Programming Language

프로그래밍 언어

- 컴퓨터 프로그램
  - ✓ 어떤 순서나 절차에 의해 나열된 항목으로 운영체제나 가상 머신(Virtual Machine)에 의해 실행되는 것
- 프로그래밍 언어
  - ✓ 인간이 생각하는 것을 프로그램으로 만들어서 컴퓨터로 실행시키기 위해 컴퓨터 또는 운영체제 또는 컴파일러가 이해할 수 있는 언어

프로그래밍 언어

❖ 프로그래밍

✓ 프로그래밍 언어를 사용하여 실행 가능한 프로그램을 개발하는 것

[프로그래밍 언어의 사용]

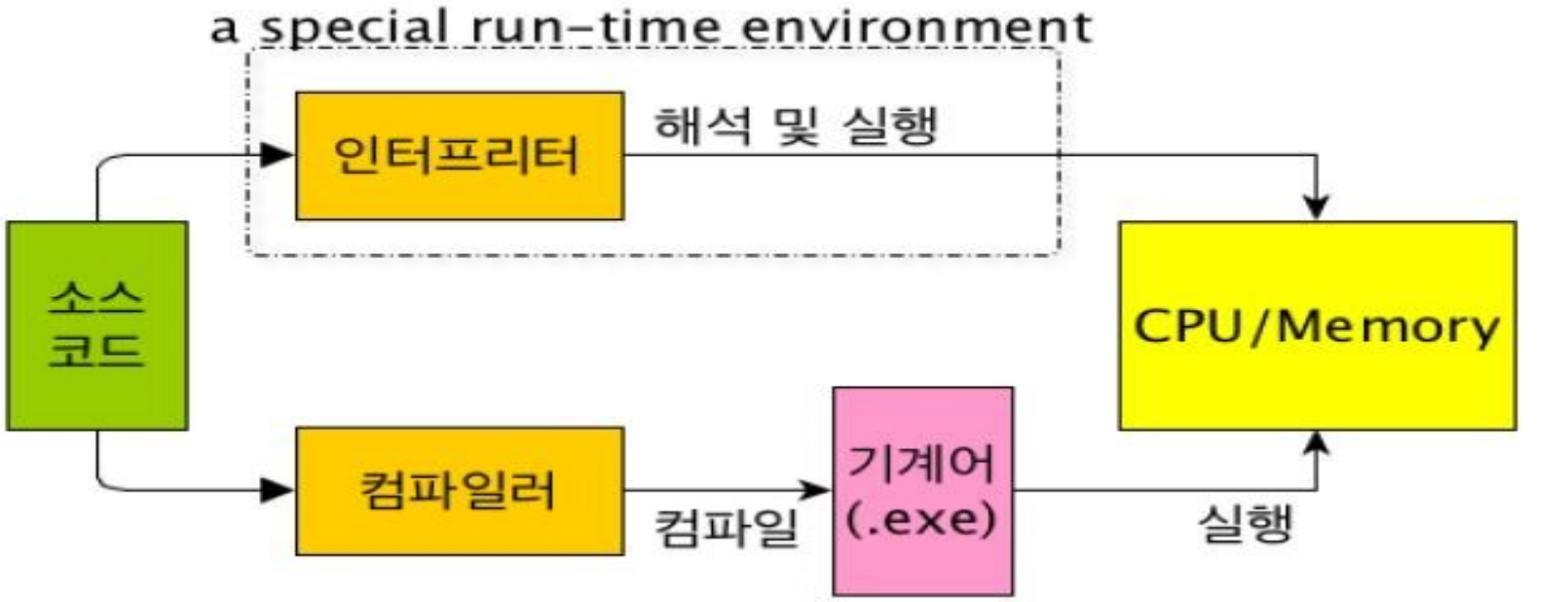
프로그래밍 언어

❖ 프로그래밍 언어 순위

Rank	Change	Language	Share	Trend
1		Python	28.29 %	-1.8 %
2		Java	17.31 %	-0.7 %
3		JavaScript	9.44 %	-0.1 %
4		C#	7.04 %	-0.1 %
5		C/C++	6.27 %	-0.4 %
6		PHP	5.34 %	-1.0 %
7		R	4.18 %	+0.3 %
8	↑↑↑	TypeScript	3.05 %	+1.5 %
9	↑↑↑	Go	2.16 %	+0.6 %
10		Swift	2.11 %	+0.5 %
11	↓↓↓	Objective-C	1.93 %	-0.0 %
12	↓↓↓	Kotlin	1.88 %	+0.0 %
13		Matlab	1.55 %	+0.1 %
14	↑↑	Rust	1.5 %	+0.7 %
15		Ruby	1.13 %	+0.1 %
16	↓↓	VBA	0.94 %	-0.3 %
17	↑	Dart	0.83 %	+0.2 %

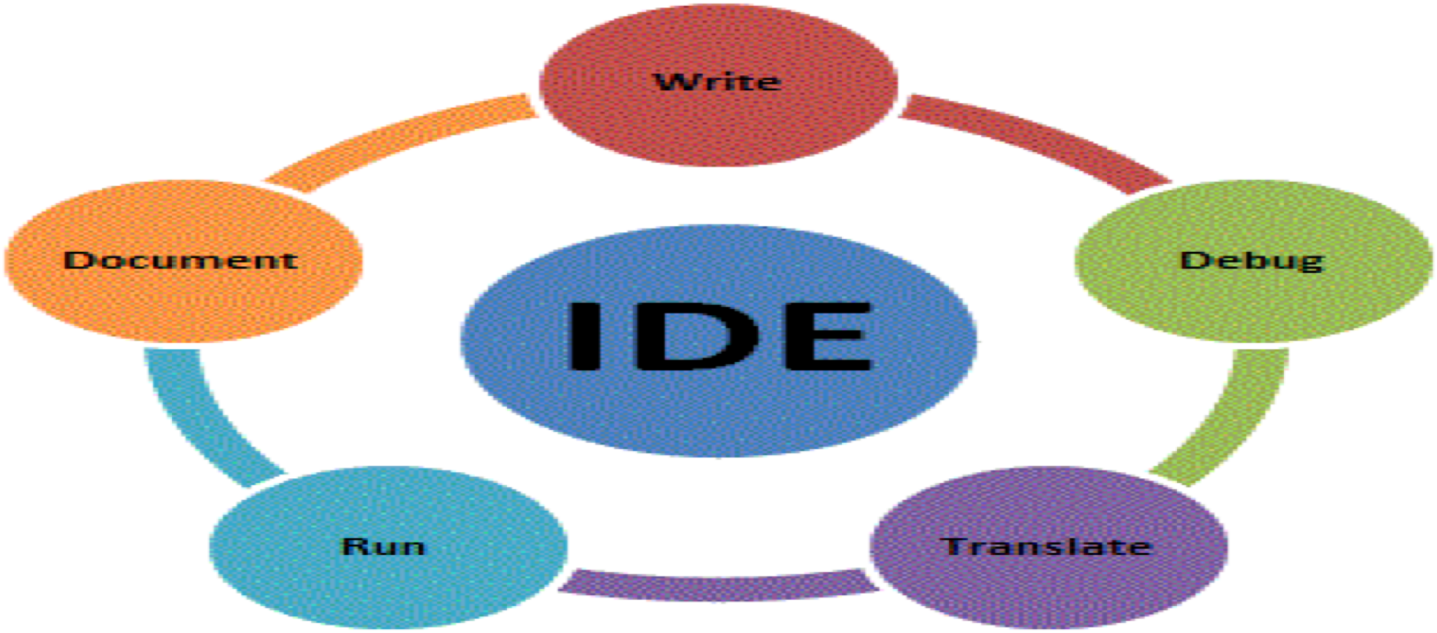
프로그래밍 언어

- 컴파일러 & 인터프리터
  - ✓ 인간이 고급 프로그래밍 언어로 작성한 것(Source Code)을 운영체제(컴퓨터가 동작할 수 있도록 해주는 시스템 소프트웨어)나 가상 머신(JVM, EVM 등)이 이해 할 수 있도록 번역해주는 프로그램 – 프로그래밍을 할 때 필수
  - ✓ 언어를 번역해주는 프로그램은 운영체제 별로 별도로 설치해야 함



프로그래밍 언어

- IDE(integrated development environment – 통합 개발 환경)
  - ✓ 코딩(작성), 디버그(오류를 찾아서 수정하는 과정), 컴파일(기계 나 운영체제가 이해할 수 있도록 번역), 빌드, 배포(실행하거나 접속할 수 있도록 해주는 과정) 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 프로그램 – 사용하지 않으면 프로그래밍 하기가 불편



