



Java 초급

프로그래밍의 이해 &
Java 소개

프로그램 program



➤ **대본**의 구성요소?

- **지시**(말한다. 움직인다. 등), **내용**(말할 내용, 움직임의 내용 등)

프로그램 program

3

```
import java.util.Scanner;
public class MainActivity11 {
    public static void main(String[] args) {
        System.out.println("[if, else 문을 사용해서 계산기 프로그램 만들기 실시]");
        int one = 0;
        int two = 0;
        String sign = "";
        Scanner scan = new Scanner(System.in);
        System.out.print("첫 번째 값 : ");
        one = scan.nextInt();
        System.out.print("두 번째 값 : ");
        two = scan.nextInt();
        System.out.print("사칙연산부호 (+, -, *, /) : ");
        sign = scan.next();
        if(sign.equals("+")) {
            System.out.println(one+"-"+two+"="+ (one+two));
        }
        else if(sign.equals("-")) {
            System.out.println(one+"-"+two+"="+ (one-two));
        }
        else if(sign.equals("*")) {
            System.out.println(one+"-"+two+"="+ (one*two));
        }
        else if(sign.equals("/")) {
            System.out.println(one+"-"+two+"="+ (one/two));
        }
        else {
            System.out.println("알수없는 연산자입니다 ... ");
        }
    }
}
//메인 종료
//클래스 종료
```

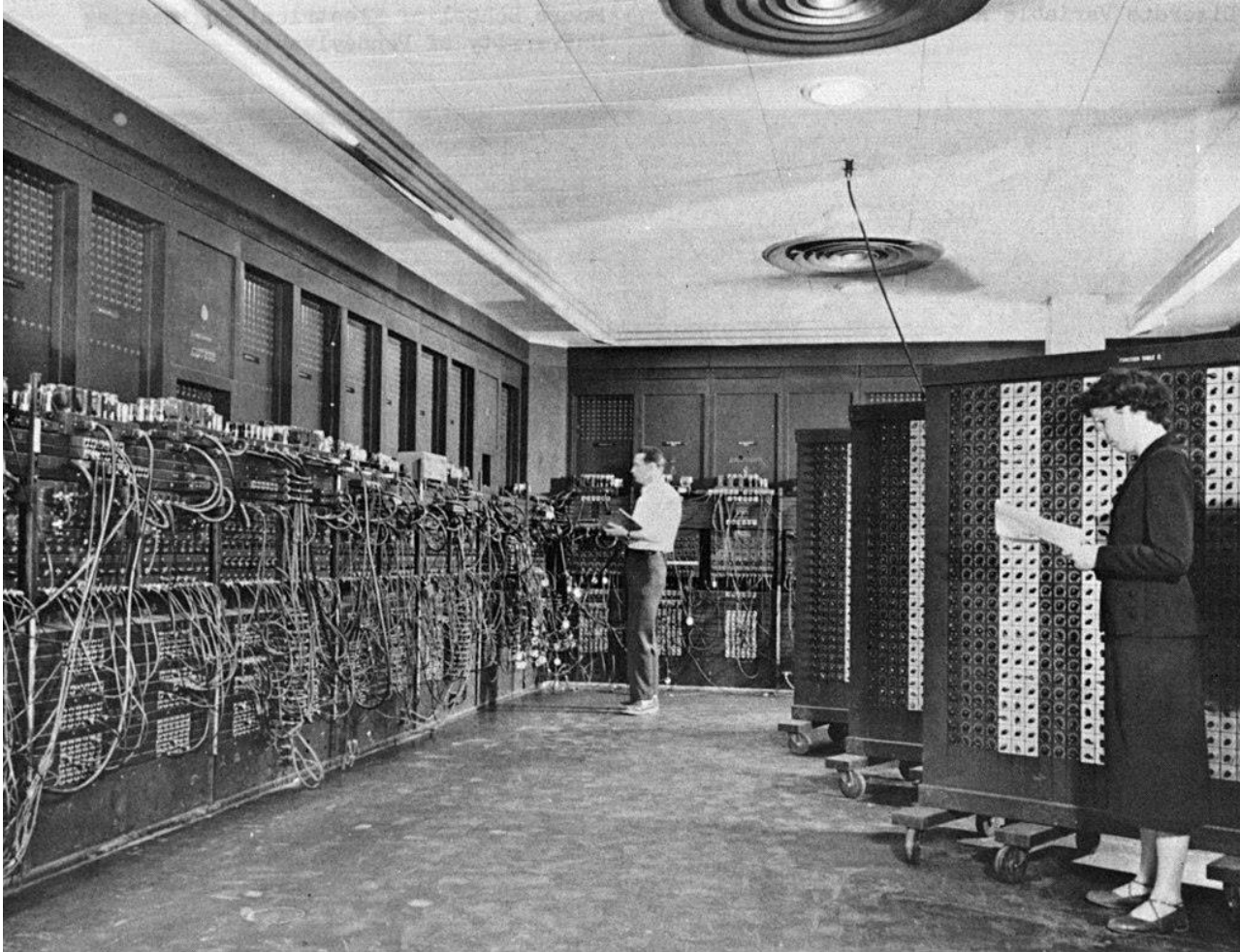


➤ 프로그램은 컴퓨터를 위한 **대본** (지시→**명령어**, 내용→**데이터**)

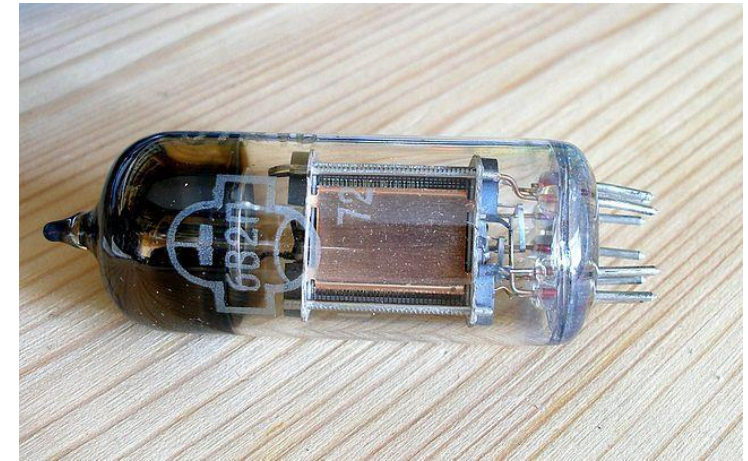
2가지 모습을 가지는 프로그램

시작은 **HW**였다.

전자식 숫자 적분 및 계산기(Electronic Numerical Integrator And Computer; ENIAC, 에니악)

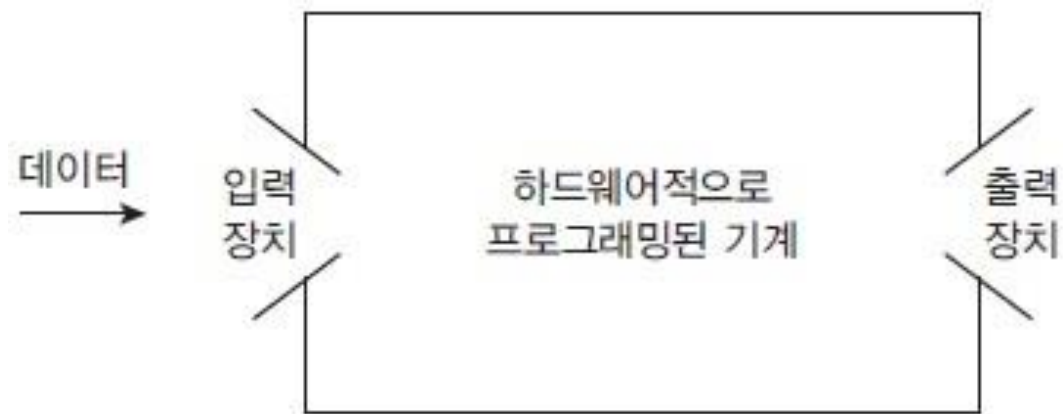


- 펜실베이니아 대학의 모클리 (J.W. Mauchli)와 에커트(J.P. Eckert) 교수에 의해 발명
- 18,000여개의 진공관이 사용됨
- 높이 5.5m, 길이 24.5m, 무게가 30톤

