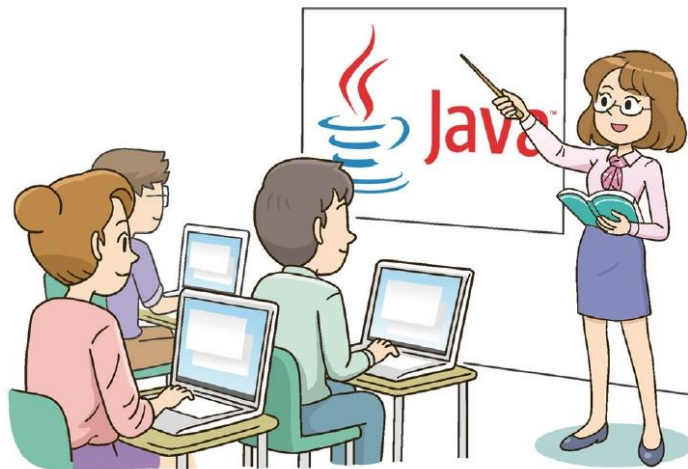


자바 프로그래밍



1주차

강의자 : 오 은 성

목차

- 수업목표
- 2023-1학기 자바 프로그래밍 소개
- 2023-1 학기 수업 소개
- 강의계획서
- 교재
- 평가계획서

2021학년도 1학기 자바 프로그래밍 소개

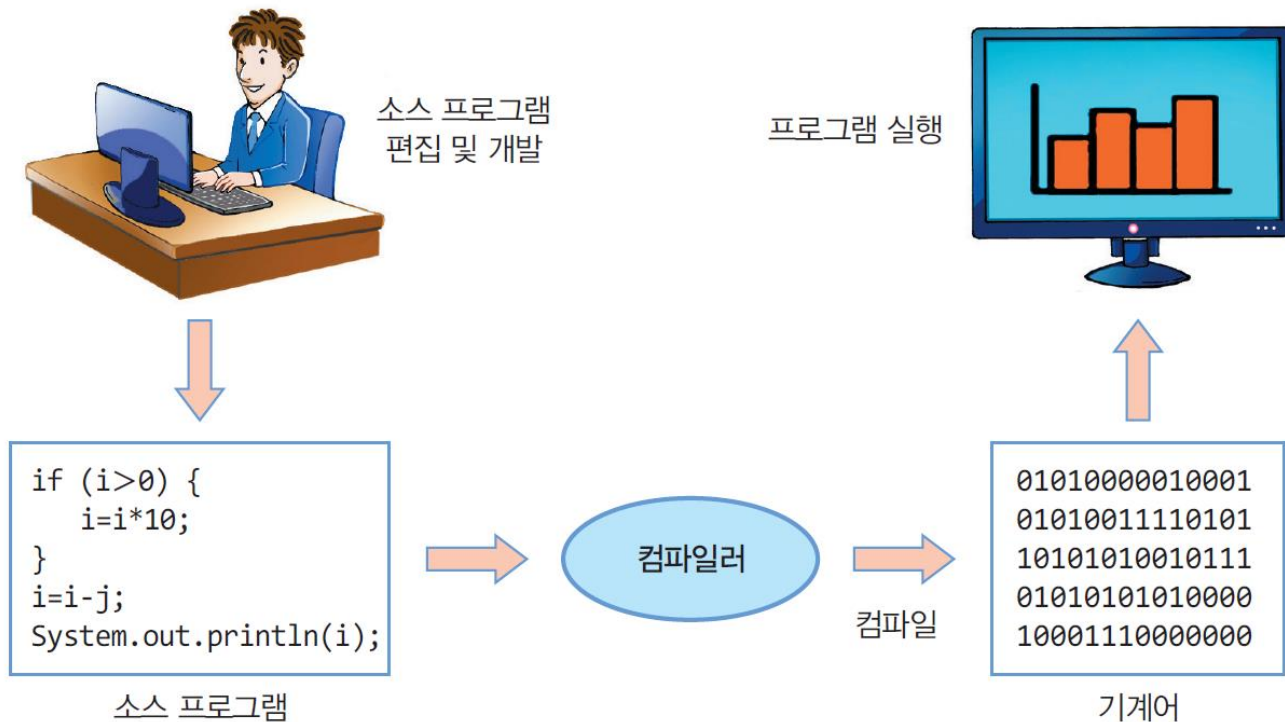
- 객체 지향의 근본적 이해 및 활용 방법, 객체 지향 언어인 자바 언어의 기본과 구조를 체계적으로 이해하고 실무에서 실제로 활용할 수 있도록 한다.
- 객체지향 프로그램의 상속에 대해 이해하고 자바를 사용하여 네트워크 프로그램이나 데이터베이스 프로그램 그래픽 프로그램을 손쉽게 작업할 수 있도록 한다.
- JAVA 의 패키지와 예외처리 인터페이스 , GUI 기초 그래픽 이벤트 처리 를 학습한다.

프로그래밍 언어

- ▶ 프로그래밍 언어
 - ▶ 프로그램 작성 언어
 - ▶ 기계어(machine language)
 - ▶ 0, 1의 이진수로 구성된 언어
 - ▶ 컴퓨터의 CPU는 기계어만 이해하고 처리가능
 - ▶ 어셈블리어
 - ▶ 기계어 명령을 ADD, SUB, MOVE 등과 같은 표현하기 쉬운 상징적인 단어인 니모닉 기호(mnemonic symbol)로 일대일 대응시킨 언어
 - ▶ 고급언어
 - ▶ 사람이 이해하기 쉽고, 복잡한 작업, 자료 구조, 알고리즘을 표현하기 위해 고안된 언어
 - ▶ Pascal, Basic, C/C++, Java, C#
 - ▶ 절차 지향 언어와 객체 지향 언어

컴파일

- 소스 : 프로그래밍 언어로 작성된 텍스트 파일
- 컴파일 : 소스 파일을 컴퓨터가 이해할 수 있는 기계어로 만드는 과정
 - ▣ 소스 파일 확장자와 컴파일 된 파일의 확장자
 - 자바 : **.java** -> **.class**
 - C : **.c** -> **.obj**-> **.exe**
 - C++ : **.cpp** -> **.obj** -> **.exe**



자바의 태동

6

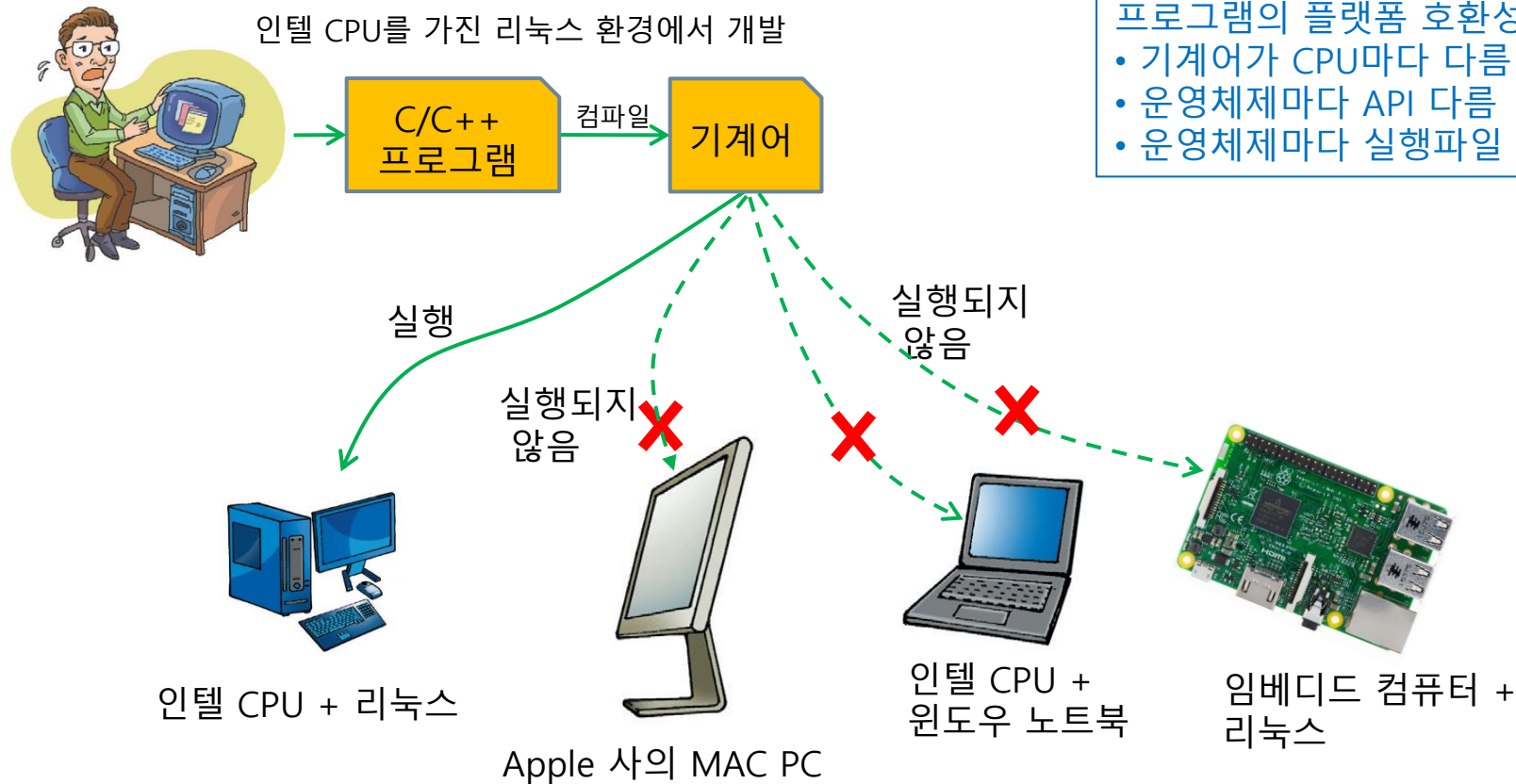
- 1991년 그린 프로젝트(Green Project)
 - ▣ 선마이크로시스템즈의 제임스 고슬링(James Gosling)에 의해 시작
 - 가전 제품에 들어갈 소프트웨어를 위해 개발
 - ▣ 1995년에 자바 발표
- 목적
 - ▣ 플랫폼 호환성 문제 해결
 - 기존 언어로 작성된 프로그램은 PC, 유닉스, 메인 프레임 등 플랫폼 간에 호환성 없음
 - 소스를 다시 컴파일하거나 프로그램을 재 작성해야 하는 단점
 - ▣ 플랫폼 독립적인 언어 개발
 - 모든 플랫폼에서 호환성을 갖는 프로그래밍 언어 필요
 - 네트워크, 특히 웹에 최적화된 프로그래밍 언어의 필요성 대두
 - ▣ 메모리 사용량이 적고 다양한 플랫폼을 가지는 가전 제품에 적용
 - 가전 제품 : 작은 량의 메모리를 가지는 제어 장치
 - 내장형 시스템 요구 충족
- 초기 이름 : 오크(OAK)
 - ▣ 인터넷과 웹의 엄청난 발전에 힘입어 퍼지게 됨
 - ▣ 웹 브라우저 Netscape에서 실행
- 2009년에 선마이크로시스템즈를 오라클에서 인수