프로그래밍 언어

- 자료구조
  - ✓ 컴퓨터 과학에서 효율적인 접근 및 수정을 가능케 하는 자료의 조직, 관리, 저장을 의미
  - ✓ 자료 구조는 데이터 값의 모임, 데이터 간의 관계, 그리고 데이터에 적용할 수 있는 함수나 명령을 의미
  - ✓ 신중히 선택한 자료구조는 보다 효율적인 알고리즘을 사용할 수 있게 함
  - ✓ 자료구조의 선택 문제는 대개 추상 자료형의 선택으로부터 시작하는 경우가 많음
  - ✓ 효과적으로 설계된 자료구조는 실행시간 혹은 메모리 용량과 같은 자원을 최소한으로 사용하면서 연산을 수행하도록 해줌
- 알고리즘
  - ✓ 알고리즘(algorithm)은 수학과 컴퓨터과학, 언어학 또는 엮인 분야에서 어떠한 문제를 풀어내기 위해 정해진 일련의 절차나 방법을 공식화한 형태로 표현한 것으로 계산을 실행하기 위한 단계적 절차를 의미
  - ✓ 문제 풀이에 필요한 계산 절차 또는 처리과정의 순서를 의미
  - ✓ 프로그램명령어의 집합을 의미하기도 함
  - ✓ 알고리즘은 연산, 데이터 마이닝(기계 학습) 또는 자동화된 추론을 수행
- 컴퓨테이셔널 씽킹
  - ✓ 컴퓨터 시스템은 하드웨어와 소프트웨어로 이뤄져 있으며 이러한 컴퓨터 시스템이 문제를 해결하는 단위
  - ✓ 본래의 추상적 문제를 컴퓨터 시스템 단위로 분해하고 이들이 모여서 본래의 커다란 문제를 해결하도록 하는 전략

프로그래밍 언어

- 알고리즘 학습 사이트
  - ✓ 프로그래머스 - https://programmers.co.kr/
  - ✓ 백준 - https://www.acmicpc.net/
  - ✓ 더블릿 - http://220.89.64.243/
  - ✓ 코딩 도장(C, Python) - https://codingdojang.com/
  - ✓ 코드 그라운드 - https://www.codeground.org/
  - ✓ 정올 - http://www.jungol.co.kr/
  - ✓ 오일러 - https://euler.synap.co.kr/
  - ✓ 알고스팟 https://algospot.com/judge/problem/list/
  - ✓ 삼성 소프트웨어 아카데이 - https://swexpertacademy.com/main/main.do
  - ✓ 구름 - https://edu.goorm.io/

Java

- JAVA
  - ✓ SUN Micro Systems에서 1995년에 발표한 객체 지향 프로그래밍 언어
  - ✓ 등장 배경: 여러 플랫폼에서 실행되는 프로그램을 모든 플랫폼마다 만들지 않도록 하는 플랫폼 호환성 문제를 해결하기 위해서 플랫폼 독립적인 언어가 필요
  - ✓ Java 이름의 유래: 인도네시아 산 커피 원료 이름 자바 에서 유래 (잠들지 않는 인터넷이라는 의미)

Java

- Java 특징
  - ✓ 운영체제에 독립적인 객체지향 언어
    - 플랫폼 독립성(Write Once, Use Anywhere)
    - OS에 상관없이 한번만 작성하면 자바가 지원되는 모든 운영체제에서 동작 – JVM 활용
  - ✓ 오픈 소스 프로젝트가 많이 구현되어 있음 - 뛰어난 Echo System
    - 자바 개발에 편리한 라이브러리 – Apache Common 프로젝트
    - 자바 서버 프레임워크 플랫폼 – Spring, Struts
    - 검색엔진 – Lucene
    - NoSQL – Cassandra
    - 분산 파일 시스템 – Hadoop
  - ✓ 자바 가상 머신 위에서 실행되는 언어가 많음
    - Jython, Scala, Kotlin, Closure, Jruby, Groovy…
  - ✓ 이식성이 높은 언어
  - ✓ Garbage Collection을 이용한 메모리 관리

Java

- ❖ 플랫폼 독립성
  - ✓ WORA(Write Once Run Anywhere)
    - 한 번 작성된 코드를 가지고 모든 플랫폼(운영체제)에서 실행할 수 있음
    - C/C++ 언어가 가진 플랫폼 종속성 극복: OS, H/W에 상관없이 한번만 작성
    - 네트워크에 연결된 어느 클라이언트에서나 실행 가능: 웹 브라우저, 분산 환경 지원
    - WORA를 가능하게 하는 자바의 특징: 바이트 코드(byte code)
      - ◆ 자바 소스 코드를 컴파일 한 목적 코드
      - ◆ CPU에 종속적이지 않은 중립적인 코드
      - ◆ JVM에 의해 해석되고 실행됨
    - JVM(Java Virtual Machine) – JRE를 설치하면 자동으로 설치됨
      - ◆ 자바 바이트 코드를 실행하는 자바 가상 기계(소프트웨어)
      - ◆ 자바 프로그램은 운영체제가 실행하지 않고 운영체제에 설치된 JVM 위에서 실행됨
      - ◆ 자바 프로그램이 실행되기 위해서는 반드시 JVM이 설치되어 있어야 함

Java

❖ 플랫폼 독립성

✓ 플랫폼 종속성

인텔 CPU를 가진 리눅스 환경에서 개발

C/C++ 응용 프로그램

실행

실행되지 않음

실행되지 않음

인텔 CPU + 리눅스

Apple 사의 MAC PC

인텔 CPU + 윈도우 노트북

플랫폼 = 하드웨어 플랫폼 + 운영체제 플랫폼

❖ 프로그램의 플랫폼 호환성 없는 이유

✓ 기계어가 CPU마다 다름

✓ 운영체제마다 API 다름

✓ 운영체제마다 실행파일 형식 다름

Java

❖ 플랫폼 독립적

Write Once !!

자바
응용 프로그램

Run Anywhere!!

실행

실행

실행

자바 가상 기계

인텔 CPU + 리눅스

자바 가상 기계

Apple 사의 MAC PC

자바 가상 기계

인텔 CPU + 윈도우 노트북

Java

- Java 개발 플랫폼
  - ✓ J2SE(Java2 Standard Edition)
    - 자바의 기본 개발 환경
    - 거의 모든 운영체제에서 사용할 수 있고 그래픽, 네트워크 등의 기능이 있음
    - android는 J2SE 플랫폼을 이용
    - JDK와 JRE를 같이 설치 할 수 있음
    - WAS(Tomcat) 나 프레임워크(Spring)를 이용하면 웹 프로그래밍을 할 수 있음
  - ✓ J2EE(Java2 Enterprise Edition)
    - 서버 프로그램을 개발할 때 사용
    - HttpServlet, JSP(Java Server Pages), EJB(Enterprise Java Beans) 사용 가능
  - ✓ J2ME(Java2 Micro Edition)
    - 휴대폰이나 PDA와 같은 embedded 관련 프로그램을 개발할 때 사용
    - 메모리 관리에는 최적화 되어 있지만 전원 처리, 장치 간의 입출력 기능 등은 제한적으로 제공

Java

- Java Environment
  - ✓ JDK(Java Developer Kit): 자바로 애플리케이션을 개발하는데 필요한 여러가지 도구를 모아서 제공 - 서버 등의 운영환경에서 디버깅 및 분석 등을 하고자 하는 경우 JDK 가 필요할 수 있음
    - 자바 API(Application Programming Interface – Software Development Kit): 자바에서 기본적으로 제공하는 클래스와 인터페이스들의 모임이며 패키지(연관성이 있는 클래스 또는 파일의 집합) 단위로 구성
    - JVM(Java Virtual Machine): 자바 프로그램을 실행할 수 있도록 물리적인 하드웨어들을 추상화 한 영역으로 명령어, 레지스터, Stack, Heap, Method 영역 등으로 구성
    - 컴파일러 등의 개발 도구

Java

- Java Environment
  - ✓ JRE(Java Runtime Environment)
    - 자바로 만들어진 응용 프로그램을 실행하기 위한 환경
    - 구성
      - 프로그램 실행에 필요한 JVM
        - 컴파일 명령으로 생성한 자바 바이트 코드를 실행할 수 있는 가상의 운영체제 및 CPU가 JVM
        - 자바 플랫폼은 다양한 플랫폼을 지원하여 Middle Ware로서의 역할과 플랫폼으로서의 역할을 동시에 수행

Java

- ❖ Java Environment
  - ✓ JRE(Java Runtime Environment)
    - ● 구성
      - ◆ 프로그램 실행에 필요한 JVM
        - ▪ 대표적인 JVM