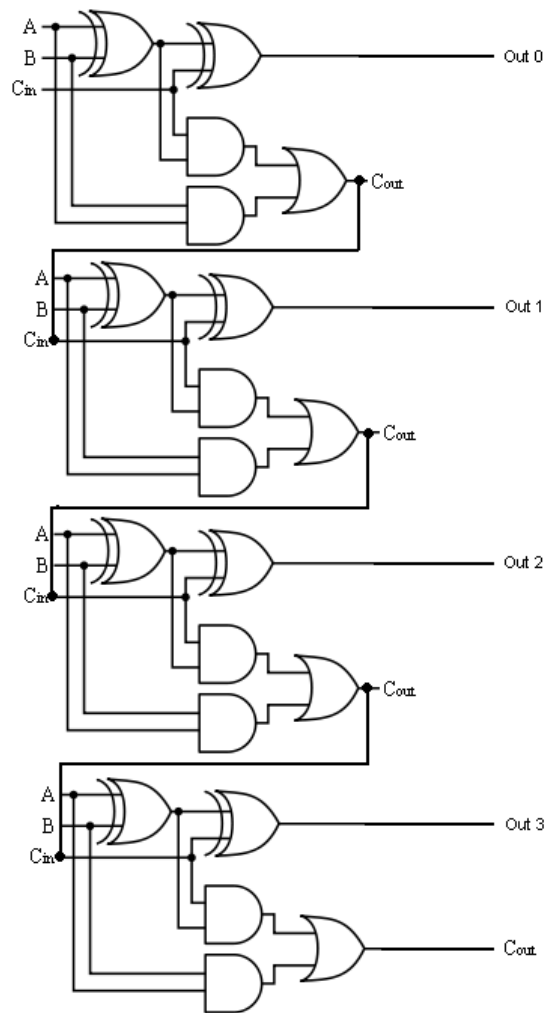


HW 프로그램

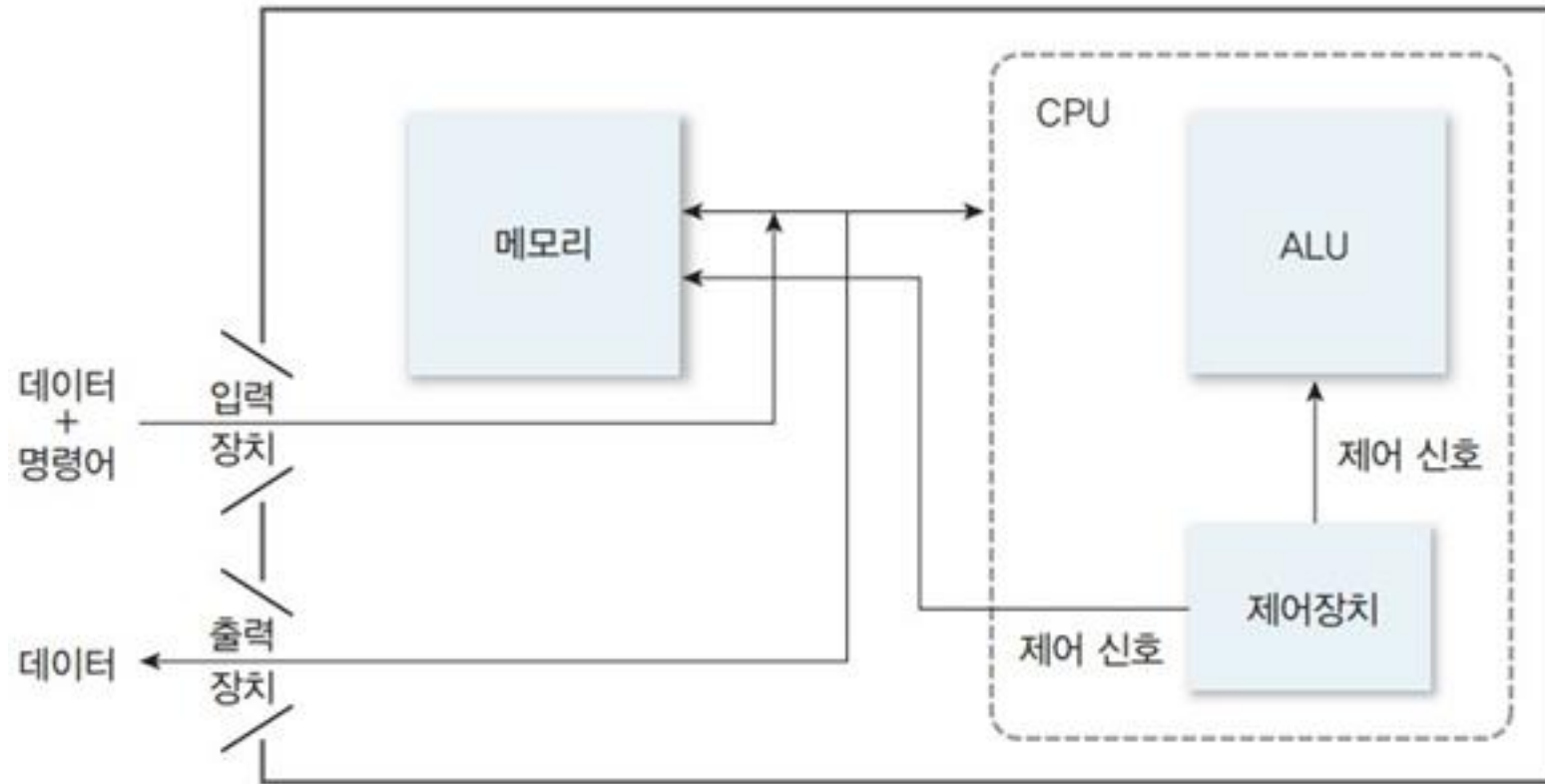
6



고정결선식 프로그램 컴퓨터



4bit 가산기

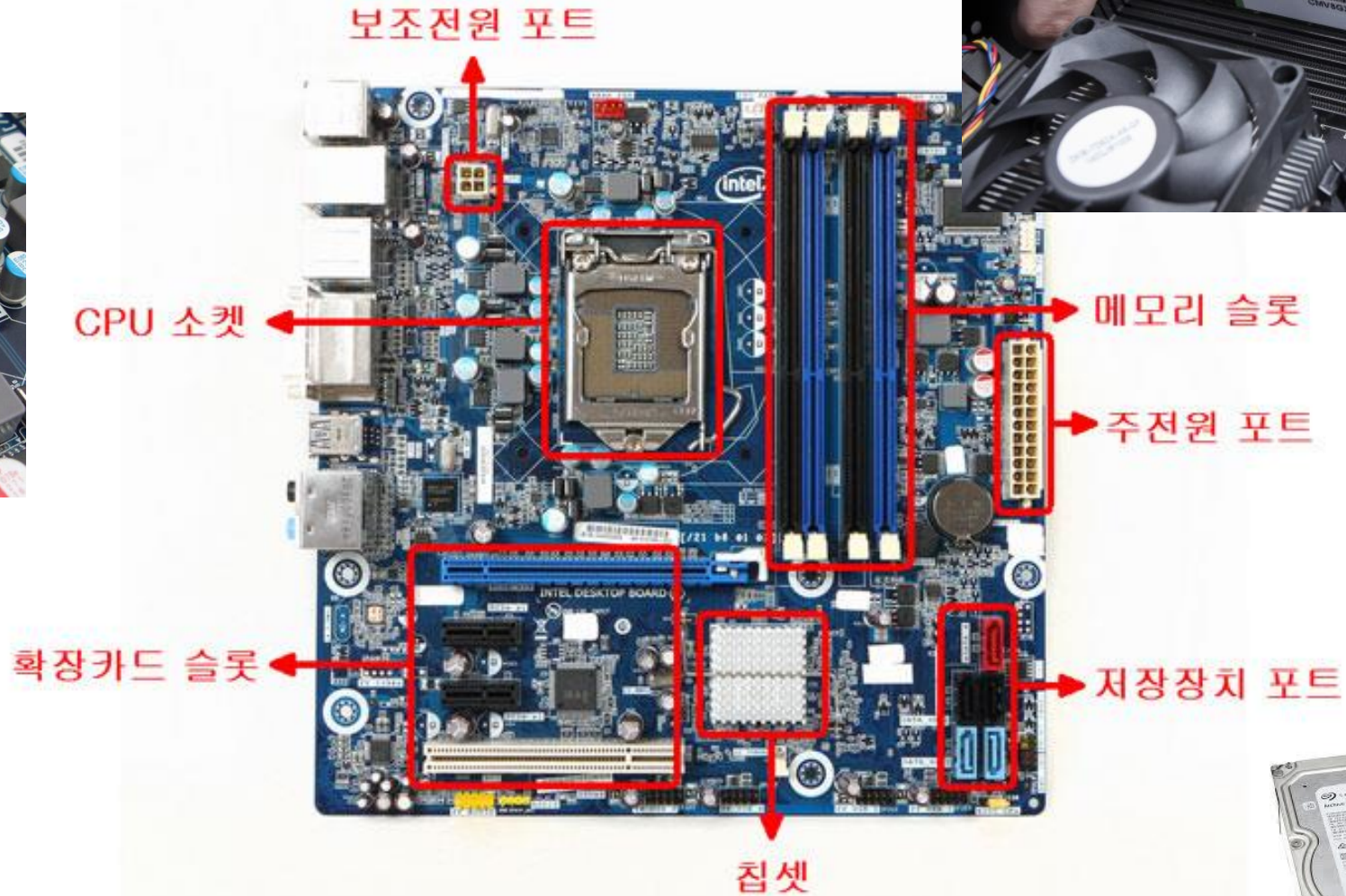


프로그램 내장식 컴퓨터
(폰 노이만 아키텍처)

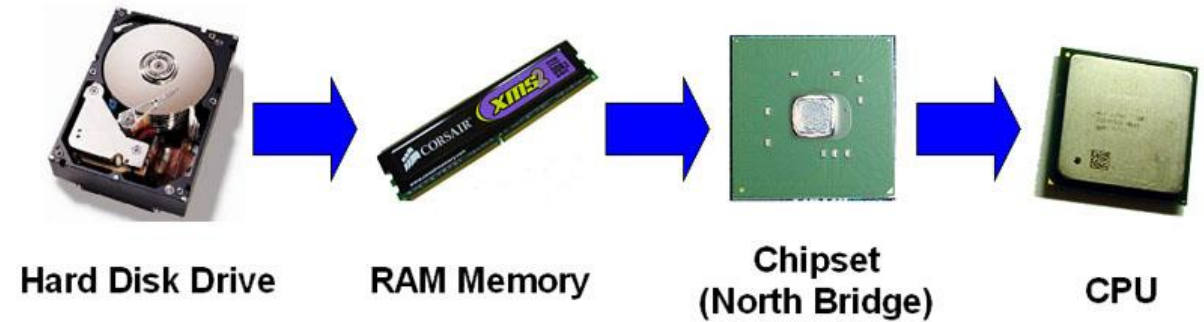
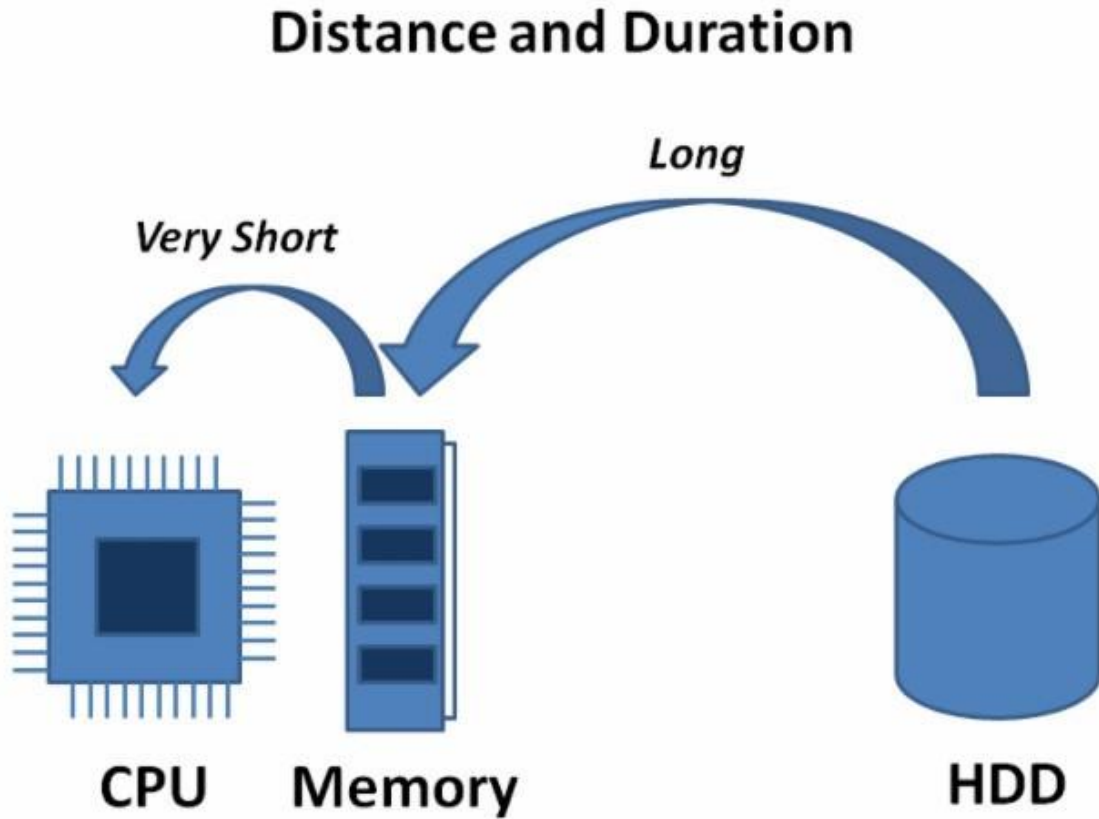
모든 **실행**은 **HW**에서 시작된다...

