

JAVA

자바 소개



프로그램이란?

프로그램은 컴퓨터를 위한 작업 지시서로서 구체적으로 컴퓨터가 특정한 작업을 하기 위한 명령어들의 리스트이다.

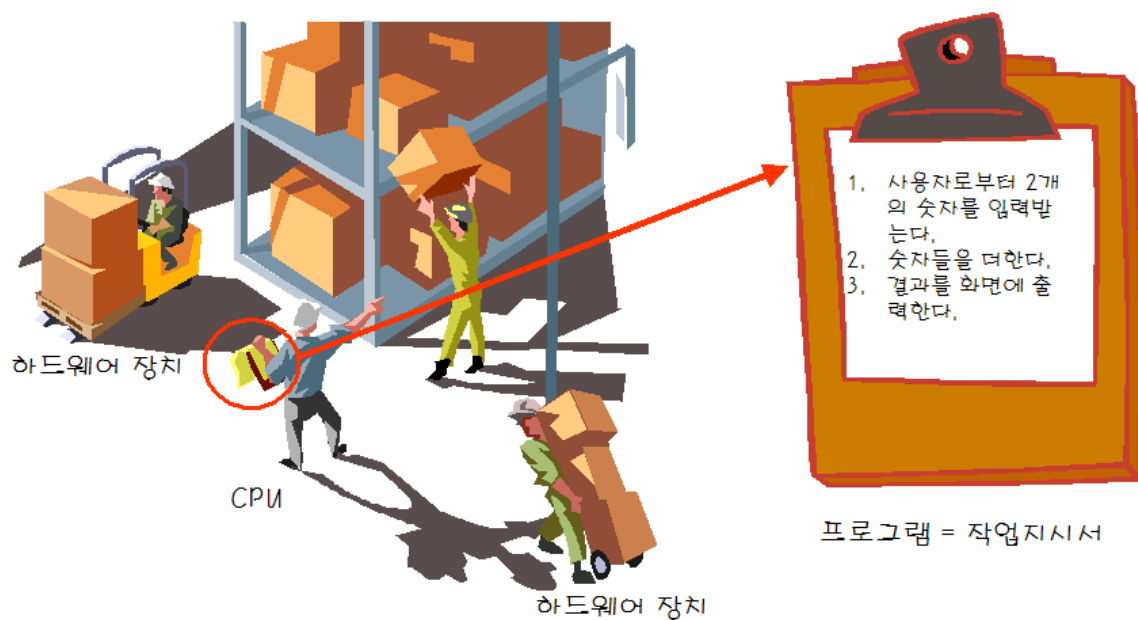


그림 1.1 프로그램은 작업 지시서와 같다.

I

프로그램은 컴퓨터상에서 어떤 일을 처리하려는 목적으로, 개발자가 프로그래밍 언어를 사용하여 어떻게 처리할 것인지를 기술한 것이다.



프로그램이란?

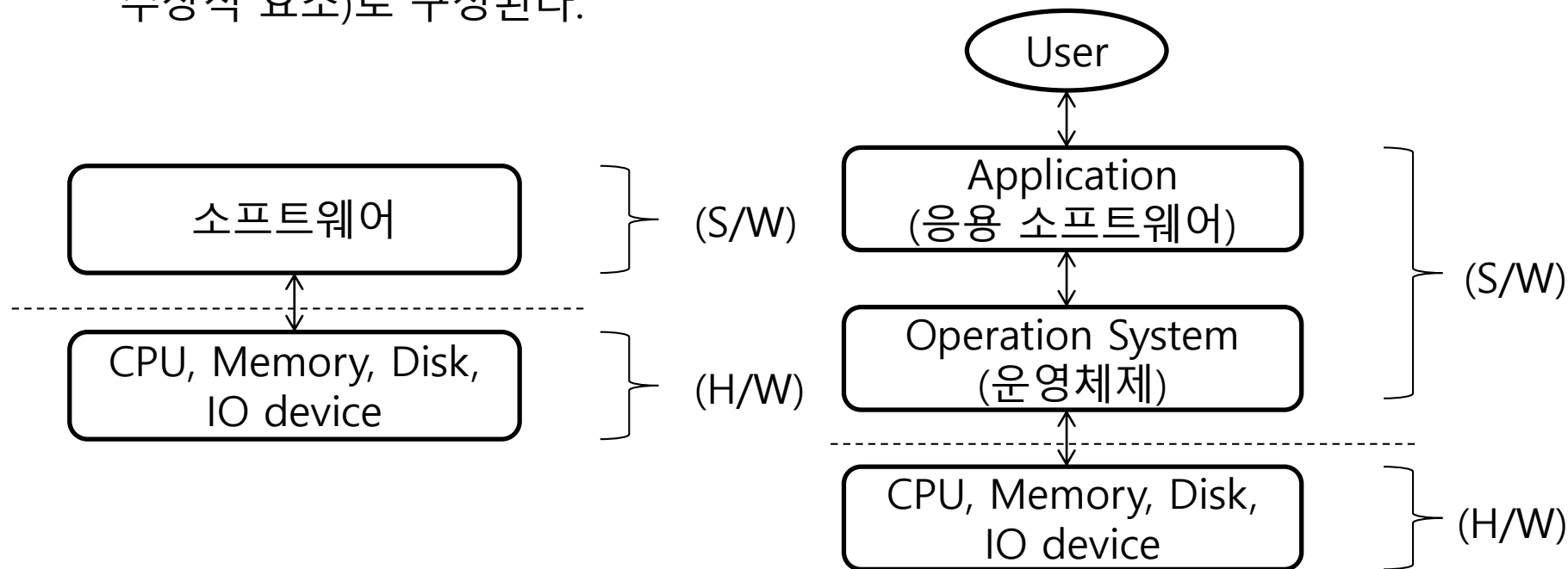
프로그램을 정의하자면 **컴퓨터를 구성하는 장치(CPU, 메모리, 입출력 장치, 하드 디스크 등)을 조작하여, 원하는 결과물을 얻도록 작업 순서를 나열해 놓은 명령어들의 집합이다.** 여기서 명령어(instruction)는 대화가 아니라, 일방적인 전달을 의미한다.

자바(JAVA)는 명령어를 표현하는 하나의 도구이다. 이를 **프로그램 언어(language)**라고 하며, C와 C++, C# 등이 있다. 정리하자면 개발자는 프로그램 언어를 사용해서 명령어를 표현하고, 표현된 명령어를 작업 순서에 맞게 나열하여 프로그램을 작성한다.

원하는 결과를 얻을 수 있도록 어떻게 명령어를 표현하고 나열해야 하는지, 즉 프로그램 작성법을 자바라는 언어를 통해 배우게 될 것이다.

컴퓨터의 구성 요소

컴퓨터는 크게 **하드웨어**(실체가 있는 요소)와 **소프트웨어**(형태가 없는 추상적 요소)로 구성된다.

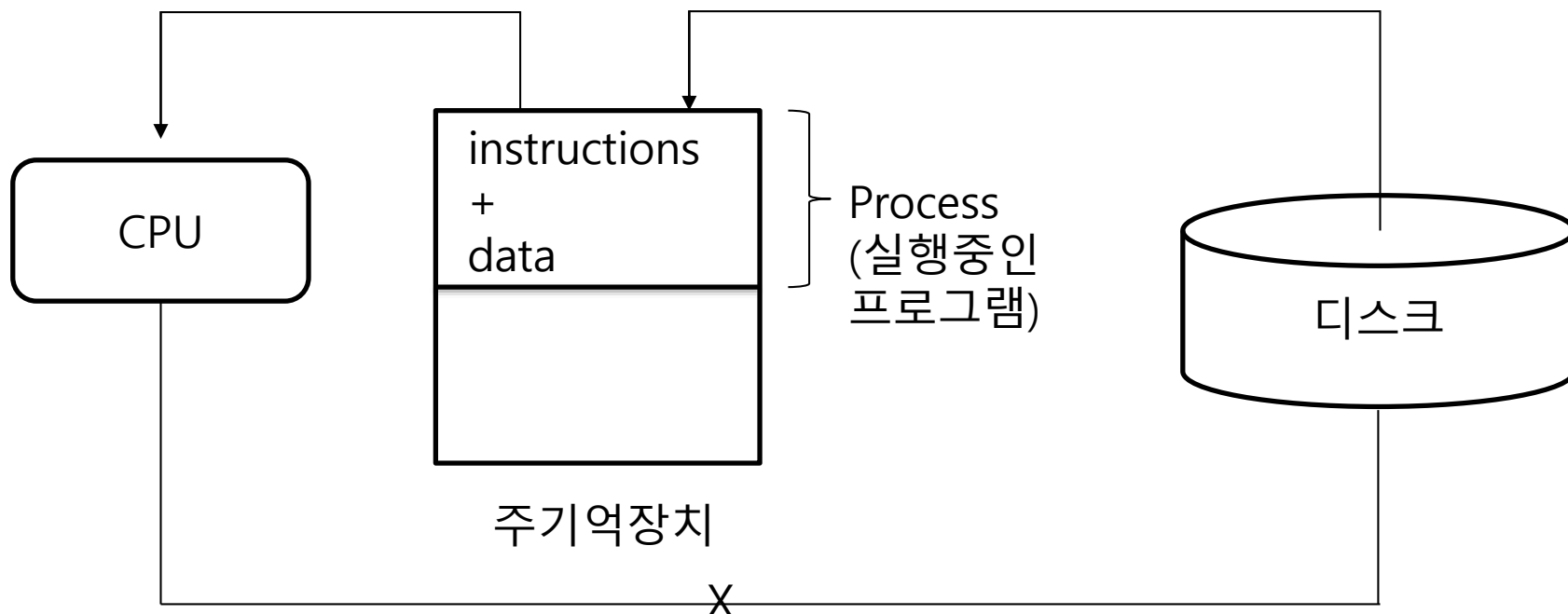


소프트웨어 모델이 구조화되면서 소프트웨어는 하드웨어와 상호 작용하는 운영체제와 같은 **시스템 소프트웨어**와 특정 작업을 보다 편리하게 처리할 목적으로 만드는 **응용 소프트웨어**로 나뉘게 된다.