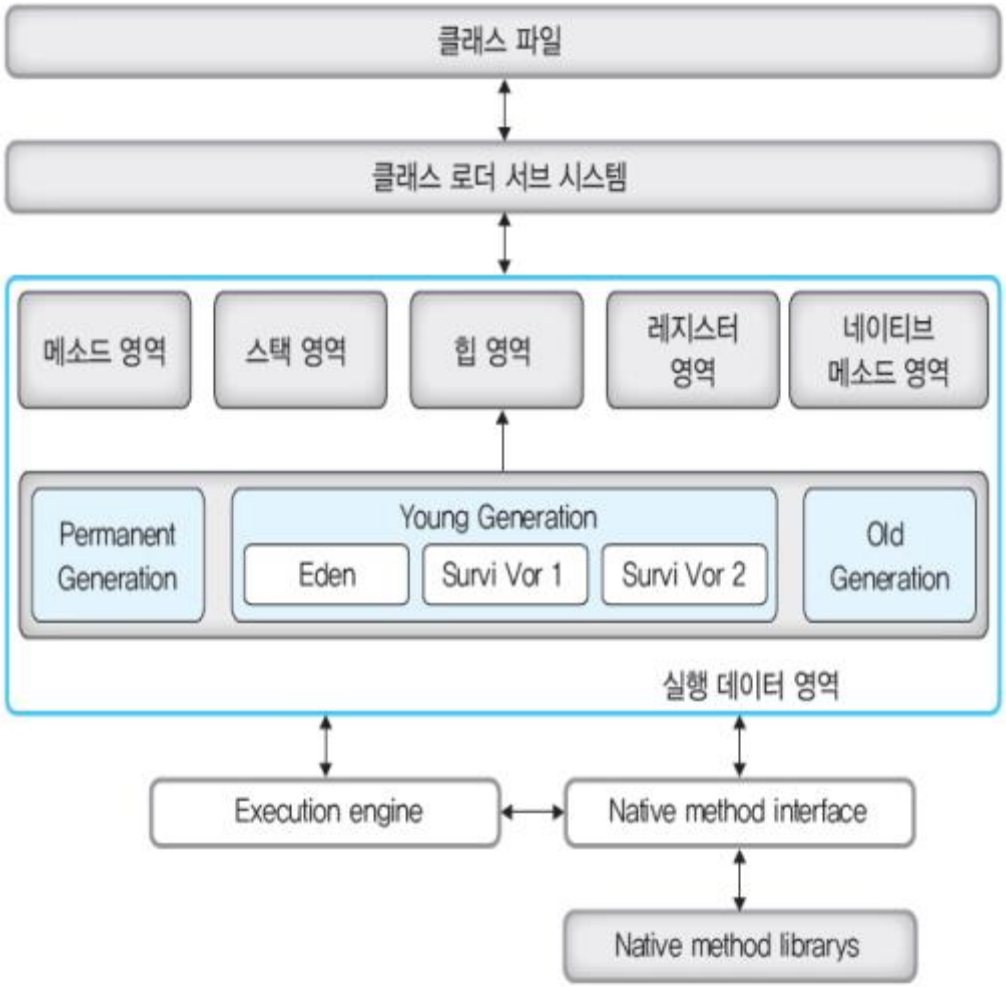
명칭	특징
HotSpot	오라클에서 제공하는 가장 널리 사용되고 있는 Java VM. 원래는 선 마이크로시스템즈가 개발했다. 하지만 오라클에 인수되어 오라클에서 개발을 계속하게 되었다.
JRockit	오라클에서 제공하는 Java VM. 원래는 Appeal Virtual Machines, BEA Systems가 개발했지만 오라클에 인수되어 오라클에서 개발을 계속하게 되었다. Java SE 8의 시점에서 Mission Control과 같은 주요 기능이 HotSpot에 통합되어 개발이 중지되었다.
IBM JVM	IBM에서 개발된 Java VM. IBM의 WebSphere 제품과 DB2 제품의 표준 VM으로 이용되고 있다.
HP-UX JVM	휴렛 패커드에서 개발된 Java VM. 휴렛 패커드의 HP-UX 시스템의 표준 VM으로 이용되고 있다.
Zing	Azul Systems에 의해 개발된 Java VM. 대량의 메모리를 처리하는 데 뛰어나다는 특징이 있다.
OpenJDK	오픈소스 버전의 Java SE. 리눅스의 주요 배포판에 기본적으로 설치되어 있다.

Java

- ❖ Java Environment
  - ✓ JRE(Java Runtime Environment)
    - ● 구성
      - ◆ 글루(glue): 플랫폼 고유의 라이브러리와 JNI 코드
      - ◆ 바이트 코드
        - ▪ JVM에서 실행 가능한 바이너리 코드
        - ▪ 바이트 코드는 컴퓨터 CPU에 의해 직접 실행되지 않고 JVM이 작동 중인 플랫폼에서 실행
        - ▪ JVM이 인터프리터 방식으로 바이트 코드 해석
        - ▪ 클래스 파일(.class)로 저장
        - ▪ 자바 바이트 코드는 자바 소스 코드를 compile해서 생성하지만 다른 언어의 컴파일러에서도 생성 가능(Kotlin은 소스 코드를 컴파일하면 자바의 바이트 코드가 만들어 짐)

Java

- ❖ JVM의 구성
  - ✓ JVM의 내부 구조를 큰 형태로 분리
    - 클래스 로더 서브 시스템
    - 실행 데이터 영역
    - Execution Engine



Java

- JVM의 구성
  - ✓ 클래스 파일
    - JDK 메서 제공하기도 하고 개발자가 만들기도 하고 다른 개발자들이 만들어서 제공
    - 자바 소스 파일의 확장자는 .java
    - .java 파일이 자바 컴파일러에 의해 컴파일 과정을 거치면 .class 파일 생성
    - 컴파일 과정을 거쳐 생성된 클래스 파일은 JVM에서 실행 가능
    - 프로그램은 클래스 파일이 서로 유기적으로 동작하면서 기능을 수행
  - ✓ 클래스 로더 서브 시스템
    - 자바 클래스 파일들은 JVM 위에서 동작
    - JVM은 실행할 클래스 파일을 읽고 JVM 메모리에 올려놓는 과정이 필요한데 이러한 과정을 클래스 로딩(Class loading)이라 하며 JVM의 클래스 로더 서브 시스템이 담당
    - 클래스 로더 서브 시스템을 줄여서 클래스 로더라 부름
    - 동적 로딩: 프로그램을 실행하는 도중에 새로운 클래스를 로딩할 수 있음
      - ◆ 로드 타임 동적 로딩(Load time dynamic loading): 프로그램 실행 초기에 클래스를 로딩하는 것
      - ◆ 런타임 동적 로딩(Runtime dynamic loading): 프로그램 실행 중간에 클래스를 로딩하는 것

Java

- JVM의 구성
  - ✓ 메서드 영역(static 영역, 클래스 영역)
    - 실행에 필요한 클래스(메서드와 코드)들이 Load되는 영역으로 클래스에 작성한 코드 저장 - MetaData
    - import 구문에 작성된 패키지의 클래스들이 Load 되고 사용자가 작성한 클래스도 Load 되고 강제로 클래스를 로드 할 수 도 있음
    - 클래스는 Load되면 프로그램이 종료될 때까지 내용이 메모리에서 소멸되지 않음
    - 상수(리터럴)도 이 영역에 저장
  - ✓ 자바 Stack
    - Method를 호출하면 Method 수행에 필요한 데이터들을 저장하는 영역
    - 가장 최근에 호출된 Method의 프레임이 가장 위에 존재하고 이 프레임이 활성 Stack 프레임
    - Method의 수행이 종료되면 자동적으로 소멸

Java

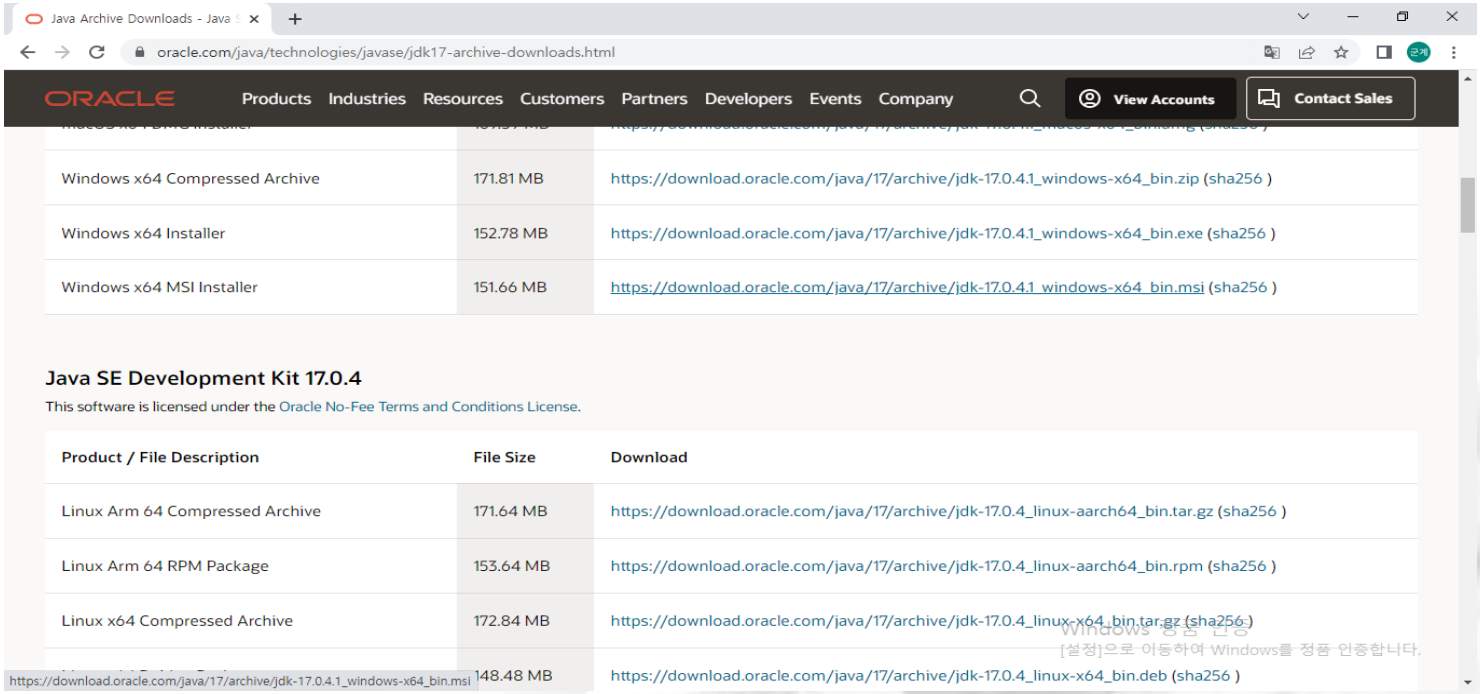
- JVM의 구성
  - ✓ Heap
    - new라는 예약어를 이용하여 동적으로 할당받는 요소들의 영역으로 Garbage Collection이 해제를 관리하는 영역으로 다시 permanent 영역과 Old, Young 영역으로 분리
  - ✓ Native Method Stacks
    - 자바는 하드웨어를 직접 제어하지 못하므로 다른 언어의 기능을 빌려서 사용해야 하는 경우가 있는데 이 때 사용되는 것이 JNI(Java Native Interface)
    - JNI는 다른 언어의 내용을 자바 바이트 코드로 변환해서 이 영역에 기록
  - ✓ Register: 다음에 실행 할 명령어 및 실행에 필요한 데이터를 저장