컴퓨터의 HW 요소

9

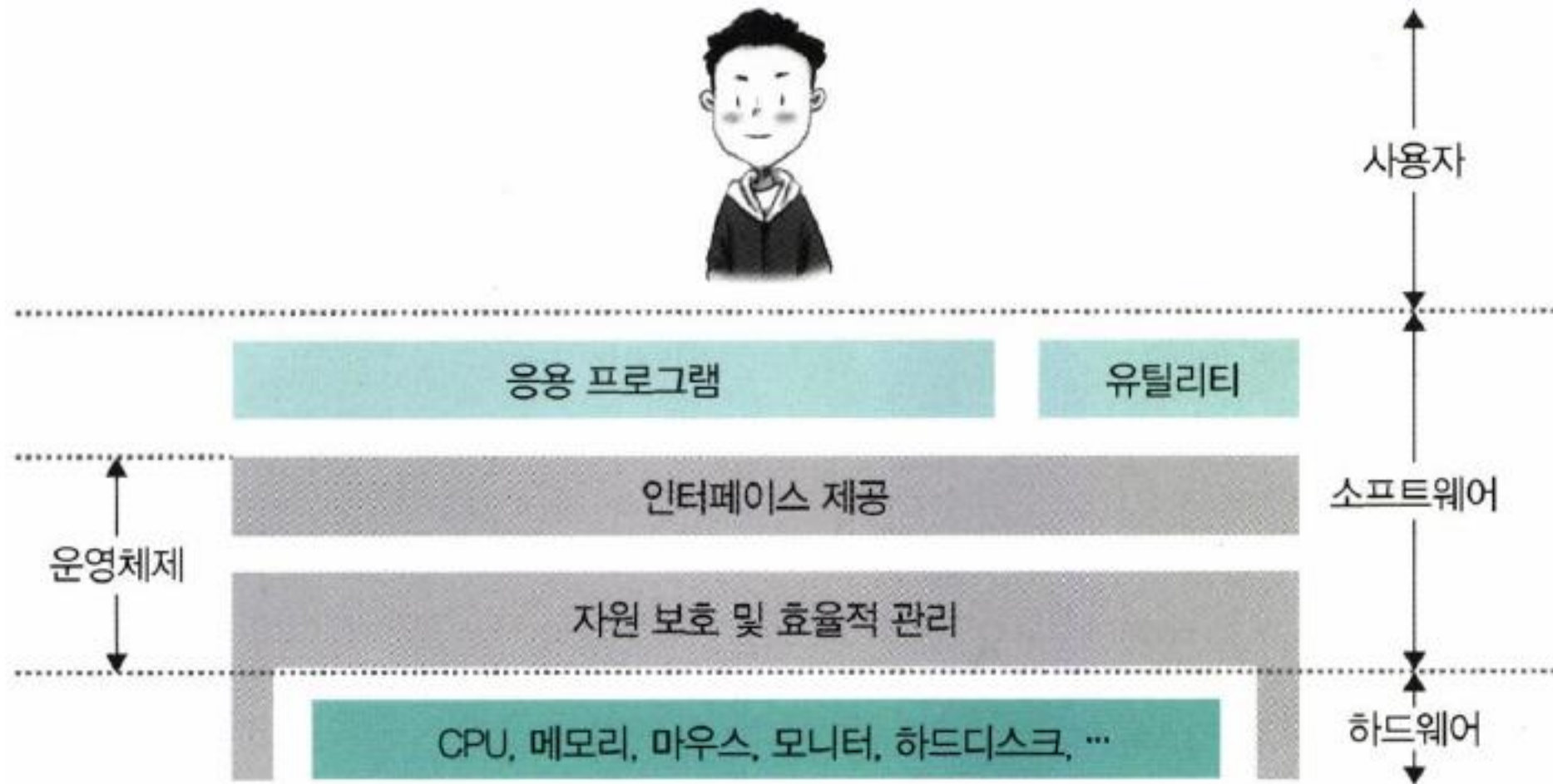


Memory의 필요성



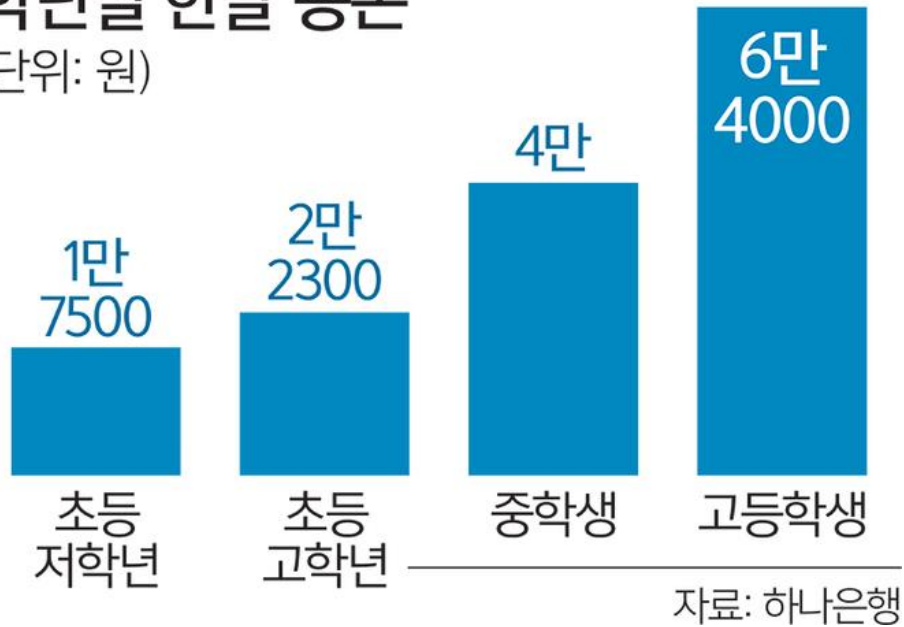
운영체제, OS, operating system

11

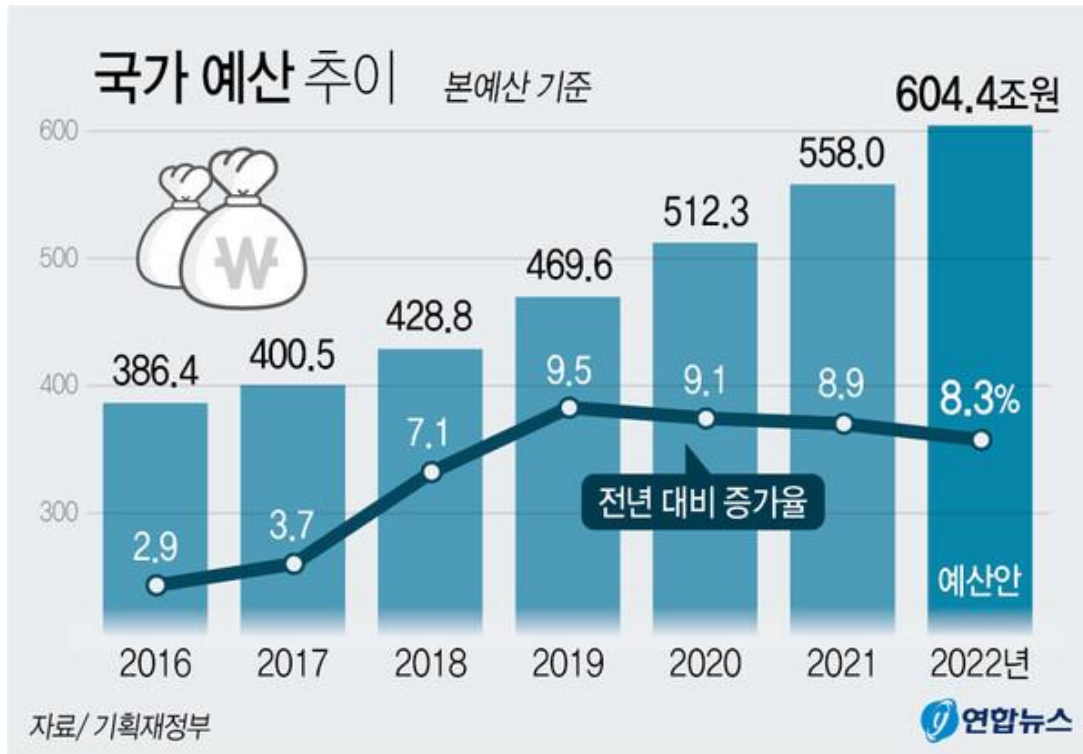


OS 왜 필요한가요?

학년별 한달 용돈
(단위: 원)



OS 왜 필요한가요?



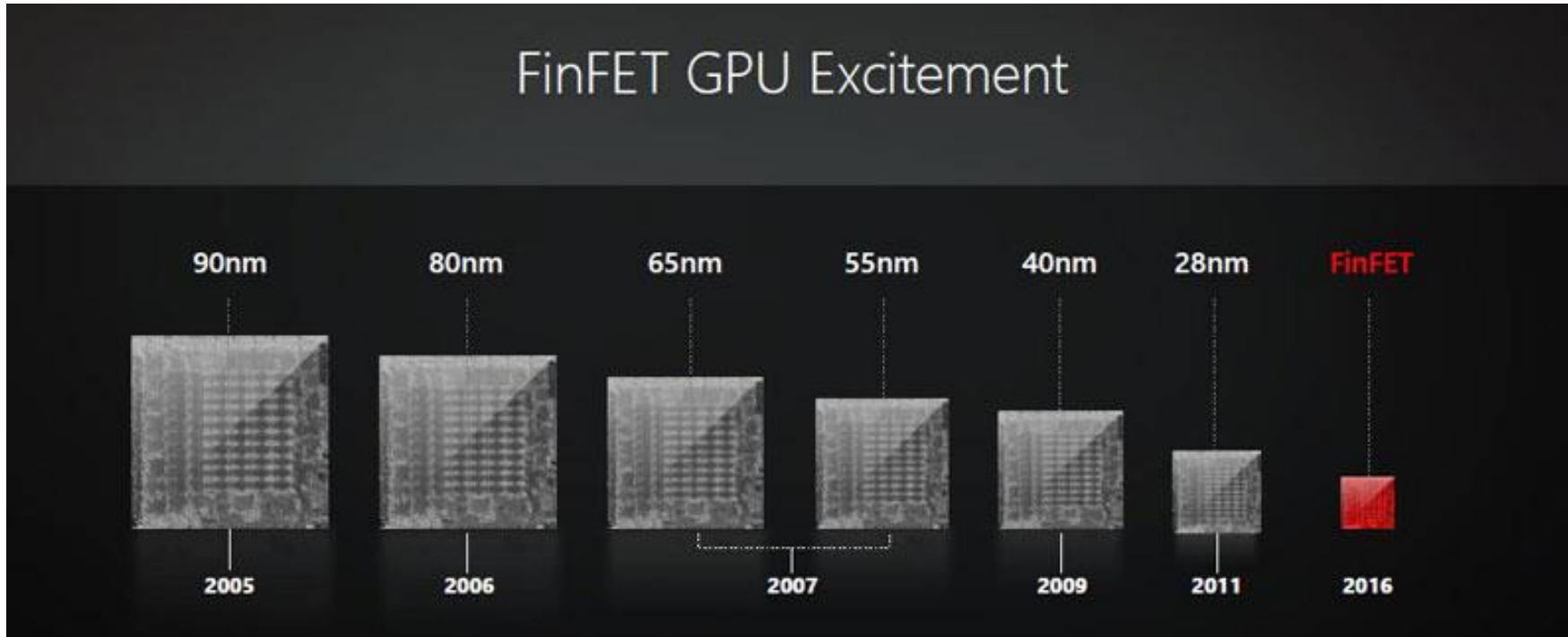
박영석 김영은 기자 / 20210831

트위터 @yonhap_graphics 페이스북 tune.kr/LeYN1

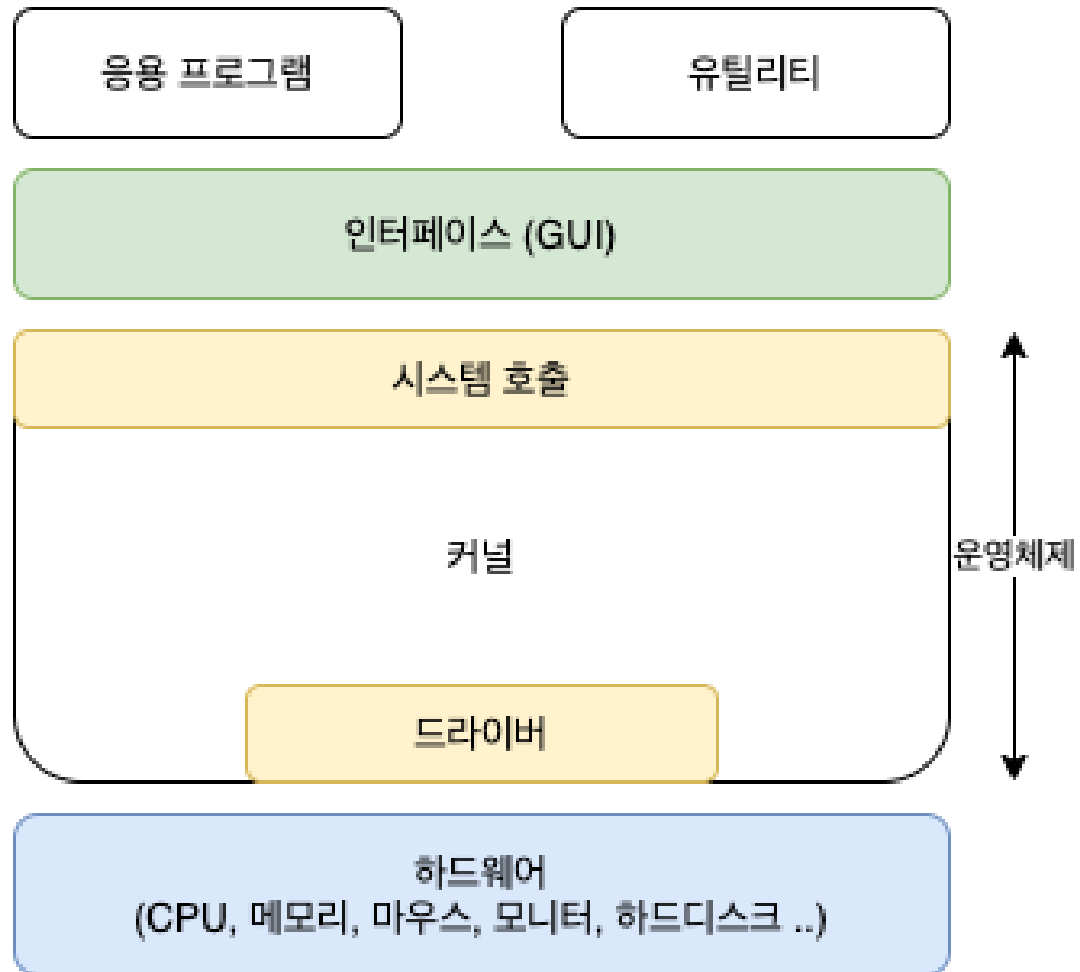


OS 왜 필요한가요?

14



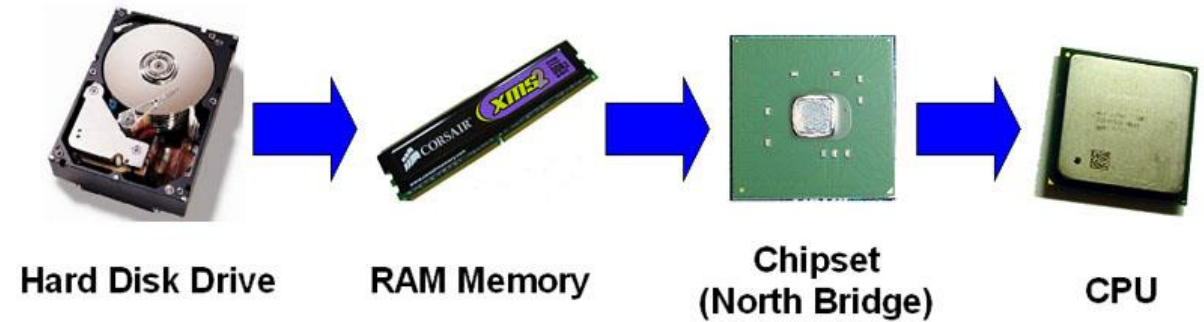
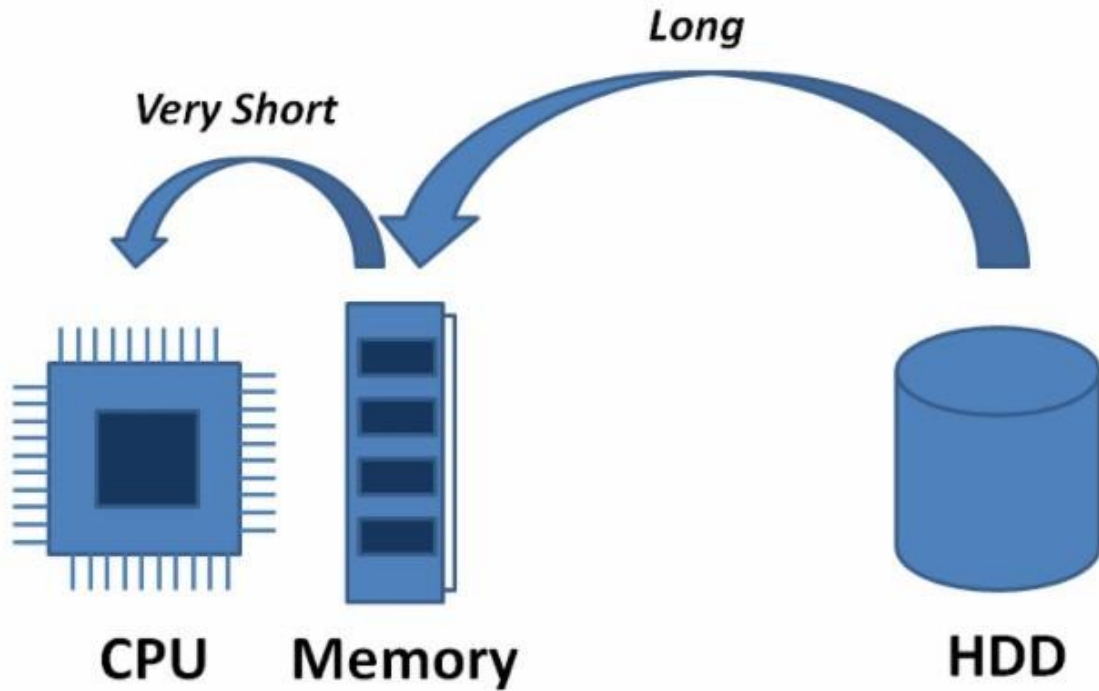
OS 왜 필요한가요?



기억합시다 → 모든 프로그램은 메모리에서 실행!

