드

|
상수 이름

|
초깃값

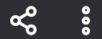
- 초기화한 뒤에는 값을 바꿀 수 없다. 바꾸려 하면 컴파일 오류가 난다.
- static final 로 선언하는 형태가 더 바람직하다. static 은 5장에서 다룬다.

예제 2-2 CircleArea 코드

```
final double PI = 3.14;  
  
double radius = 10.2;  
  
double circleArea = radius * radius * PI;  
  
System.out.print("반지름 " + radius + ", ");  
  
System.out.println("원의면적 = " + circleArea);
```

CircleArea 실행 결과

Output

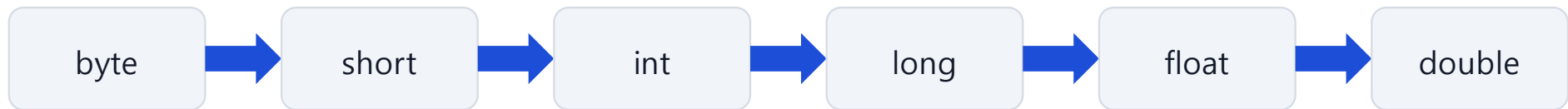


반지름 10.2, 원의면적 = 326.68559999999997

=== Code Execution Successful ===

326.6856 이 아닌 이유는 잠시 뒤에

자동 타입 변환



왼쪽에서 오른쪽으로는 값이 손실되지 않는다.

```
long m = 25;
```

int 가 long 으로 바뀐다

```
double d = 3.14 * 10;
```

10 이 10.0 으로 바뀐다

작은 타입에서 큰 타입으로는 컴파일러가 알아서 바꾼다.

char 도 산술 연산에서 int 로 올라간다. 'A' + 1 은 66, 'c' + 1 은 100 (int)

강제 타입 변환

```
int n = 300;  
byte b = (byte)n;
```

b 는 44

byte 범위를 넘어 잘린다

```
int n = (int)1.9;
```

n 은 1

소수부를 버린다

큰 타입에서 작은 타입으로는 손실이 생기므로 (타입) 을 직접 적는다

내 키를 m 로 바꿔 보기

```
int cm = 175;  
  
System.out.println(cm / 100);  
  
System.out.println(cm / 100.0);  
  
System.out.println( (double) cm / 100);
```

175 를 자기 키로 바꾼다

먼저 예측

결과가 어떻게 나올지
예측해 보고 실행한다.

내 키 변환 실행 결과

```
Output
```

```
1  
1.75  
1.75  
  
=== Code Execution Successful ===
```

한쪽만 실수여도 결과는 실수가 된다.

문제: 타입과 리터럴

1

다음 중 자바의 기본 타입은?

`byte` `Double` `bool` `String`

2

365 인 YEAR, 30 인 MONTH, 7 인 WEEK 를 int 상수로 선언하라

3

기본 타입 리터럴이 아닌 것은?

`'가'` `012` `0X1A` `"a"` `'\b'` `true` `0.23E-4`

326.68559999999997 의 정체

컴퓨터는 실수를 2진수로 저장한다.
그래서 3.14 도 딱 떨어지지 않는다.

1/3 을 소수로 쓰면

0.333... 끝나지 않는다

10진수 0.1 도 2진수로는

0.0001100110011... 끝나지 않는다

IEEE 754 로 정해진 동작이다. 파이썬·자바스크립트·C 도 결과가 같다.



바늘은 눈금 사이에 멈춘다

실제로 확인하기

```
Output
0.30000000000000004
false
326.68559999999997

=== Code Execution Successful ===
```

$0.1 + 0.2$, $0.1 + 0.2 == 0.3$, $3.14 * 10.2 * 10.2$ 의 결과. 종이 계산과 컴퓨터 계산이 다르다

그래서 어떻게 하나

1

원 단위 정수

```
int won = 100 + 200;
```

출력 300

2

BigDecimal

```
new BigDecimal("0.1")  
.add(new BigDecimal("0.2"))
```

출력 0.3

돈 계산에 **double** 을 쓰지 않습니다.

1교시 실습 살펴보기

헬로2030

"헬로"+n 은 n 이 문자열로 바뀌어 붙는다

자기소개

println 한 번이 한 줄. 줄바꿈은 \n

타입 선언 모음

21억을 넘는 정수 리터럴은 L 을 붙인다

문자열 연결

1 + 2 + "3" 은 33, "1" + 2 + 3 은 123. 왼쪽부터

진법 리터럴

015 는 13. 앞의 0 은 8진수 표기다

CircleArea

final 은 재대입 불가. 뒷자리 ...97 은 오차다

내 키

cm/100 은 1, cm/100.0 은 1.75. 한쪽만 실수면 실수 나눗셈

2교시

키 입력과 연산자

키보드에서 값을 읽는다. 연산자로 계산한다.