Java

- JVM의 구성 – Heap Area
  - ✓ Young Generation
    - 프로그램 내부에 어떤 코드가 실행되면서 새롭게 생긴 데이터가 저장되는 부분
    - 저장된 데이터의 사용이 끝나고 일정 시간 동안 사용되지 않으면(연결된 이름이 존재하지 않으면) 그 데이터는 지워짐
  - ✓ Old 영역
    - 빈번하게 사용되는 데이터는 Young 영역에서 Old 영역으로 이동
    - Old 영역에서 실행되는 Garbage Collector의 알고리즘은 Young 영역보다 수행 횟수가 적으며 속도도 느림
  - ✓ Permanent 영역
    - Permanent 영역은 클래스들의 정보가 저장되는 영역
    - JVM에서 클래스 정보를 바탕으로 메모리에 인스턴스를 빠르게 생성하기 위해 저장

Java 개발 환경 구성

- JDK & JRE 설치
  - ✓ JDK(자바 개발 도구) SE 버전 다운로드 – 운영체제에 맞는 버전 다운로드
    - https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html
    - 32bit 버전을 설치하는 경우 자바로 만들어진 모든 프로그램(Tomcat 등)도 32bit 버전이 설치되어야 함



Java 개발 환경 구성

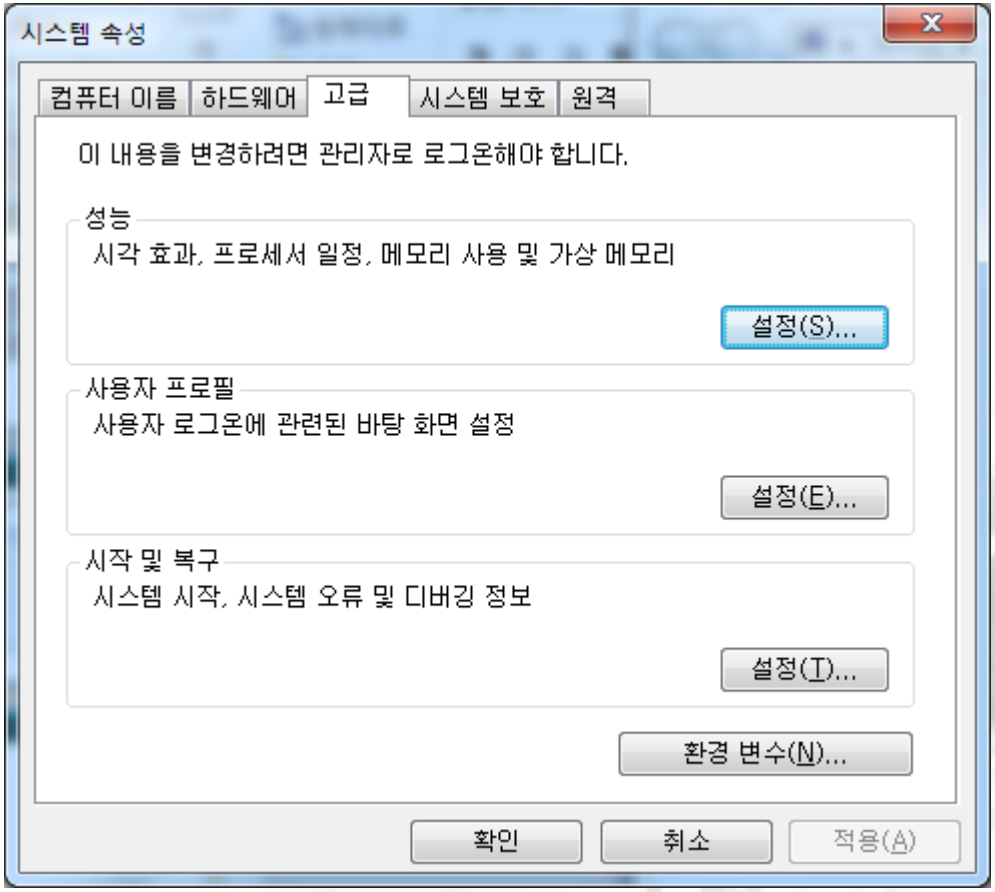
- JDK & JRE 설치
  - ✓ 윈도우에서 환경 변수 설정 – 각 항목은 ;으로 구분
    - 탐색기에서 [내 PC]를 선택하고 마우스 오른쪽 클릭 후 [속성]을 선택한 후 – [고급 시스템 설정] 메뉴를 누르고 [고급] 탭에서 [환경변수]를 선택
    - path
      - 콘솔(터미널)에서 명령어를 입력해서 프로그램을 실행할 때 프로그램을 찾는 경로를 설정하는 환경 변수
      - JDK가 설치된 경로의 bin 디렉토리 위치를 추가
    - JAVA_HOME: 다른 환경 변수를 설정할 때 java 경로를 편리하게 입력하기 위해서 java가 설치된 경로(윈도우에서 기본은 C:₩Program Files₩Java₩jdk버전)를 등록하고 path에는 %JAVA_HOME%₩bin; 으로 입력하기 도 함
    - IDE(Eclipse, Intelli J)를 사용해서 개발할 때는 위의 설정을 하지 않아도 개발은 가능
  - ✓ Mac 에서는 이러한 작업을 하지 않아도 환경 변수에 자동 설정됨
  - ✓ Linux 에서는 Open JDK를 apt-get을 이용해서 설치하면 자동으로 설정됨

Java 개발 환경 구성

- JDK & JRE 설치
  - ✓ 윈도우에서 환경 변수 설정 – 각 항목은 ;으로 구분
    - 탐색기에서 [내 PC]를 선택하고 마우스 오른쪽을 눌러서 속성을 선택

