

기초사항

514760-1
2017년 가을학기
8/28/2017
박경신

자바란 무엇인가?

- 1991년 그린 프로젝트(Green Project)
 - 선마이크로시스템즈의 제임스 고슬링(James Gosling)에 의해 시작
- 가전 제품에 들어갈 소프트웨어를 위해 개발
- 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
- 2009년에 선마이크로시스템즈를 오라클에서 인수

Java Version

- | | |
|--|--|
| <ul style="list-style-type: none">□ Java 1.0<ul style="list-style-type: none">■ 1996■ 211개 클래스■ applet□ Java 1.1-1.4<ul style="list-style-type: none">■ 1997-2004■ 2700개 클래스■ ME, SE, EE version■ 내부클래스, AWT, 자바빈즈, Reflection, Collection, Swing, 핫자바 JVM, JavaSound, JNDL, assertion, regular expression, XML parser | <ul style="list-style-type: none">□ Java 1.5-1.6<ul style="list-style-type: none">■ 2004-2006■ 3700개 클래스■ Generic class, foreach, 가변 인수, 오토 박싱, 메타 데이터, 열거형, 정적 import, interface□ Java 1.7-1.8<ul style="list-style-type: none">■ 2011-2017■ 4200개 클래스■ Lambda expression, Parallel Array Sorting, Base64 Encoding & Decoding API, Date and Time API, Password-Based-Encryption(PBE) |
|--|--|

자바의 특징



자바의 특징

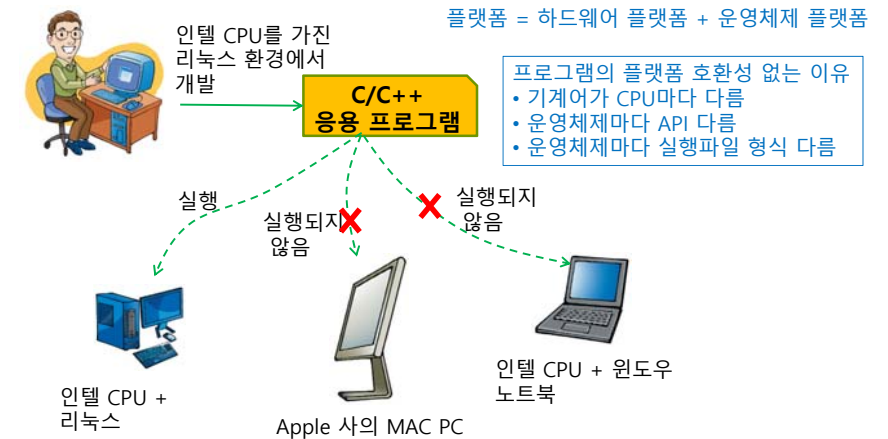
□ WORA(Write Once Run Anywhere)

- 한번 작성된 코드는 모든 플랫폼에서 바로 실행되는 자바의 특징
- 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 중립적인 코드
 - JVM에 의해 해석되고 실행됨
- **JVM(Java Virtual Machine)**
 - 자바 바이트 코드를 실행하는 자바 가상 기계(소프트웨어)

플랫폼 종속성 (Platform Dependency)

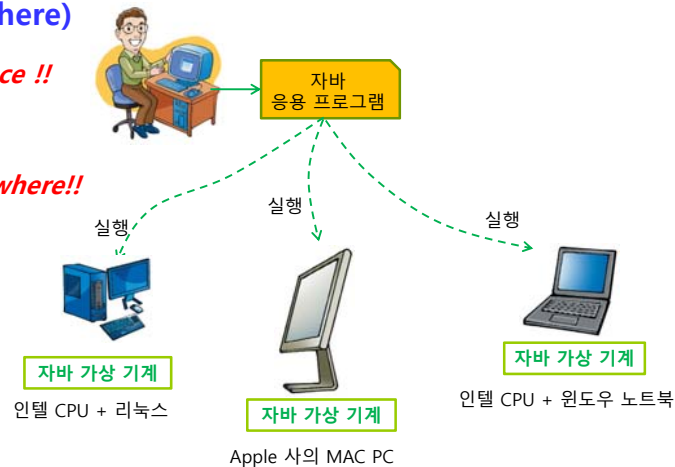


플랫폼 독립성

□ 자바는 플랫폼 독립성 - WORA(Write Once Run Anywhere)