System.in과 Scanner

산술 · 비트 · 시프트

비교 · 논리 · 삼항

예제 2-3 TypeConversion 코드

여덟 줄의 출력을 먼저 적는다.

```
1  byte b = 127;
2  int i = 100;
3  System.out.println(b+i);
4  System.out.println(10/4);
5  System.out.println(10.0/4);
6  System.out.println((char)0x12340041);
7  System.out.println((byte)(b+i));
8  System.out.println((int)2.9 + 1.8);
9  System.out.println((int)(2.9 + 1.8));
10 System.out.println((int)2.9 + (int)1.8);
```

이 열 줄은 public class Main 의 main() 안에 넣는다.

TypeConversion 실행 결과

Output

227

2

2.5

A

-29

3.8

4

3

=== Code Execution Successful ===

괄호 위치에 따라 마지막 세 줄이 3.8, 4, 3 으로 갈린다.

타입 변환 결과 풀이

`b + i` **227**

byte 가 int 로 자동 변환

`10 / 4` **2**

정수끼리 나누면 정수

`10.0 / 4` **2.5**

한쪽이 실수면 실수 나눗셈

`(char)0x12340041` **'A'**

하위 2바이트 0x0041 만 남는다

`(byte)(b + i)` **-29**

227 의 하위 8비트 0xE3

`(int)2.9 + 1.8` **3.8**

2.9 만 잘려 2 + 1.8

`(int)(2.9 + 1.8)` **4**

4.7 을 통째로 잘라 4

`(int)2.9 + (int)1.8` **3**

각각 잘려 2 + 1

괄호가 캐스팅 범위를 정한다.

System.in



System.in

java.lang.System 클래스의 필드

- 키보드 장치와 직접 연결되는 표준 입력 스트림 객체
- 키 값을 바이트 정보로 바꾸어 제공하는 저수준 스트림
- 이것만 쓰면 바이트를 문자·숫자로 바꾸는 일을 직접 해야 한다

그래서 바이트를 타입별로 바꿔 주는 도우미 클래스가 필요합니다.

Scanner 클래스

키 입력을 문자·정수·실수·문자열 등 원하는 타입으로 바꿔 주는 고수준 스트림 클래스입니다.

System.in

저수준 · 바이트를 줍니다.

Scanner

고수준 · 타입으로 바꿔 줍니다.

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

Scanner 클래스

scanner 객체

System.in 을 넘긴다

키보드, System.in, Scanner 의 관계



바이트로 들어와 타입으로 바뀐다.

import 문

```
import java.util.Scanner;
```

- 맨 앞줄에 둔다
- Scanner 클래스의 경로명이 java.util.Scanner 임을 알려 준다
- 없으면 컴파일러가 Scanner 클래스를 찾지 못한다

템플릿 3줄째에 이미 있다

토큰 단위로 읽는다

공백.탭.개행으로 구분되는 토큰 단위로 읽는다

키보드 입력

Kim

Seoul

20

65.1

true

```
String name = scanner.next(); // "Kim"  
String city = scanner.next(); // "Seoul"  
int age = scanner.nextInt(); // 20  
double weight = scanner.nextDouble(); // 65.1  
boolean single = scanner.nextBoolean(); // true
```

구분 문자 ' ' · '\t' · '\r' · '\n'

Scanner 클래스의 주요 메소드

메소드	설명
String next()	다음 토큰을 문자열로 리턴
byte nextByte()	다음 토큰을 byte 타입으로 변환하여 리턴
int nextInt()	다음 토큰을 int 타입으로 변환하여 리턴
long nextLong()	다음 토큰을 long 타입으로 변환하여 리턴
double nextDouble()	다음 토큰을 double 타입으로 변환하여 리턴
boolean nextBoolean()	다음 토큰을 boolean 으로 변환하여 리턴
String nextLine()	'\n' 을 포함하는 한 라인을 읽고 '\n' 을 버린 나머지 리턴
void close()	Scanner 사용 종료

Programiz 에서 입력하는 방법

1

Run 을 누른다

Run 버튼을 누른다.
프로그램이 시작되고
입력을 기다린다.

2

Output 창 클릭

오른쪽 Output 창을
마우스로 클릭한다.
커서가 깜빡인다.

3

값 입력 후 Enter

값을 입력하고
Enter 를 누른다.
순서는 코드와 같게

입력한 글자는 Output 에 그대로 보인다.