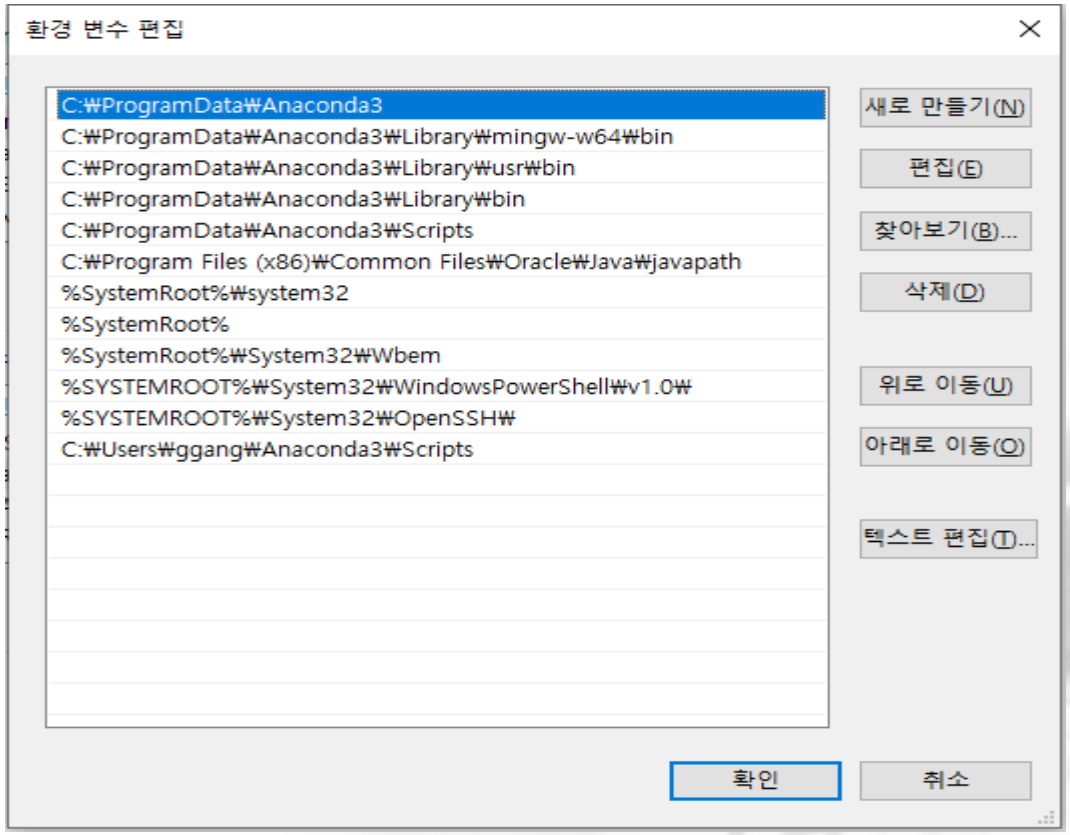
Java 개발 환경 구성

- JDK & JRE 설치
  - ✓ 윈도우에서 환경 변수 설정 – 각 항목은 ;으로 구분
    - 고급 시스템 설정을 클릭



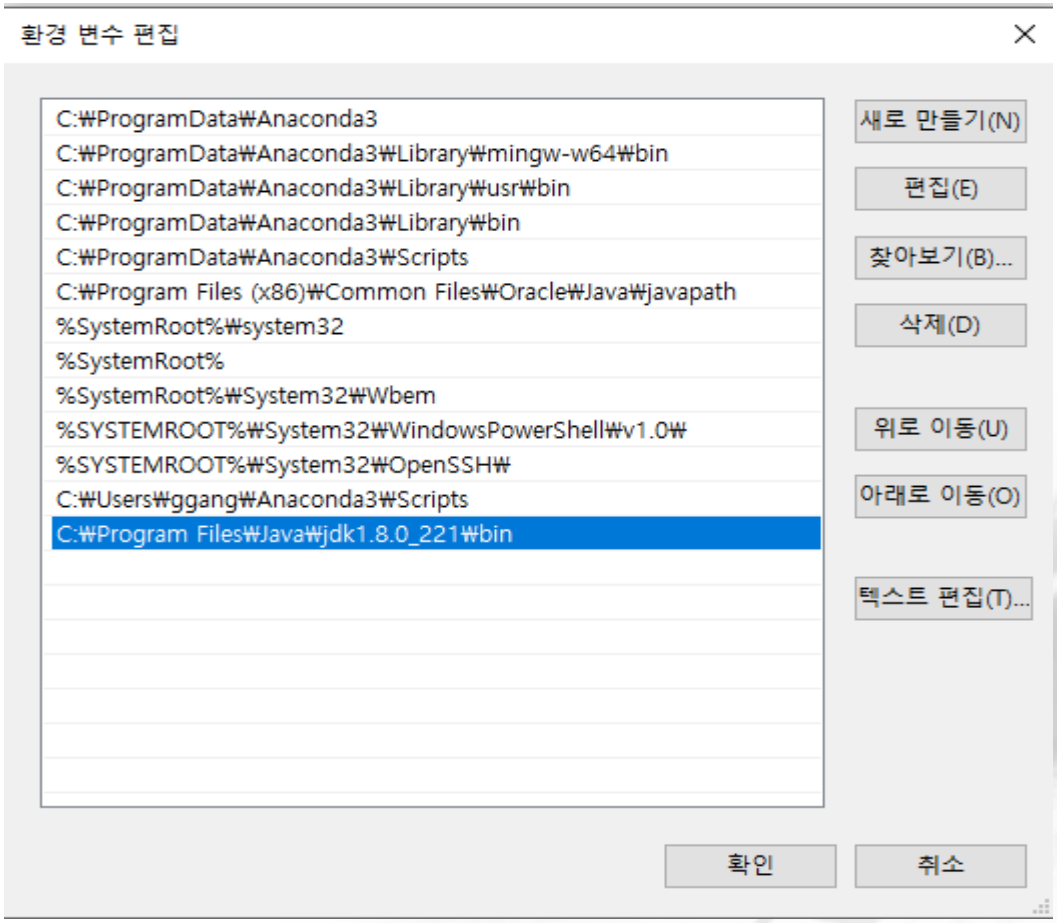
Java 개발 환경 구성

- JDK & JRE 설치
  - ✓ 윈도우에서 환경 변수 설정 – 각 항목은 ;으로 구분
    - 사용자 변수에 작성하고자 할 때는 새로 만들기를 클릭해서 작업
    - 시스템 변수에 작성하고자 할 때는 path 항목을 선택하고 편집을 클릭해서 작업



Java 개발 환경 구성

- JDK & JRE 설치
  - ✓ 윈도우에서 환경 변수 설정 – 각 항목은 ;으로 구분
    - 새로 만들기를 선택하고 Java가 설치된 경로의 jdk₩bin 디렉토리를 설정

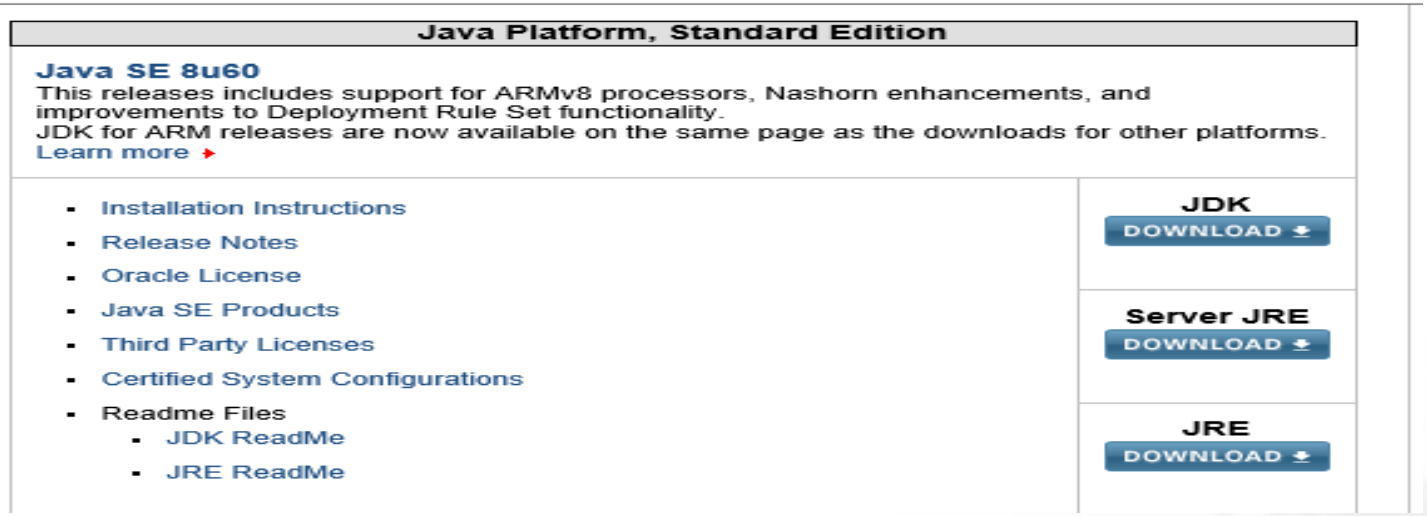


Java 개발 환경 구성

- JDK & JRE 설치
  - ✓ 설치 확인
    - 명령 프롬프트(cmd, 터미널) 실행
    - java –version 라고 입력하면 JRE 버전이 출력됨
    - javac-version라고 입력하면 jdk 버전이 출력됨 – path 설정이 안되어서 명령이 없다고 하기도 함

Java 개발 환경 구성

- API(Application Programming Interface)
  - ✓ JAVA API: https://docs.oracle.com/en/java/javase/17/docs/api/index.html



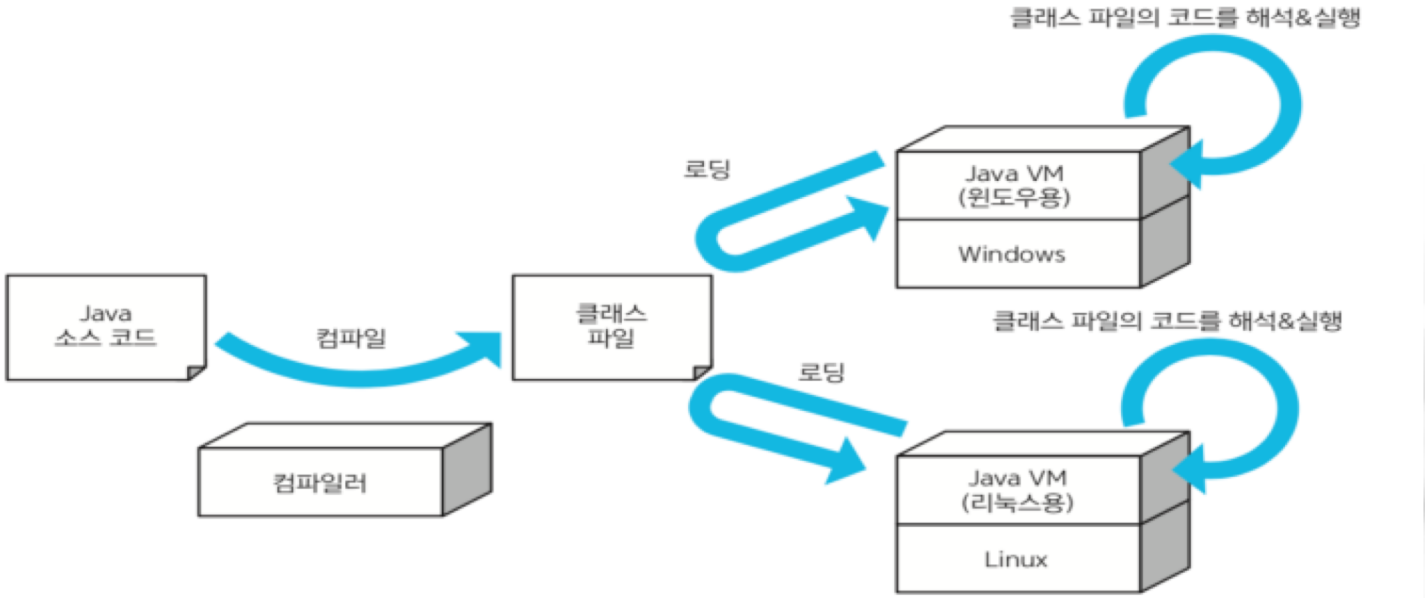
- 예제 코드
  - ✓ http://www.java2s.com/Code/JavaAPI/CatalogJavaAPI.htm
  - ✓ http://www.java2s.com/Tutorial/Java/CatalogJava.htm