우리가 개발하려는 프로그램은 사용자가 편리하게 기능을 수행할 수 있도록 작성하는 프로그램 즉 응용 소프트웨어(프로그램)에 해당한다.

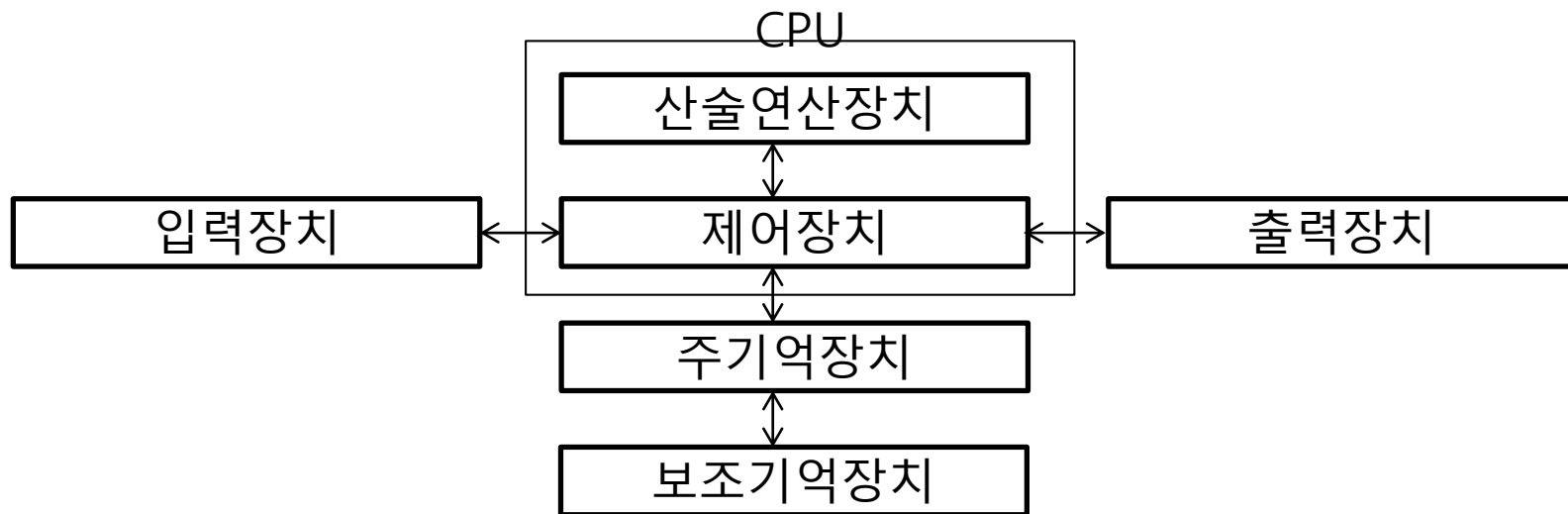
메모리와 프로세스의 관계

명령은 반드시 CPU에 의해 실행되어야 결과가 나온다. **명령이 CPU에 의해 실행되려면 반드시 명령이 주기억 장치(메모리)에 적재되어야 한다.** 명령은 주기억 장치에 적재되기 전에는 하드웨어에 파일 형태로 저장된다. 이것을 프로그램이라 한다. 프로그램은 명령어들의 집합이라고 했다. 그런데 **프로그램을 주기억장치(메모리)에 적재하여 실행하면 프로세스**라 한다. 다시 말해서 프로세스는 실행 중인 프로그램이라 할 수 있다. 결국, 프로그램과 프로세스는 주기억 장치에 적재되었는지 여부로 구분한다.



메모리와 프로세스의 관계

주기억 장치에 적재된 명령들의 집합을 실행하는 주체는 CPU이다. CPU가 실행하는 명령어들을 나열한 것이 프로그램이다. **CPU는 제어장치와 연산장치로 구성되어 있다.** 즉 사칙 연산은 CPU의 연산 장치가 수행하며, 주변 장치의 상호 작용은 제어장치를 통해 이루어진다.



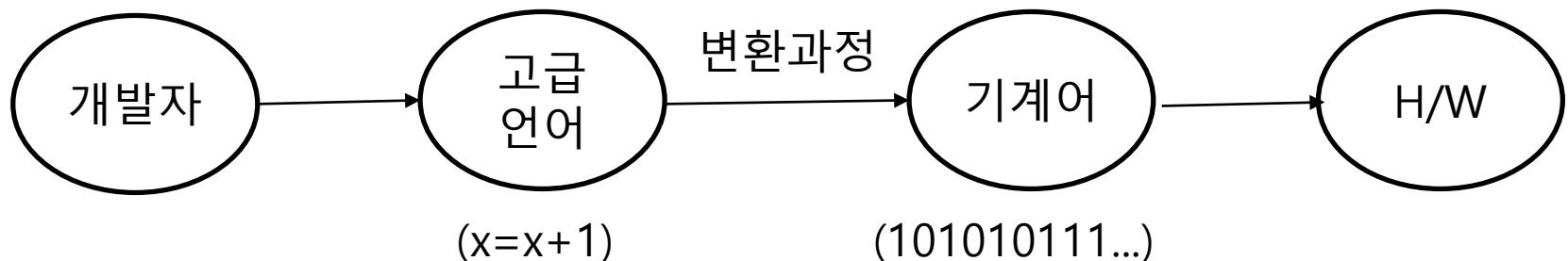
이러한 제어 순서로 해당 명령어를 나열하면 프로그램이 되는 것이다. 그러면 명령어를 어떻게 표현해야 할까? 이러한 명령어를 표현하는 도구가 프로그램 언어이다. 자바(JAVA)라는 표현도구(프로그램 언어)를 이용해서 프로그램을 작성할 것이다.

프로그램 언어

프로그램 언어는 명령어를 표현하는 도구라고 했다. 대개 **고급언어**와 **저급언어**로 구분한다. 프로그램을 구성하는 명령어가 하드웨어에 전달되려면 2진수 형태(010101...)로 저장되고서, 최종적으로는 전기 신호로 변환되어야 한다.

소프트웨어 초기에는 프로그램을 하드웨어에 적합한 2진수 형태의 기계어로 작성했다. 하지만 프로그램을 편리하게 작성하기 위해서는 사람이 이해하기 쉬운 문자 형태의 명령어가 필요했다. 그래서 사람이 이해하기 쉬운 프로그램 언어를 **고급언어**라 하며 반대로 형태가 기계어에 가까운 프로그램 언어를 **저급언어**라 한다

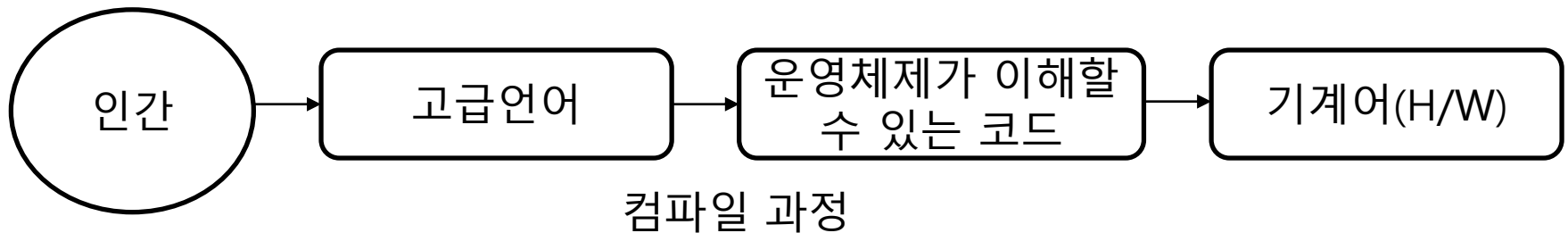
고급언어를 사용해서 명령어를 작성하면 사람은 편리하지만, 컴퓨터는 이해할 수 없는 형태이므로 반드시 변환 과정을 거쳐 기계어로 만들어야 한다. 이러한 **변환 과정을 컴파일(compile)이라 한다**. 컴파일을 통해 변환된 프로그램은 기계어를 통해 하드웨어에 전달되어 실행된다. 그리고 원하는 결과를 내게 된다.





프로그램 언어

응용프로그램(응용 소프트웨어)을 사람의 언어와 형태가 닮은 언어(고급언어)로 작성한 후, 운영체제가 이해하는 코드로 변환해서 운영체제에 전달한다. 그러면 운영체제는 다시 하드웨어(기계어)가 이해하는 수준의 코드(기계어)로 변환한다.



자바는 형식이 사람의 언어에 가깝다. 그래서 고급언어에 속한다. 그렇다면 자바를 이용해서 프로그램을 작성하면 반드시 변환 과정을 거쳐야 한다.

운영체제가 윈도우를 사용하고 있다면 자바로 작성한 프로그램은 윈도우가 이해하는 코드로 변환해서 전달해야 한다. 리눅스라면 리눅스에 맞게 변환해야 한다. 다행히 변환을 위해 **JDK(자바개발환경)**에 포함되어 제공되고 있으니 여러분은 변환을 걱정하지 않아도 된다.