예: 정수를 입력하세요:4000

예제 2-4 ScannerEx 코드

```
1 System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));
2 System.out.println("이름, 도시, 나이, 체중, 독신 여부를 입력하세요.");
3 Scanner scanner = new Scanner(System.in);
4 String name = scanner.next();
5 System.out.println("당신의 이름은 " + name + "입니다.");
6 String city = scanner.next();
7 System.out.println("당신이 사는 도시는 " + city + "입니다.");
8 int age = scanner.nextInt();
9 System.out.println("당신의 나이는 " + age + "살입니다.");
10 double weight = scanner.nextDouble();
11 System.out.println("당신의 체중은 " + weight + "kg 입니다.");
12 boolean single = scanner.nextBoolean();
13 System.out.println("당신은 독신 여부는 " + single + "입니다.");
14 scanner.close();
```

ScannerEx 실행 결과

```
Output
이름, 도시, 나이, 체중, 독신 여부를 입력하세요.
Kim Seoul 20 65.1 true
당신의 이름은 Kim입니다.
당신이 사는 도시는 Seoul입니다.
당신의 나이는 20살입니다.
당신의 체중은 65.1kg 입니다.
당신은 독신 여부는 true입니다.

=== Code Execution Successful ===
```

둘째 줄은 내가 친 것이 그대로 보인다

Scanner 닫기

```
scanner.close();
```



close() 가 없으면

이클립스는 경고를 낸다.
실행에는 문제가 없다.



프로그램이 끝나면

Scanner 가 자동으로 닫힌다.
그래도 close() 는 습관으로 붙인다.

내 학생증 출력해 보기

학번, 이름, 학과를 한 줄로 입력받아 학생증처럼 출력

```
System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));

Scanner scanner = new Scanner(System.in);
System.out.print("학번 이름 학과를 입력하세요: ");
int id = scanner.nextInt();
String name = scanner.next();
String dept = scanner.next();
System.out.println("==== 학생증 =====");
System.out.println("학번: " + id);
System.out.println("이름: " + name);
System.out.println("학과: " + dept);
scanner.close();
```

입력 예

20261234 김철수 인공지능학과

자기 학번·이름으로

학생증 실행 결과

```
Output
학번 이름 학과를 입력하세요: 20261234 김철수 인공지능학과
===== 학생증 =====
학번: 20261234
이름: 김철수
학과: 인공지능학과

=== Code Execution Successful ===
```

입력한 세 토큰이 순서대로 학번·이름·학과 변수에 들어간다.

틀린 코드 찾기: Scanner

어디가 틀렸나

1

```
// 파일 첫 줄  
public class Main {  
    ...  
    Scanner scanner =  
        new Scanner(System.in);
```

2

```
import java.util.Scanner;  
public class Main {  
    ...  
    Scanner scanner =  
        new Scanner();
```

3

```
...  
Scanner scanner =  
    new Scanner(System.in);  
int age = scanner.nextInt();  
// 입력: 스무살
```

실행 중 오류 화면

Output

Age: 스무살

ERROR!

```
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
    at Main.main(Main.java:10)
```

=== Code Exited With Errors ===

③ 의 실행 화면. 컴파일 오류는 Run 직후 javac 가 잡고,
실행 오류는 프로그램이 돌다가 멈춘다

nextInt() 다음의 nextLine()



```
int age = sc.nextInt();
```

```
String name = sc.nextLine();
```

```
// name 에 빈 문자열이 들어온다
```

nextInt() 는 숫자 토큰만 가져간다
Enter 는 버퍼에 남는다

해결 방법

nextLine() 한 번 더

next() 로 통일

자바의 연산 종류와 연산자

연산의 종류	연산자
증감	++ --
산술	+ - * / %
시프트	>> << >>>
비교	> < >= <= == !=
비트	& ^ ~
논리	&& ! ^
조건	? :
대입	= *= /= += -= &= ^= = <<= >>= >>>=

식과 연산자

a + 5

▲ 피연산자 ▲ 연산자 ▲ 피연산자

산술 연산식

a > b

▲ 피연산자 ▲ 연산자 ▲ 피연산자

비교 연산식

x = y * 2

▲ 피연산자 ▲ 연산자 ▲ 피연산자

대입과 산술이 함께 있는 식

식을 계산하여 결과를 얻는 과정이 연산이다.

산술 연산자

+

-

*

/

%

/ 는 몫, % 는 나머지

정수끼리 나누면 소수점이 사라진다

예

$$69 / 10 \rightarrow 6$$

$$69 \% 10 \rightarrow 9$$

나머지 연산의 쓰임



자릿수 분리

69 / 10 은 6
69 % 10 은 9

10 으로 나눈 몫과 나머지



홀짝 판별

```
int r = n % 2;  
1 이면 홀수  
0 이면 짝수
```

2 로 나눈 나머지로 판별



배수 판별

```
int s = n % 3;  
0 이면 3의 배수
```

3 으로 나눈 나머지가 0

나눗셈 네 줄 비교해 보기

출력값을 먼저 적고 Run 을 누른다

- 1 `System.out.println(10 / 4);`
- 2 `System.out.println(10 / 4.0);`
- 3 `System.out.println(10 % 4);`
- 4 `System.out.println(10.0 / 4);`