- 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)

플랫폼 종속성(platform dependency)

8

플랫폼 = 하드웨어 플랫폼 + 운영체제 플랫폼

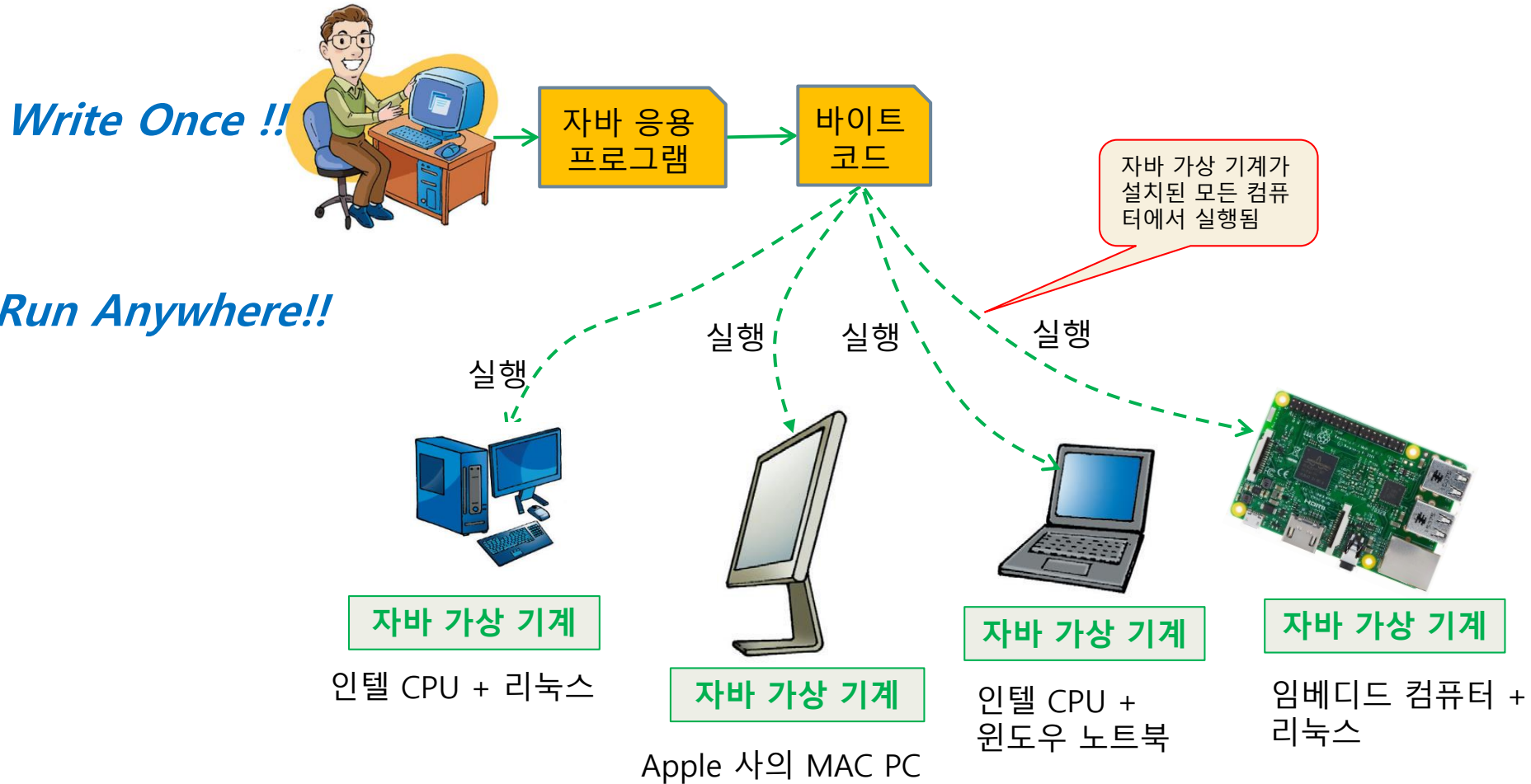


프로그램의 플랫폼 호환성 없는 이유

- 기계어가 CPU마다 다름
- 운영체제마다 API 다름
- 운영체제마다 실행파일 형식 다름

자바의 플랫폼 독립성, WORA

9



바이트 코드와 자바 가상 기계

10

- 바이트 코드
 - ▣ 자바 가상 기계에서 실행 가능한 바이너리 코드
 - 바이트 코드는 컴퓨터 CPU에 의해 직접 실행되지 않음
 - 자바 가상 기계가 작동 중인 플랫폼에서 실행
 - 자바 가상 기계가 인터프리터 방식으로 바이트 코드 해석
 - ▣ 클래스 파일(.class)에 저장
- 자바 가상 기계(JVM : Java Virtual Machine)
 - ▣ 동일한 자바 실행 환경 제공
 - 각기 다른 플랫폼에 설치
 - ▣ 자바 가상 기계 자체는 플랫폼에 종속적
 - 자바 가상 기계는 플랫폼마다 각각 작성됨
 - 예) 리눅스에서 작동하는 자바 가상 기계는 윈도우에서 작동하지 않음
 - ▣ 자바 가상 기계 개발 및 공급
 - 자바 개발사인 오라클, IBM 등
- 자바 응용프로그램 실행
 - ▣ 자바 가상 기계가 응용프로그램을 구성하는 클래스 파일(.class)의 바이트 코드 실행

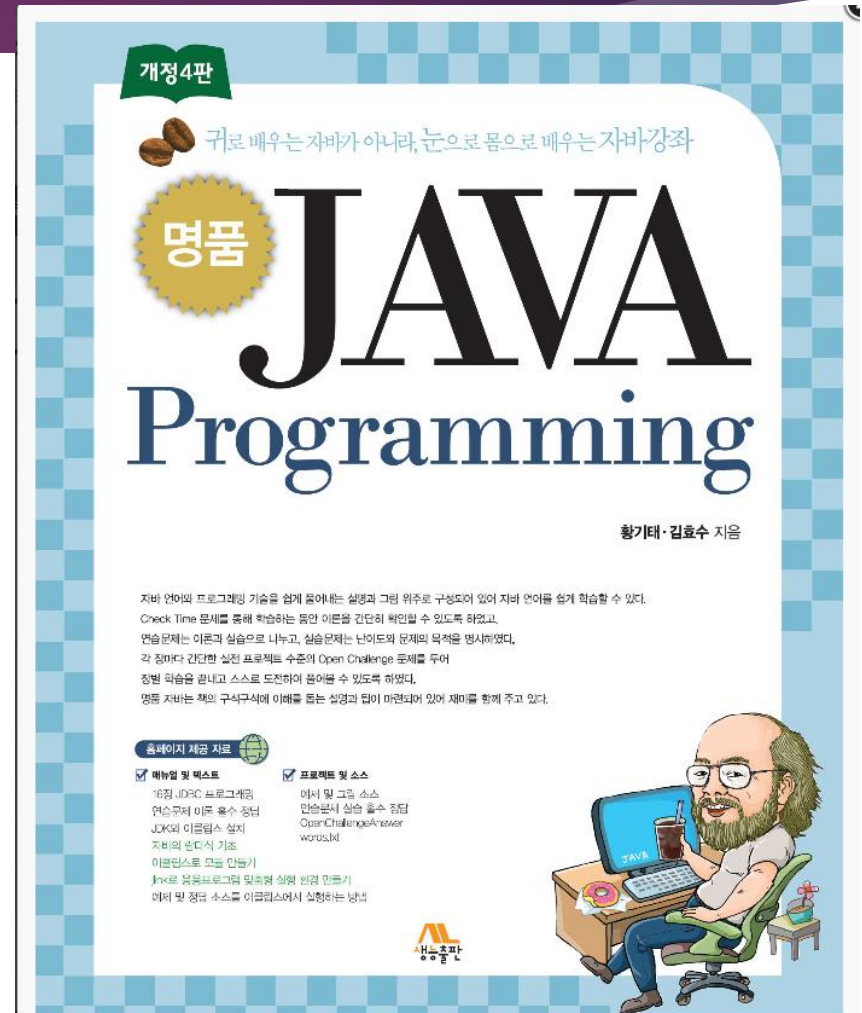
2023학년도 1학기 수업소개

학습목표

객체지향 개념을 이해하고 객체지향언어 JAVA 프로그래밍을 기반으로 객체지향 구조로 인해 소스 코드 관리가 쉽고 소프트웨어의 생산성이 높기 때문에 어떤 컴퓨터 운영체제에도 실행되는 JAVA 프로그램을 숙지하고 응용프로그램을 제작한다

교재

- 도서명: 명품 JAVA Programing
- 저자: 황기태, 김효수
- 출판사: 생능 출판사
- 교재에 없는 건 교안드림



2023학년도 1학기 강의 계획서

1주	객체지향 프로그래밍 소개/자바프로 그래밍 환경설정 과 실습	9주	메소드 오버라이딩 메서드 중복 정의
2주	객체지향 프로그래밍 기초와 기본 문법(변수 식별자 제어문)	10주	중간 개인 프로젝트
3주	자바 기본 프로그래밍(기본문법) 반복문과 배열 그리고 예외 처리	11주	인터페이스
4주	클래스와 객체 멤버변수 메서드	12주	모듈과 패키지 개념, 자바 기본 패키 지
5주	클래스 만들고 객체정의하기 객체 배 열	13주	제네릭과 컬렉션
6주	상속 / 상속과 생성자	14주	입출력 스트림과 파일 입출력
7주	추상 클래스	15주	기말고사
8주	중간고사	파란색 프로그래밍 기본	

대면 강의 수업 방법



이론 과 실습 수업을 병행



실무프로젝트 실제
예제를 통해 확인학습을
확인한다



출석은 반드시 스스로
경기대출석 스마트 앱을
이용하여 스스로 한다.