컴퓨터가 이해하는 언어

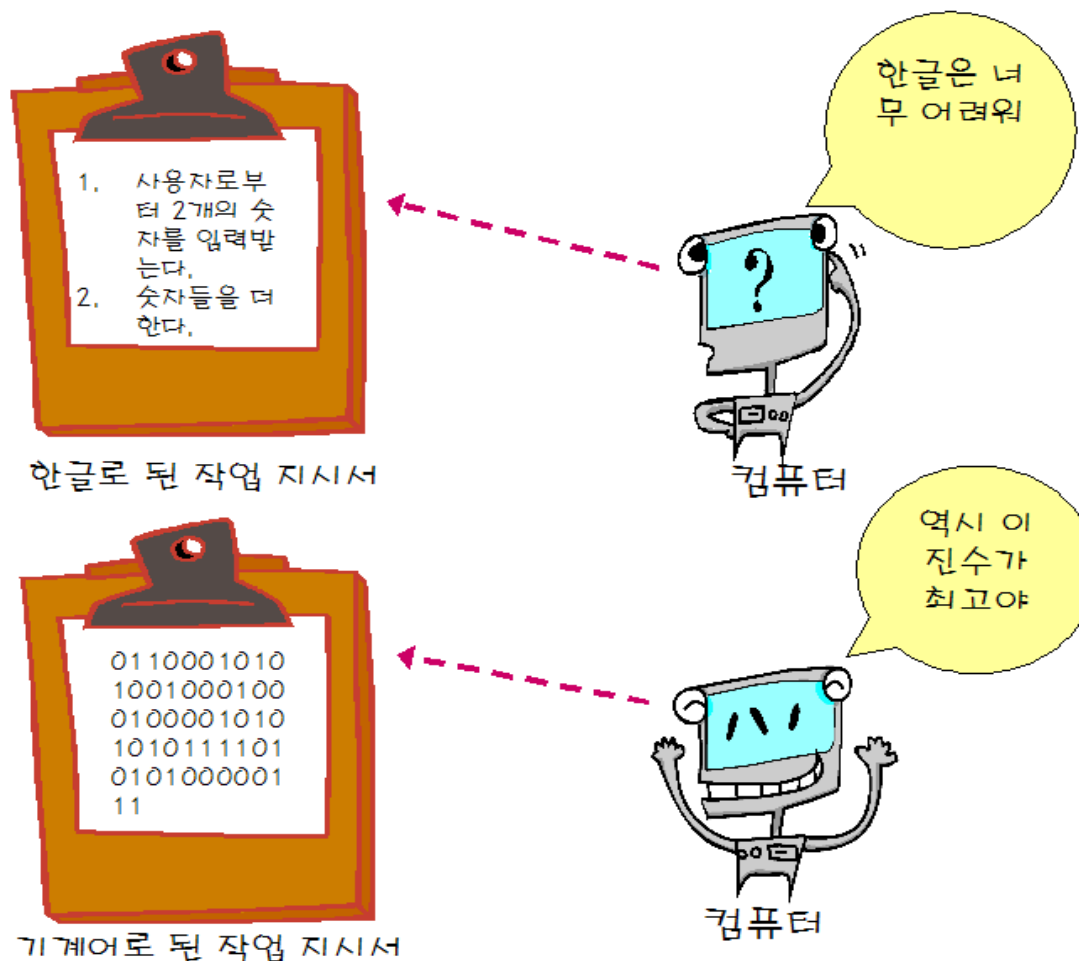


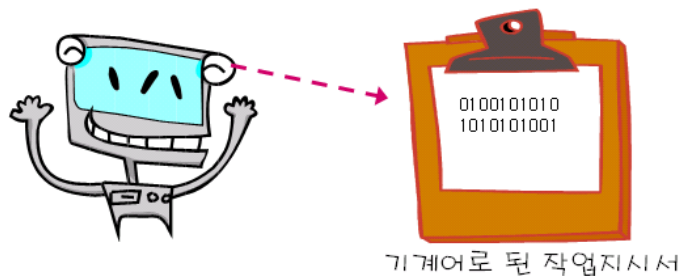
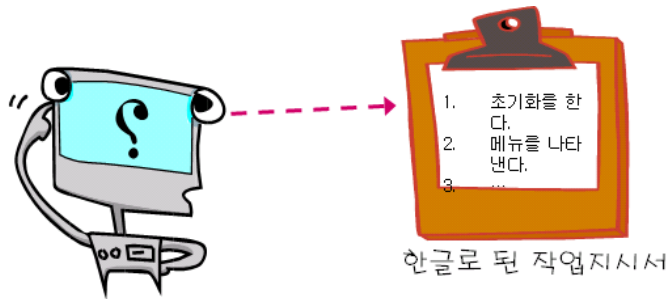
그림 1.2 컴퓨터는 한글로 된 작업 지시서는 이해하지 못하는 반면 기계어로 된 작업 지시서는 이해할 수 있다.

기계어

Q) 컴퓨터가 이해할 수 있는 언어는 어떤 것인가?

A) 컴퓨터가 알아듣는 언어는 한가지이다. 즉 0과 1로 구성되어 있는 “001101110001010...”과 같은 기계어이다.

A) 컴퓨터는 모든 것을 0과 1로 표현하고 0과 1에 의하여 내부 스위치 회로들이 ON/OFF 상태로 변경되면서 작업을 한다.

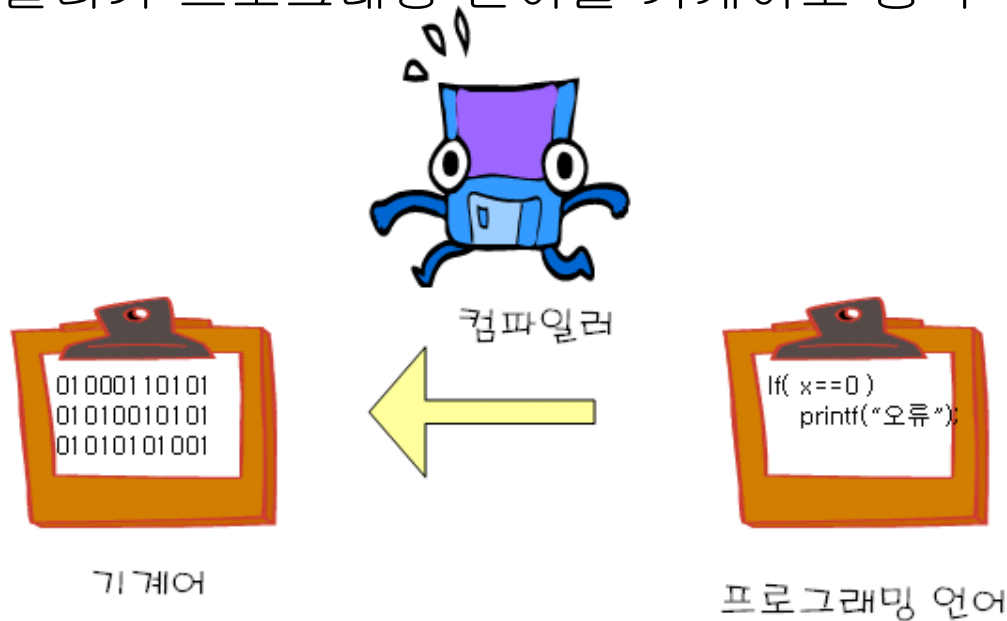


기계어는 프로그램이 CPU에 내리는 명령들을 표현하며, 이러한 명령들은 실제로 0과 1로만 이루어져 있다.

프로그래밍 언어의 필요성

Q) 그렇다면 인간이 기계어를 사용하면 어떤가?


- 기계어를 사용할 수는 있으나 이진수로 프로그램을 작성하여야 하기 때문에 아주 불편하다.
- 프로그래밍 언어는 자연어와 기계어 중간쯤에 위치
- 컴파일러가 프로그래밍 언어를 기계어로 통역



자바는 프로그래밍 언어의 일종

프로그램을 개발할 때 가장 먼저 프로그래밍 언어로 명령들을 구현한다.

구현이 완료되면 파일로 저장해야 한다. 이 파일을 "소스파일(Source File)"이라고 한다.



0100011010
1010100101
0101010101
0010111111

기계어

if(x > 0)
x--;
else
x = 100;

프로그래밍 언어

시작이 좋으면
끝도 좋다.
...

한국어



번역가

A good
beginning
makes a
good ending.
...