예상 출력

나눗셈 실행 결과

```
Output
2
2.5
2
2.5

=== Code Execution Successful ===
```

하나만 실수여도 실수 나눗셈

예제 2-5 초를 시분초로 코드

```
1  Scanner scanner = new Scanner(System.in);
2  System.out.print("정수를 입력하세요:");
3  int time = scanner.nextInt();

4  int second = time % 60;
5  int minute = (time / 60) % 60;
6  int hour = (time / 60) / 60;

7  System.out.print(time + "초는 ");
8  System.out.print(hour + "시간, ");
9  System.out.print(minute + "분, ");
10 System.out.println(second + "초입니다.");
11 scanner.close();
```

초를 시분초로 실행 결과

```
Output
정수를 입력하세요:4000
4000초는 1시간, 6분, 40초입니다.

=== Code Execution Successful ===
```

입력값 4000

비트 연산자

연산자	이름	의미
$a \& b$	AND	두 비트가 모두 1일 때만 1
$a b$	OR	두 비트가 모두 0일 때만 0
$a \wedge b$	XOR	서로 다르면 1
$\sim a$	NOT	1 을 0 으로 0 을 1 로

&

=

0 1 1 0 1 0 1 0

1 1 0 0 1 1 0 1

0 1 0 0 1 0 0 0

|

=

0 1 1 0 1 0 1 0

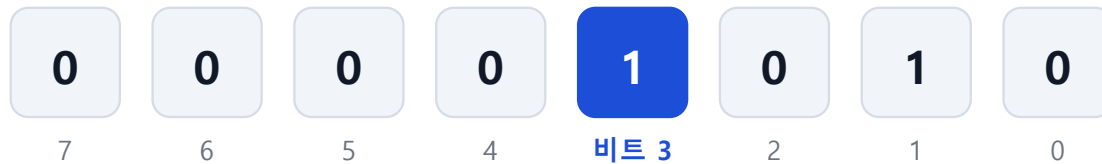
1 1 0 0 1 1 0 1

1 1 1 0 1 1 1 1

비트 검사의 쓰임

```
byte flag = 0b00001010;  
if ((flag & 0b00001000) == 0) System.out.print("온도는 0도 이하");  
else System.out.print("온도는 0도 이상");
```

flag = 0b00001010



결과 · 온도는 0도 이상

비트 3 을 제외한 자리는
AND 결과가 모두 0

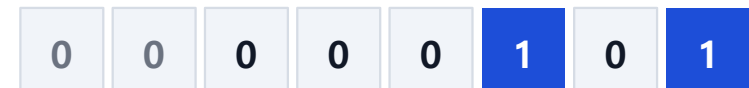
시프트 연산자

연산자	설명
<code>a >> b</code>	오른쪽으로 b 번, 빈자리는 최상위 비트
<code>a >>> b</code>	오른쪽으로 b 번, 빈자리는 0
<code>a << b</code>	왼쪽으로 b 번, 빈자리는 0

byte a = 20



>> 2



(byte)(a >> 2) = 5

한 칸 시프트는 $\times 2$ 또는 $\div 2$. float·double·boolean 은 불가

>>> 는 자바가 만든 연산자

> > >

C/C++ 에는 없다. 자바가 만들었다.

- C/C++ 는 unsigned int 로 부호 없는 정수를 구분했다
- 자바는 설계 단계에서 부호 없는 정수 타입을 없앴다
- **타입으로 구분할 수 없게 되자 연산자를 하나 더 만들었다**

<<< 는 없다. 왼쪽 시프트는 빈 자리를 항상 0 으로 채운다.

비교 연산자

연산자	결과가 true 가 되는 경우
$a < b$	a 가 b 보다 작으면 true
$a > b$	a 가 b 보다 크면 true
$a \leq b$	a 가 b 보다 작거나 같으면 true
$a \geq b$	a 가 b 보다 크거나 같으면 true
$a == b$	a 와 b 가 같으면 true
$a != b$	a 와 b 가 같지 않으면 true

논리 연산자

연산자	설명
<code>!a</code>	NOT, a 가 true 면 false
<code>a ^ b</code>	XOR, 같으면 false
<code>a b</code>	OR, 둘 다 false 일 때만 false
<code>a && b</code>	AND, 둘 다 true 일 때만 true

복합 조건식

20대

$(age \geq 20) \ \&\& \ (age < 30)$

대문자

$(c \geq 'A') \ \&\& \ (c \leq 'Z')$

사각형 안

$(x \geq 0) \ \&\& \ (y \geq 0) \ \&\& \ (x \leq 50) \ \&\& \ (y \leq 50)$

20 <= age < 30 은 오류

두 식을 && 로 연결한다

예제 2-7 LogicalOperator 코드

출력 10줄을 먼저 적는다

```
System.out.println(3 <= 2);
System.out.println(3 >= 2);
System.out.println(-1 < 0);
System.out.println(3.45 <= 2);
System.out.println(3 == 2);
System.out.println(3 != 2);
System.out.println(!(3 != 2));
System.out.println((3 > 2) && (3 > 4));
System.out.println((3 != 2) || (-1 > 0));
System.out.println((3 != 2) ^ (-1 > 0));
```