프로그램 작성 및 실행

- 작성 및 실행
  - ✓ source code 작성(확장자는 java)
    - 메모장과 같은 편집기에서 Source Code 작성
    - java는 파일 이름이 클래스 이름과 동일해야 함
    - 클래스 이름과 파일이름이 같은지 확인
  - ✓ 컴파일: Source Code -> JVM이 인식할 수 있는 바이트 코드로 변환
    - [시작] – [실행] – [cmd]를 입력하고 확인
    - 프롬프트를 java 파일을 저장한 위치로 이동
    - (cd ..을 입력하면 상위 디렉토리로 이동, cd 경로를 입력하면 경로로 이동)
      javac 파일명.java 라고 입력하고 Enter
    - 이 과정에서 실패하면 문법적인 오류가 있는 것이므로 오류 메시지를 확인하고 수정
  - ✓ 실행
    - jvm을 구동시켜서 java 클래스를 기계어로 만들고 실행(javaw라는 프로세스가 구동)
      java 클래스명
    - main 메서드를 찾아서 실행하는 역할을 수행
    - 클래스 이름을 잘못 입력했거나 main 메서드가 없거나 예외가 발생한 경우

프로그램 작성 및 실행

- 자바는 프로그램을 실행할 때 Java VM이 중간 코드를 해석하면서 실행하기 때문에 C 언어처럼 CPU가 이해할 수 있는 기계어 프로그램을 생성해서 실행하는 언어보다 처리 속도가 늦다라고 여기던 시대가 있었는데 Java VM에 의한 오버헤드(처리에 걸리는 부하)는 있지만 한편으론 JIT(Just In Time) 컴파일러라는 런타임 시에 최적화하는 기술이 진보한 덕택에 C언어와 비교해도 손색이 없을 정도의 속도까지 향상되있음



프로그램 작성 및 실행

❖ Source Code 작성 및 저장

✓ Editor를 열어서 작성

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

✓ 저장(First.java)

- 파일 이름은 파일에 클래스가 1개만 있는 경우에는 class의 이름과 동일해야 함
- 하나의 파일에 2개 이상의 클래스가 있는 경우 public 클래스가 있다면 public 클래스 이름으로 파일 이름이 만들어져야 하며 public 클래스가 없는 경우 아무 클래스 이름이나 파일명이 될 수 있지만 main Method를 소유한 클래스 이름으로 파일명을 만들어야 실행할 수 있음
- 하나의 파일에 public 클래스는 1개만 존재해야 함

프로그램 작성 및 실행

❖ java 파일의 구조

package 패키지이름;

import ...
import static ...

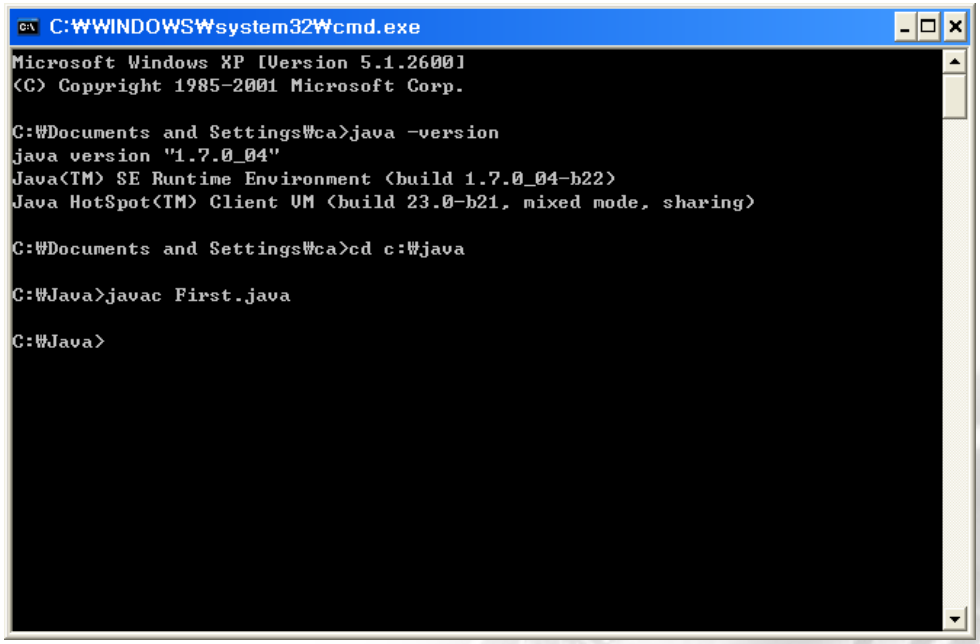
class ...

- ✓ package는 자신의 소속을 설정하는 것으로 1번만 나와야 하며 패키지가 없다면 생략 가능
- ✓ import는 클래스의 위치를 생략해서 사용하기 위한 구문으로 여러 번 나와도 되며 없는 경우도 있을 수 있음
- ✓ class는 클래스를 선언하기 위한 구문으로 하나의 파일에 반드시 1번 이상 나와야 하며 여러 번 사용할 수 있음
- ✓ 순서는 반드시 package => import => class

프로그램 작성 및 실행

- 컴파일 (javac)
  - ✓ console 프롬프트에서 파일이 저장된 디렉토리로 이동한 후 아래와 같이 입력하고 Enter

    javac First.java
  - ✓ 성공하면 아무런 메시지 없이 명령 프롬프트가 보임
  - ✓ 에러가 없다면 파일이 저장된 디렉토리에 가보면 First.class가 만들어짐
  - ✓ 윈도우에서 파일명은 대소문자 구별을 하지 않음

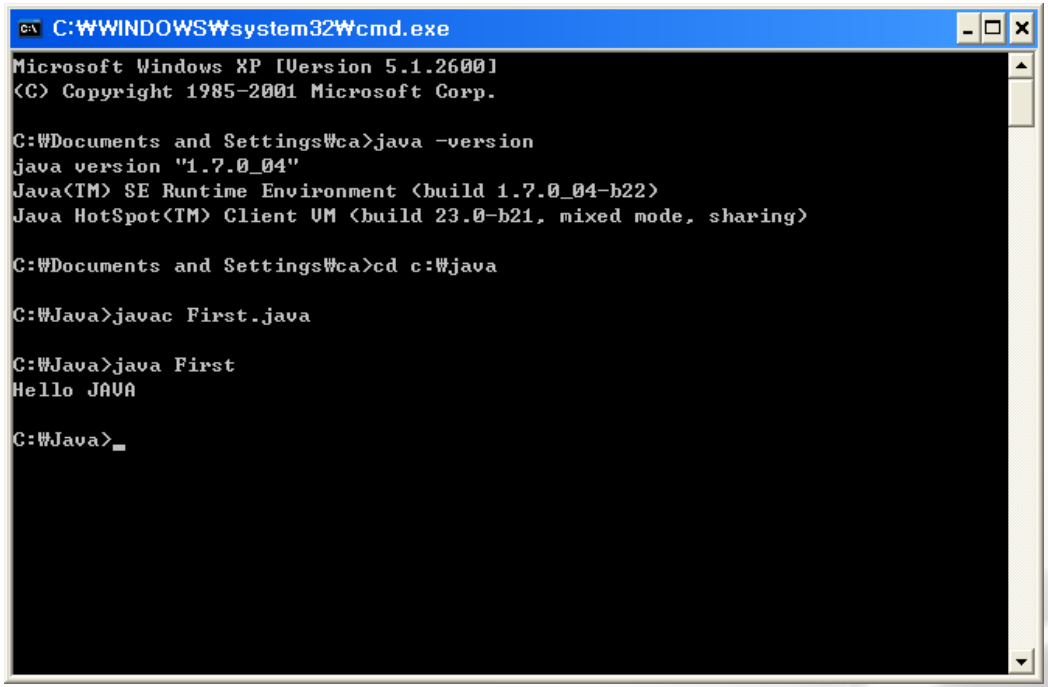


프로그램 작성 및 실행

❖ 실행

java First ==> 확장자는 입력하지 않음

✓ 실행 시 입력하는 클래스 이름은 대소문자 구별



프로그램 작성 및 실행

- 소스의 내용
  - ✓ 1행: 클래스 선언
  - ✓ 3행: main() Method로 String은 문자열 Data Type이고 args[]는 argument 배열을 나타냄
  - ✓ 5행: System.out.println은 출력문으로 괄호 안의 내용을 출력하고 커서를 다음 줄로 이동시키는 메서드

프로그램 작성 및 실행

- 소스 파일 작성 시 주의 사항
  - ✓ 소스 코드에서는 대소문자 구별
  - ✓ 한번에 실행되어야 하는 문장의 끝은 ;
  - ✓ 행의 개념은 없으므로 행의 분리는 의미가 없음
  - ✓ 클래스 및 메서드 등의 하나의 블럭은 { 로 시작해서 }로 끝

프로그램 작성 및 실행

- ❖ 실행 문장과 세미콜론(;)
  - ✓ 표현식(Expression)
    - 변수 선언, 값 저장, 메서드 호출에 해당하는 코드
    - 하나의 실행 문장 끝에는 세미콜론(;)을 붙여 한 번에 실행되는 문장의 끝 표시 – 이 표시가 있는 곳 까지 한 번에 읽어서 수행해 달라는 기호
    - 하나의 실행 문장을 표시하는 기호가 있는 언어는 코드를 작성할 때 한 줄에 여러 개의 실행 문장을 작성해도 되고 하나의 실행 문장을 여러 줄에 작성하는 것도 가능

```
int x = 1;                    //변수 x를 선언하고 1을 저장
int y = 2;                    //변수 y를 선언하고 2를 저장
int result = x + y;           //변수 result를 선언하고 변수 x와 y를 더한 값을 저장
System.out.println(result);   //콘솔에 출력하는 메소드 호출
```