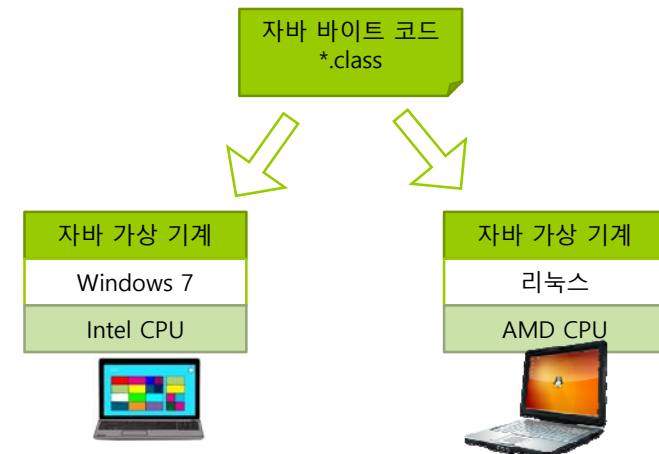
Write Once !!

Run Anywhere!!



자바 가상 기계

- 자바 컴파일러는 특정한 컴퓨터가 아닌 가상적인 기계(virtual machine)를 위한 바이트 코드를 생성함



자바 가상 기계

자바 가상 기계 (JVM : Java Virtual Machine)

- 각기 다른 플랫폼에 설치
- 동일한 자바 실행 환경 제공
- 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 외 IBM, MS 등 다양한 회사에서 제작 공급

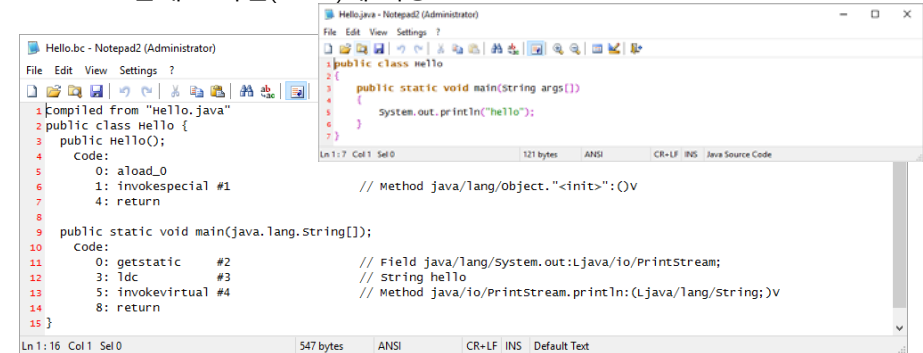
자바의 실행

- 자바 가상 기계가 클래스 파일(.class)의 바이트 코드 실행

바이트 코드

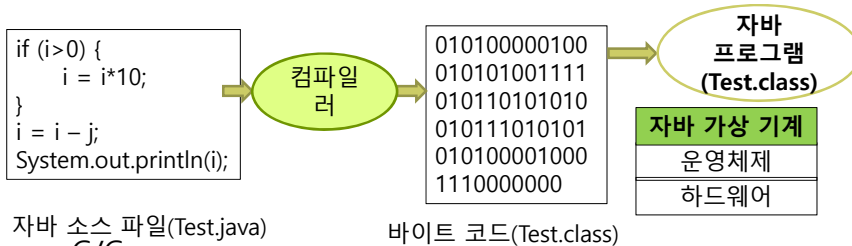
바이트 코드

- 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

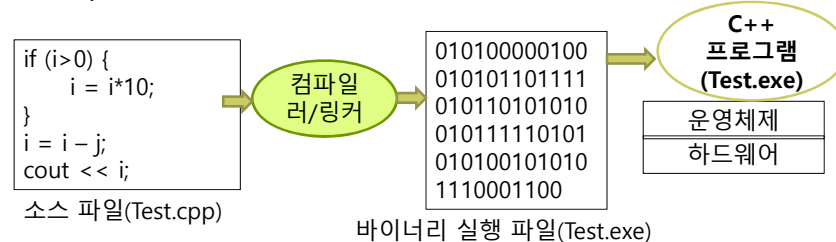


자바와 C/C++의 실행 환경 차이

자바



C/C++



자바와 C/C++의 실행 환경 차이

자바

- 자바는 링크 과정 없이 컴파일러가 바로 바이트 코드 생성
- 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
- 목적 코드 및 실행 파일은 플랫폼에 따라 다름
 - 플랫폼이 바뀌면 다시 컴파일 및 링크

자바의 종류

- 오라클은 개발 환경에 따라 다양한 자바 배포판 제공

□ Java SE

- 자바 표준 배포판(Standard Edition)