매 수업 시간마다 그날
실습 예제를 통해 그 주
수업을 확인한다.

오프라인 수업 방법

- ▶ 이론과 실습을 통해 예제를 이해하고 프로그램을 실행한다.
- ▶ 매 수업시간 확인 학습을 통해 수업의 이해도를 높인다.
- ▶ 중간 개인프로젝트를 통해 실무예제를 작성해 본다.
- ▶ 중간 평가를 통해 객체의 개념을 활용한 실습 예제를 작성해 본다.
- ▶ 모든 수업시간의 질문은 그 시간에 반드시 한다.(이해하지 못한 내용 그냥 넘어가지 않기)

평가 방법

평가

- ▶ 출결(20점)
- ▶ 중간고사 (30점)
- ▶ 기말고사(30점)
- ▶ 과제 및 실습 예제 확인학습 (10점)
- ▶ 중간 실무 예제 확인학습(10점)

공지사항

1. 질문은 메일로 언제든지
 - ▶ **ohes74@hanmail.net**
2. 문자나 카톡으로 문의
 - ▶ **010-8764-0939**
3. 메일을 보낼 때 문자 보낼 때 본인을 밝힐 것!!!
 - ▶ (메일제목) 경기대학교 2023학번 홍길동입니다.
 - ▶ 수업 혹은 과제 관련 질문 있습니다.

제 1 장 자바란?

▶ 개발자 : 제임스 고슬링(1995) 외

▶ 용도 :

웹 어플리케이션

시스템 통합 프로젝트(SI)

모바일 어플리케이션(안드로이드)

▶ 특징 :

1. C++ 기반의 객체지향 프로그래밍 언어

2. 컴파일과 인터프리팅 방식을 모두 사용하는 언어

3. 플랫폼에 독립적 (Write Once, Run Anywhere)

제 2 장 절차적언어 VS 객체지향언어

▶ 절차적 언어

: 프로그램의 흐름(절차)을 따라 코드를 작성하는 방식을 사용

C언어, COBOL, ALGOL 등 초기 세대의 프로그래밍 언어가 이에 속함

▶ 객체 지향 언어

: 기능 별로 객체화 하여, 객체마다 코드를 작성하는 방식을 사용

절차적 언어 이후에 나온 방식이며, 자바, C++, Python 등 현재 주로 사용되는 언어들은 객체 지향 언어에 속함

- (1) 캡슐화
- (2) 상속
- (3) 다형성

제 2 장 절차적언어 VS 객체지향언어

▶ 절차적 방식으로 자동차 만들기

1. 자동차의 실행 절차를 계획한다.
2. 처음 부분(엔트리 포인트)를 시작으로 시동이 멈추기까지의 흐름을 코딩한다.
e.g. 시동을 건다 → 안전벨트를 맨다 → 기어를 드라이브에 둔다
→ 브레이크를 뗀다 → 핸들을 움직인다. → 원하는 곳으로 이동한다. → 시동을 끈다.

장점 :

세심한 프로그래밍이 가능하다. (메모리와 하드웨어를 계획적으로 사용할 수 있다.)
따라서 작은 프로그램이나, 퍼포먼스가 중요한 프로그램(임베디드, 펌웨어 등)에서는
절차적 언어를 사용한다.

단점 :

유지 보수가 어렵고 (하나의 기능을 고치려면 프로그램을 전반적으로 수정해야 한다.)
개발 기간이 길다. (여러 명이 동시 작업하기가 까다롭다.)
유연한 실행이 불가능하다. (절차에 벗어날 수 없다.)

➔ 이러한 단점을 극복하기 위해 '객체지향 프로그래밍 패러다임' 이 등장

제 2 장 절차적언어 VS 객체지향언어

▶ 객체지향 방식으로 자동차 만들기

1. 자동차를 부품(객체) 별로 제작 (e.g. 엔진, 핸들, 브레이크 등)
2. 이 부품(객체)들을 하나의 자동차(프로그램)로 조립한다.

장점 :

1. 빠른 프로그램 제작이 가능하다. (동시 작업 가능, 부품 재사용 가능)
2. 유지보수가 편하다. (문제가 생긴 부품만 고치면 된다.)
3. 모든 기능들이 독립적인 부품으로 존재하기 때문에 유연한 실행이 가능하다.
(핸들이 고장나도 브레이크는 사용할 수 있다.)

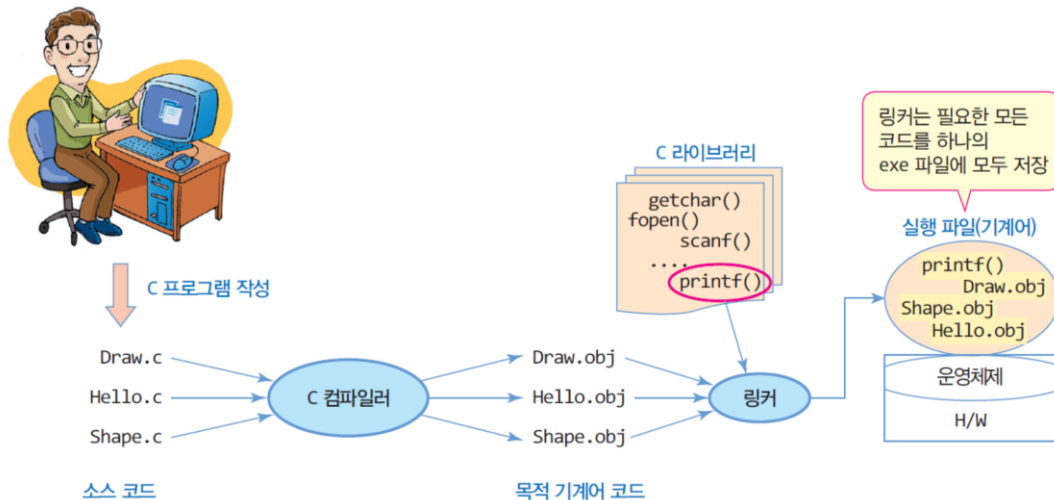
단점 :

무겁다. (불필요한 기능이 있거나, 필요 이상으로 자원을 소비할 수 있다.)

➔ 현대 하드웨어는 성능이 좋아 이를 신경 쓰지 않는 것이다.

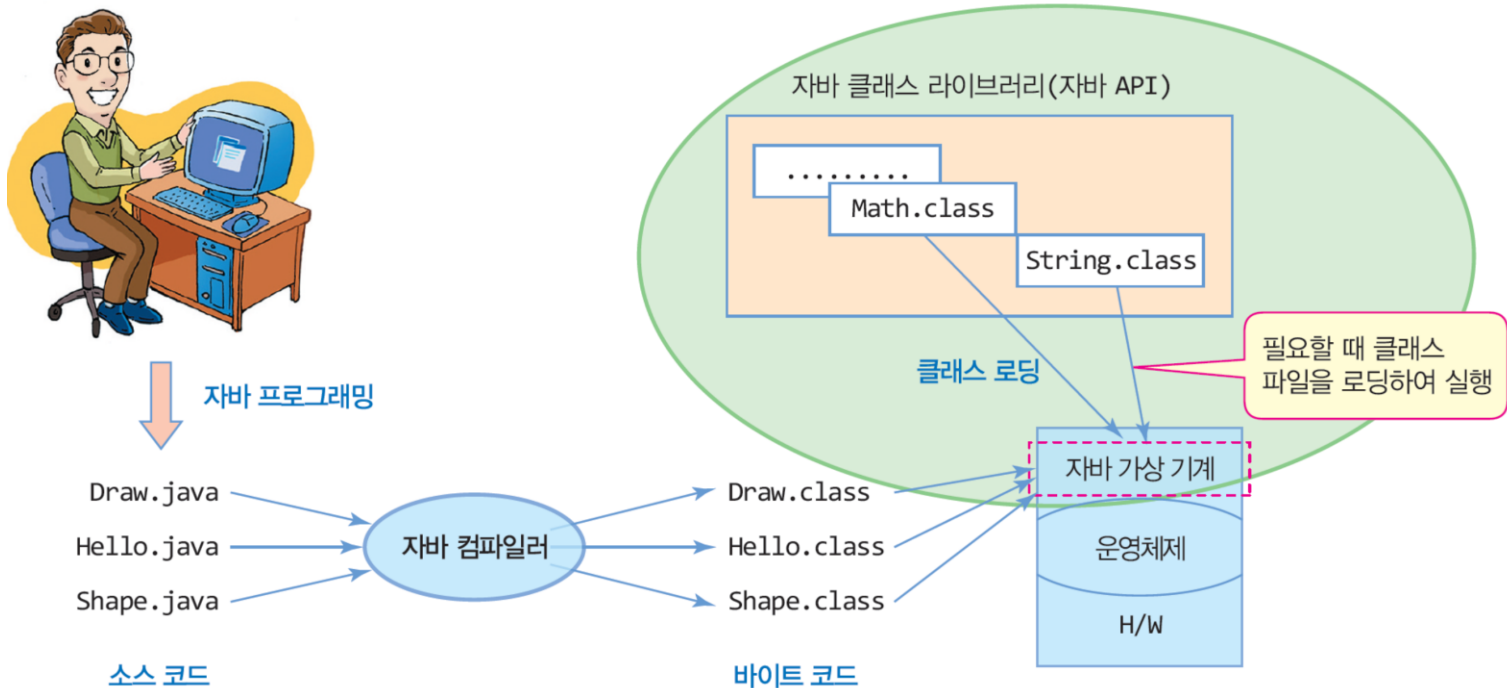
C/C++ 프로그램의 개발 및 실행 환경

- ▶ C/C++ 프로그램의 개발
 - ▶ 여러 소스(.c) 파일로 나누어 개발
 - ▶ 링크를 통해 실행에 필요한 모든 코드를 하나의 실행 파일(.exe)에 저장
- ▶ 실행
 - ▶ 실행 파일(exe)은 모두 메모리에 올려져야 실행, 메모리가 적은 경우 낭패



자바의 개발 및 실행 환경

- ▶ 자바 프로그램의 개발
 - ▶ 여러 소스(.java)로 나누어 개발
 - ▶ 바이트 코드(.class)를 하나의 실행 파일(exe)로 만드는 링크 과정 없음
- ▶ 실행
 - ▶ main() 메소드를 가진 클래스에서 부터 실행 시작
 - ▶ 자바 가상 기계는 필요할 때, 클래스 파일 로딩, 적은 메모리로 실행 가능
자바 실행 환경(JRE)