16

Distance and Duration



**컴퓨터의 부팅 과정을 통해
메모리 기반 컴퓨팅을 생각해 봅시다**

^^*

컴퓨터 부팅(booting) 과정

ROM, read only memory → HW

BIOS, basic input output system → SW

전원 ON



BIOS의 주요 동작 내용

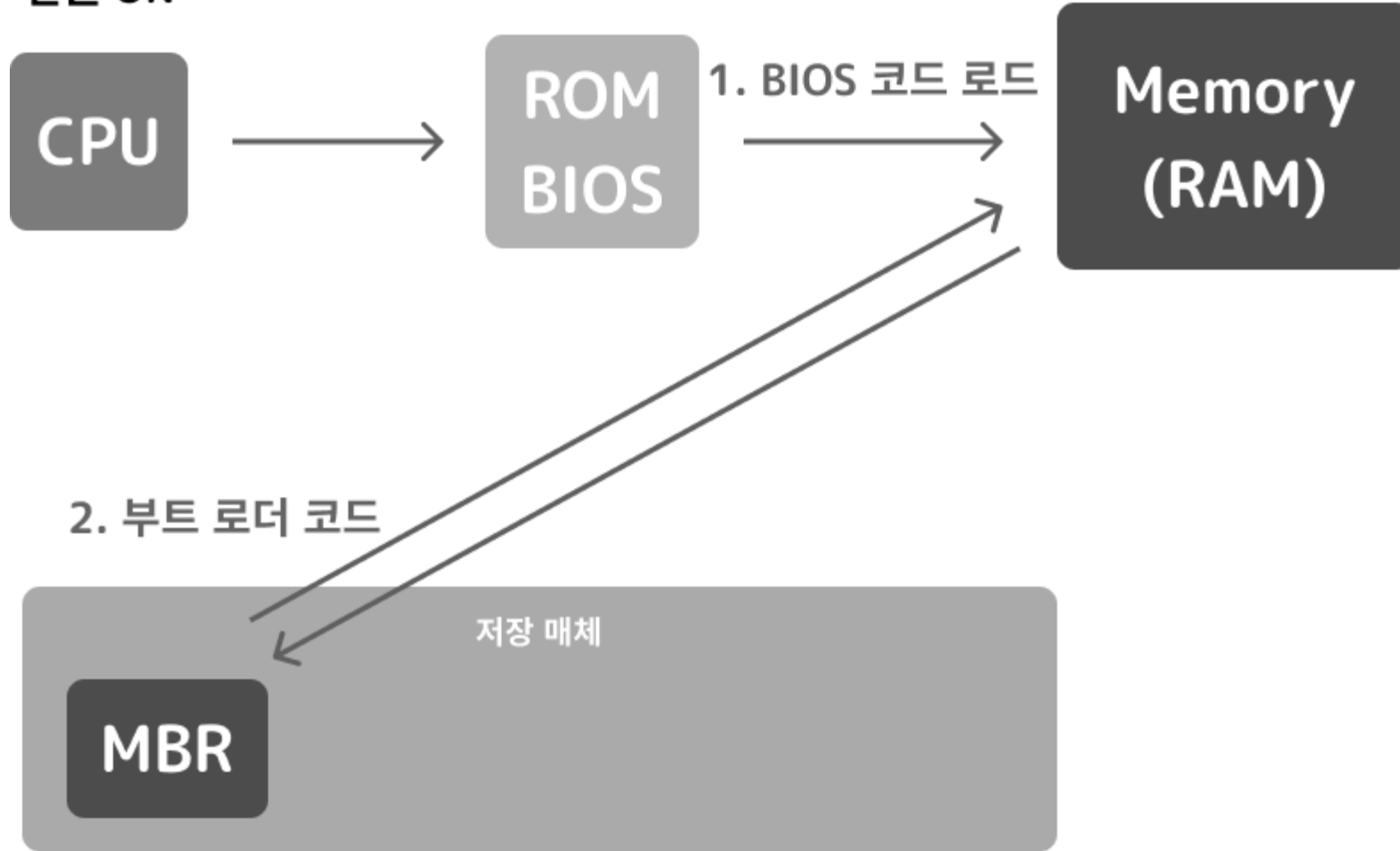
1) 하드웨어 초기화

2) 저장매체의 MBR(master boot record) 읽기

*MBR은 파티션 정보를 저장하고 있는 저장매체의 첫 부분으로 부트로더(boot loader) 프로그램이 또한 저장되어 있음

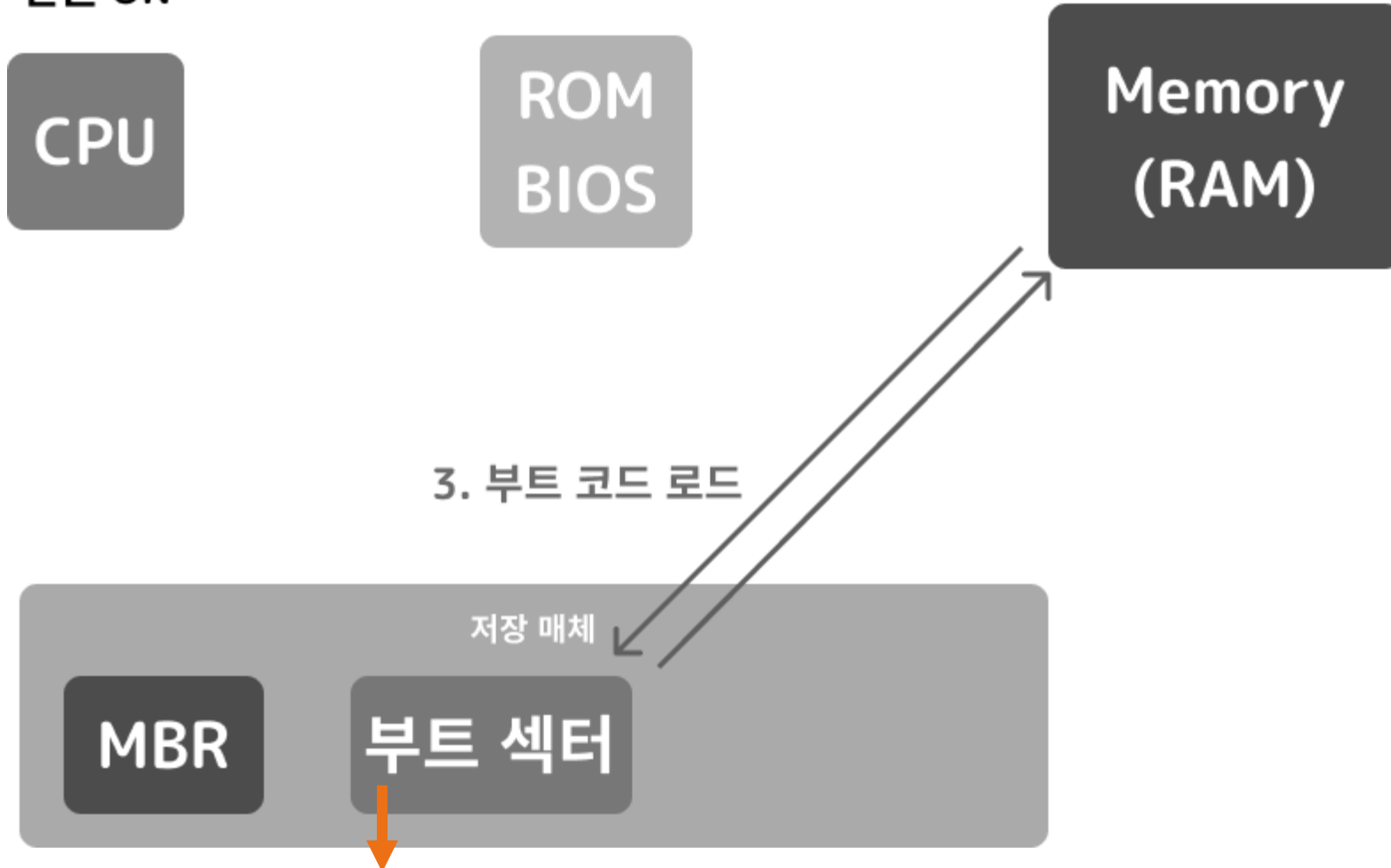
컴퓨터 부팅(booting) 과정

전원 ON



컴퓨터 부팅(booting) 과정

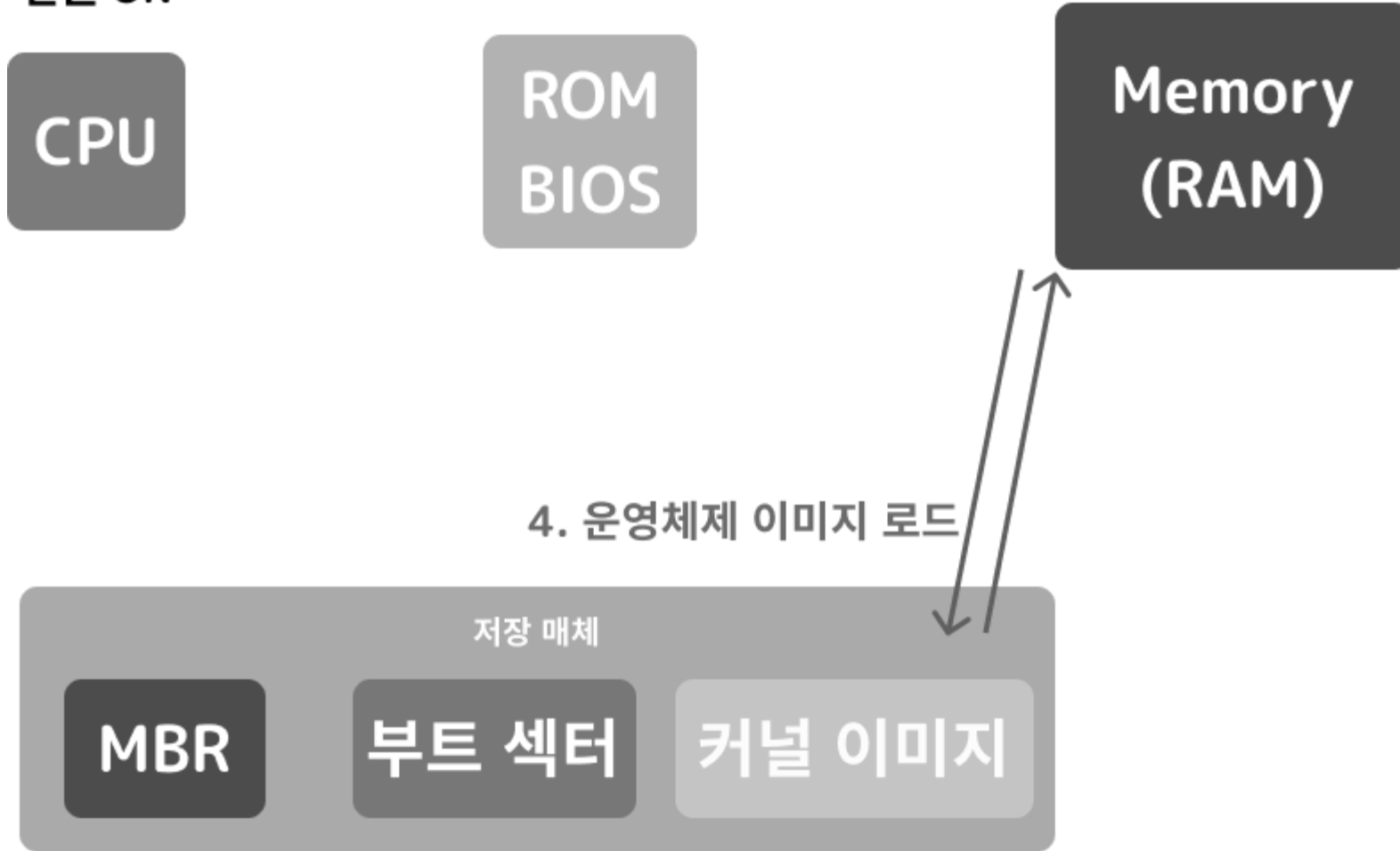
전원 ON



운영체제 커널 이미지를 메모리로 올리는 부트코드가 저장된 곳

컴퓨터 부팅(booting) 과정

전원 ON



앞서 배웠던 “프로그램”의 개념

```
import java.util.Scanner;
public class MainActivity11 {
    public static void main(String[] args) {
        System.out.println("[if, else 문을 사용해서 계산기 프로그램 만들기 실시]");
        int one = 0;
        int two = 0;
        String sign = "";
        Scanner scan = new Scanner(System.in);
        System.out.print("첫 번째 값 : ");
        one = scan.nextInt();
        System.out.print("두 번째 값 : ");
        two = scan.nextInt();
        System.out.print("사칙연산부호 (+, -, *, /) : ");
        sign = scan.next();
        if(sign.equals("+")) {
            System.out.println(one+"-"+two+"="+ (one+two));
        }
        else if(sign.equals("-")) {
            System.out.println(one+"-"+two+"="+ (one-two));
        }
        else if(sign.equals("*")) {
            System.out.println(one+"*"+two+"="+ (one*two));
        }
        else if(sign.equals("/")) {
            System.out.println(one+"/"+two+"="+ (one/two));
        }
        else {
            System.out.println("알수없는 연산자입니다 ... ");
        }
    }
}
//메인 종료
//클래스 종료
```



➤ 프로그램은 컴퓨터를 위한 **대본** (지시→**명령어**, 내용→**데이터**)

프로그래밍 programming

프로그램

```
import java.util.Scanner;
public class MainActivity11 {
    public static void main(String[] args) {
        System.out.println("[if, else 문을 사용해서 계산기 프로그램 만들기 실시]");
        int one = 0;
        int two = 0;
        String sign = "";
        Scanner scan = new Scanner(System.in);
        System.out.print("첫번째 값 : ");
        one = scan.nextInt();
        System.out.print("두번째 값 : ");
        two = scan.nextInt();
        System.out.print("사칙연산부호 (+, -, *, /) : ");
        sign = scan.next();
        if(sign.equals("+")) {
            System.out.println(one+"-"+two+"="+ (one+two));
        }
        else if(sign.equals("-")) {
            System.out.println(one+"-"+two+"="+ (one-two));
        }
        else if(sign.equals("*")) {
            System.out.println(one+"*"+two+"="+ (one*two));
        }
        else if(sign.equals("/")) {
            System.out.println(one+"/"+two+"="+ (one/two));
        }
        else {
            System.out.println("알수없는 연산자입니다 ... ");
        }
    }
} //메인 종료
} //클래스 종료
```



프로그래밍



**컴퓨터는 어떤 언어를 사용할까요?
사람의 언어를 어떻게 이해할까요?**

컴퓨터의 언어 → 기계어

1000 1011 0100 0101 1111 1000

1000 0011 1100 0100 0000 1100

0000 0011 0100 0101 1111 1100

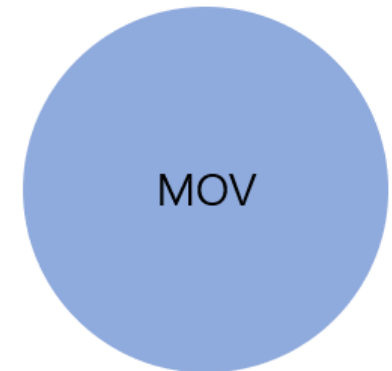
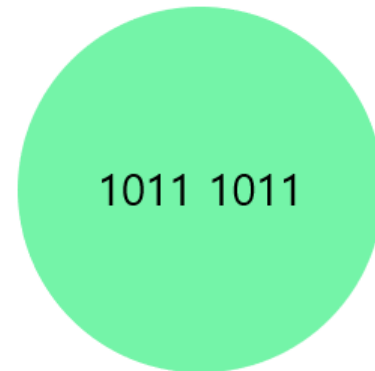
해석을...?