- 데스크탑과 서버 응용 개발 플랫폼

□ Java EE

- 자바 기업용 배포판

- 자바를 이용한 다중 사용자, 기업용 응용 개발을 위한 플랫폼

- Java SE + 인터넷 기반의 서버사이드 컴퓨팅 관련 API 추가

- 응용 서버, 웹서버, J2EE API, 엔터프라이즈 자바 빈즈(JavaBeans) 지원, 자바 서블릿 API 와 JSP 등을 포함

□ 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



자바의 종류

□ Java FX

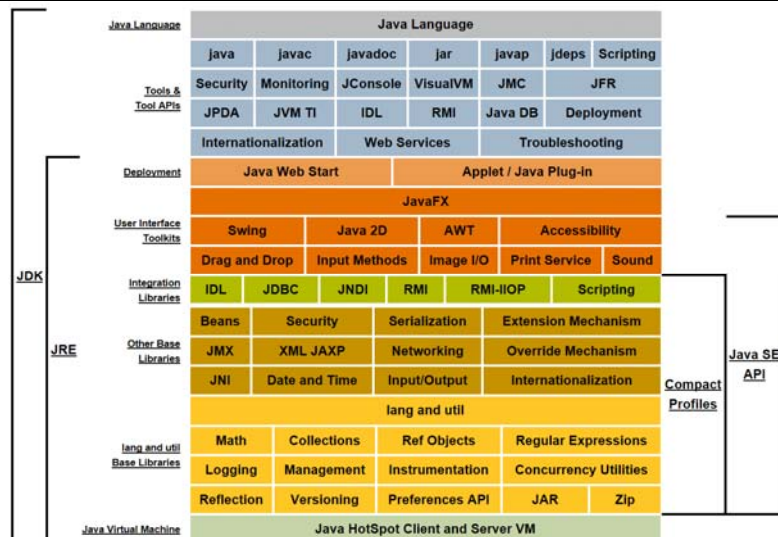
- 자바 리치-인터넷-애플리케이션(RIA: Rich Internet Application)을 생성하고 배포하기 위한 자바 클라이언트 플랫폼

- RIA은 서로 인터넷 상의 다양한 플랫폼에서도 동일한 외관으로 실행

- JavaFX 플랫폼은 자바 기술에 기반을 둔 고성능의 하드웨어 가속 그래픽과 미디어 엔진 API를 제공



Java SE



JDK와 JRE

□ JDK(Java Development Kit)

- 자바 응용 개발 환경. 개발에 필요한 도구 포함

- 컴파일러, JRE (Java Runtime Environment), 클래스 라이브러리, 샘플 등 포함

□ JRE(Java Runtime Environment)

- 자바 실행 환경. JVM 포함

- 자바 실행 환경만 필요한 경우 JRE만 따로 다운 가능

□ JDK와 JRE의 개발 및 배포

- 오라클의 Technology Network의 자바 사이트에서 다운로드

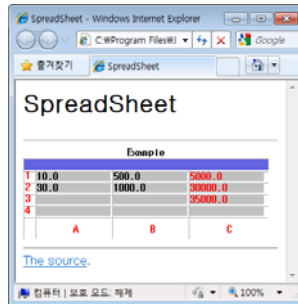
- <http://www.oracle.com/technetwork/java/index.html>

□ JDK의 bin 디렉터리에 포함된 주요 개발 도구

- javac - 자바 소스를 바이트 코드로 변환하는 컴파일러
- java - JRE의 bin 디렉터리에 있는 자바 응용프로그램 실행기
- jar - 자바 아카이브 파일 (JAR)의 생성 및 관리하는 유틸리티
- jdb - 자바 디버거
- appletviewer - 웹 브라우저 없이 애플릿을 실행하는 유틸리티

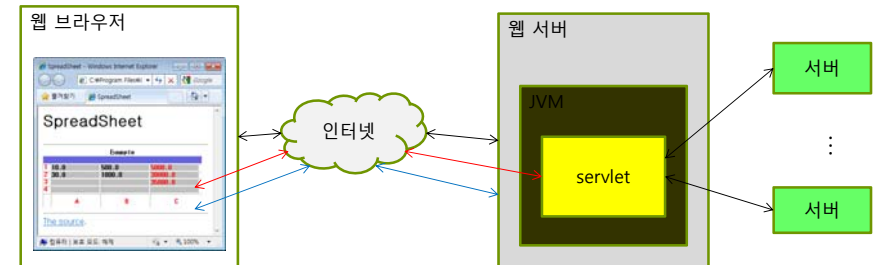
자바로 만들 수 있는 것

- 자바 애플리케이션(Java application)
 - 독립적으로 실행될 수 있는 일반 응용 프로그램
- 자바 애플릿(Java applet)
 - 웹 브라우저 안에서 실행되는 작은 프로그램
 - 애플릿은 사용할 수 있는 자원 접근에 제약 있음



자바로 만들 수 있는 것

- 자바 서블릿(Java servlet)
 - 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 구현시 사용. 웹 서버에 의해 실행 통제 받음.