LogicalOperator 실행 결과

Output



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

```
=== Code Execution Successful ===
```

틀린 코드 찾기: 조건식

어디가 틀렸나

- ① `if (20 <= age < 30) { ... }`
- ② `if (age = 20) { ... }`
- ③ `if (age == 20 || 30) { ... }`

대입 연산자

연산자	의미
a = b	b 를 a 에 대입
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a %= b	a = a % b

연산자	의미
a &= b	a = a & b
a ^= b	a = a ^ b
a = b	a = a b
a <<= b	a = a << b
a >>= b	a = a >> b
a >>>= b	a = a >>> b

앞에서 배운 연산자에 = 을 붙여 줄여 쓴 형태다.

증감 연산자

연산자	의미
<code>a++</code>	먼저 사용한 후 1 증가
<code>++a</code>	먼저 1 증가한 후 사용
<code>a--</code>	먼저 사용한 후 1 감소
<code>--a</code>	먼저 1 감소한 후 사용

```
int a, b = 4; a = b++; // a=4, b=5
```

```
int a, b = 4; a = ++b; // a=5, b=5
```

조건 연산자

condition ? opr2 : opr3

condition 이 참이면 앞의 opr2

거짓이면 뒤의 opr3

```
int x = 5, y = 3;  
int big = (x > y) ? x : y;  
// big 은 5
```

```
int a = 3, b = 5;  
System.out.println("두 수의 차는 "  
    + ((a > b) ? (a - b) : (b - a)));  
// 두 수의 차는 2
```

삼항 연산자로 성인 판정해 보기

```
Scanner scanner = new Scanner(System.in);  
System.out.print("나이: ");  
int age = scanner.nextInt();  
System.out.println((age >= 19) ? "성인" : "미성년");  
scanner.close();
```

성인 판정 실행 결과

```
Output
나이: 19
성인

=== Code Execution Successful ===
```

18 이면?

문제: 연산자

1

int x = 2, y = 10, z = 0;
z = x++*2 + --y - 5 + x*(y%2); 실행 후 z 는?

2

System.out.println(8 >> 1); 출력은?

3

int opr = 4; System.out.println(opr++); 출력은?

2교시 실습 살펴보기

TypeConversion (int)2.9+1.8 은 3.8, (int)(2.9+1.8) 은 4. 괄호 안만 캐스팅

ScannerEx next() 는 공백 앞까지 한 토큰. 순서와 타입을 맞춘다

학생증 학번은 nextInt(), 이름은 next(). 공백이 토큰을 가른다

나눗셈 네 줄 10 / 4 는 몫 2, 10 % 4 는 나머지 2

초를 시분초로 (time / 60) % 60 이 분. 몫을 다시 60 으로 나눈 나머지

LogicalOperator 'a' > 'b' 는 97 과 98 의 비교라 false

성인 판정 (age >= 19) ? "성인" : "미성년" 은 값. println 안에 넣는다

3교시

조건문

조건으로 실행할 문장을 고른다. 끝에는 가위바위보.

if와 if-else

다중 · 중첩 if-else

switch와 break

조건식은 불린 값



논리 타입 변수 하나만 써도 조건식이 된다.

조건문은 조건의 참·거짓에 따라 다른 작업을 한다.

조건식은 비교·논리 연산이 섞인 식이다.
결과는 언제나 불린 값.

자바의 조건문은 if-else 문과 switch 문이다.

단순 if 문

구문

```
if(조건식) {  
    ... 실행 문장 ...  
}
```

동작

- 조건식이 참이면 실행 문장을 실행한다
- 거짓이면 실행 문장을 건너뛰고 if 문을 벗어난다
- 실행 문장이 여러 개면 중괄호로 묶는다

한 문장이면 중괄호를 생략할 수 있습니다. 이 수업에서는 항상 씁니다.

if 문 코드 사례

문장이 둘이면 중괄호

나머지 연산

```
if (n % 2 == 0) {  
    System.out.print(n);  
    System.out.println("은 짝수입니다.");  
}
```

문장이 하나면 중괄호 생략

비교 연산 · 논리 연산

```
if (score >= 80 && score <= 89)  
    System.out.println("학점은 B입니다.");
```

if-else 문

구문

```
if(조건식) {  
    실행 문장1  
} else {  
    실행 문장2  
}
```

사례

```
if (score >= 90)  
    System.out.println("합격입니다.");  
else  
    System.out.println("불합격입니다.");
```

true 면 문장1, false 면 문장2 를 실행한 뒤 벗어난다

예제 2-10 Twenties 코드

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("나이를 입력하시오: ");
3 int age = scanner.nextInt();

4 if ((age >= 20) && (age < 30)) {

5     System.out.print("20대입니다. ");
6     System.out.println("20대라서 행복합니다!");
7 }
8 else
9     System.out.println("20대가 아닙니다.");
10 scanner.close();
```

main() 안에 넣는 10줄. 입력 23

Twenties 실행 결과

```
Output
나이를 입력하시오: 23
20대입니다. 20대라서 행복합니다!