8b 45 f8

83 c4 0c

03 45 fc



사람어 → 기계어 변환

result = a + b;



mov eax, DWORD PTR [ebp-8]

add esp, 12

add eax, DWORD PTR [ebp-4]

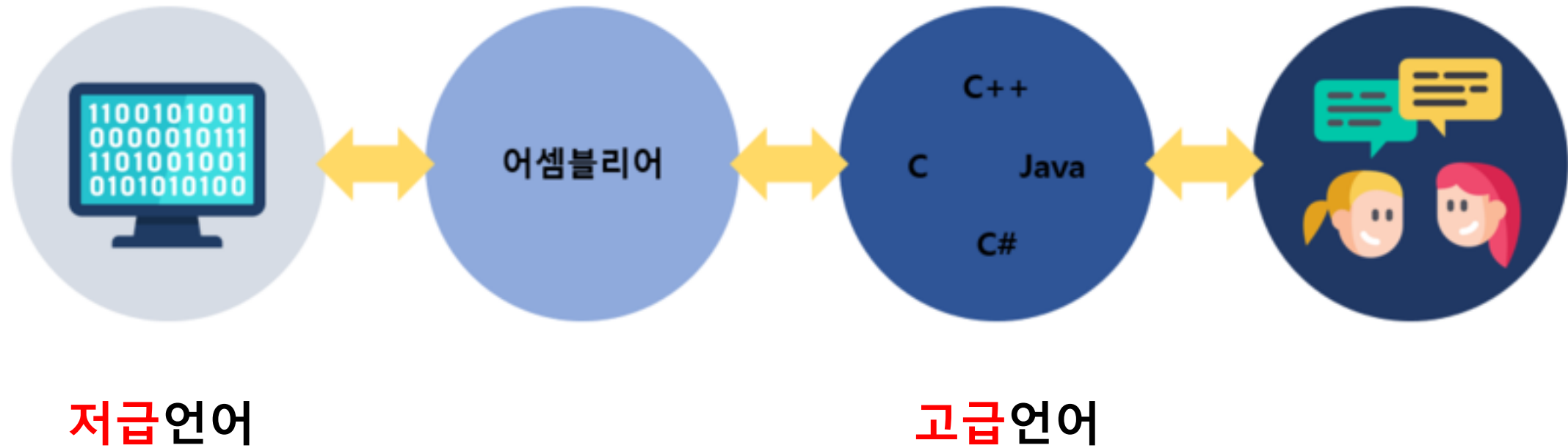


1000 1011 0100 0101 1111 1000

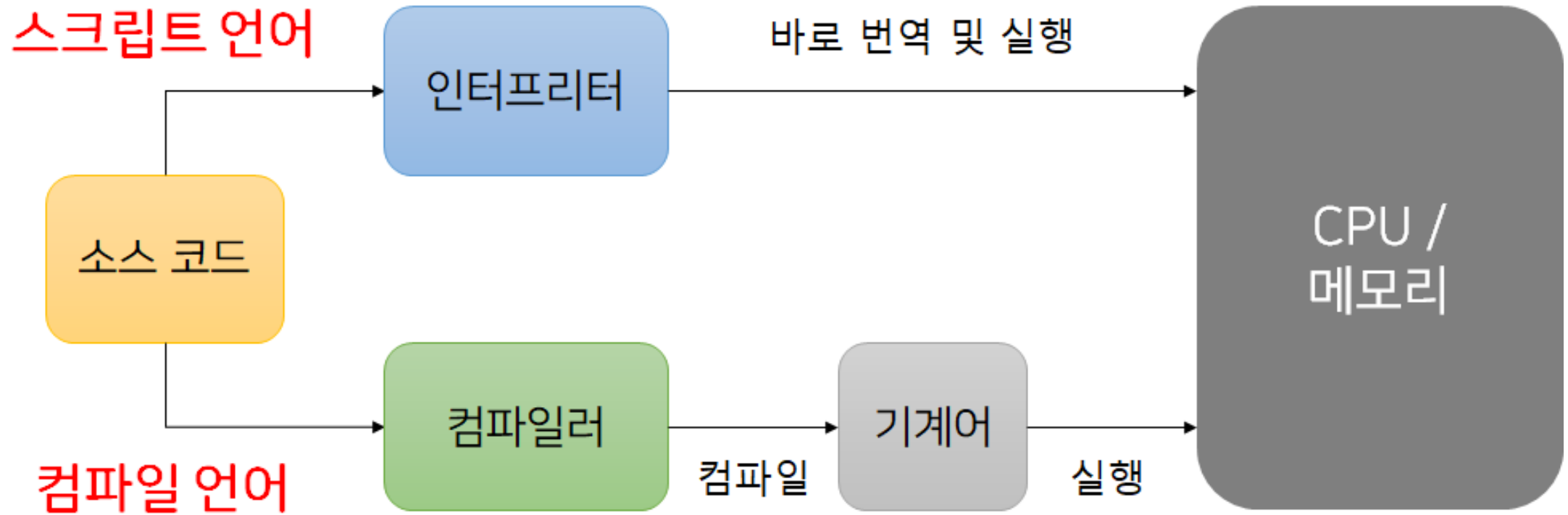
1000 0011 1100 0100 0000 1100

0000 0011 0100 0101 1111 1100

프로그래밍 언어 변환

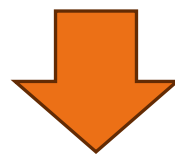


사람어 To 기계어



프로그래밍을 잘하는 방법이 있을까요?

네! 있습니다^^



전체 작성 과정의 흐름을 이해하는 것
입니다.



프로그래밍의 수행 과정 예시

데이터 : 중학생 1학년의 수학성적

처리 : 평균계산 =
학생들의 수학성적의 총합 / 학생 수

표현 : 모니터 화면에 평균 점수 나타냄.

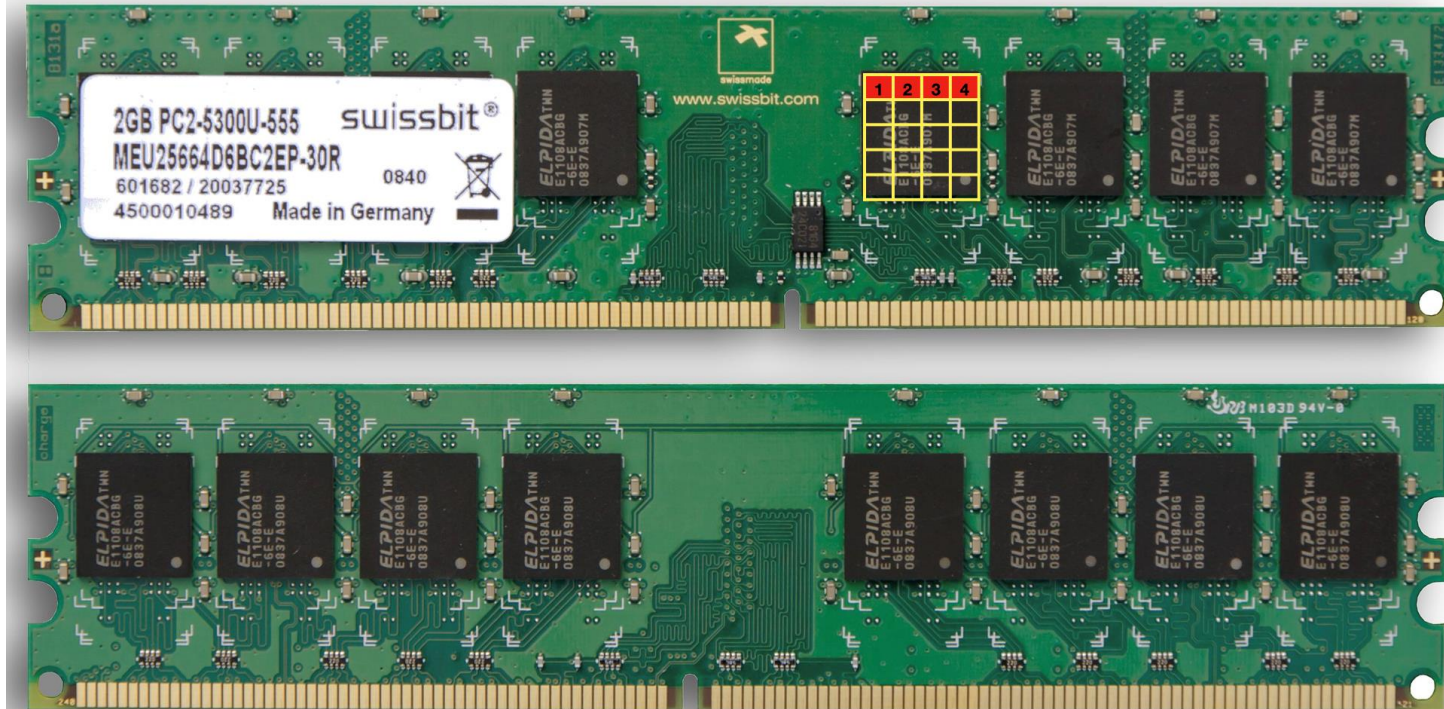
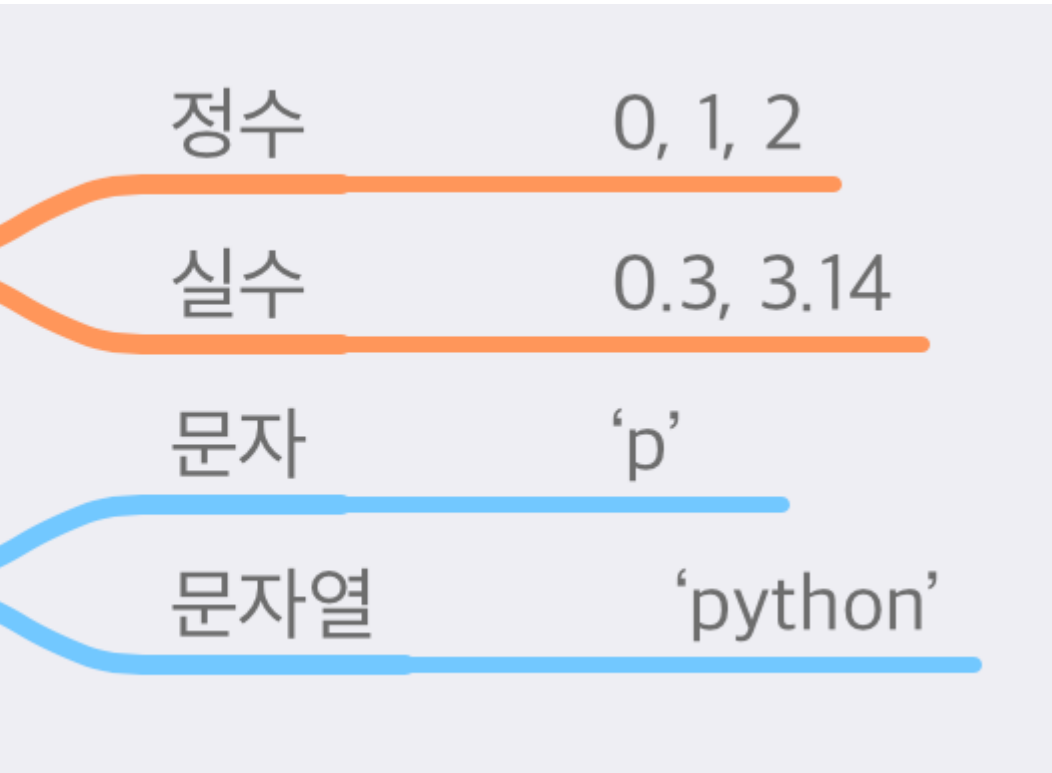
시작은 데이터를 저장하는 것!