자바로 만들 수 있는 것

- JSP (Java Server Pages)
 - HTML안에 자바 스크립트 코드를 넣으면 웹페이지를 사용자와 상호작용하도록 만들 수 있음
 - JSP는 서버에서 실행되고 클라이언트 요청에 따라서 동적으로 HTML, XML 웹페이지를 생성

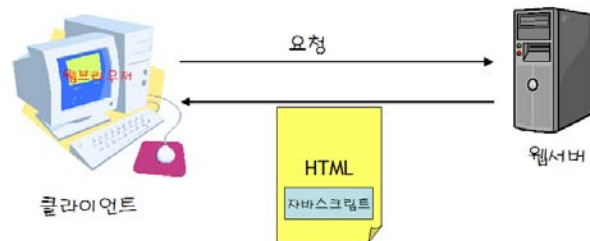


그림 1.14 JSP의 실행 과정

자바로 만들 수 있는 것

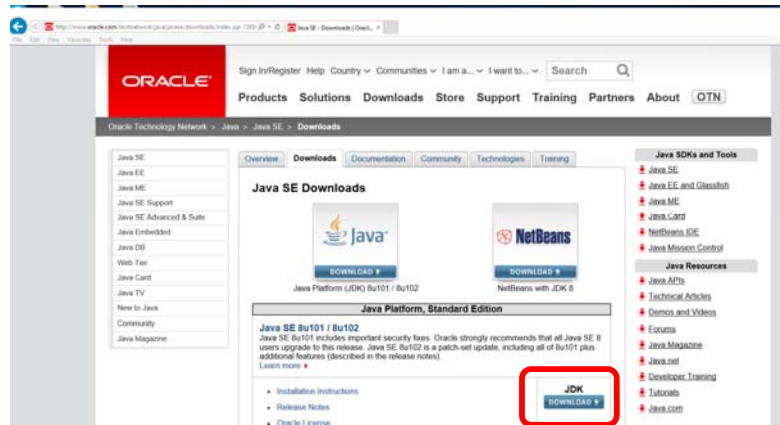
- 안드로이드 애플리케이션
 - 안드로이드 개발자들은 자바의 SE 버전 중에서 AWT와 스윙(swing)을 제외한 거의 모든 패키지를 사용함
 - 개발 언어는 자바를 사용하나 효율성때문에 자체적인 달빅(Dalvik) 가상 머신을 구현함. 달빅은 기존 바이트 코드와 호환성이 없어 변환이 필요함.