영어

프로그래밍 언어의 분류

기계어(machine language)
어셈블리어(assembly language)
고급 언어(high-level language): 자바, C++, C언어

- 컴퓨터가 이해하는 단 하나의 언어
- 기계어는 프로그램이 CPU에 내리는 명령들을 표현
- '0' 과 '1'의 숫자로 구성

- 인간이 이해하기에 용이한 명령문이나 기호를 사용하여 구성된 프로그래밍 언어

- 기계어에 해당되는 명령을 기호를 이용해서 나타낸 기호 언어
- 어셈블리어는 기본적으로 기계어와 일대일로 대응되며,
- 기계 중심적인 언어로서 프로그램 시에 하드웨어에 대한 지식이 요구됨



컴퓨터

기계어

어셈블리어

고급언어



인간

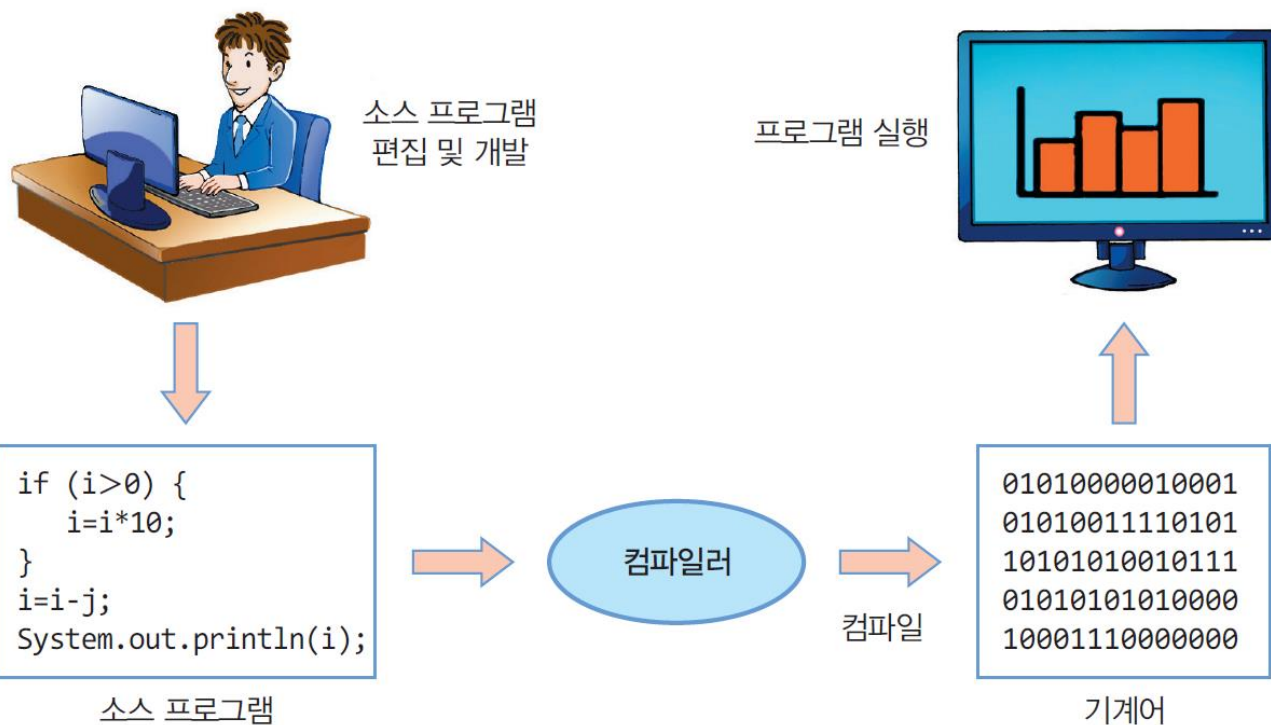
컴파일

소스 : 프로그래밍 언어로 작성된 텍스트 파일

컴파일 : 소스 파일을 컴퓨터가 이해할 수 있는 기계어로 변환하는 과정

- 소스 파일 확장자와 컴파일 된 파일의 확장자

- 자바 : .java -> .class
- C : .c -> .obj -> .exe
- C++ : .cpp -> .obj -> .exe



중간 점검

1. 컴퓨터가 직접 이해할 수 있는 단 하나의 언어는 기계어이다.
2. 프로그래밍 언어를 기계어로 변환시켜주는 프로그램을 컴파일러라 한다





자바의 역사

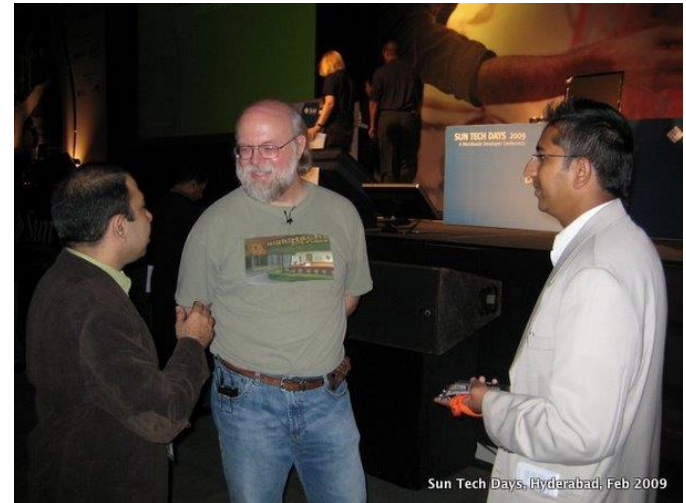
자바는 선마이크로시스템즈의 제임스 고슬링(James Gosling)이 이끄는 연구팀이 개발했다. 자바의 전신은 1991년 그린 프로젝트로 탄생한 객체 지향 언어인 오크(Oak)이다.

오크는 냉장고, TV 등 가전제품에 장착한 컨트롤러 칩에 각종 기능을 추가하는 프로그래밍 언어로 사용할 예정이었다. 하지만 당시 가전제품용 메모리나 컨트롤러 칩이 프로그램을 구동할 만큼 성능을 갖추지 못했기에 시장에서는 시기상조였다. 또 선마이크로시스템즈의 주요 계약이 다른 회사로 넘어가는 등 그린 프로젝트가 많은 어려움에 봉착했었다.

그러나 인터넷과 웹이 엄청난 속도로 발전하면서 오크에 활력을 불어넣기 시작했다. 선마이크로시스템즈에서는 인터넷 환경에 적합하도록 오크를 새롭게 설계한 후 1995년 자바라는 이름으로 발표했다. 2009년 오라클이 선마이크로시스템즈를 인수하면서 현재는 오라클이 자바 플랫폼을 지원한다.

자바는 누가 만들었을까?

James Gosling(제임스 고슬링)



자바 언어의 전 세계적인 활용도

TIOBE 인덱스(www.tiobe.com/tiobe-index)

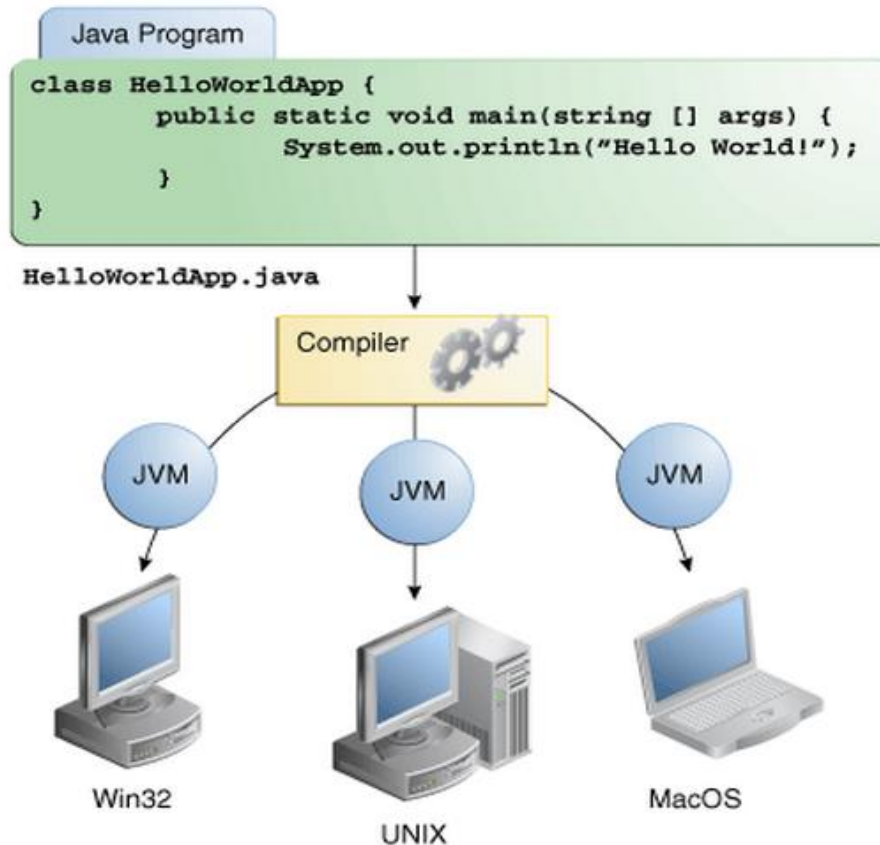
– 프로그래밍 언어의 인기 순위를 매기는 사이트

자바는 지난 10년 동안 1위

Dec 2020	Dec 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.48%	+0.40%
2	1	▼	Java	12.53%	-4.72%
3	3		Python	12.21%	+1.90%
4	4		C++	6.91%	+0.71%
5	5		C#	4.20%	-0.60%
6	6		Visual Basic	3.92%	-0.83%
7	7		JavaScript	2.35%	+0.26%
8	8		PHP	2.12%	+0.07%
9	16	▲▲	R	1.60%	+0.60%
10	9	▼	SQL	1.53%	-0.31%
11	22	▲▲	Groovy	1.53%	+0.69%
12	14	▲	Assembly language	1.35%	+0.28%
13	10	▼	Swift	1.22%	-0.27%
14	20	▲▲	Perl	1.20%	+0.30%
15	11	▼	Ruby	1.16%	-0.15%
16	15	▼	Go	1.14%	+0.15%
17	17		MATLAB	1.10%	+0.12%
18	12	▼	Delphi/Object Pascal	0.87%	-0.41%
19	13	▼	Objective-C	0.81%	-0.39%
20	24	▲	PL/SQL	0.78%	+0.04%

자바의 가장 큰 장점

자바는 동일한 프로그램이 다양한 컴퓨터에서 실행이 가능하다는 점이다. (**Write once, Run everywhere!**)



기존의 언어들의 실행 방식

윈도우에서 개발된 C 프로그램

```
#include <stdio.h>
int main(void)
{
    printf("Hello World! \n");
}
```