➤ 데이터로서 저장되는 내용의 종류

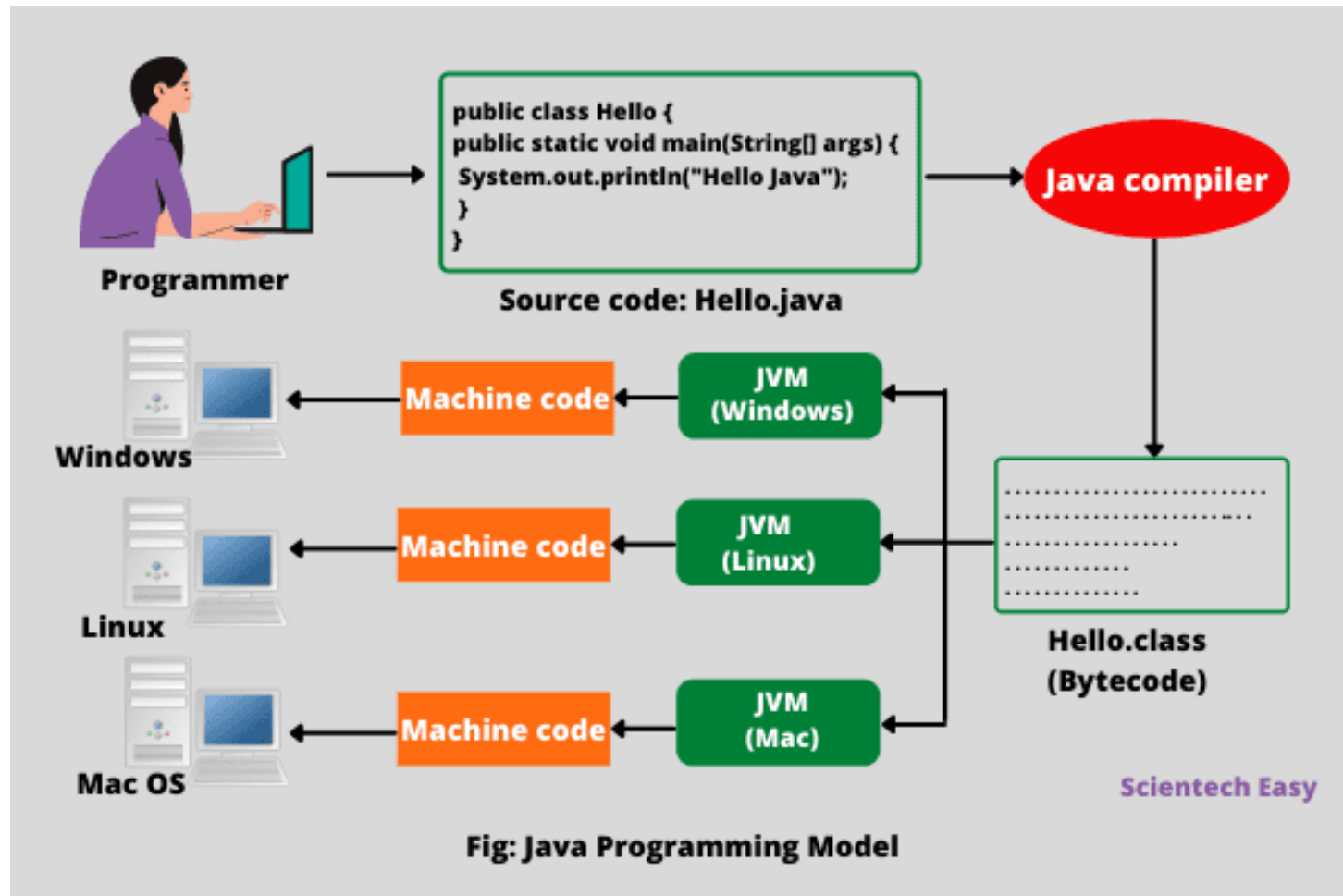


모든 실행은 메모리에서!