JDK 설치

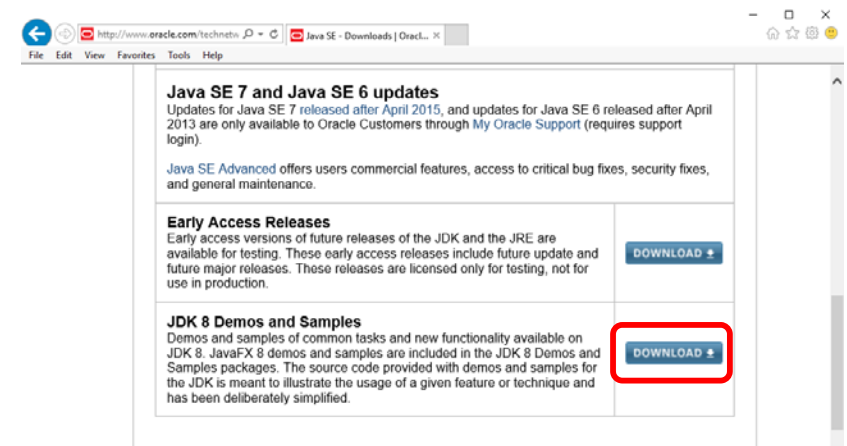
Java SE Download

- http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

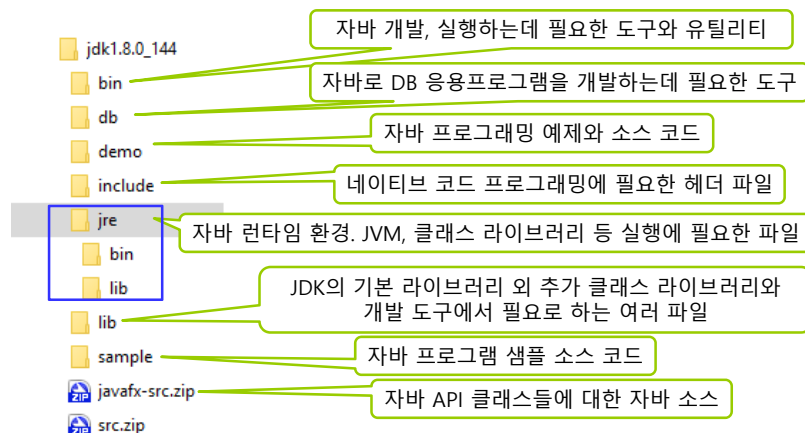


JDK 설치

JDK8 Demos and Samples Download

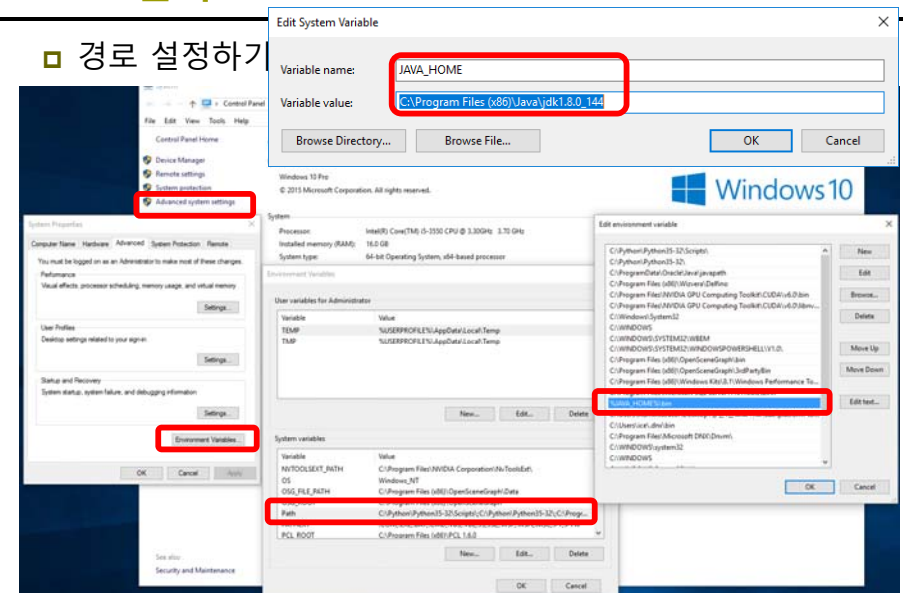


JDK 설치 후 디렉터리 구조



JDK 설치

경로 설정하기



JDK 설치

설치된 자바 버전 확인

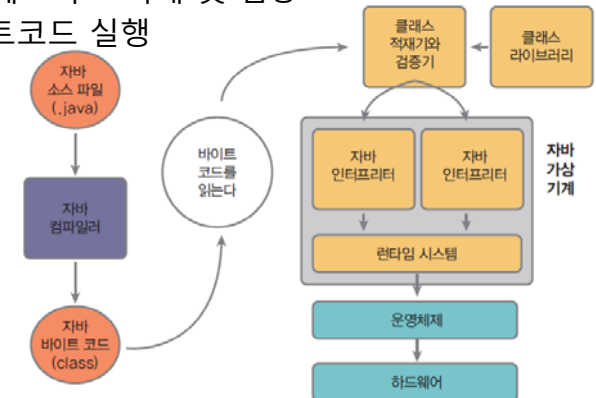
```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\park>javac -version
javac 1.8.0_144

C:\Users\park>
```

자바 프로그램 개발 단계

- 소스 파일 .java
- 컴파일 후 바이트코드로 변환 .class
- 바이트코드를 메모리로 적재 및 검증
- JVM에서 바이트코드 실행



자바 프로그램 개발 단계

소스코드 Hello.java

```
Hello.java - Notepad2 (Administrator)
File Edit View Settings ?
1 public class Hello
2 {
3     public static void main(String args[])
4     {
5         System.out.println("hello");
6     }
7 }
```

컴파일 javac Hello.java

실행 java Hello

```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\park>javac Hello.java
C:\Users\park>java Hello
Hello

C:\Users\park>
```