* 자바는 하나의 실행 파일(exe)로 만드는 링크 과정 없음

자바와 C/C++의 실행 환경 차이

```
if (i>0) {  
    i = i*10;  
}  
i = i - j;  
System.out.println(i);
```

자바 소스 파일(Test.java)

컴파일러

```
01010000010001  
010100111110101  
10101010010111  
01010101010000  
10001110000000
```

바이트 코드(Test.class)

자바 프로그램
(Test.class)

자바 가상 기계

운영체제

하드웨어

```
if (i>0) {  
    i = i*10;  
}  
i = i - j;  
cout << i;
```

소스 파일(Test.cpp)

컴파일러
/링커

```
01010000010001  
010110111110101  
10101010010111  
11010101010010  
10101110001100
```

바이너리 실행 파일(Test.exe)

C++ 프로그램
(Test.exe)

운영체제

하드웨어

Tip: 자바와 C/C++ 실행 환경 및 과정

▶ 자바

- ▶ 컴파일러가 바로 바이트 코드한 후 링크 과정 없음
- ▶ 바이트 코드는 JVM에서만 실행 가능
- ▶ 자바는 필요한 클래스들을 프로그램 실행 중에 동적으로 로딩
 - ▶ 동적 로딩은 JVM에 포함된 클래스 로더에 의해 이루어짐
 - ▶ ClassLoader 클래스를 이용하여 개발자가 직접 클래스 로딩가능

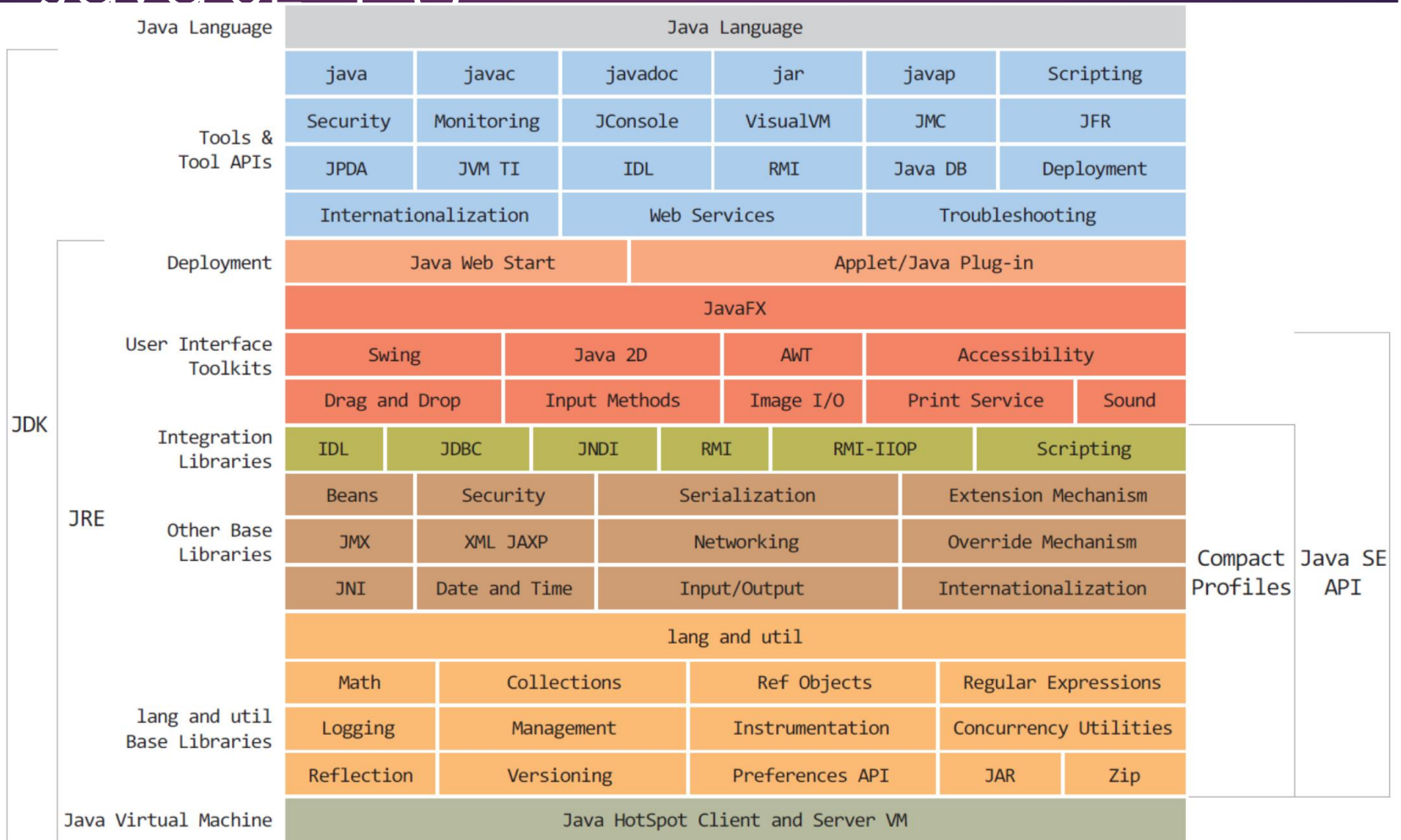
▶ C/C++

- ▶ 컴파일
 - ▶ C/C++에서는 컴파일러가 중간 단계인 목적 코드를 생성
- ▶ 링크
 - ▶ 링커가 목적 코드와 라이브러리 연결, 실행 가능한 최종 실행 파일 생성
 - ▶ 정적 라이브러리는 실행 파일에 포함
 - ▶ 실행 파일 크기가 커짐
 - ▶ 동적 라이브러리의 경우는 실행 중에 동적 링크
- ▶ 목적 코드 및 실행 파일은 플랫폼에 따라 다름
 - ▶ 플랫폼이 바뀌거나 다른 플랫폼에서 실행시키려면 다시 컴파일 및 링크

JDK와 JRE

- ▶ JDK(Java Development Kit)
 - ▶ 자바 응용 개발 환경. 개발에 필요한 도구 포함
 - ▶ 컴파일러, 컴파일된 자바 API 클래스들이 들어 있는 모듈 파일들, 샘플 등 포함
- ▶ JRE(Java Runtime Environment)
 - ▶ 자바 실행 환경. JVM 포함
 - ▶ 컴파일된 자바 API 등이 들어 있는 모듈 파일
 - ▶ 개발자가 아닌 경우 JRE만 따로 다운 가능
- ▶ JDK와 JRE의 개발 및 배포
 - ▶ 오라클의 Technology Network의 자바 사이트에서 다운로드
 - ▶ <http://www.oracle.com/technetwork/java/index.html>
- ▶ JDK의 bin 디렉터리에 포함된 주요 개발 도구
 - ▶ javac - 자바 소스를 바이트 코드로 변환하는 컴파일러
 - ▶ java - 자바 응용프로그램 실행기. 자바 가상 기계를 작동시켜 자바프로그램 실행
 - ▶ javadoc - 자바 소스로부터 HTML 형식의 API 도큐먼트 생성
 - ▶ jar - 자바 클래스들(패키지포함)을 압축한 자바 아카이브 파일(.jar) 생성 관리
 - ▶ jmod: 자바의 모듈 파일(.jmod)을 만들거나 모듈 파일의 내용 출력
 - ▶ jlink: 응용프로그램에 맞춘 맞춤형(custom) JRE 제공
 - ▶ jdb - 자바 응용프로그램의 실행 중 오류를 찾는 데 사용하는 디버거
 - ▶ javap - 클래스 파일의 바이트 코드를 소스와 함께 보여주는 디어셈블러

Java SE 구성



Java SE의 구성(출처: <http://www.oracle.com/technetwork/java/javase/tech/index.html>)

JDK 설치 후 디렉터리 구조

Java

jdk-11.0.2

> bin

자바 개발, 실행에 필요한 도구와 유틸리티 명령

> conf

여러 종류의 배치 파일

> include

네이티브 코드 프로그래밍에 필요하는 C 언어 헤더 파일

jmods

컴파일된 모듈 파일들

> legal

각 모듈에 대한 저작권과 라이선스 파일

> lib

실행 시간에 필요한 라이브러리 클래스들

자바의 배포판 종류

- ▶ 오라클은 개발 환경에 따라 다양한 자바 배포판 제공
- ▶ Java SE
 - ▶ 자바 표준 배포판(Standard Edition)
 - ▶ 데스크탑과 서버 응용 개발 플랫폼
- ▶ Java ME
 - ▶ 자바 마이크로 배포판
 - ▶ 휴대 전화나 PDA, 셋톱박스 등 제한된 리소스를 갖는 하드웨어에서 응용 개발을 위한 플랫폼
 - ▶ 가장 작은 메모리 풋프린트
 - ▶ Java SE의 서브셋 + 임베디드 및 가전 제품을 위한 API 정의
- ▶ Java EE
 - ▶ 자바 기업용 배포판
 - ▶ 자바를 이용한 다중 사용자, 기업용 응용 개발을 위한 플랫폼
 - ▶ Java SE + 인터넷 기반의 서버사이드 컴퓨팅 관련 API 추가

제 2 장 절차적언어 VS 객체지향언어

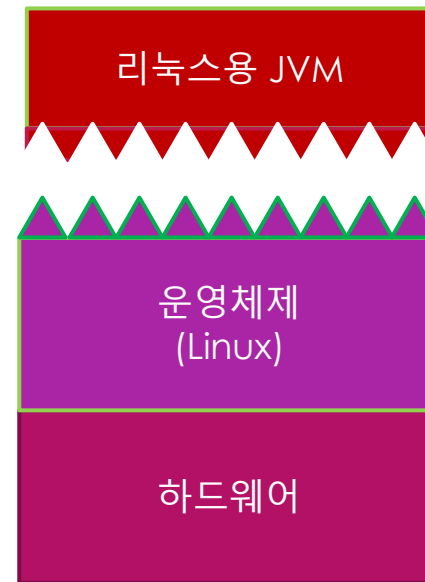
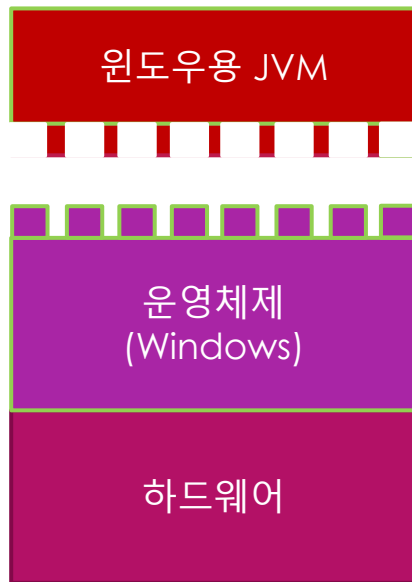
▶ 컴파일 방식

- ▶ 소스코드 -> 번역(compile) -> 실행 프로그램 -> 실행
- ▶ 장점 : 실행 속도가 빠르다.
- ▶ 단점 : 운영체제(플랫폼)에 종속적이다.