➤ Thus, 메모리에 **저장된 데이터만** 사용할 수 있음



이제 Java를 살펴 봅시다^^*



*JVM = Java Virtual Machine

Java 프로그램의 차이

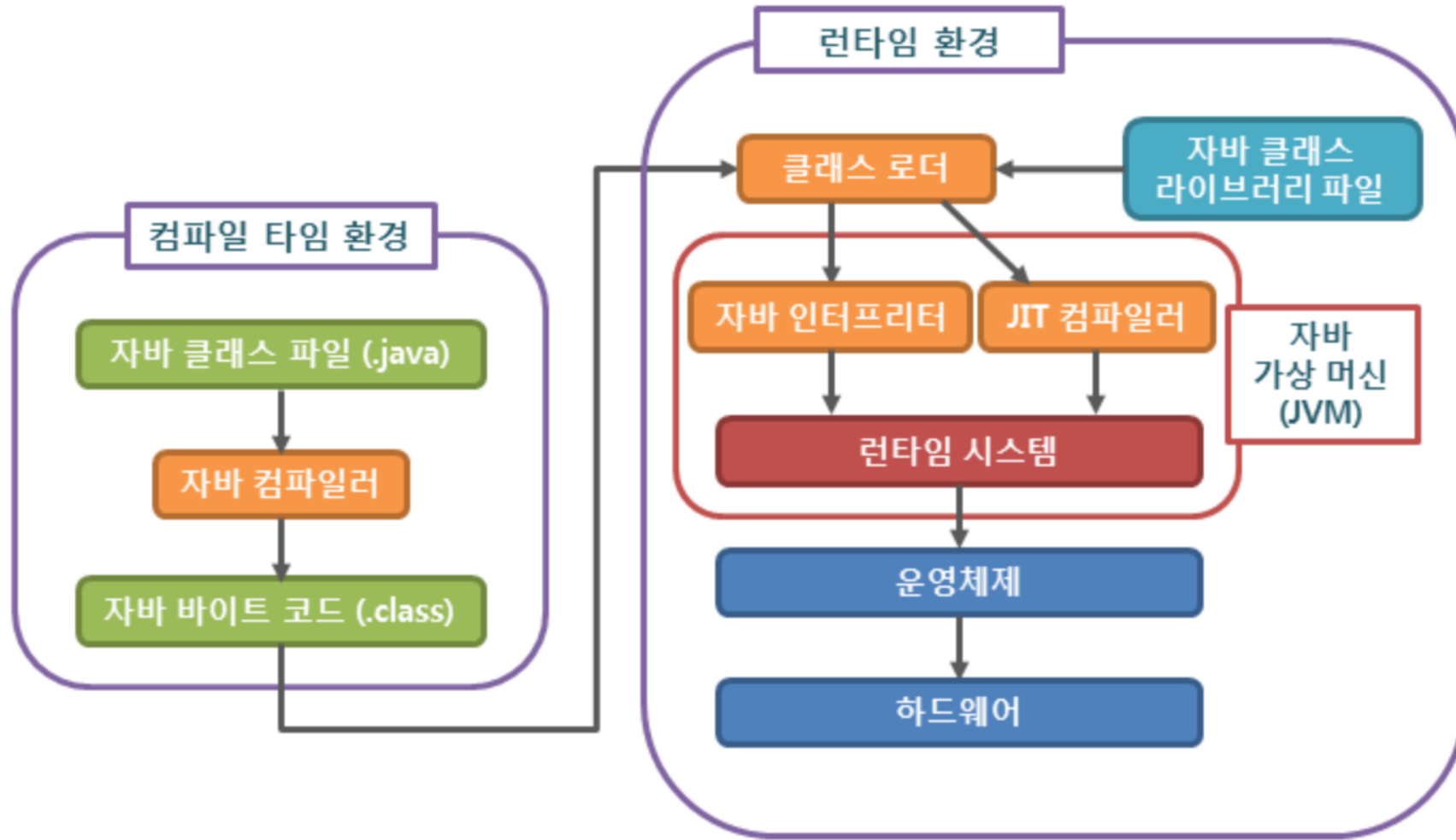


< 일반적인 프로그램 >



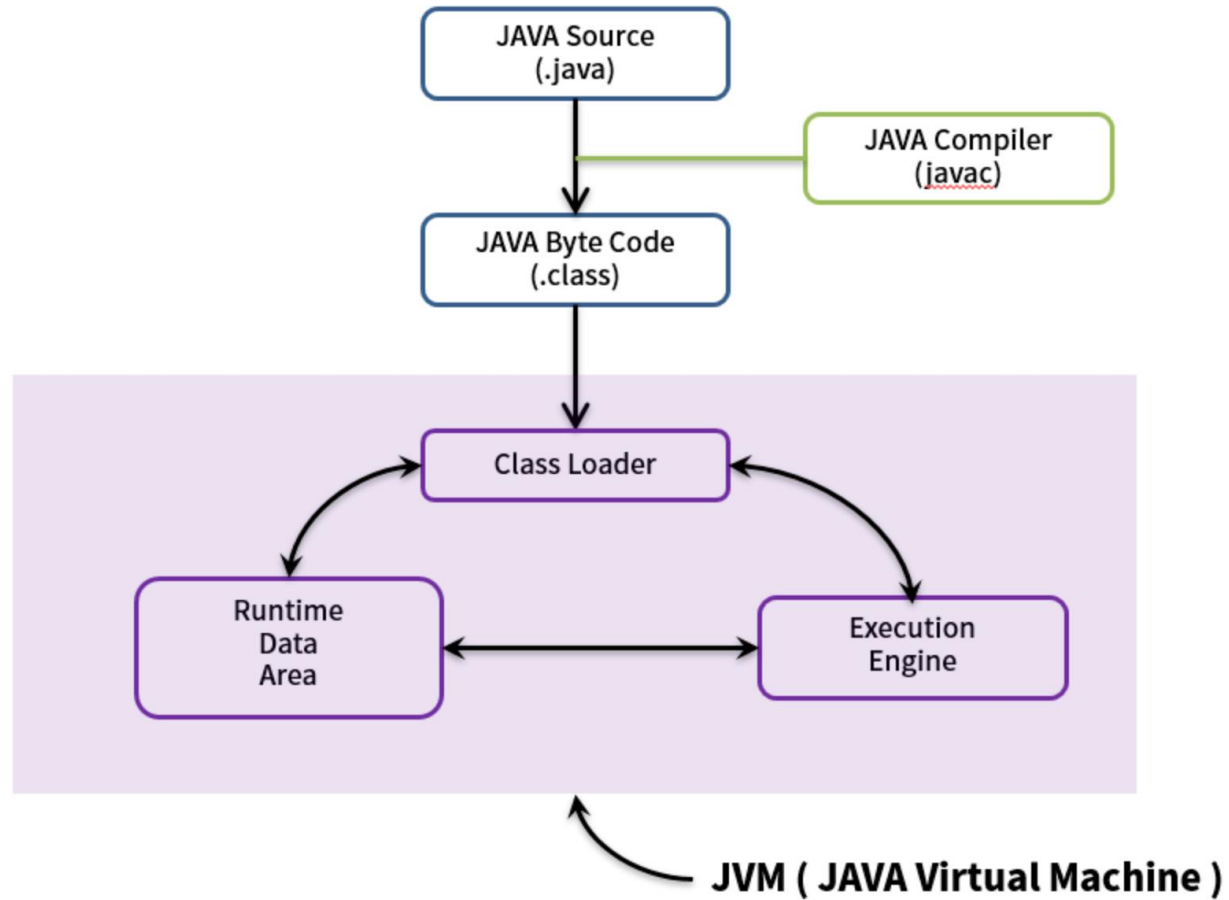
< JAVA 프로그램 >

Java compile & runtime process



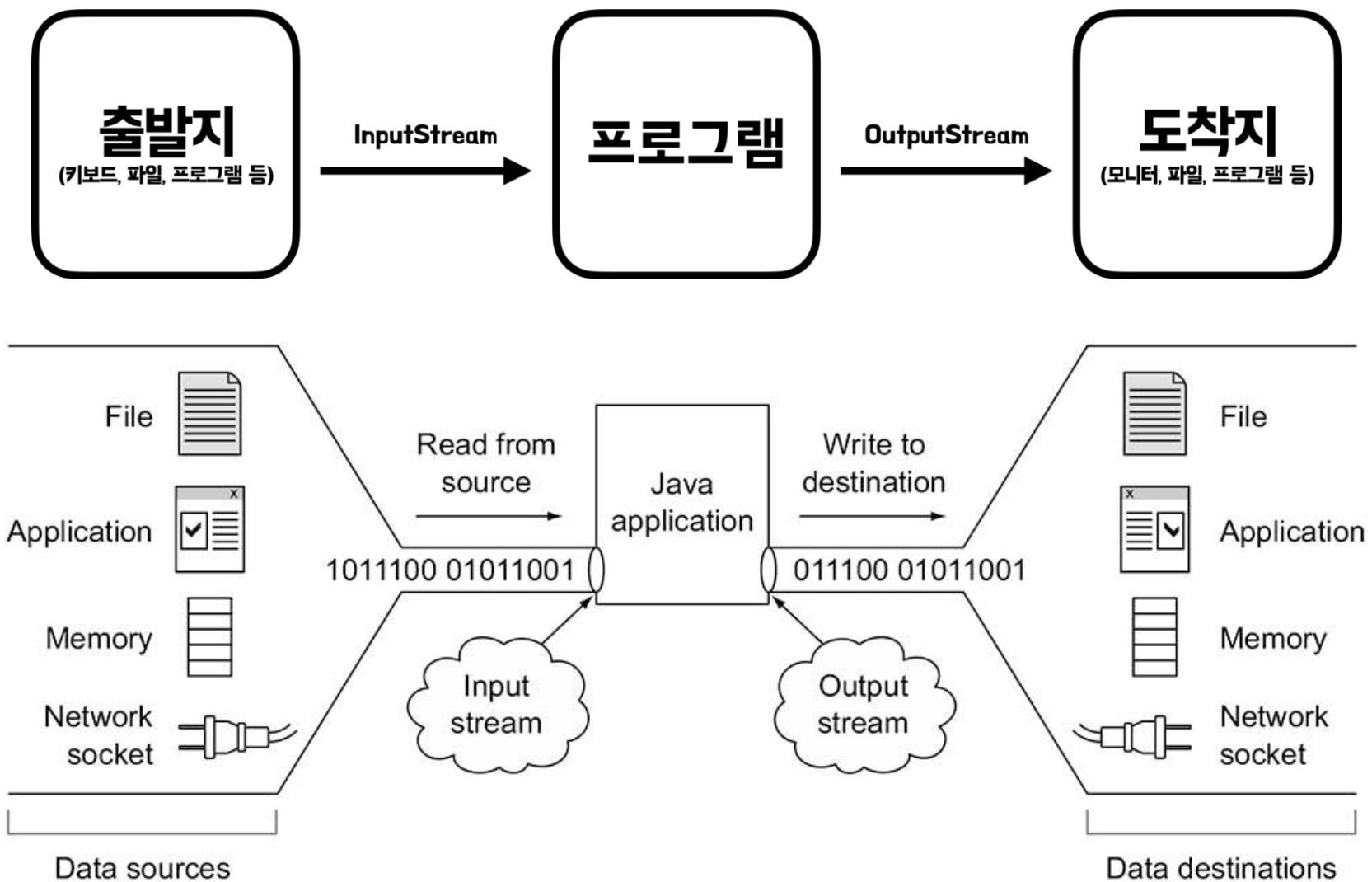
*JIT = Just-in-time

자바 코드(JAVA Code) 실행 과정

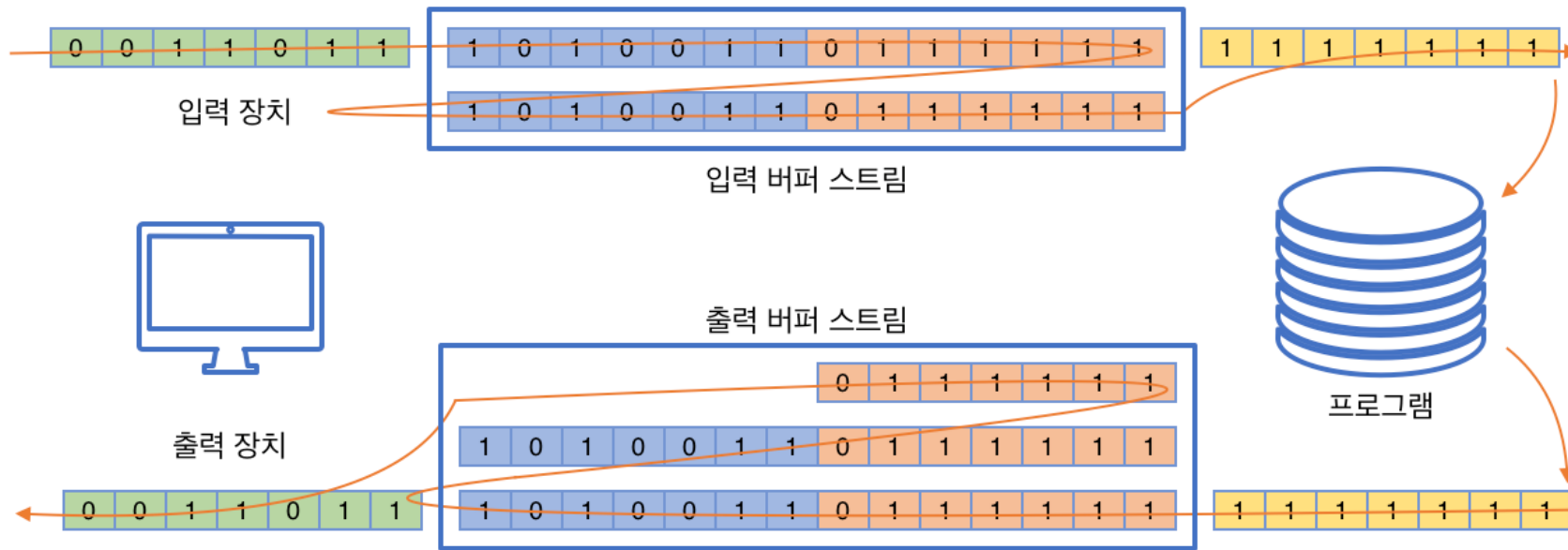
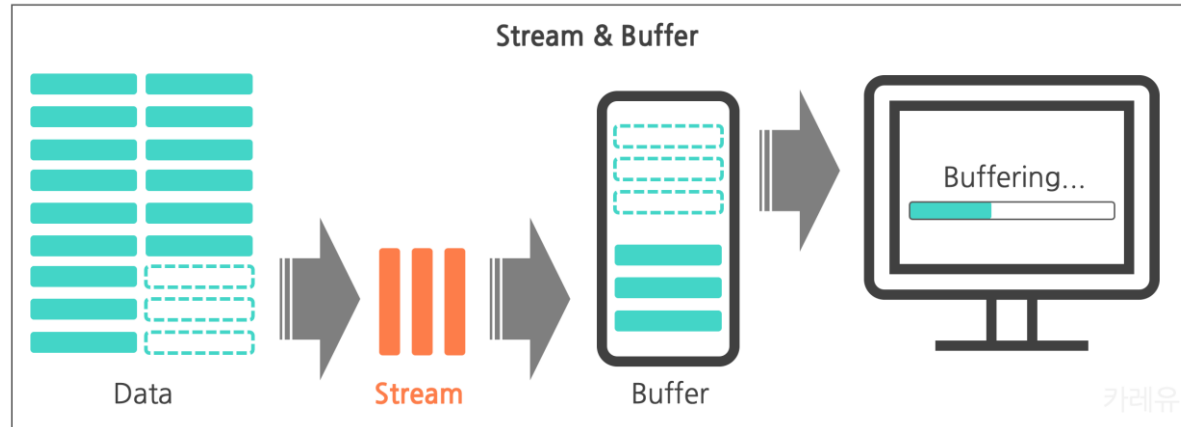


Java 특징

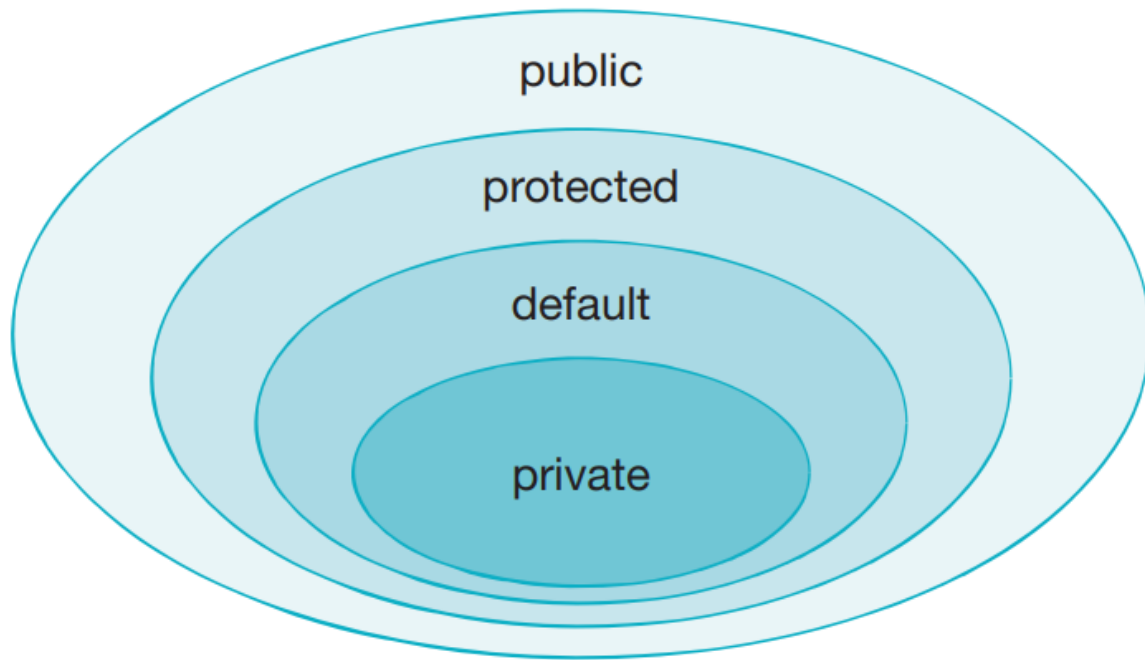
1. 운영체제에 독립적이다. → JVM
2. OOP(Object-Oriented Programming) 언어 → data 지향 언어
3. 자동 메모리 관리 → Garbage Collection → JVM
4. 네트워크와 분산처리를 잘 지원함 → 풍부한 관련 라이브러리 제공
5. Multi-Thread 지원 → 운영체제 호환성 + 관련 라이브러리
6. Dynamic Loading 지원 → 실행시 효율적인 로딩 프로세스 제공
7. 효율적 코드 컴파일 → 클래스 변경에 따른 컴파일 부하가 적음
8. 상대적으로 느린 동작 → 소스코드를 바이트 코드로 변환한 후 다시 기계어 변환함으로...



Stream



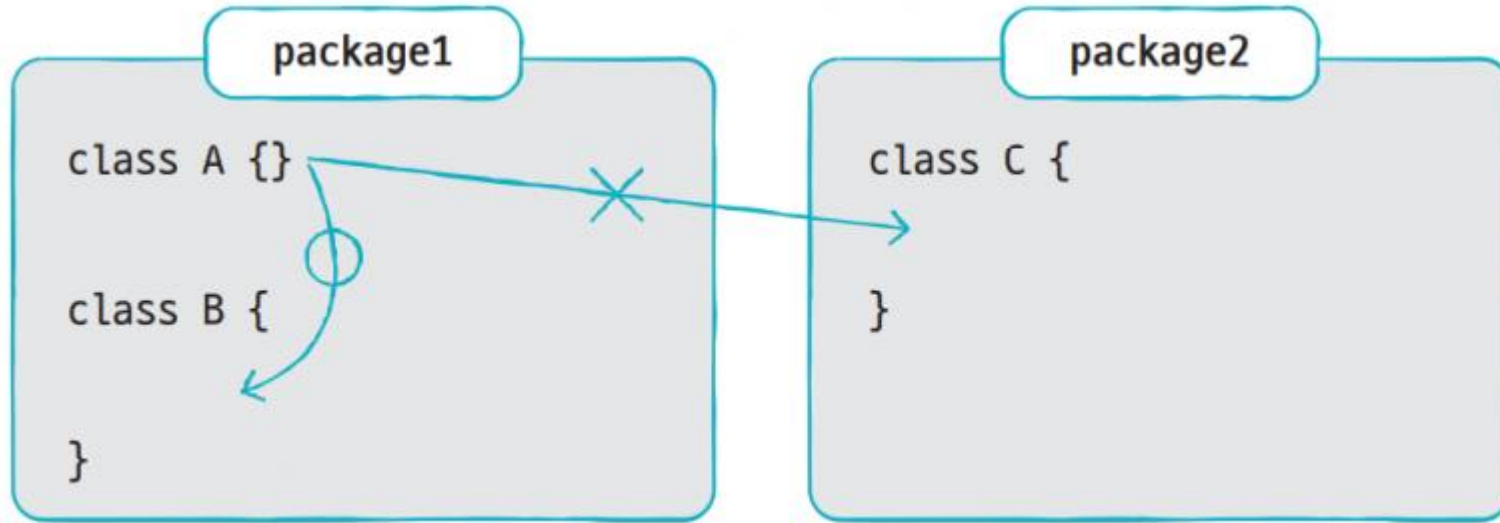
접근자(Access Modifier)



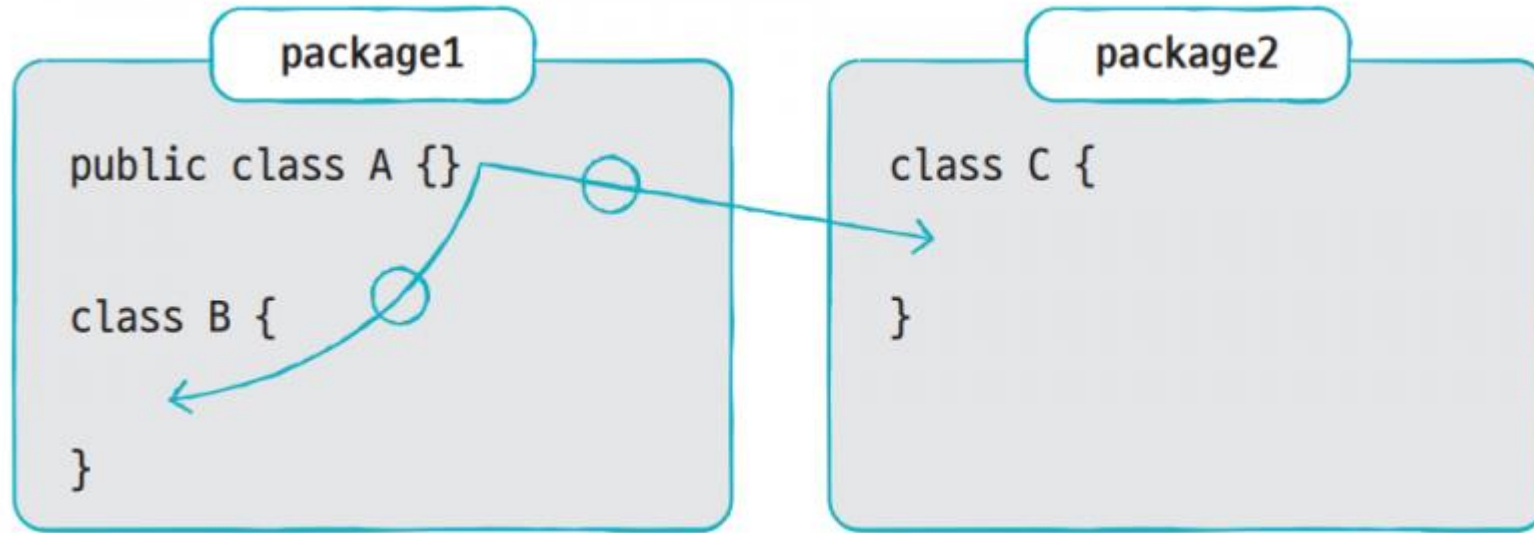
접근 제한이 강화

- **public 접근 제한자:** 외부 클래스가 자유롭게 사용 가능
- **protected 접근 제한자:** 같은 패키지 또는 자식 클래스에서 사용 가능
- **private 접근 제한자:** 외부에서 사용안됨
- **default 접근 제한:** 같은 패키지에 소속된 클래스에서만 사용 가능