자바 프로그램 개발 단계

Disassemble

- 클래스 파일에 들어있는 바이트 코드를 텍스트로 볼 수 있게 변환하는 작업 (JDK의 javap.exe 이용)
- Hello.class 파일을 디스어셈블한 결과 Hello.bc 파일 생성

```
C:\Users\park> javap -c Hello > Hello.bc
```

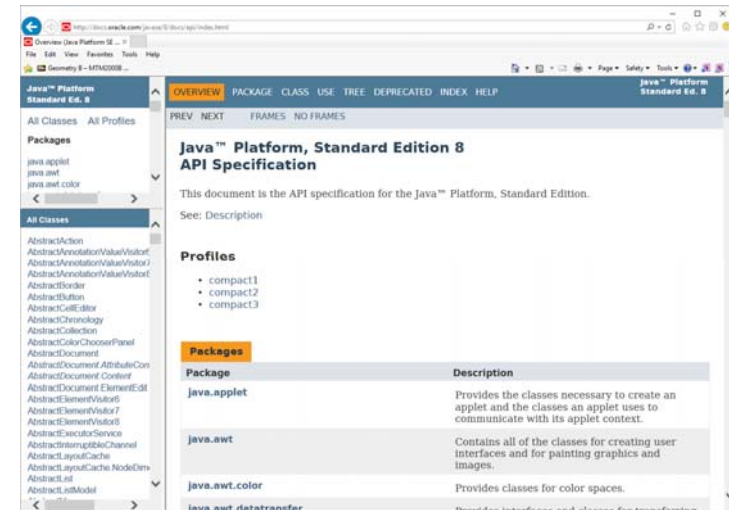
```
Hello.bc - Notepad2 (Administrator)
File Edit View Settings ?
1 compiled from "Hello.java"
2 public class Hello {
3     public Hello();
4     code:
5         0: aload_0
6         1: invokespecial #1           // Method java/lang/Object."<init>":()V
7         4: return
8
9     public static void main(java.lang.String[]);
10    code:
11        0: getstatic  #2             // Field java/lang/System.out:Ljava/io/PrintStream;
12        3: ldc       #3              // String hello
13        5: invokevirtual #4          // Method java/io/PrintStream.println:(Ljava/lang/String;)V
14        8: return
15 }
```

자바 API

- 자바 API(Application Programming Interface)
 - JDK에 포함된 클래스 라이브러리
 - 주요한 기능들을 미리 구현한 코드(클래스 라이브러리)의 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

자바 API 문서

- <http://docs.oracle.com/javase/8/docs/api/index.html>

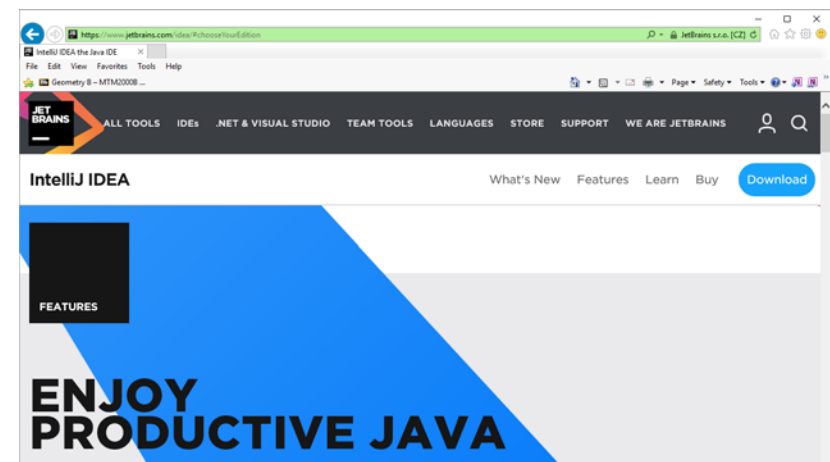


자바 IDE 소개와 설치

- 자바 IDE (Integrated Development Environment)
 - 통합 개발 환경
 - 편집, 컴파일, 디버깅을 한번에 할 수 있는 통합된 개발 환경
- 이클립스(Eclipse)
 - 자바 응용 프로그램 개발을 위한 통합 개발 환경
 - IBM에 의해 개발된 오픈 소스 프로젝트
 - <http://www.eclipse.org/downloads/> 에서 다운로드
- 넷빈즈(Netbeans)
 - <https://netbeans.org/>
- IntelliJ Idea
 - <http://www.jetbrains.com/idea/download/>

IntelliJ 설치하기

- IntelliJ Community (ideaIC-2016.2.2.exe)
<http://www.jetbrains.com/idea/download/> 다운로드