▶ 인터프리터 방식

- ▶ 소스코드 -> 번역(compile) & 실행
- ▶ 장점 : 운영체제에 자유롭다.
- ▶ 단점 : 실행 속도가 느리다.

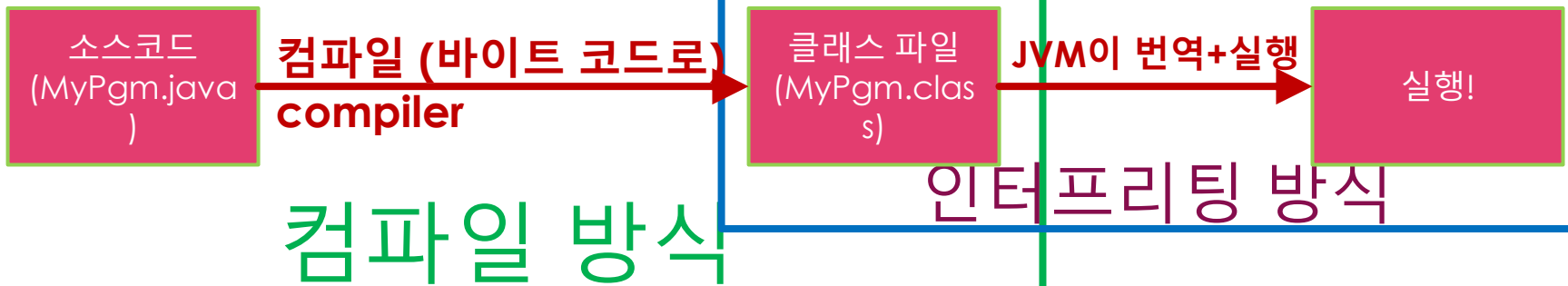
제 4 장 자바의 JVM (Java Virtual Machine)



제 5 장 자바의 컴파일과 실행 과정



제 5 장 자바의 컴파일과 실행 과정



제 6 장 JRE와 JDK

1. JRE (Java Runtime Environment; 자바 실행 환경)

- ▶ JVM이 자바 프로그램을 동작 시킬 때 필요한 라이브러리 파일들과 기타 파일들을 가지고 있다. (창고의 역할)
- ▶ **사용자의 컴퓨터에 설치**해야 함.

2. JDK (Java Development Kit; 자바 개발 도구)

- ▶ 자바 개발에 사용되는 컴파일러, 개발 환경을 지원하는 프로그램
- ▶ Javac 등의 컴파일 명령어들이 소속되어 있다.
- ▶ 개발자가 작성한 소스 코드를 JVM에게 전달할 바이트 코드로 변환하여 .class 파일을 생성한다.
- ▶ **개발자의 컴퓨터에 설치**해야 하며, JDK가 설치되면 JRE도 함께 설치 됨.

자바 API

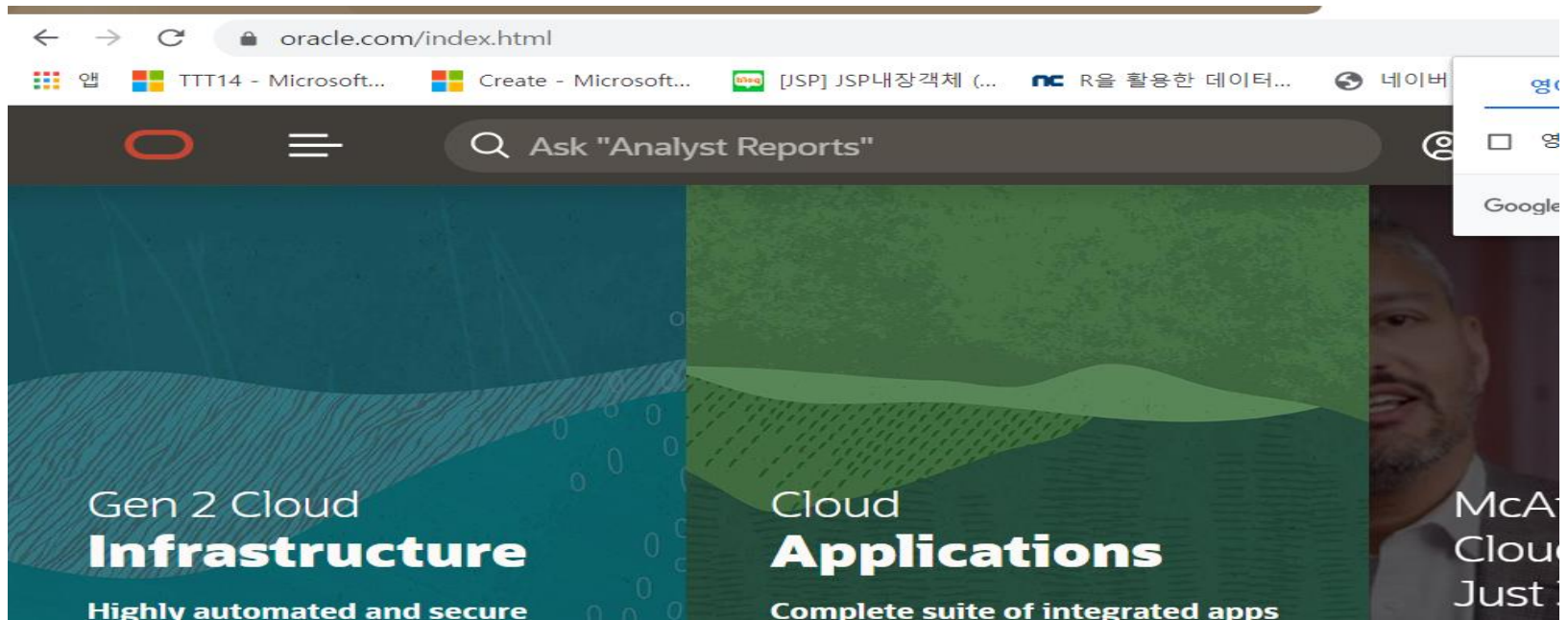
- ▶ 자바 API(Application Programming Interface)란?
 - ▶ JDK에 포함된 클래스 라이브러리
 - ▶ 주요한 기능들을 미리 구현한 클래스 라이브러리의 집합
 - ▶ 개발자는 API를 이용하여 쉽고 빠르게 자바 프로그램 개발
 - ▶ API에서 정의한 규격에 따라 클래스 사용
- ▶ 자바 패키지(package)
 - ▶ 서로 관련된 클래스들을 분류하여 묶어 놓은 것
 - ▶ 계층구조로 되어 있음
 - ▶ 클래스의 이름에 패키지 이름도 포함
 - ▶ 다른 패키지에 동일한 이름의 클래스 존재 가능
 - ▶ 자바 API(클래스 라이브러리)는 JDK에 패키지 형태로 제공됨
 - ▶ 필요한 클래스가 속한 패키지만 import하여 사용
 - ▶ 개발자 자신의 패키지 생성 가능








자바를 사용하려면 무엇이 필요한가?

- ▶ 명칭: JDK (Java Development Kit)
- ▶ 설명: 자바 개발 도구
- ▶ 다운로드 위치: www.oracle.com

1. JDK(Java Development Kit) 다운로드 및 설치

- ▶ JDK다운로드
- ▶ <https://www.oracle.com/java/technologies/downloads>

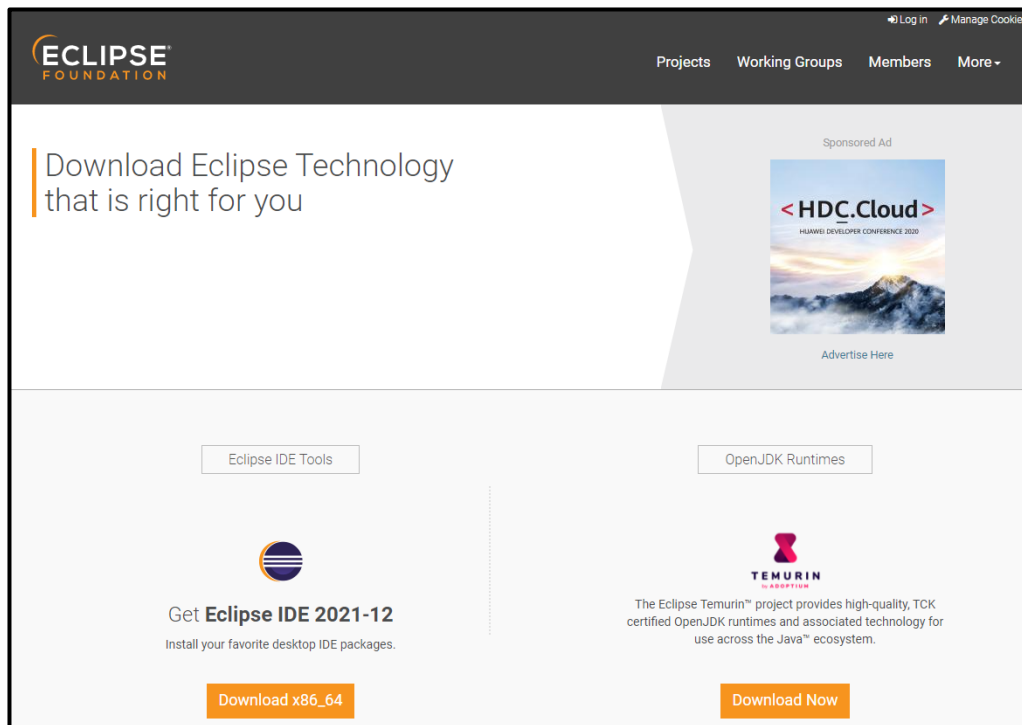


Linux x64 Debian Package	146.17 MB	 jdk-16.0.2_linux-x64_bin.deb
Linux x64 RPM Package	153.01 MB	 jdk-16.0.2_linux-x64_bin.rpm
Linux x64 Compressed Archive	170.04 MB	 jdk-16.0.2_linux-x64_bin.tar.gz
macOS Installer	166.6 MB	 jdk-16.0.2_osx-x64_bin.dmg
macOS Compressed Archive	167.21 MB	 jdk-16.0.2_osx-x64_bin.tar.gz
Windows x64 Installer	150.58 MB	 jdk-16.0.2_windows-x64_bin.exe
Windows x64 Compressed Archive	168.8 MB	 jdk-16.0.2_windows-x64_bin.zip