=== Code Execution Successful ===
```

실제 자기 나이로 다시 해보기

우산 챙기기를 코드로

비 올 확률이 60% 이상이면 우산을 챙긴다.

```
Scanner scanner = new Scanner(System.in);
System.out.print("오늘 비 올 확률(%): ");
int rain = scanner.nextInt();

if (rain >= 60) {
    System.out.println("우산을 챙긴다");
}

else {
    System.out.println("우산은 두고 간다");
}
scanner.close();
```

main() 안에 넣는 10줄. 입력 70, 이어서 30

우산 챙기기 실행 결과

Output

오늘 비 올 확률(%): 70

우산을 챙긴다

=== Code Execution Successful ===

60 을 넣으면? > 로 바꾸면?

다중 if-else 문

```
if (조건식1) {  
    실행 문장1  
} else if (조건식2) {  
    실행 문장2  
} else if (조건식m) {  
    ...  
} else {  
    실행 문장n  
}
```

위에서부터 조건식이 참인 것을
찾아 그 실행 문장을 실행한다

실행 후 다중 if-else 전체를 벗
어난다

다중 if-else 코드 사례

```
if (score >= 90) grade = 'A'; // 90~100  
else if (score >= 80) grade = 'B'; // 80~89  
else if (score >= 70) grade = 'C'; // 70~79  
else if (score >= 60) grade = 'D'; // 60~69  
else grade = 'F'; // 60 미만
```

위에서 걸리는 순서

90 이상은 첫 줄에서
이미 걸러진다.
그래서 다음 조건은
80 이상만 본다.

score = 85 → B

두 번째 조건이 `score >= 80` 이면 되는 이유는 90 이상이 위에서 이미 걸렸기 때문이다

예제 2-11 Grading 코드

```
1 char grade;
2 Scanner scanner = new Scanner(System.in);
3 System.out.print("점수를 입력하세요(0~100): ");
4 int score = scanner.nextInt();
5 if(score >= 90)
6     grade = 'A';
7 else if(score >= 80)
8     grade = 'B';
9 else if(score >= 70)
10    grade = 'C';
11 else if(score >= 60)
12    grade = 'D';
13 else
14    grade = 'F';
15 System.out.println("학점은 " + grade + "입니다.");
16 scanner.close();
```

코드 읽는 순서

- 1행 선언만 한다
- 4행 점수를 읽는다
- 5~14행 학점을 정한다
- 15행 결과를 출력한다

grade 는 선언만 해 두고
조건에서 값을 넣는다.

main() 안에 넣는 16줄. 입력 89

Grading 실행 결과

Output



점수를 입력하세요(0~100): 89

학점은 B입니다.

=== Code Execution Successful ===

점수를 입력하세요(0~100): 89 → 학점은 B입니다.

순서를 뒤집으면

```
if(score >= 60) grade = 'D';  
else if(score >= 70) grade = 'C';  
else if(score >= 80) grade = 'B';  
else if(score >= 90) grade = 'A';  
else grade = 'F';
```

질문

89 를 넣으면?

예제 2-11 은 B 였다.

예제 2-12 NestedIf 코드

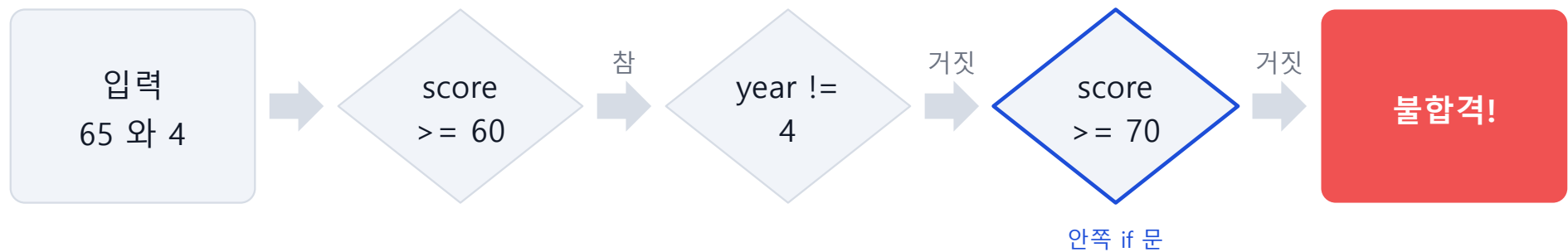
```
1  import java.io.PrintStream;
2  import java.nio.charset.StandardCharsets;
3  import java.util.Scanner;
4
5
6  public class Main {
7      public static void main(String[] args) {
8          System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));
9
10         Scanner scanner = new Scanner(System.in);
11         System.out.print("점수와 학년을 입력하세요 > > ");
12         int score = scanner.nextInt();
13         int year = scanner.nextInt();
14         if (score >= 60) {
15             if (year != 4) System.out.println("합격!");
16             else if (score >= 70) System.out.println("합격!");
17             else System.out.println("불합격!");
18         }
19         else System.out.println("불합격!");
20     }
21 }
```

바깥 조건: 60점 이상

안쪽 조건: 4학년은 70점 이상

입력 65. NestedIf. 60점 이상 합격, 4학년은 70점 이상

65점 4학년의 경로



if 문이나 else 문의 실행 문장 안에 if 문을 내포할 수 있다.

if-else 를 조건 연산자로

if-else 문