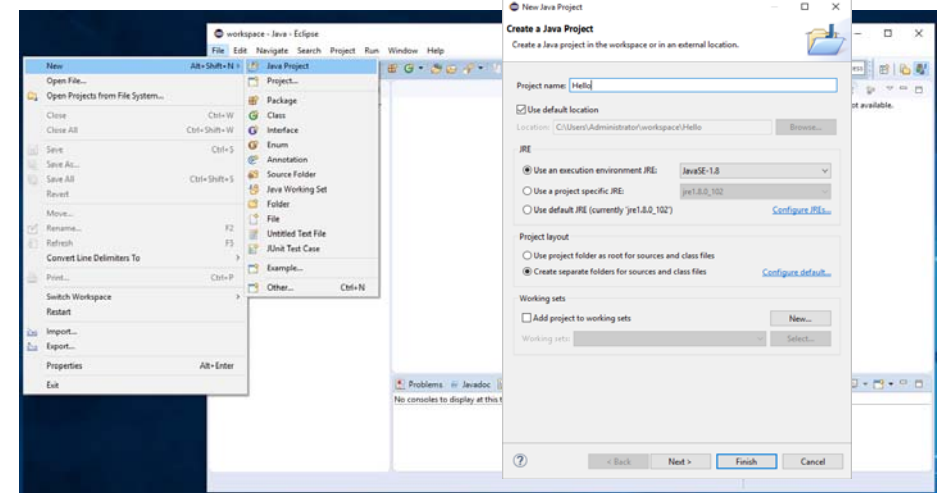
Eclipse 설치하기

- 이클립스 Oxygen (eclipse-java-oxygen-R-win32.zip)
<https://www.eclipse.org/downloads/> 다운로드
- eclipse.exe 실행