컴파일러



윈도우



리눅스



MacOS

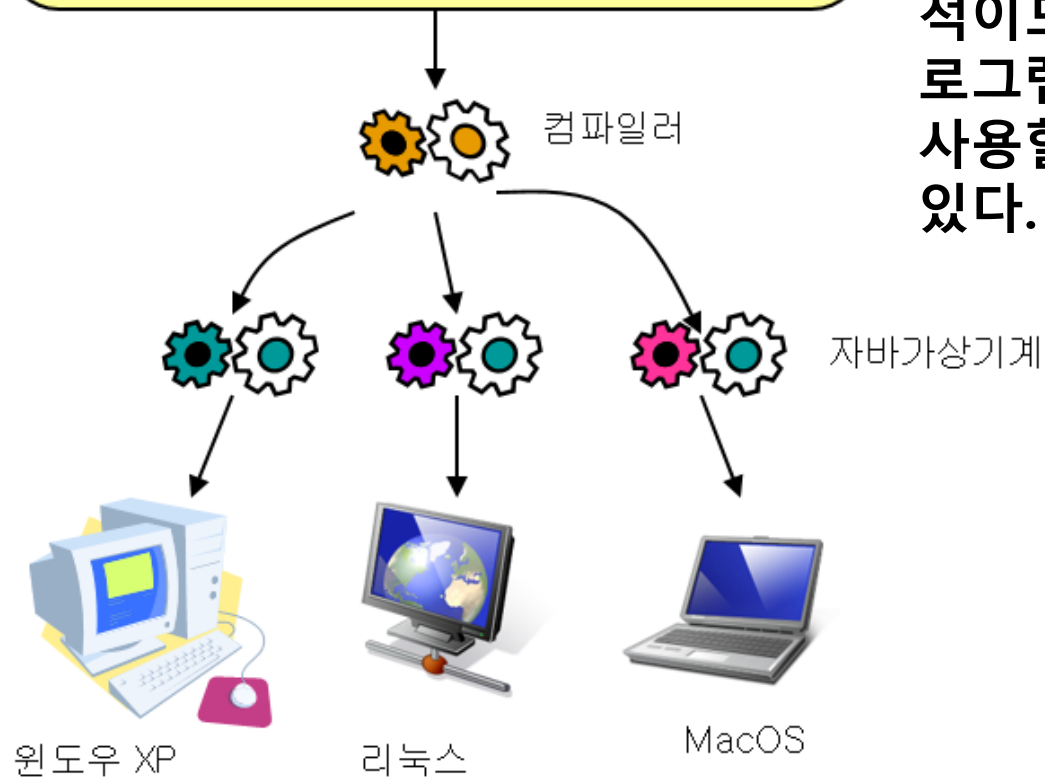
자바 가상 기계

자바 프로그램

```
class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Write Once!

JAVA 언어는 운영체제에 독립적이도록 설계되었다. 한번 프로그램을 작성하면 어디서든 사용할 수 있다는 원리를 담고 있다.



Run Everywhere!

자바 가상 기계(Java Virtual Machine)

자바 컴파일러는 특정한 컴퓨터가 아닌 가상 머신(Virtual Machine)을 위한 코드를 생성한다.

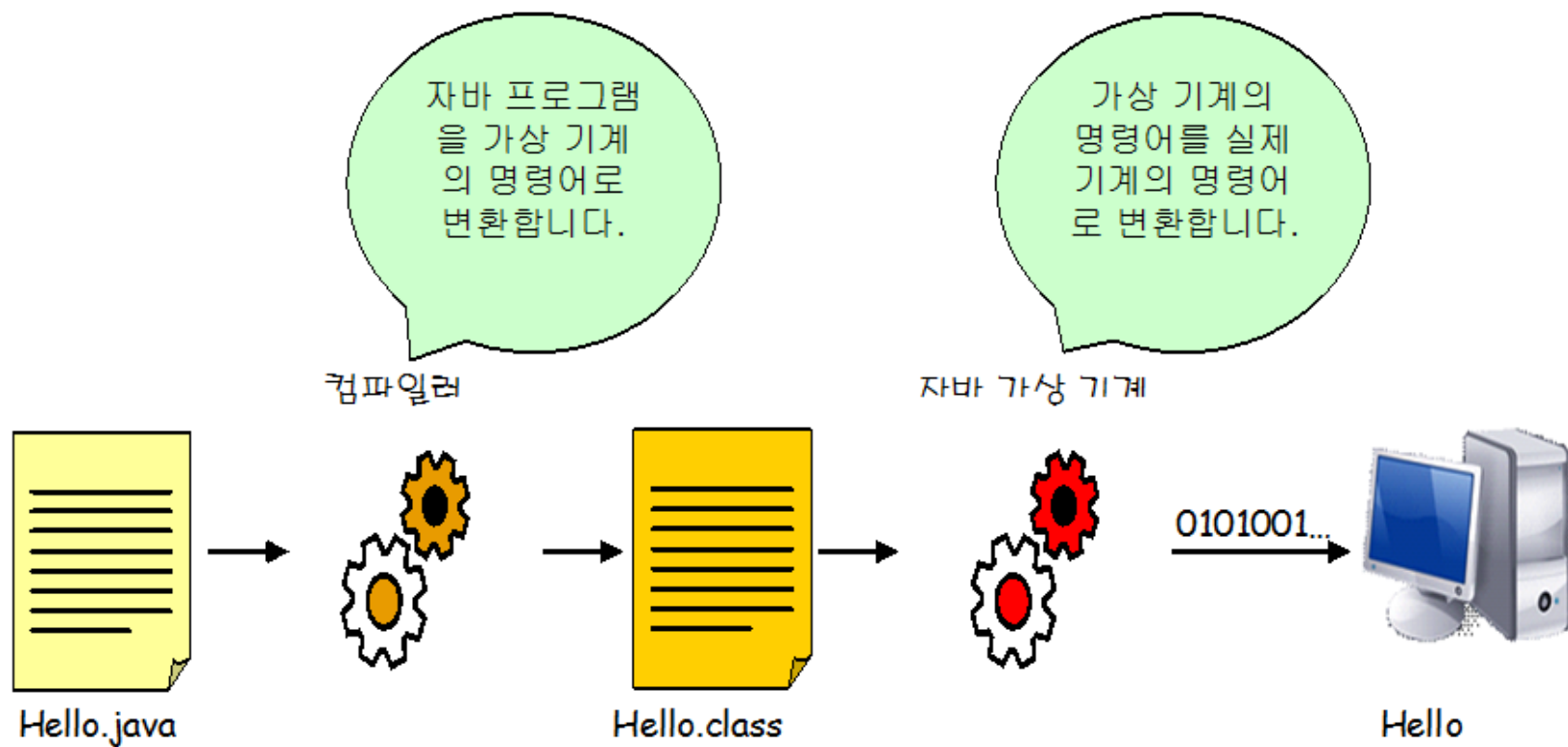
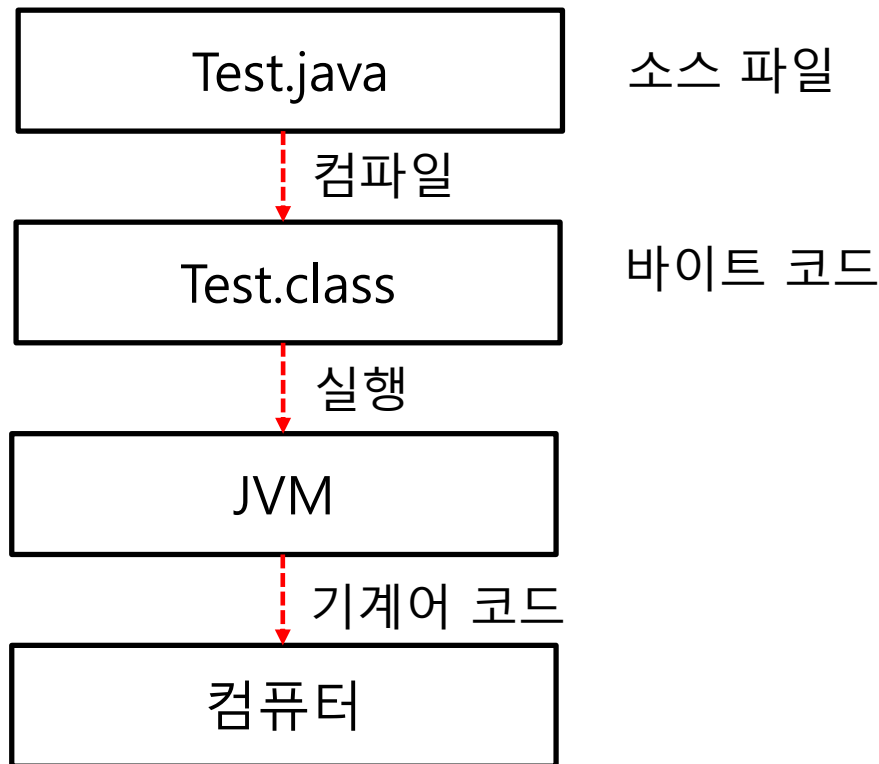


그림 1.4 자바의 실행 과정

자바 가상 기계

자바 소스 파일이 컴파일 작업을 거쳐 생성된 파일은 컴퓨터에서 실행 가능한 기계어 코드가 아니라 '**바이트 코드(byte code)**'이다. 바이트 코드는 기계어로 변환하기 전 중간 단계의 코드이다.

바이트 코드를 기계어로 변환하여 자바 프로그램을 실행하게 된다. 이와 관련된 모든 작업은 '**자바 가상 머신(JVM Java Virtual Machine)**'에서 담당한다.



바이트 코드

자바 프로그램

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
    }  
}
```

바이트 코드

Compiled from "Hello.java"

```
public class Hello extends java.lang.Object{  
    public Hello();
```

Code:

0: aload_0

1: invokespecial #1; //Method java/lang/Object.<init>:()V

4: return

```
public static void main(java.lang.String[]);
```

Code:

0: getstatic #2; //Field java/lang/System.out:Ljava/io/PrintStream;

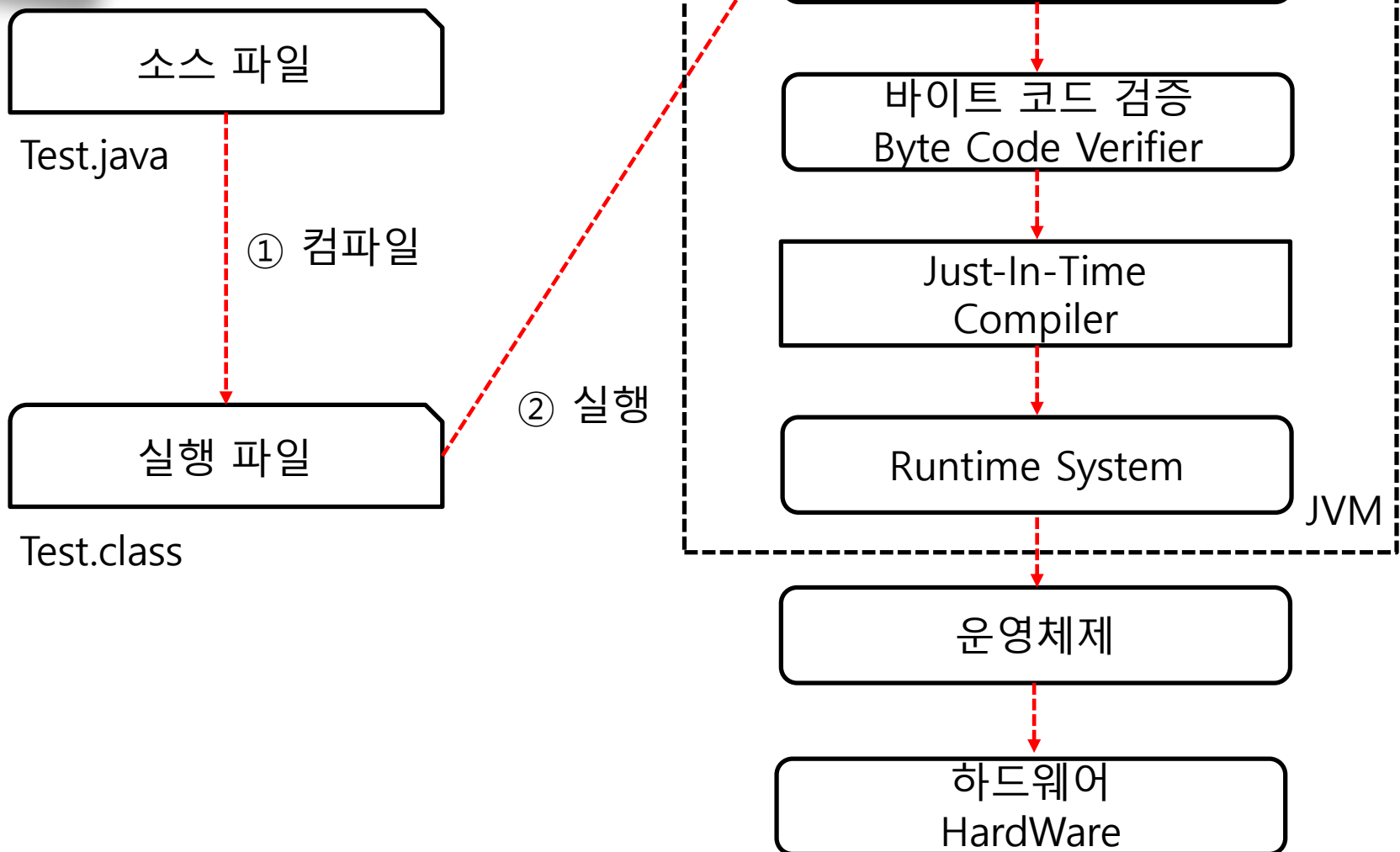
3: ldc #3; //String Hello World!

5: invokevirtual #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V

8: return

```
}
```

JVM 실행 환경





JVM 실행 환경

(1) 클래스 로드(Class loader)

자바 프로그램이 실행될 때 가장 먼저 클래스 로드가 동작한다. 클래스 로드는 실행에 필요한 모든 실행 파일(.class)을 찾아준다.

(2) 바이트 코드 검증(Byte Code Verifier)

이 파일의 코드가 올바른지 검증한다.

(3) JIT 컴파일러

기계어로 변환하는 방식에서 인터프리터(Interpreter) 방식과 JIT(Just-In-Time) 방식이 있다. 인터프리터 방식은 기계어로 변환하는 작업을 명령문 단위로 처리하고, JIT 방식은 소스 파일을 실행 파일로 변환하는 것처럼 파일 전체를 한번에 기계어로 변환한다. 자바 초기에는 인터프리터 방식을 사용했는데 이 방식은 처리 속도가 느리다는 단점이 있다. JIT는 이를 보완하기 위해서 나온 방식으로 미리 컴파일해 놓고 실행하므로 처리 속도가 빠르다.



WORA(정리)

WORA(Write Once Run Anywhere)

- 한번 작성된 코드는 모든 플랫폼에서 바로 실행
- C/C++ 등 기존 언어가 가진 플랫폼 종속성 극복
 - OS, H/W에 상관없이 자바 프로그램이 동일하게 실행
- 네트워크에 연결된 어느 클라이언트에서나 실행
 - 웹 브라우저, 분산 환경 지원

WORA를 가능하게 하는 자바의 특징

- 바이트 코드(byte code)
 - 자바 소스를 컴파일한 목적 코드
 - CPU에 종속적이지 않은 독립적인 코드
 - JVM에 의해 해석되고 실행됨
- JVM(Java Virtual Machine)
 - 자바 바이트 코드를 실행하는 자바 가상 기계(소프트웨어)



자바 플랫폼

자바 플랫폼은 JAV A API(Application Programming Interface)와 JVM(JAVA Virtual Machine)으로 구성된다. 여기서 플랫폼이란 프로그램을 개발하거나 실행하는 하드웨어나 소프트웨어 환경을 말한다. 자바 플랫폼은 자바 프로그램을 개발하고 실행하기 위한 환경이다.

JVAV API

응용 프로그램이나 시스템 프로그램이 시스템 자원을 사용하려면 운영체제나 프로그램 언어가 제공하는 기능을 사용한다. 이를 기능을 제어하는 매개체가 API이다. 개발 관점에서 보면 JAVA API는 자바를 사용하여 프로그램을 쉽게 구현하도록 해주는 클래스 라이브러리의 집합이다.

JVM

개발 과정에서는 작성한 자바 프로그램(.java)은 자바 컴파일러(javac.exe)로 변환하여 바이트코드(.class)로 저장된다. 그리고 나서 JVM(java.exe)에 의해서 바이트 코드가 실행된다.

자바 플랫폼

자바 소스파일(*.java)

javac.exe(컴파일러)

컴파일된 파일(*.class)

java.exe(인터프리터)

JVM For OS

OS

javac.exe

자바 컴파일러이다
자바 컴파일러는 자바 프로그램을 바이트 코드로
컴파일한다

java.exe

자바 인터프리터이다
자바 인터프리터는 바이트 코드를 변환하여
실행한다

중간 점검

1. 자바 컴파일러가 소스 코드를 컴파일하면 바이트 코드가 생성된다.
2. 바이트 코드를 해석하여 실행하는 소프트웨어는 자바가상머신이다.
3. 자바가 어떤 컴퓨터에서나 실행이 가능한 이유는 무엇인가?

특정한 컴퓨터가 아닌 중간적인 코드를 생성하고 이것을 해석하여 실행하는 구조로 되어 있기 때문이다.

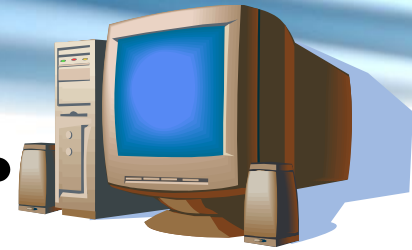


자바의 에디션

Java SE(Standard Edition)

Java EE(Enterprise Edition)

Java ME(Micro Edition)





자바의 에디션

Java SE(Standard Edition)

- 자바 표준 플랫폼, JVM 운영 환경과 API2에 대한 개발 환경을 갖추고 있다. 일반 데스크톱 환경에서 응용 프로그램을 개발할 때 사용.