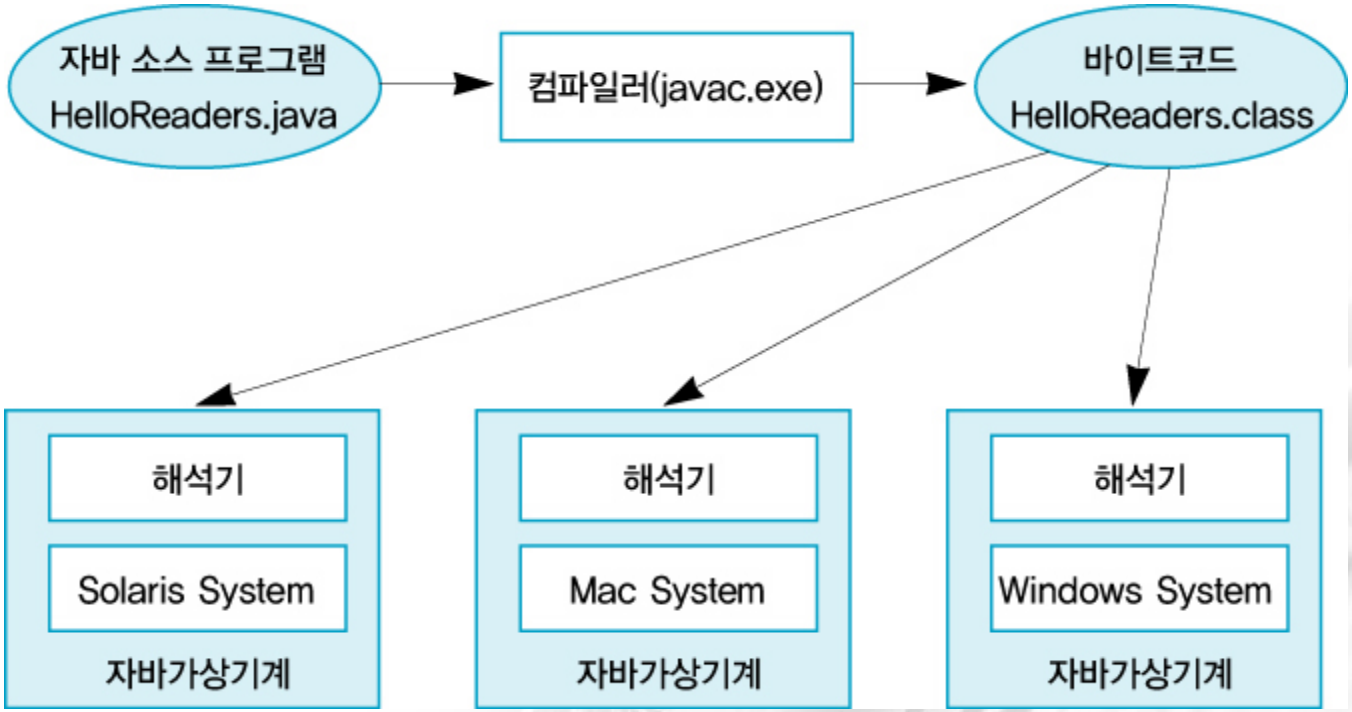
```
int x = 1; int y = 2;
int result =
                x + y;
```

프로그램 작성 및 실행

- 실행 문장과 세미콜론(;)
  - ✓ Statement(문장)
    - 한 번에 실행되는 명령어의 모음
    - ;으로 구분
  - ✓ block
    - 독립적으로 코드를 작성할 수 있는 영역으로 { }로 생성
    - 제어 구문, 예외처리 구문, method 선언, class 선언 등이 하나의 block
    - block 안에서 만든 이름들은 블록 안에서만 사용이 가능

프로그램 작성 및 실행

- 실행 과정
  - ✓ 자바 해석기를 포함한 시스템을 자바 가상 기계(Java Virtual Machine)라 하며 바이트 코드(확장자명 class)는 시스템의 운영체제 및 하드웨어에 관계없이 자바 해석기를 포함한 어떤 시스템에서도 실행 가능
  - ✓ 자바 응용 프로그램은 main() method의 첫번째 명령어부터 차례대로 실행



Java 프로그램

- Name
  - ✓ 예약어(Keyword)
    - 자바가 기능을 설정해 놓은 명령어

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

    - 리터럴 예약어(false, true, null)

Java 프로그램

- Name
  - ✓ 식별자(Identifier)
    - 코드 내에서 프로그래머가 기능을 지정하는 이름
    - 규칙
      - 예약어(char, int, class 등)는 식별자로 사용 불가능
      - 특수문자 중에서 _(언더스코어),$,? 는 사용 가능
      - Camel 표기법 – 이름을 만들 때 사용하는 방식 중 하나
        - 식별자의 첫 번째 글자는 반드시 문자로 하고 그 다음부터 숫자와 문자(한글, 영문) 혼합 가능
        - 클래스 이름은 대문자로 시작하며 2개 단어 이상의 조합일 때는 두번째 단어부터는 대문자로 시작(Dto, Member, MemberDto, MemberDao, MemberBean, MemberImpl..) – Pascal Case
        - 변수와 메서드 이름은 소문자로 작성하되 2개 이상의 단어 조합일 때는 두번째 단어부터는 대문자로 시작(insert, insertPlayer, insertMember, getName, setName…)
      - 상수(final)는 구분하기 위해서 모두 대문자를 사용하며 2개 이상의 단어 조합일 때는 _를 이용해서 구분(PI=3.14, COUNT=1, MAX_COUNT=1000…) – Snake 표기법
      - 변수나 상수는 명사 메서드 이름은 동사로 하는 것을 권장
      - 자주 사용하는 단어는 다른 의미를 부여하는 것을 권장

Java 프로그램

- Data
  - ✓ Literal: 소스 코드의 고정된 값
    - 논리형 리터럴 - true, false
    - 문자 리터럴 – 하나의 문자를 ' ' 안에 기재
    - 문자열 리터럴 – 0개 이상의 문자 집합으로 " " 안에 기재
    - 정수형 리터럴 – 10진 정수, 16진수(0xa4), 8진수(030), 천단위 구분 기호로 _ 이용(123_456)
    - 실수형 리터럴 – 1.7, 1.7f, 0.17e1, 3E-5
    - 레퍼런스 리터럴 - null이 있음(숫자로 0에 해당, 레퍼런스 변수는 현재 아무것도 가르키고 있지 않는다는 의미)
  - ✓ Variable - 변수
    - 데이터 (정수, 실수, 문자, 문자열)를 담아 놓을 때 사용하는 메모리 상의 이름
    - 생성한 후 사용해야 함
    - 타입과 이름으로 생성
    - 타입: 변수를 만들 때 저장할 수 있는 데이터의 종류를 의미
  - ✓ Constant – 상수
    - 변경할 수 없는 데이터
    - 변수를 만들 때 앞에 final을 추가

Java 프로그램

- 데이터 이외의 구성 요소
  - ✓ 연산자(Operator): 산술, 논리 연산 등 을 수행하게 하는 부호 또는 명령어
  - ✓ 제어문(Control Statement): 프로그램의 실행 흐름을 변경하는 명령어
  - ✓ 배열과 클래스: 여러 개의 데이터 모임
    - 배열: 동일한 타입의 데이터 모임
    - 클래스: 사용자 정의 데이터 모임
      - 클래스를 이용해서 메모리를 할당한 공간
      - 속성: 저장 공간
      - 메서드: 수행할 작업
      - 생성자: 인스턴스를 생성하기 위한 특별한 메서드
  - ✓ 인터페이스: 메서드의 이름과 상수 만을 가진 개체
  - ✓ enum: 나열형 상수
  - ✓ Annotation: @로 시작하는 문자열로 소스 코드 안의 요소를 설명하거나 특정 기능을 갖는 인스턴스를 자동 생성해주는 기능
  - ✓ Pakage: 유사한 역할을 수행하는 클래스, 인터페이스, enum, annotation 의 집합

Java 프로그램

- 데이터 이외의 구성 요소
  - ✓ 주석(comment): 보충 설명
    - 코드의 가독성을 높이기 위해서 작성
    - 작성 방법
      - // 한 줄 주석
      - /* */ 여러 줄 주석을 작성할 때 사용
      - /** ~ */ 사용자의 Document를 만들 때 입력되는 주석

Java 프로그램

- 콘솔 출력
  - ✓ System.out.print는 내용을 모아서 콘솔에 출력한 후 줄 바꿈을 하지 않음
  - ✓ System.out.println은 명령마다 바로 실행하고 매 번 줄 바꿈을 함
  - ✓ 메서드의 매개변수(argument - 메서드를 호출할 때 괄호 안에 작성하는 데이터)
    - "메시지": " "안에 출력할 문자열을 입력
    - 데이터: 데이터를 출력할 때는 기본형의 경우는 그 데이터를 그렇지 않은 경우는 toString()의 리턴 값을 출력
    - 문자열 과 다른 데이터를 +로 결합해서 출력 가능
  - ✓ 포맷을 설정해서 출력할 수 있는 System.out.printf 사용 가능