```
if (a > b)
    abs = a - b;
else
    abs = b - a;
```



조건 연산자

```
abs = (a > b) ? a - b : b - a;
```

(조건) ? 참일 때 : 거짓일 때

간단한 if-else 는 조건 연산자로 줄일 수 있다.

틀린 코드 찾기: if 문

어디가 틀렸나

①

```
if (score >= 60)
    System.out.println("합격");
    System.out.println("축하");
```

②

```
if (score >= 60);
{
    System.out.println("합격");
}
```

③

```
if (score >= 60)
    if (score >= 90)
        System.out.println("A");
else
    System.out.println("탈락");
```

switch 문

구문

```
switch (식) {  
    case 값1:  
        실행 문장1;  
        break;  
    case 값2:  
        실행 문장2;  
        break;  
    default:  
        실행 문장n;  
}
```

break 를 만나면 switch 를 빠져나온다.

사례. grade 가 'B' 이면 출력은 좋아요.

```
char grade = 'B';  
switch (grade) {  
    case 'A':  
        System.out.println("축하합니다.");  
        System.out.println("잘했습니다.");  
        break;  
    case 'B':  
        System.out.println("좋아요.");  
        break;  
    case 'C':  
        System.out.println("노력하세요.");  
        break;  
    default:  
        System.out.println("탈락입니다!");  
}
```

switch 문에서 break 의 역할

```
char grade = 'A';
switch (grade) {
    case 'A':
        System.out.println("90~100점입니다.");
        // break; 가 없다
    case 'B':
        System.out.println("80~89점입니다.");
        break;
}
```

실행 결과

90~100점입니다.
80~89점입니다.

break 가 없으면 아래 case 의 실행 문장을 break 를 만날 때까지 계속 실행한다

case 문의 값

허용

case 1:

case '+':

case "예":

문자열은 JDK 1.7 부터 됩니다

오류

case a:

case a > 3:

case a == 1:

변수 · 식 · 조건식은 안 됩니다

case 값은 리터럴만 쓸 수 있습니다. 변수나 식은 안 됩니다.

예제 2-13 Season 코드

```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("월(1~12)을 입력하시오: ");
3 int month = scanner.nextInt();
4 switch(month) {
5     case 3:
6     case 4:
7     case 5:
8         System.out.println("봄입니다.");
9         break;
10    case 6: case 7: case 8:
11        System.out.println("여름입니다.");
12        break;
13    case 9: case 10: case 11:
14        System.out.println("가을입니다.");
15        break;
16    case 12: case 1: case 2:
17        System.out.println("겨울입니다."); break;
18    default:
19        System.out.println("잘못된 입력입니다.");
20 }
21 scanner.close();
```

Season 실행 결과

```
Output
월(1~12)을 입력하시오: 3
봄입니다.

=== Code Execution Successful ===
```

13 을 넣으면 default 로 갑니다.

break 를 지우면

case 3 의 break; 를 지운 코드

```
case 3:
    System.out.println("봄입니다.");

    // break 를 지움

case 6:
    System.out.println("여름입니다.");
    break;
```

**3 을 넣으면
몇 줄이 나오나**

case 6 의 break; 는
그대로 두었다.

학점 메시지, if-else 로

A·B 는 Excellent, C·D 는 Good, F 는 Bye

```
String grade = scanner.next();  
if (grade.equals("A") || grade.equals("B"))  
    System.out.println("Excellent");  
else if (grade.equals("C") || grade.equals("D"))  
    System.out.println("Good");  
else if (grade.equals("F"))  
    System.out.println("Bye");
```

문자열 비교는 equals()

학점 메시지, switch 로

같은 문제, A·B 는 Excellent · C·D 는 Good · F 는 Bye.

switch 버전

```
Scanner scanner = new Scanner(System.in);
System.out.print("학점을 입력하세요: ");
String grade = scanner.next();
switch (grade) {
    case "A": case "B":
        System.out.println("Excellent"); break;
    case "C": case "D":
        System.out.println("Good"); break;
    case "F":
        System.out.println("Bye"); break;
    default:
        System.out.println("잘못된 입력입니다.");
}
scanner.close();
```

main() 안에 넣는 14줄. 입력 C

switch 버전 실행 결과