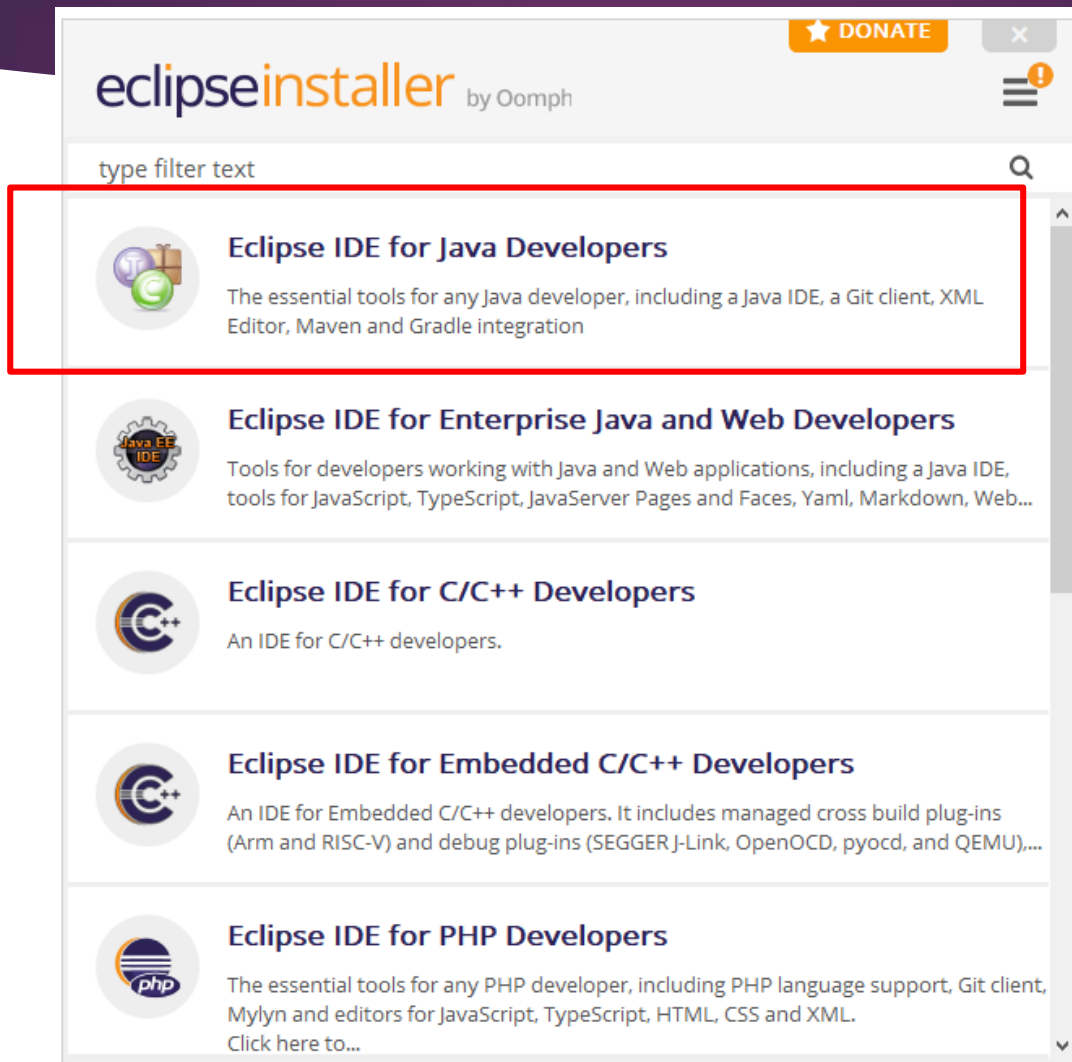
본인 컴퓨터 사양에 맞는 부분을 다운로드 합니다.

이클립스 다운로드

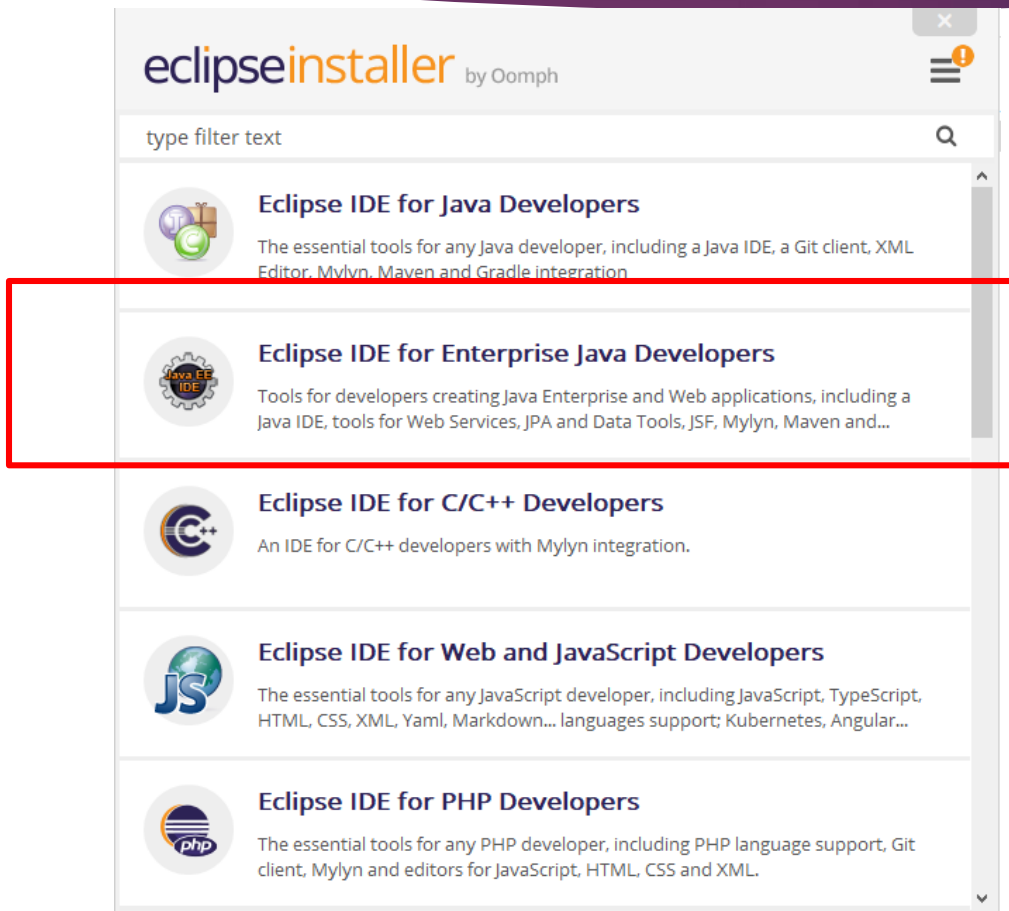
▶ <https://www.eclipse.org/downloads/>



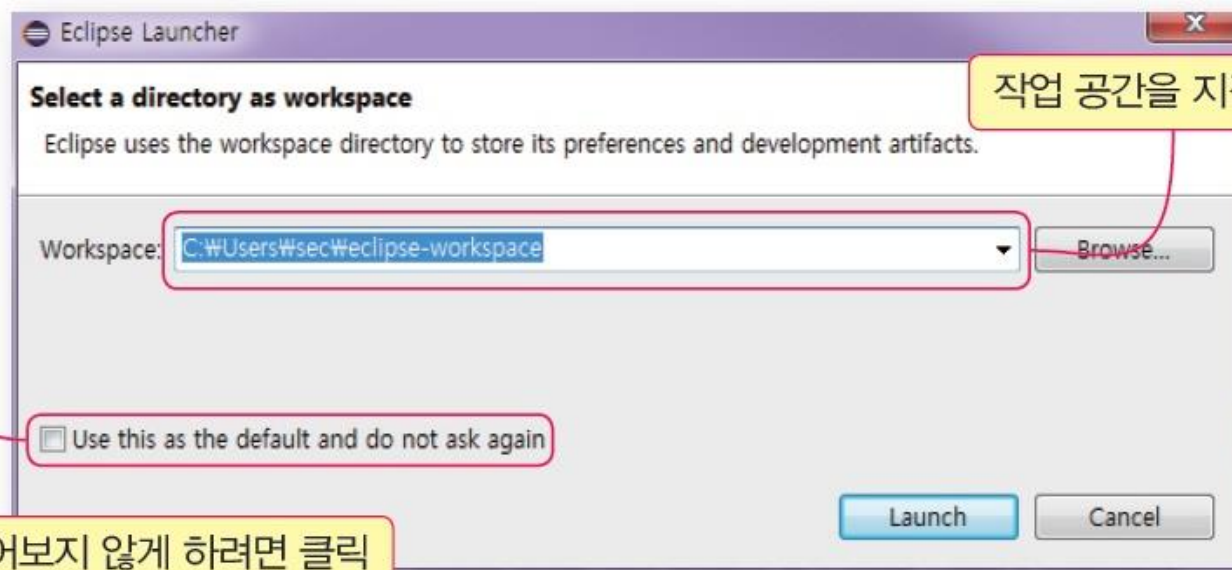
이클립스 설치 (java만 사용)



이클립스 설치(웹프로그래밍 모바일 프로그래밍, 스프링)