Java EE(Enterprise Edition)

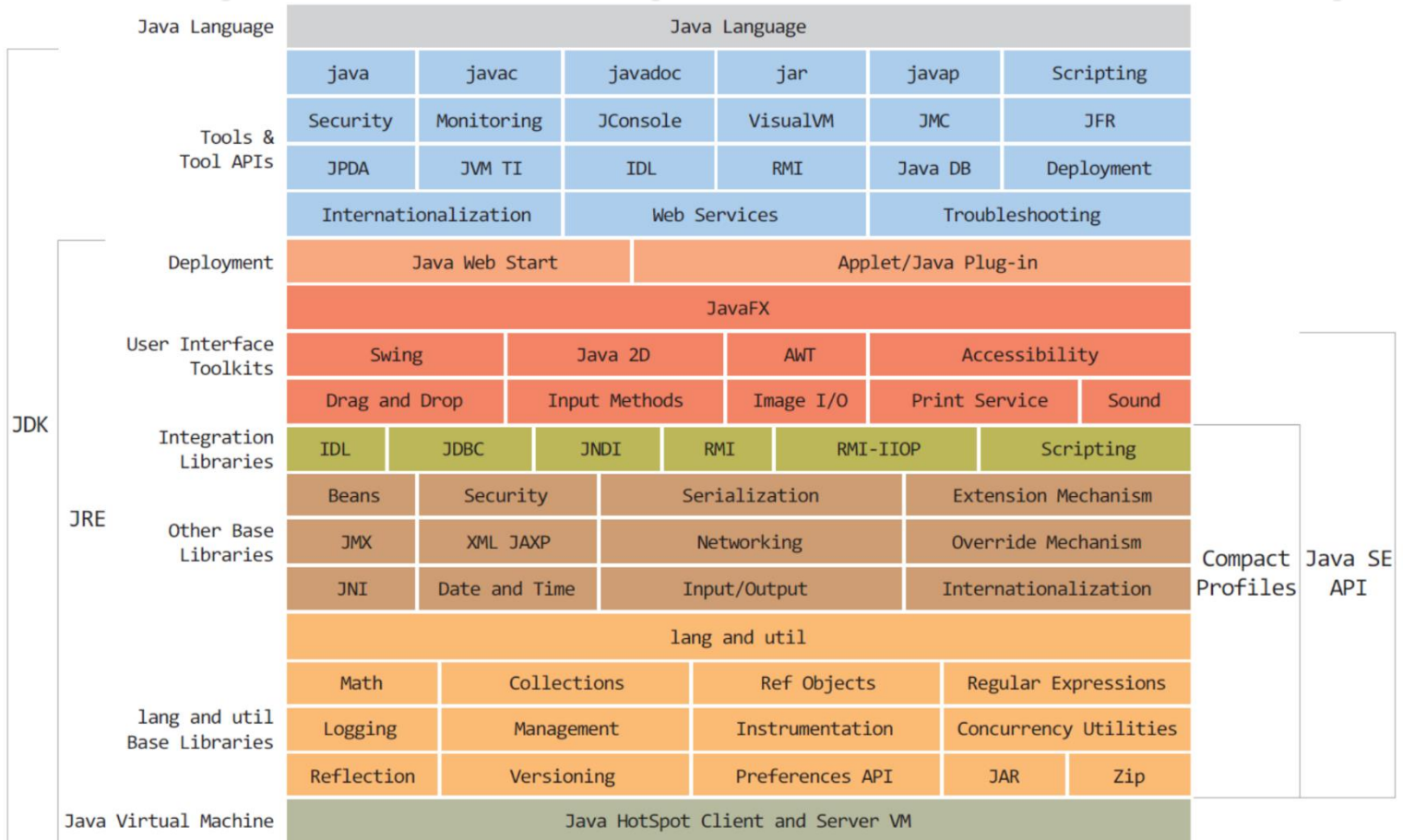
- 서버 기능 개발을 위한 플랫폼, SE를 바탕으로 여러 컴퓨터간의 분산처리와 서블릿/JSP 개발에 사용하는 환경.

Java ME(Micro Edition)

- 셋탑 박스와 같은 임베디드 장비를 위한 개발에 사용되는 플랫폼. 휴대폰이나 PDA, 프린터 등 소형 전자 제품의 응용 프로그램 개발할 때 사용.

Java SE

Java SE는 데스크탑과 서버에서 자바 애플리케이션을 개발하고 실행할 수 있게 해주며 임베디드 환경(embedded environment)과 실시간 환경(real-time environments)도 지원



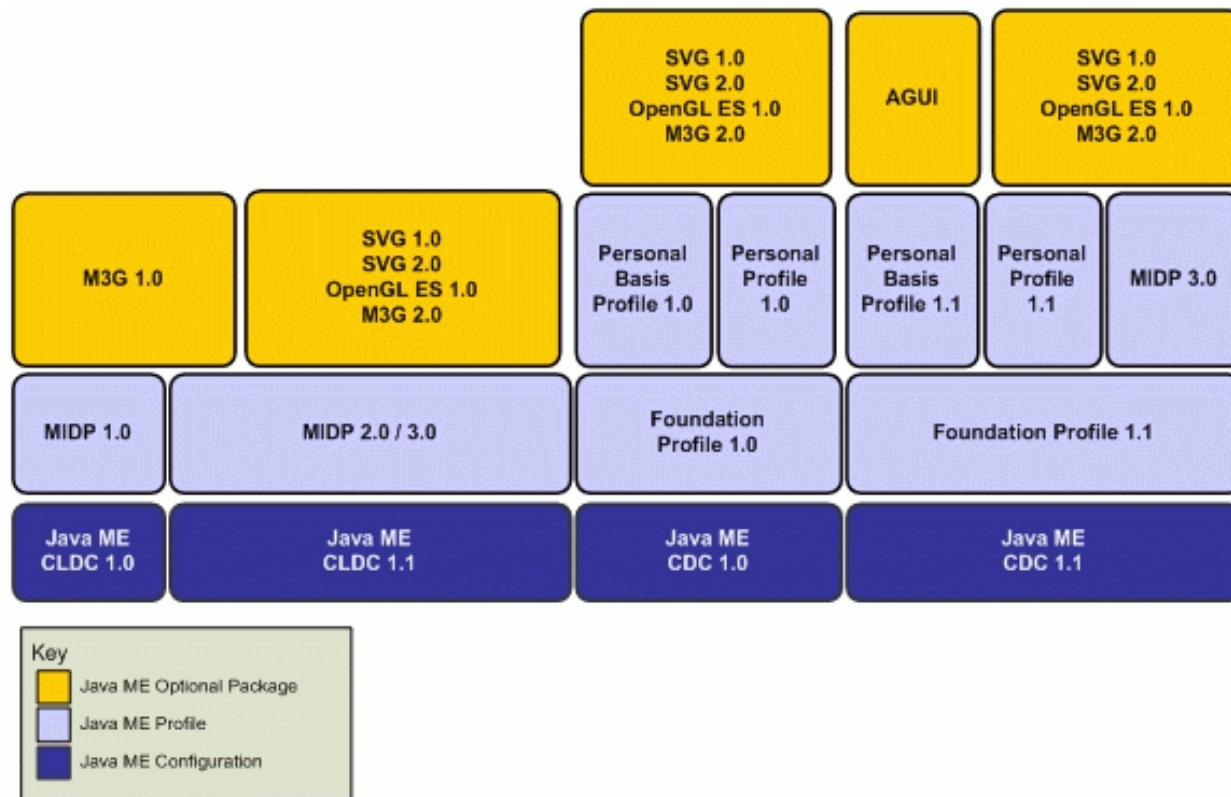


Java EE

Java EE는 기업용 애플리케이션을 개발하는 데 필요한 여러 가지 도구 및 라이브러리들을 모아 놓은 것
응용 서버, 웹서버, J2EE API, 엔터프라이즈 자바 빈즈 (JavaBeans) 지원, 자바 서블릿 API 와 JSP 등을 포함

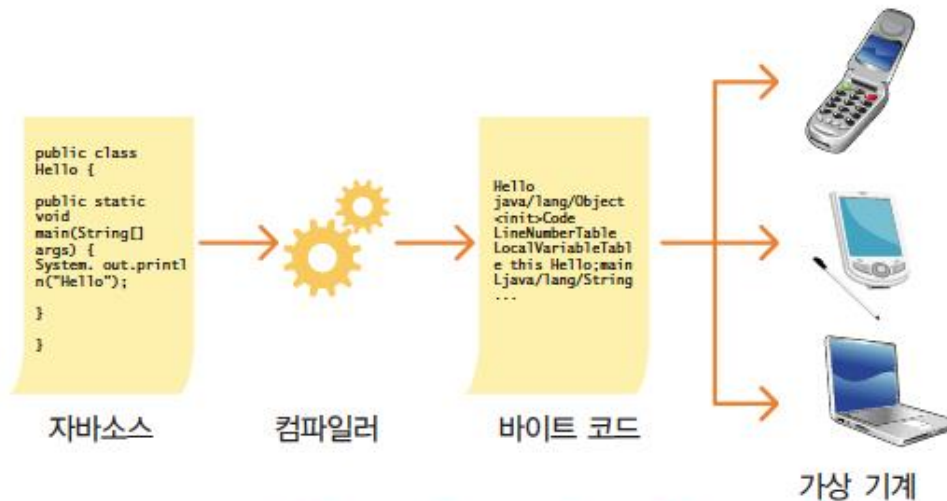
Java ME

Java ME는 PDA, TV 셉톱박스, 프린터와 같은 모바일 기기나 다른 임베디드 장치들에서 실행되는 애플리케이션을 위한 강인하고 유연한 환경을 제공



자바의 특징

자바를 기존의 다른 프로그래밍 언어와 달리 특색 있게 만드는 것은 인터넷 홈페이지 상에서 수행이 가능하고 어떤 컴퓨터 기종에서도 수행이 가능하다는 점이다. 이것을 잘 설명하는 문장이 "한번만 작성하고 모든 곳에서 수행시키세요(Write One, Run Everywhere)"라는 슬로건이다.



자바는 한번 작성하면
여러 가지 장치에서
실행할 수 있습니다.



Write Once, Run Everywhere!



자바의 특징

단순하지만 강력하다

- 꼭 필요로 하는 기능만을 포함시키고 복잡하고 많이 쓰이지 않는 기능은 삭제
- 포인터 연산, 연산자 오버로딩, 다중 상속 등의 복잡한 기능을 삭제
- 자동 메모리 관리 기능, 멀티 스레드, 방대한 라이브러리 제공

객체 지향적(Object Oriented)

- 객체별로 코드를 작성하고 객체들을 조합하여 전체 프로그램을 완성하는 프로그램 설계 방법론.
- 기본 데이터 타입을 제외한 거의 모든 것이 객체로 표현

분산 환경 지원

- 네트워크상에서 동작되는 것을 기본으로 설계
- 쉽게 네트워크 관련 프로그램을 개발



자바의 특징

견고하다

- 오류를 만들 수 있는 원인들을 제거
- (예) 포인터 개념을 삭제하였으며 컴파일시에 강력하게 데이터 타입을 검사

안전하다.