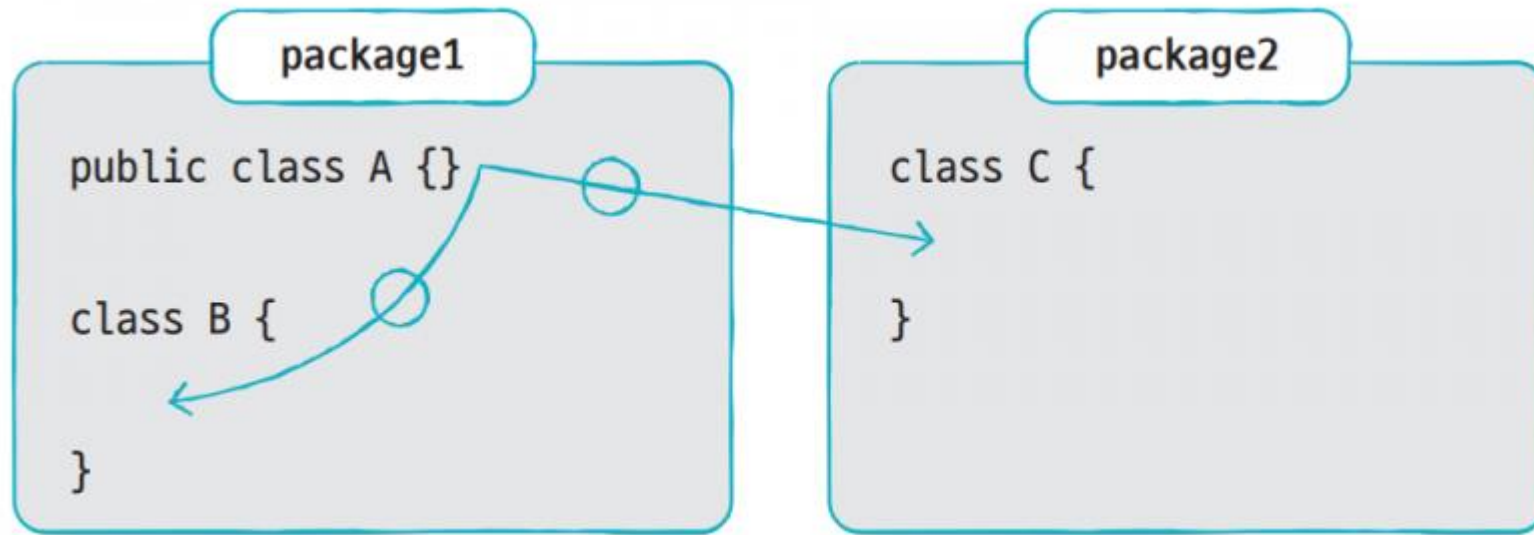
default 접근 제한자



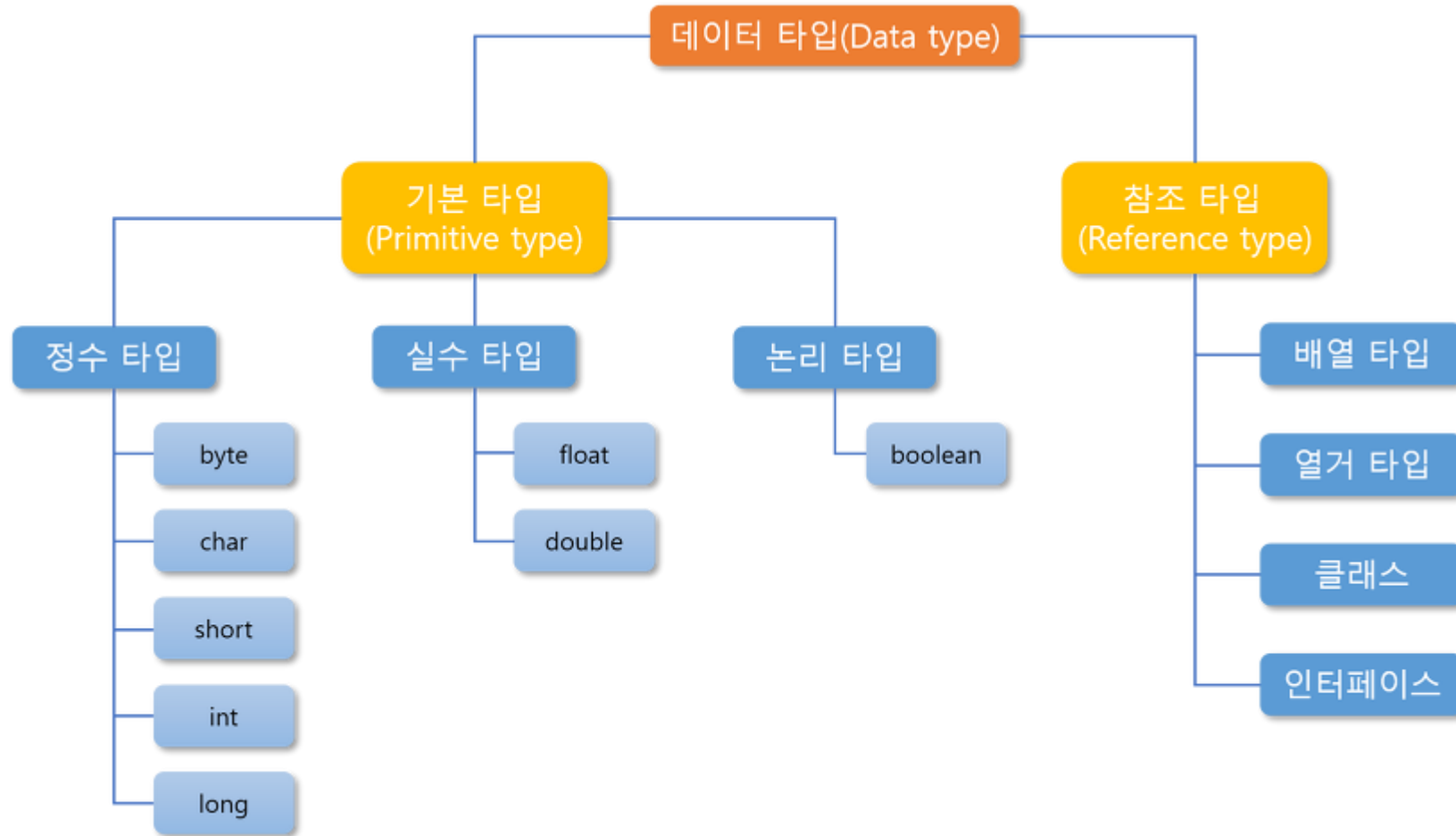
public 접근 제한자



public 접근 제한자



Data type



Data type

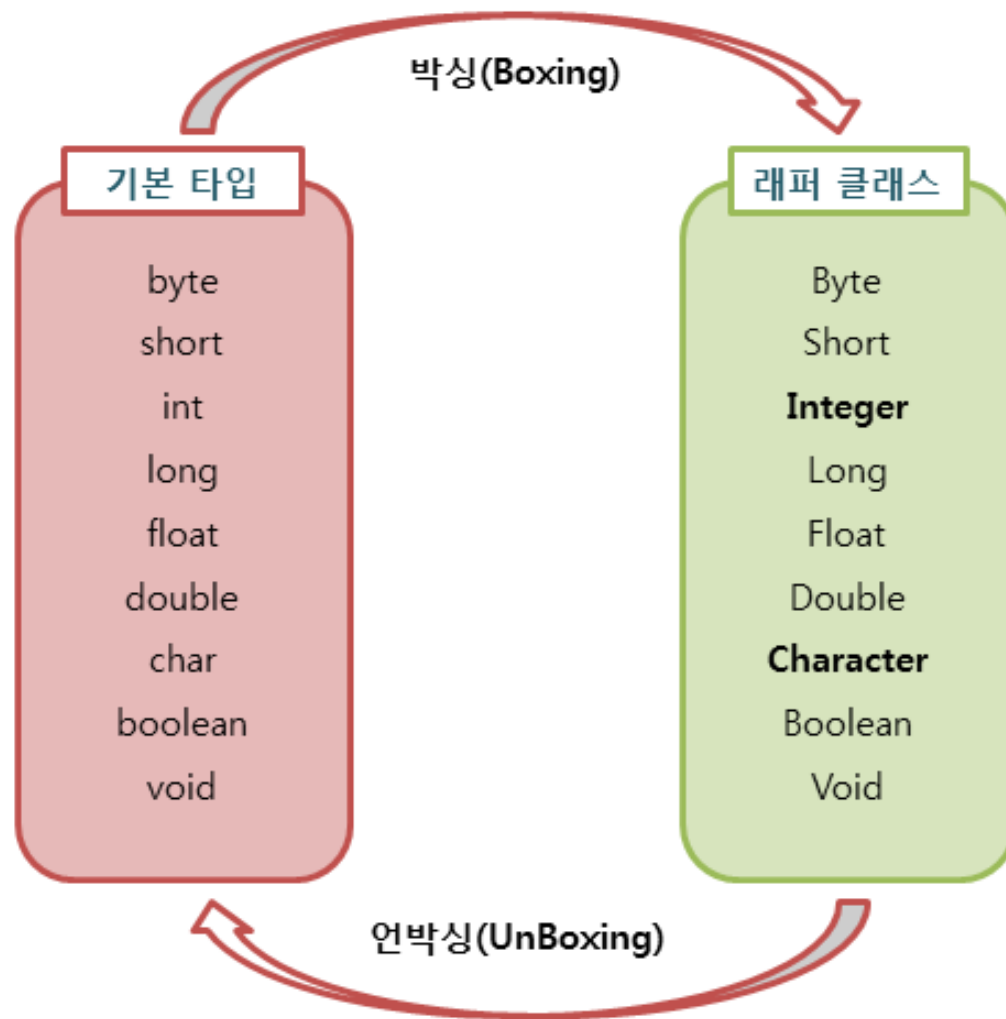
| 값의 종류 | 기본 타입 | 메모리 사용 크기 | | 저장되는 값의 범위 |
|-------|---------|-----------|--------|--|
| 정수 | byte | 1 byte | 8 bit | $-2^7 \sim (2^7 - 1)$ |
| | char | 2 byte | 16 bit | $0 \sim (2^{16} - 1)$, 유니코드 |
| | short | 2 byte | 16 bit | $-2^{15} \sim (2^{15} - 1)$ |
| | int | 4 byte | 32 bit | $-2^{31} \sim (2^{31} - 1)$ |
| | long | 8 byte | 64 bit | $-2^{63} \sim (2^{63} - 1)$ |
| 실수 | float | 4 byte | 32 bit | $(+/-)1.4E-45 \sim (+/-)3.4028235E38$ |
| | double | 8 byte | 8 bit | $(+/-)4.9E-324 \sim (+/-)1.7976931348623157E308$ |
| 논리 | boolean | 1 byte | 8 bit | true, false |

Boxing, Unboxing

객체화

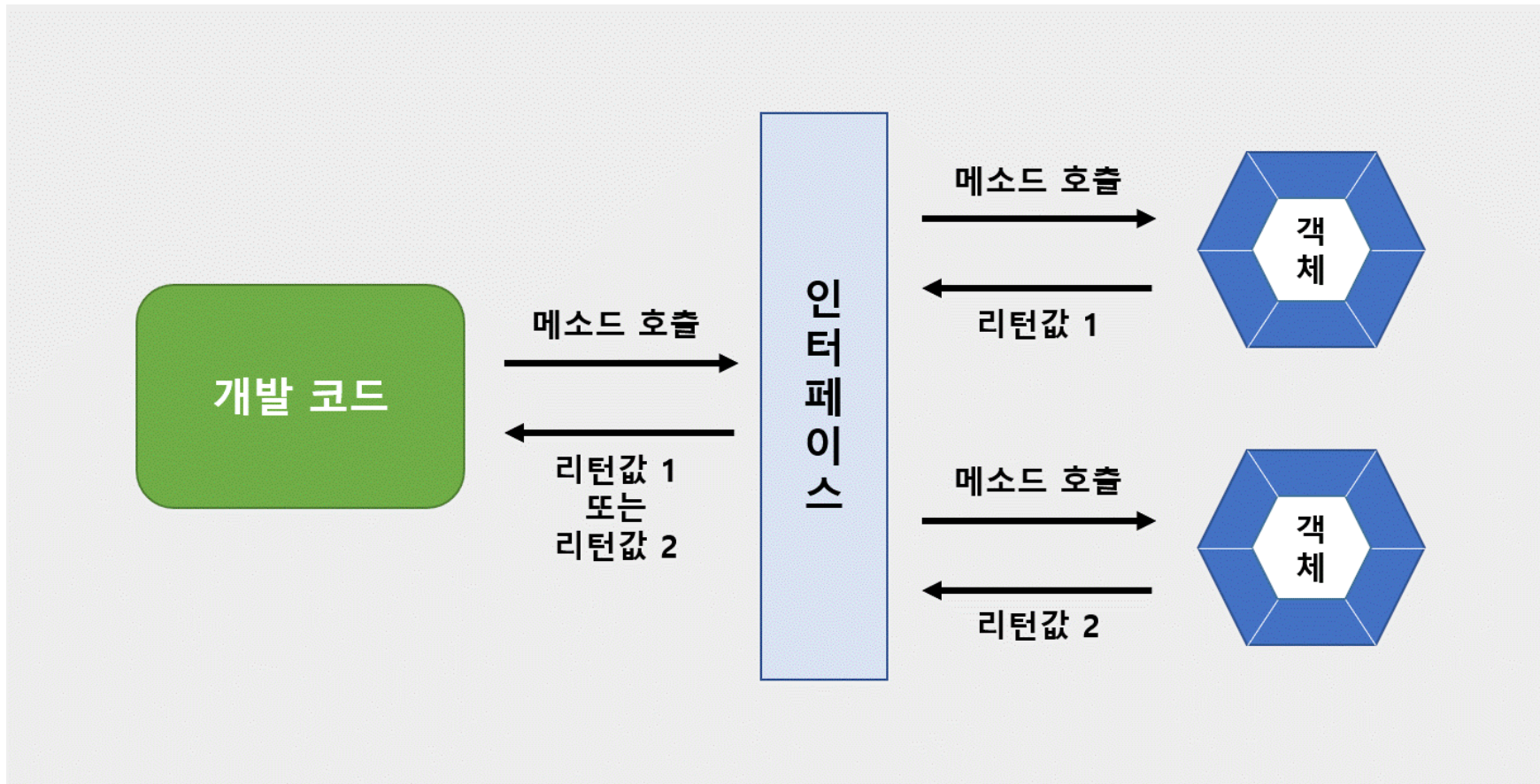


| 기본 자료형 | Wrapper 클래스 |
|---------|-------------|
| byte | Byte |
| char | Character |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |



Interface

- 자바는 다중상속을 지원하지 않음 → 이를 보완하기 위해 인터페이스를 통한 다중상속 지원
- 다른 클래스를 작성할 때 기본이 되는 틀을 제공
- 다른 클래스 사이의 중간 매개 역할 담당할 수 있음 → 일종의 추상 클래스



Generic

- data type을 일반화 시키는 문법

```
class Box<T>{  
    T box;  
}
```

타입매개변수(Type Parameter)

클래스 내부가 아닌 외부에서
자료형 지정

타입인자(Type Argument)

```
Box<String> box1 = new Box<String>();  
Box<Integer> box2 = new Box<Integer>();
```

매개변수화타입
(Parameterized Type)

고생하셨습니다 ^^*