```
Output
학점을 입력하세요: C
Good

=== Code Execution Successful ===
```

C 를 넣으면 Good 이 출력된다.

if-else 와 switch 의 선택



if-else

범위로 나뉘면

- 점수 90 이상은 A
- 나이가 20대인지 판별



switch

값으로 나뉘면

- 월에서 계절 구하기
- 학점 문자에서 메시지

범위는 if-else, 값은 switch

틀린 코드 찾기: switch

어디가 틀렸나

①

```
int a = 3;  
switch (n) {  
    case a:  
        break;  
}
```

②

```
double rate;  
switch (rate) {  
    case 3.5:  
        break;  
}
```

③

```
switch (n) {  
    case 3: x = 1;  
    case 6: x = 2;  
    default: x = 3;  
}
```

break 하나가 전화망을 9시간 멈춘 사건



1990년 1월 15일, AT&T 장거리 전화망이 9시간 멈췄다.

원인은 새로 배포한 C 코드의 switch 문이었다.

break 한 줄이 빠져 흘러내린 코드가 데이터를 지웠다.

최근 언어들은 자동으로 break 되도록 문법을 바꿨다.

3교시 실습 살펴보기

Twenties

`(age >= 20) && (age < 30)`. `20 <= age < 30` 은 boolean 비교라 오류

우산 챙기기

`>= 60` 과 `> 60` 은 경계값 60 에서만 다르다

Grading

else if 순서가 조건의 일부. 60 부터 검사하면 89 도 D

Season

case 3: case 4: case 5: 는 묶기. break 없으면 흘러내린다

switch 버전

문자열 switch 는 JDK 1.7 부터. case 값은 리터럴만

심화학습

이제부터는 조금 복잡하게

가위바위보 게임

미션 조건

- 클래스: Main
- 입력: 철수 · 영희의 가위 · 바위 · 보
- 출력: 승자 이름 한 줄

Main.java, 템플릿 아래에 이어서

```
Scanner scanner = new Scanner(System.in);
System.out.print("철수 >> ");
String p1 = scanner.next();
System.out.print("영희 >> ");
String p2 = scanner.next();
if (p1.equals(p2))
    System.out.println("무승부");
scanner.close();
```

가위바위보 힌트 네 가지

1

print() 는 줄을 바꾸지 않는다

```
System.out.print("철수 >> ");
```

2

문자열은 next() 로 읽는다

```
String p1 = scanner.next();
```

3

문자열 비교는 equals()

```
if (p1.equals("가위"))
```

4

승패는 한 갈래부터 쓴다

```
if (p1.equals("가위") && p2.equals("보"))
```

안 될 때 여기부터

틀림

- 메인 클래스 이름이 Main 이 아님
- 입력했는데 반응 없음
- 문자열 비교에 == 사용

맞음

- ✓ 메인 클래스는 Main 으로
- ✓ Output 창을 먼저 클릭하고 Enter
- ✓ 문자열 비교는 equals() 사용

오류 메시지는 줄 번호부터 읽는다

오류 메시지 읽기

```
Main.java
1 import java.io.PrintStream;
2 import java.nio.charset.StandardCharsets;
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) {
7         System.setOut(new PrintStream(System.out, true,
8             StandardCharsets.UTF_8));
9         int n = 2030
10     }
11 }
```

Output

ERROR!
Main.java:8: error: ';' expected
int n = 2030
^

1 error
=== Code Exited With Errors ===

① 줄 번호 ② 메시지 ③ 캐럿 자리 순서로 읽는다. 오류가 여러 개면 첫 번째부터

자주 보는 오류 메시지

메시지	원인
<code>';' expected</code>	세미콜론 누락
<code>cannot find symbol</code>	이름 오타 또는 import 누락
<code>class X is public, should be declared in a file named X.java</code>	메인 클래스 이름이 Main 이 아님

실행 중 오류 `InputMismatchException` 은 입력 타입 불일치

== 로 비교했는데 왜 어떤 때는 맞나



글자가 같은지 보려면 equals()

코드에 적은 문자열

```
String a = "가위", b = "가위";  
a == b 는 참
```

입력으로 받은 문자열

```
String s = sc.next();  
s == "가위" 는 거짓
```

오늘 공부한 내용

클래스 안에 변수 · 메소드 · main()

기본 타입 8개 + 레퍼런스 타입 1개

Scanner 는 토큰 단위로 읽는다

연산자: 산술 · 비교 · 논리 · 삼항

조건문: if-else 와 switch

다음 주엔 반복문부터



다음 주 예고

- for · while · do-while
- 배열과 배열 활용



과제

- 가위바위보 완성 코드
- 2begoodlife@gmail.com