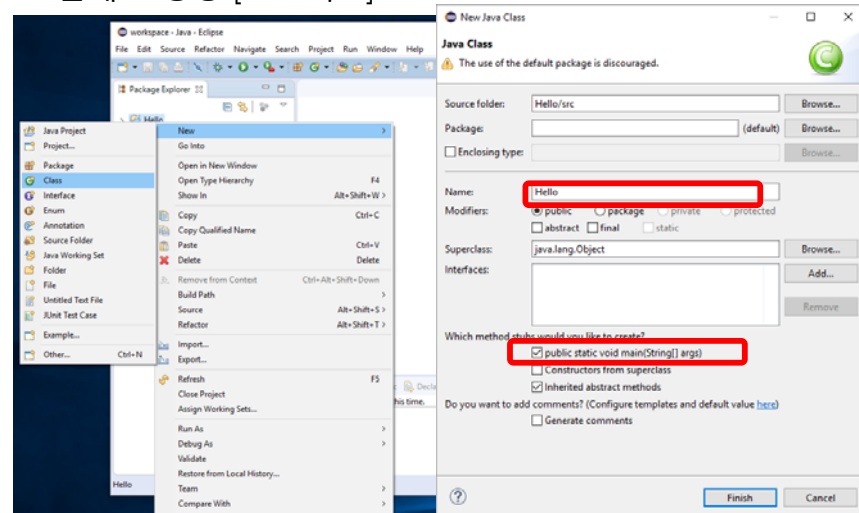
자바 프로그램 작성

- 이클립스 프로젝트 생성 File->New->Java Project



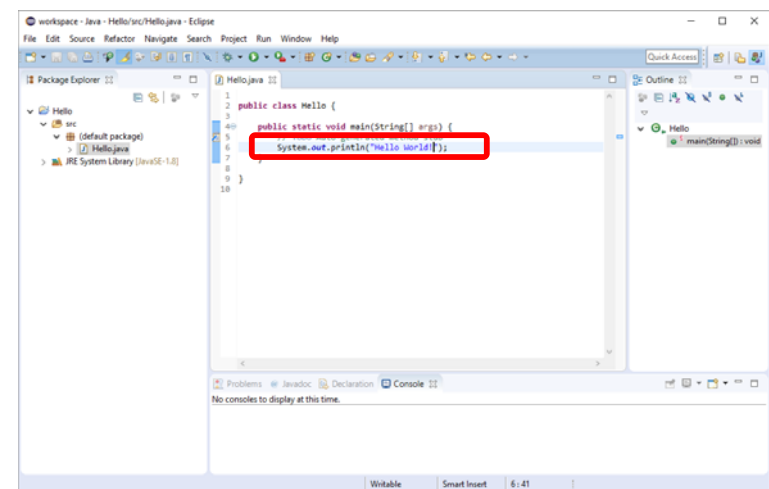
자바 프로그램 작성

- 클래스 생성 [프로젝트]->New->Class



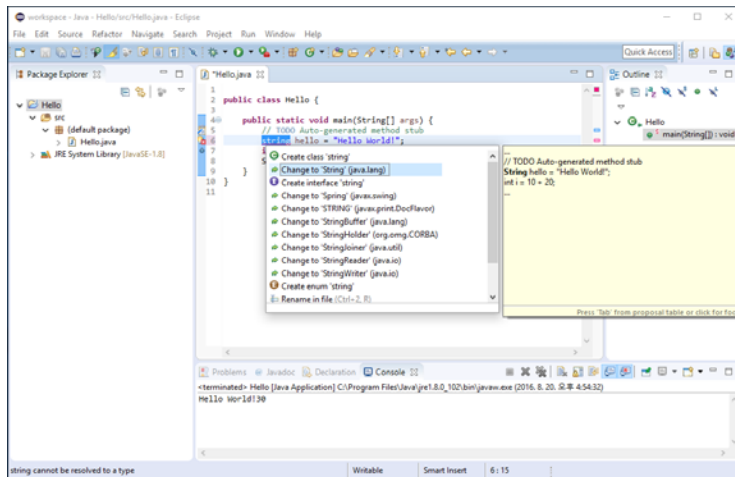
자바 프로그램 작성

- 클래스에 소스 코드 추가



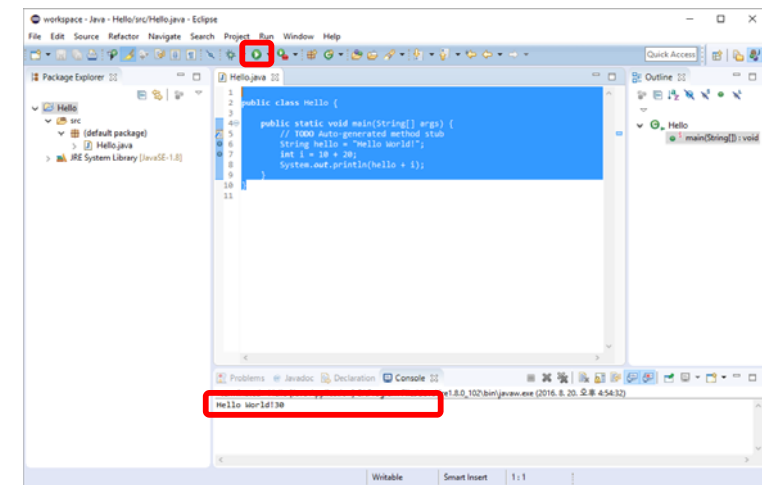
자바 프로그램 컴파일 오류

- 컴파일 오류시 quick fix를 눌러서 수정



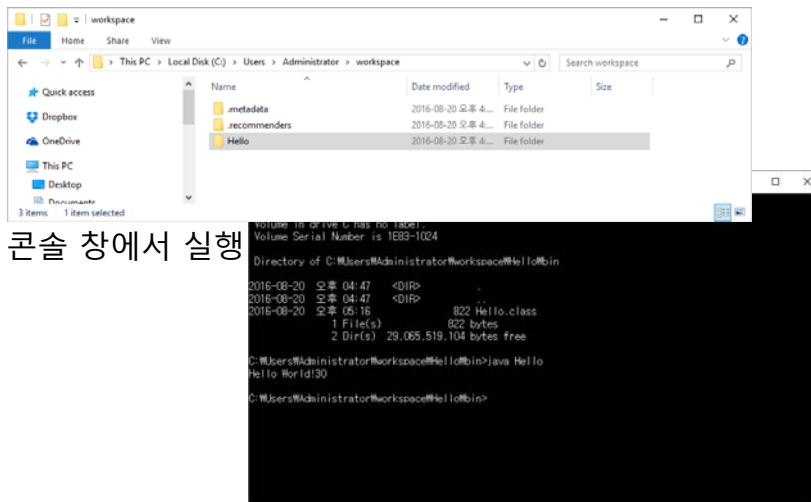
자바 프로그램 실행

- 프로그램 실행 Run (Ctrl+F11)



자바 프로그램 실행

- 이클립스 프로젝트 디렉토리



- 콘솔 창에서 실행

LAB0

- HelloWorld 프로그램을 작성한다.

- JDK 설치
- IDE 설치
- 환경설정
- 이클립스를 사용한 자바 프로그램 작성
- 프로젝트 디렉토리 안에 보고서 (1~2장)를 넣고 Lab0_학번_이름.zip 압축한 후 제출