- 바이러스, 파일의 삭제나 수정, 데이터 파괴 작업이나 컴퓨터 오류 연산 등을 방지하면서 실행되도록 설계되었다.

컴퓨터 구조에 독립적이다.

- 컴퓨터 구조에 독립적인 바이트 코드로 번역
- 이러한 바이트 코드 특성 때문에 인터넷에 연결된 서로 다른 기종의 컴퓨터에서도 자바는 실행될 수 있다.



자바의 특징

이식성이 높다

- 서로 다른 실행 환경을 가진 시스템 간에 프로그램을 옮겨 실행할 수 있는 것을 말한다.
- 자바 언어로 개발된 프로그램은 소스파일을 다시 수정하지 않아도 자바 실행환경(JRE)이 설치되어 있는 모든 운영체제에서 실행 가능하다.

멀티스레딩 지원

- 자바는 언어 수준에서 멀티스레딩(multithreading)을 지원
- 멀티스레딩이란 많은 작업을 동시에 할 수 있음을 의미한다. 멀티스레딩을 이용하면 CPU를 유휴 시간없이 효율적으로 사용할 수 있다.

동적이다(Dynamic).

- 라이브러리들은 실행 파일에 영향을 끼치지 않고 자유롭게 새로운 기능들을 추가할 수 있다.
- 자바는 실행되기 직전에 라이브러리를 동적으로 링크하므로 실행할 때 변경된 라이브러리가 자동적으로 참조된다.

자바로 만들 수 있는 것

가장 전형적인 자바 응용프로그램

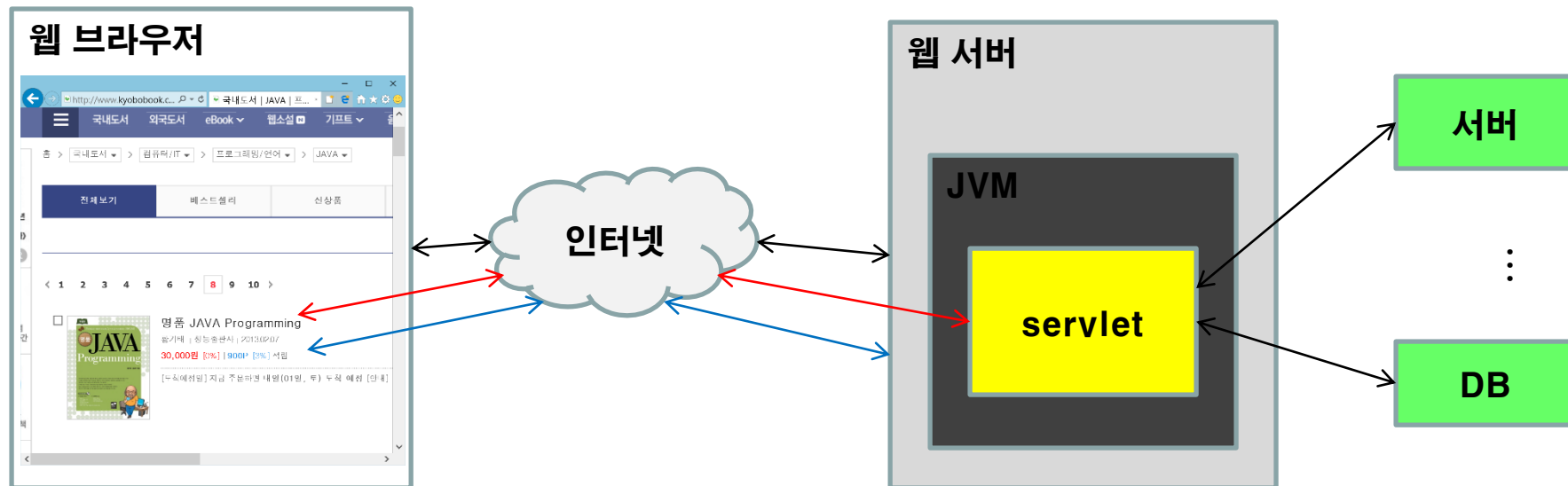
- PC 등의 데스크톱 컴퓨터에 설치되어 실행
- 자바 실행 환경(JRE)이 설치된 어떤 컴퓨터에서도 실행
 - 다른 응용프로그램의 도움 필요 없이 단독으로 실행



자바로 만들 수 있는 것

서블릿(servlet)

- 웹 서버에서 실행되는 자바 프로그램
 - 서블릿은 웹브라우저에서 실행되는 자바스크립트 코드와 통신
- 웹서버에서 동작하는 서버 모듈로서 클라이언트의 요구를 받아서 그에 대한 처리를 한 후에, 실행 결과를 HTML 문서 형태로 클라이언트 컴퓨터로 전송
- 데이터베이스 서버 및 기타 서버와 연동하는 복잡한 기능 구현 시 사용
- 사용자 인터페이스가 필요 없는 응용
- 웹 서버에 의해 실행 통제 받음



자바로 만들 수 있는 것

JSP

- HTML안에 자바 코드를 넣으면 웹 페이지를 사용자와 상호작용하도록 만들 수 있다. JSP는 서버에서 실행되고 결과를 사용자에게 보여준다

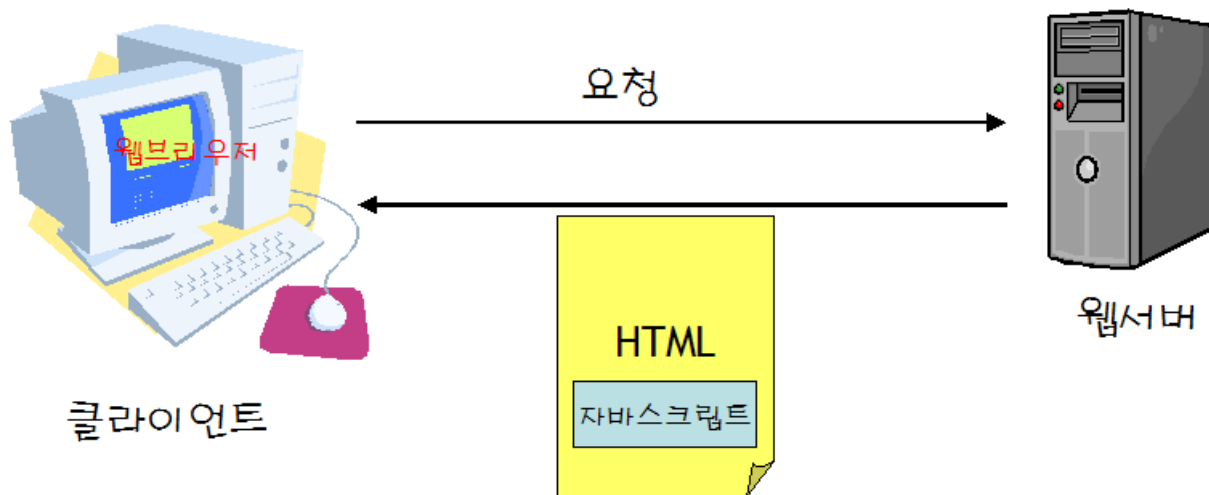
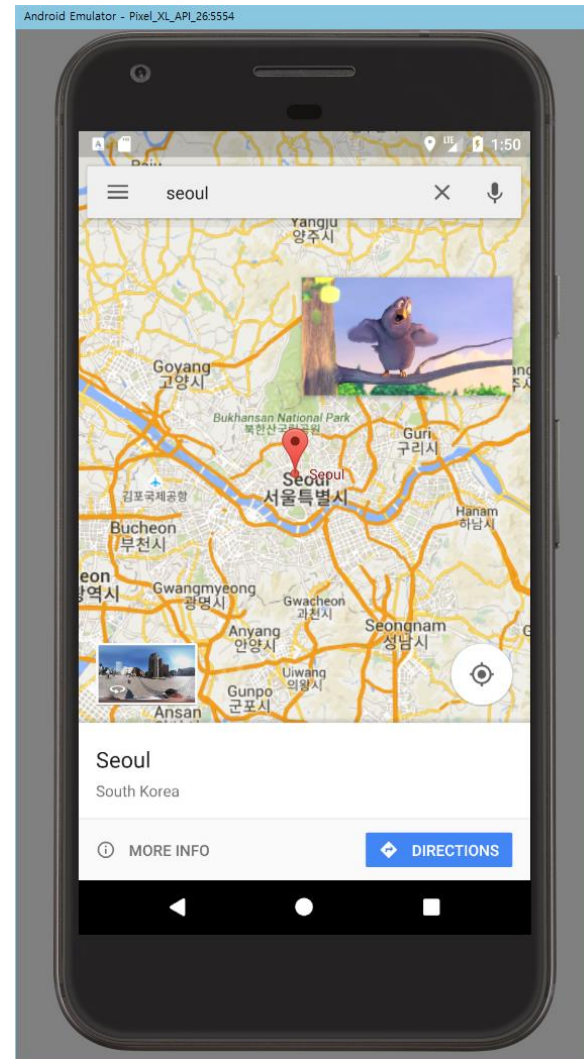


그림 1.14 JSP의 실행 과정

자바 모바일 응용 : 안드로이드 앱

안드로이드

- 구글의 주도로 여러 모바일 회사가 모여 구성한 OHA(Open Handset Alliance)에서 만든 무료 모바일 플랫폼
- 개발 언어는 자바를 사용하나 JVM에 해당하는 Dalvik은 기존 바이트 코드와 호환성이 없어 변환 필요





Thank You