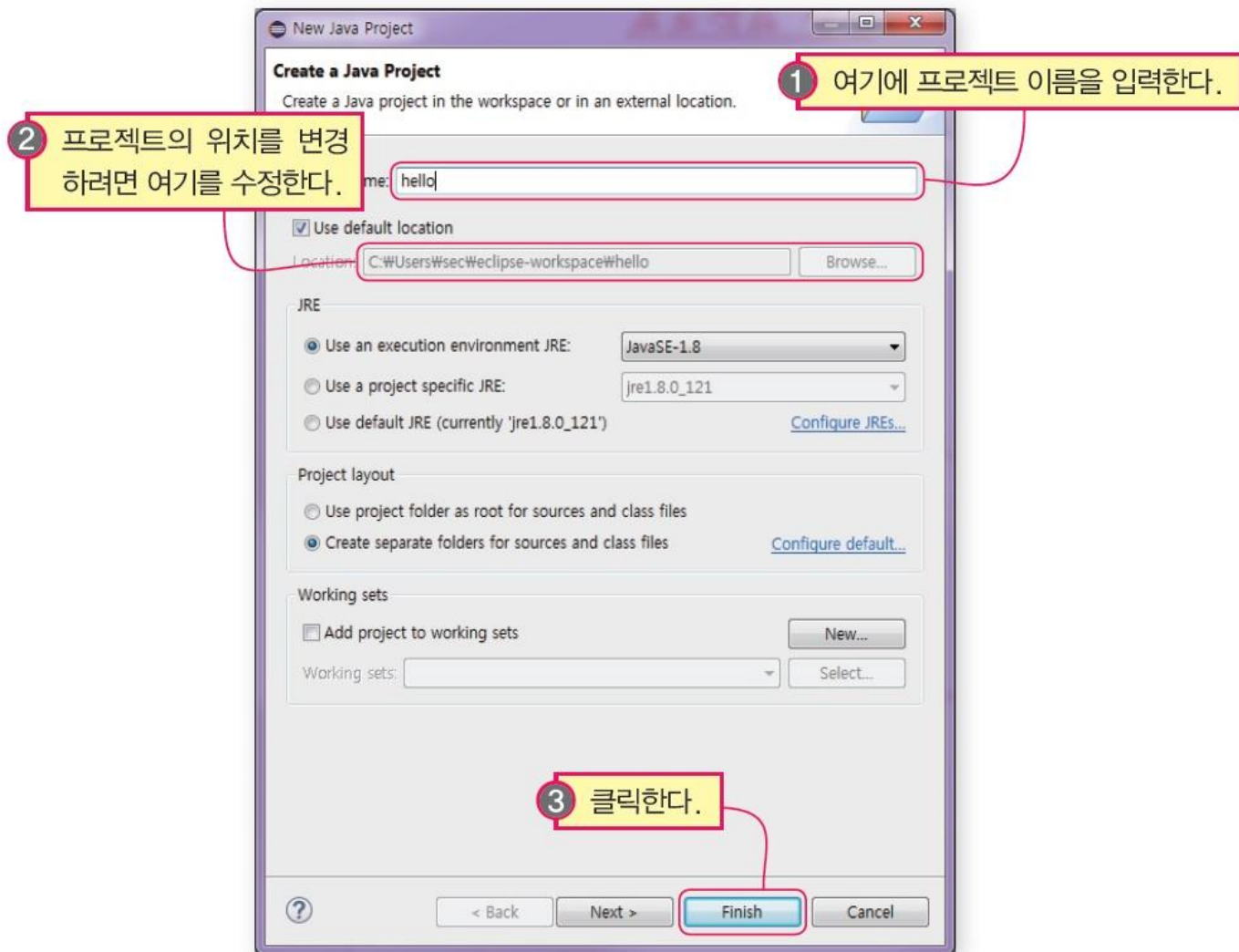
이클립스 실행



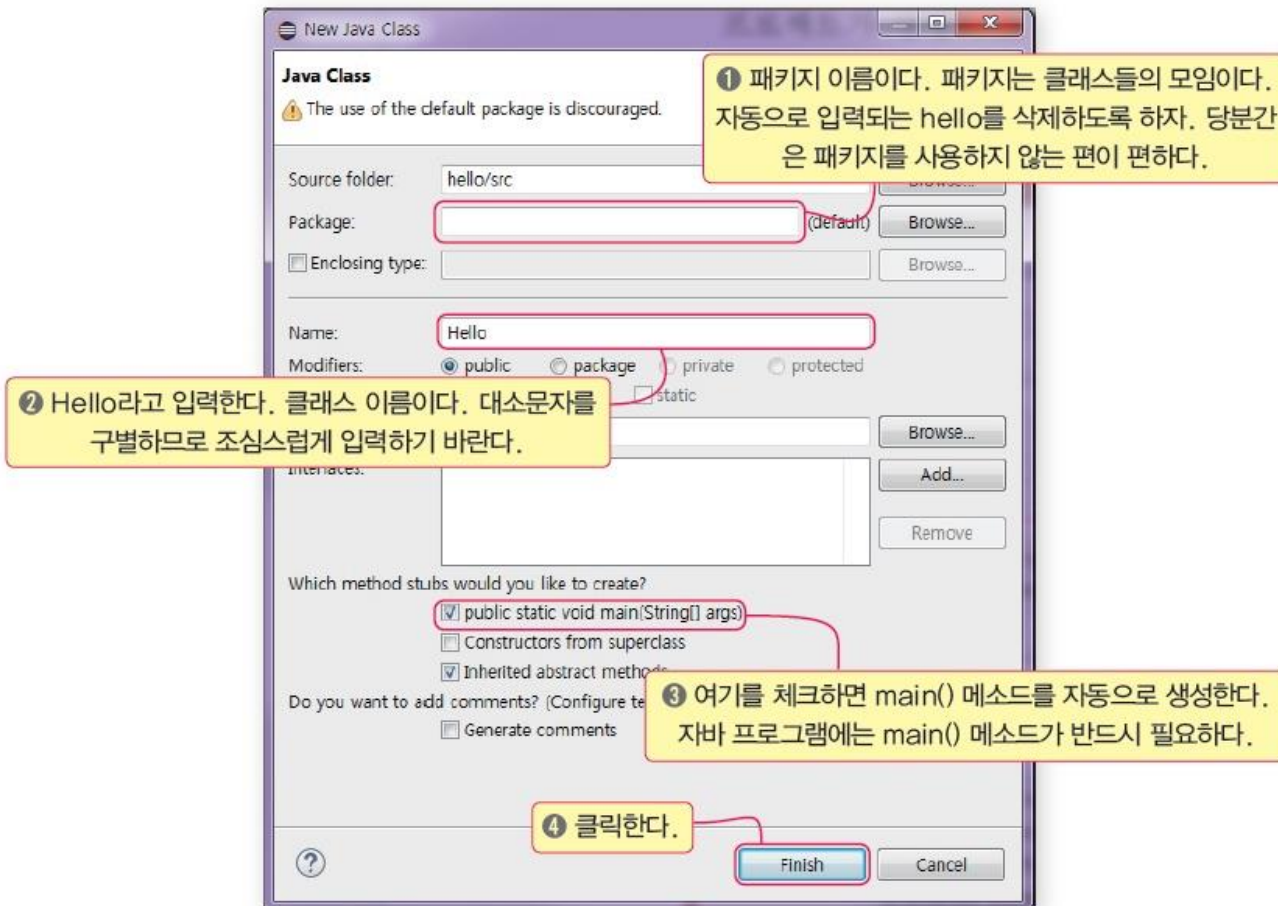
이클립스 첫화면



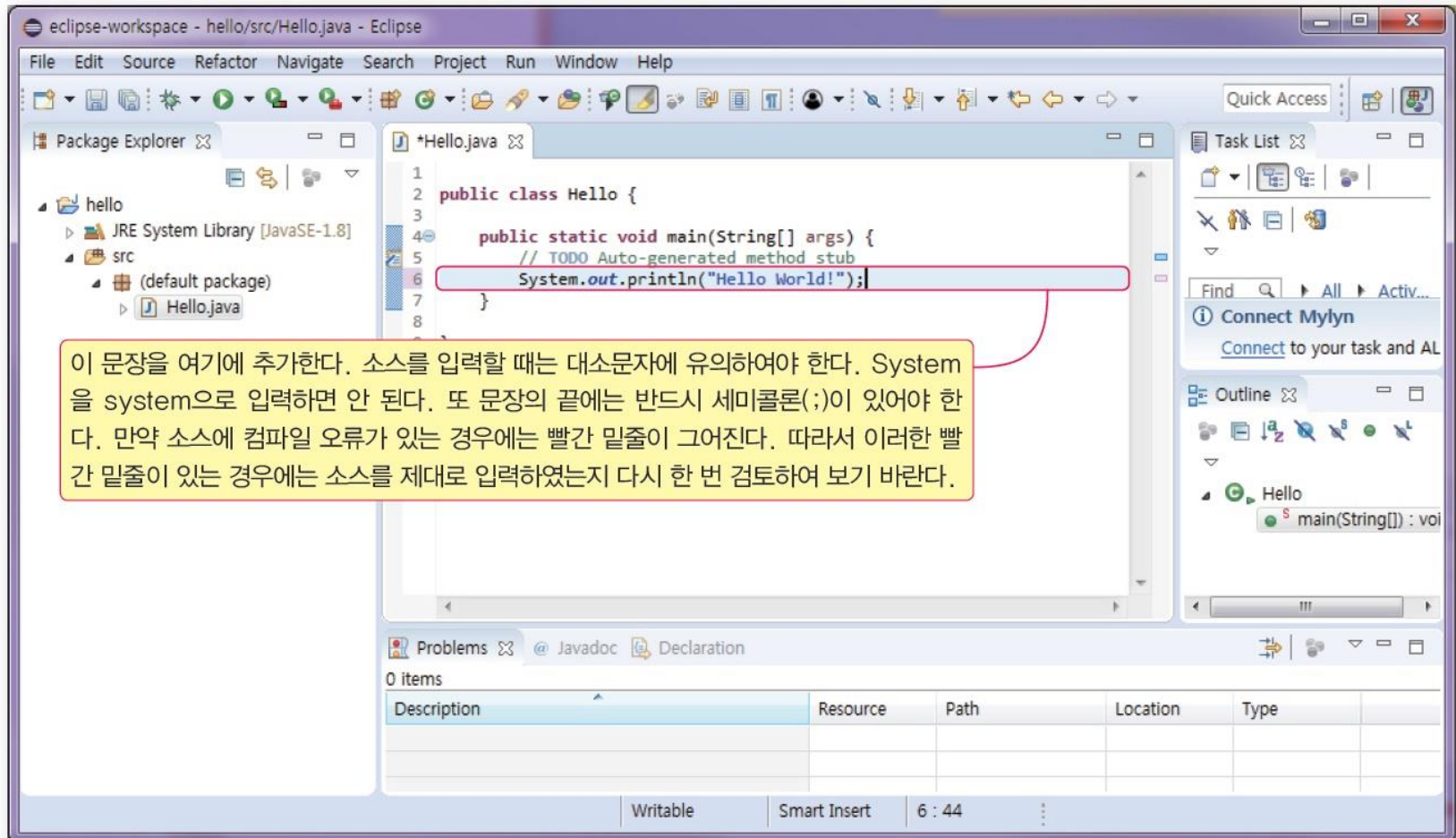
프로젝트 생성



클래스 생성



소스 코드 입력



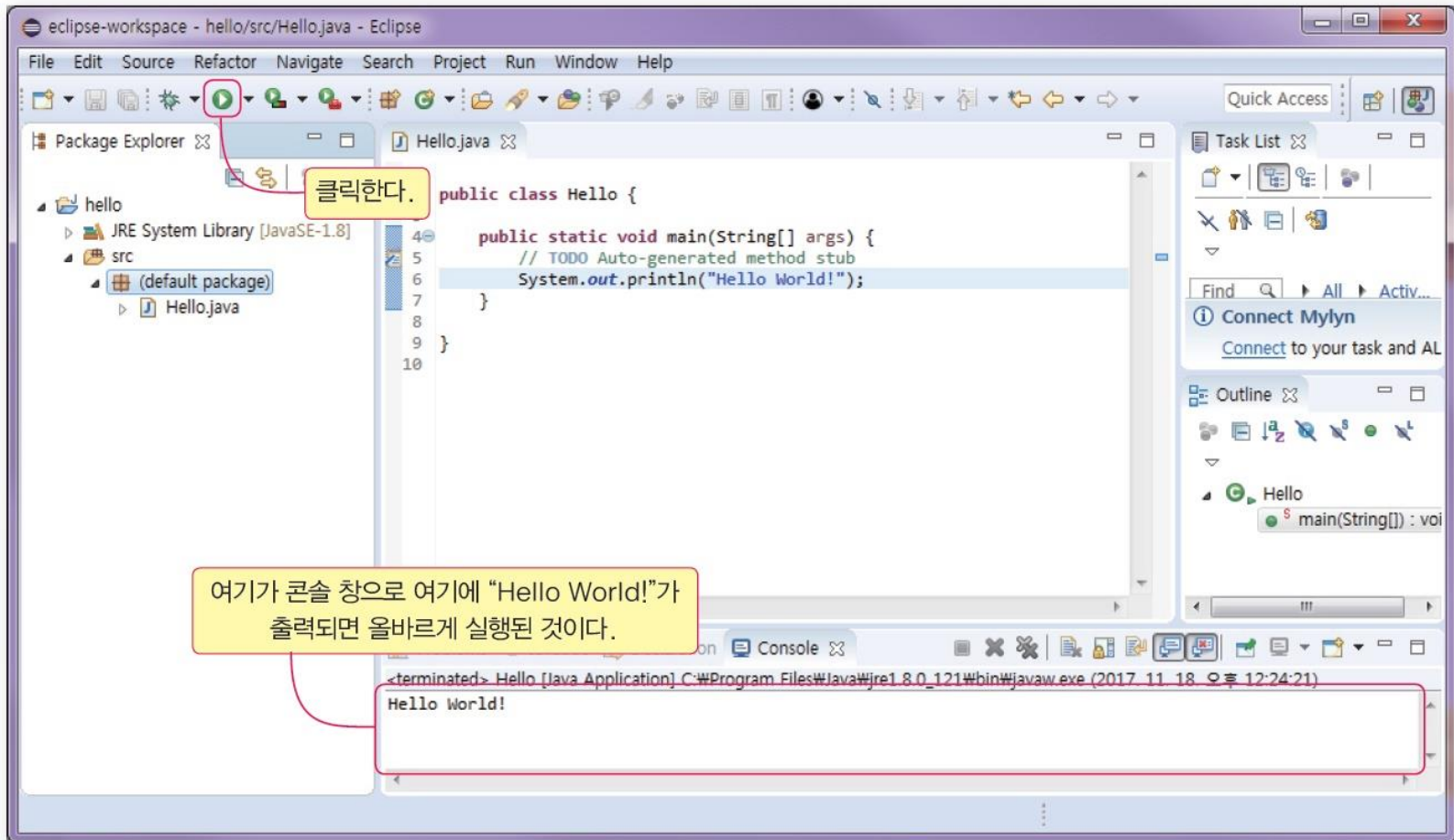
이 문장을 여기에 추가한다. 소스를 입력할 때는 대소문자에 유의하여야 한다. System을 system으로 입력하면 안 된다. 또 문장의 끝에는 반드시 세미콜론(;)이 있어야 한다. 만약 소스에 컴파일 오류가 있는 경우에는 빨간 밑줄이 그어진다. 따라서 이러한 빨간 밑줄이 있는 경우에는 소스를 제대로 입력하였는지 다시 한 번 검토하여 보기 바란다.

```
1 public class Hello {
2
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         System.out.println("Hello World!");
7     }
8 }
```

Description	Resource	Path	Location	Type

Writable Smart Insert 6 : 44

프로그램 실행



클래스의 정의

이 클래스는 누구든지
사용 가능

클래스를 선언하는 키워드

클래스 이름

클래스 시작

전체적인 구조



형식

```
public class Hello {
```

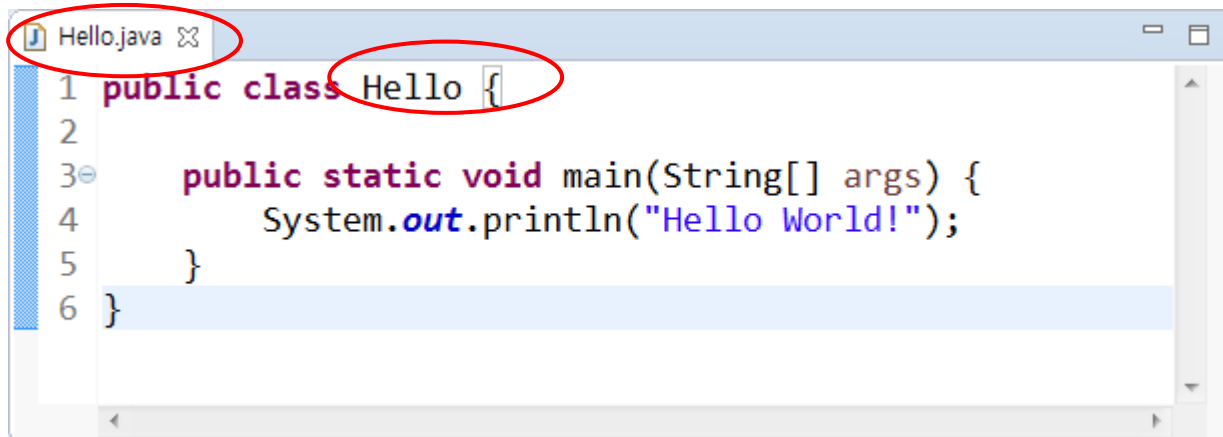
여기에 우리가 원하는
문장을 추가한다.

클래스 종료

```
}
```


소스 파일과 클래스 이름

- ▶ 소스 안에 public 클래스가 있다면 반드시 소스 파일의 이름은 public 클래스의 이름과 일치하여야 한다.
- ▶ 하나의 소스 파일 안에 public 클래스가 2개 이상 있으면 컴파일 오류가 발생한다.



```
1 public class Hello {  
2  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!");  
5     }  
6 }
```

자바 응용의 종류 : 데스크톱 응용 프로그램

▶ 가장 전형적인 자바 응용프로그램

▶ PC 또는 데스크톱 컴퓨터에 설치되어 실행



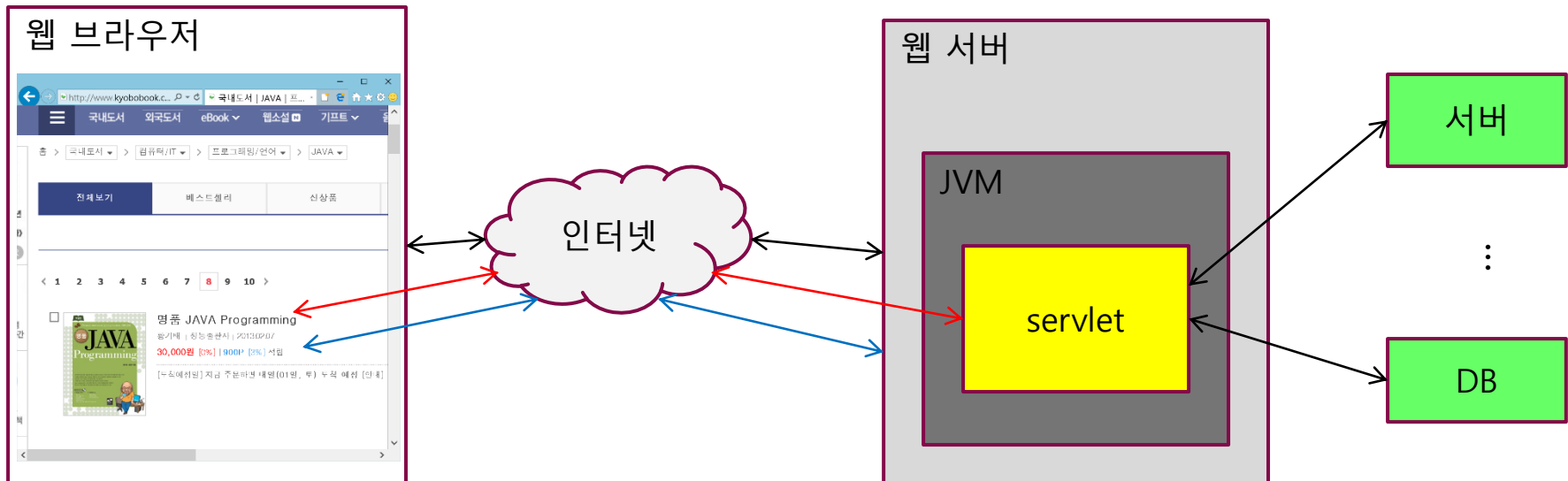
어떤 컴퓨터에서도 실행

없이 단독으로 실행

자바 응용의 종류 : 서블릿 응용프

▶ 서블릿 (Servlet)

- ▶ 웹 서버에서 실행되는 자바 프로그램
- ▶ 서블릿은 웹브라우저에서 실행되는 자바스크립트 코드와 통신
- ▶ 데이터베이스 서버 및 기타 서버와 연동하는 복잡한 기능 구현 시 사용
- ▶ 사용자 인터페이스가 필요 없는 응용
- ▶ 웹 서버에 의해 실행 통제 받음



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