Java 프로그램

❖ 콘솔 출력

✓ Source Code 작성

```
public class Second {
    public static void main(String args[]) {
            System.out.println("안녕하세요");
            System.out.println("박문석입니다.");
            System.out.println("오늘은 " + 1 + "일차" + " 수업입니다");
            System.out.println("3 + 4 = " + 3 + 4);
            System.out.println("3 + 4 = " + (3 + 4));
            System.out.println("3 * 4 = " + 3 * 4);
    }
}
```

Java 프로그램

- 콘솔 출력
  - ✓ 결과

```
안녕하세요
박문석입니다.
오늘은 1일차 수업입니다
3 + 4 = 34
3 + 4 = 7
3 * 4 = 12
```

Java 프로그램

- ❖ 매개변수를 활용한 실행
  - ✓ public static void main(String args[])
    - 자바 응용 프로그램이 실행을 하기 위해서는 main()이 반드시 하나 존재해야 함
    - 괄호 안의 내용은 argument(매개변수, 인자)로 실행 할 때 주는 옵션

      java 실행클래스명 argument나열
    - 매개변수는 공백으로 구분해서 여러 개 대입이 가능
    - args[0]부터 차례대로 문자열로 대입

Java 프로그램

- 매개변수를 활용한 실행
  - ✓ 소스 코드 작성

```
public class Third {
    public static void main(String args[]) {
            System.out.println(args[0]);
            System.out.println(args[1]);
            System.out.println(args[2]);
    }
}
```

Java 프로그램

❖ 매개변수를 활용한 실행

✓ 결과

위와 같이 작성하고 컴파일(javac Third.java) 한 후

java Third 우리나라 대한민국 Korea

라고 console 창에 입력하면

우리나라

대한민국

Korea

라고 화면에 출력

Java 프로그램

- Java 명령어
  - ✓ javac.exe : 컴파일
  - ✓ java.exe: 실행
  - ✓ javap.exe : 역어셈블(컴파일 된 클래스 파일을 원래의 소스로 변환)

    javap 클래스이름 > 소스파일명
  - ✓ javadoc.exe : java api 문서와 같은 형식의 문서를 생성(public class 만 가능)

    javadoc 소스파일명

    javadoc –private –encoding UTF-8 –charset UTF-8 –docencoding UTF-8 소스파일명
  - ✓ jar.exe : 압축을 하거나 압축을 해제할 때 사용

    압축을 해제하고자 하는 경우: jar cvf [jar 파일 경로]

    압축을 하고자 하는 경우: jar xvf [대상디렉토리 | class 파일들]
  - ✓ jps: 현재 실행 중인 자바 프로세스 이름과 프로세스 ID 출력

Java 프로그램

- 디어셈블
  - ✓ 클래스 파일에 들어 있는 바이트 코드를 텍스트로 볼 수 있게 변환하는 작업
  - ✓ JDK의 javap.exe 이용(-c 옵션을 이용하면 역어셈블된 코드로 저장)

    javap -c First > First.bc

```
Compiled from "First.java"
public class First {
  public First();
    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     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #13                 // String Hello JAVA
       5: invokevirtual #15                 // Method
java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

Java 프로그램

- 도큐먼트 생성
  - ✓ 소스 코드 작성 – Third.java 파일 수정

```
public class Third {
    public static void main(String args[]) {
            /**
            프로그램을 만든 사람: 박문석
            프로그램을 만든 장소: 회사
            */
            System.out.println(args[0]);
            System.out.println(args[1]);
            System.out.println(args[2]);
    }
}
```

  - ✓ 도큐먼트 생성
    - 콘솔 창에서 javadoc Third.java 라고 입력
    - 컴파일하고 파일이 저장된 디렉토리에 가서 확인

IDE

- Java IDE
  - ✓ Eclipse: IBM, Borland 등의 업체를 중심으로 오픈 소스로 개발되고 있는 통합 개발 도구
  - ✓ IntelliJ
    - IntelliJ IDEA는 JetBrains사에서 제작한 상용 자바 통합 개발 환경
    - Eclipse 재단 의 Eclipse와 썬 마이크로시스템즈의 넷빈즈로 대표되는 무료 자바 통합개발환경에서 볼랜드(/코드기어)의 제이빌더(JBuilder)와 함께 얼마 안 되는 상용 개발 도구 가운데 하나
  - ✓ NetBeans
    - 넷빈즈(NetBeans)는 자바 기반의 사용자 프로그램을 개발하기 위한 플랫폼 또는 자바(Java), 자바스크립트(Javascript), PHP, 그루비(Groovy), C, C++ 등을 개발하기 위한 통합 개발 환경(Integrated Development Environment, IDE)
    - 넷빈즈는 자바로 개발되어 있어 자바 가상 머신(JVM)이 갖추어진 환경이면 사용 가능하며 윈도, 리눅스, 솔라리스(x86/x64, sparc), 맥 OS X의 경우에는 편리한 사용을 위하여 설치 파일 제공

IDE

- Eclipse
  - ✓ 사용하기 편리하고 프로그램 개발에 필요한 각종 기능을 플러그 인 형태로 제공
  - ✓ 다운로드 사이트: http://www.eclipse.org/downloads/
  - ✓ 다운로드 받은 파일을 받아서 설치

IDE

- Eclipse
  - ✓ Eclipse 설치
    - Eclipse IDE for Java Developers 는 Java Application 만 개발
    - Eclipse IDE for Enterprise Java Developers 는 Java Web Application 까지 개발

IDE

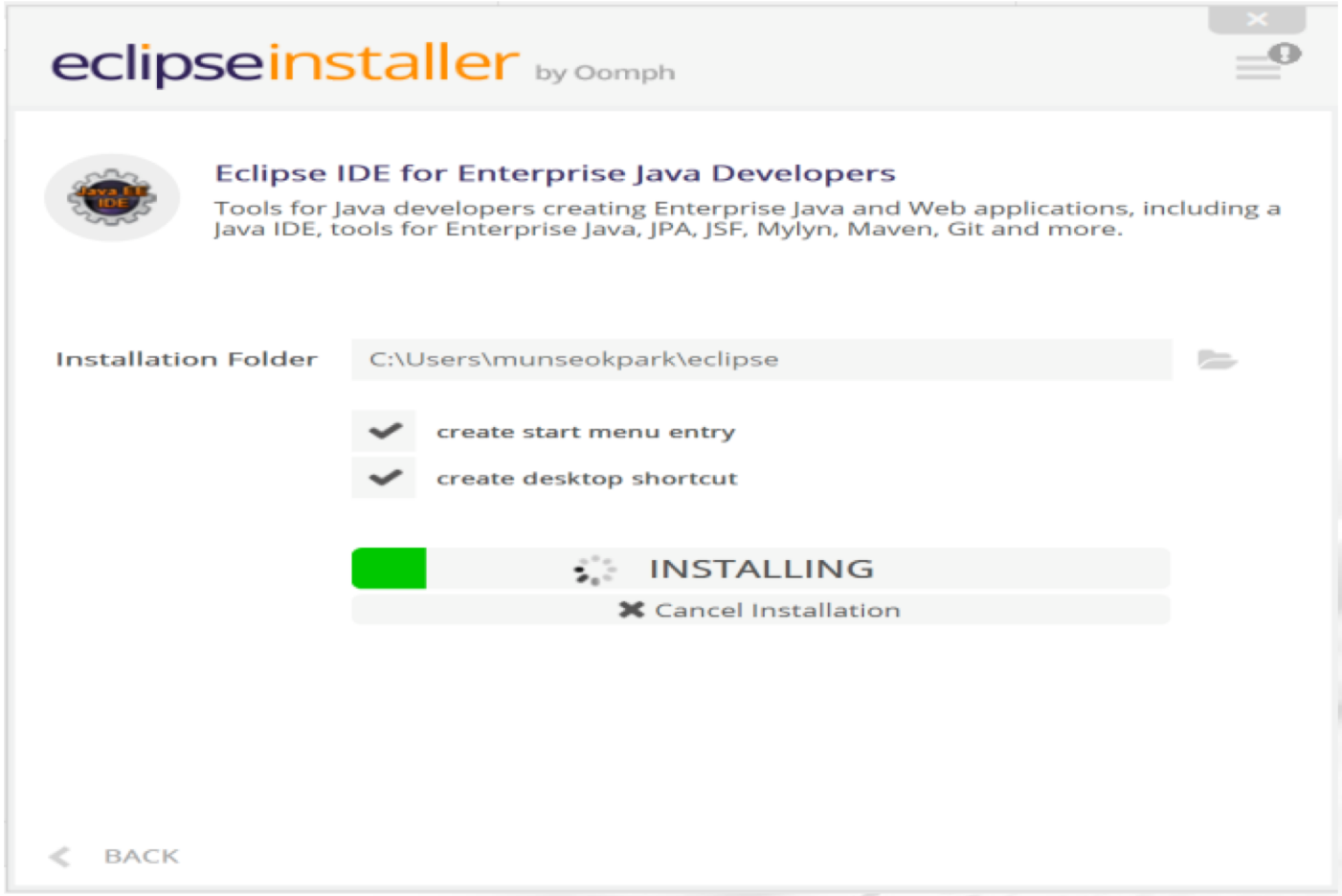
- Eclipse
  - ✓ Eclipse 설치
    - 설치 위치를 설정하고 INSTALL 클릭

IDE

- Eclipse
  - ✓ Eclipse 설치
    - 사용권 동의

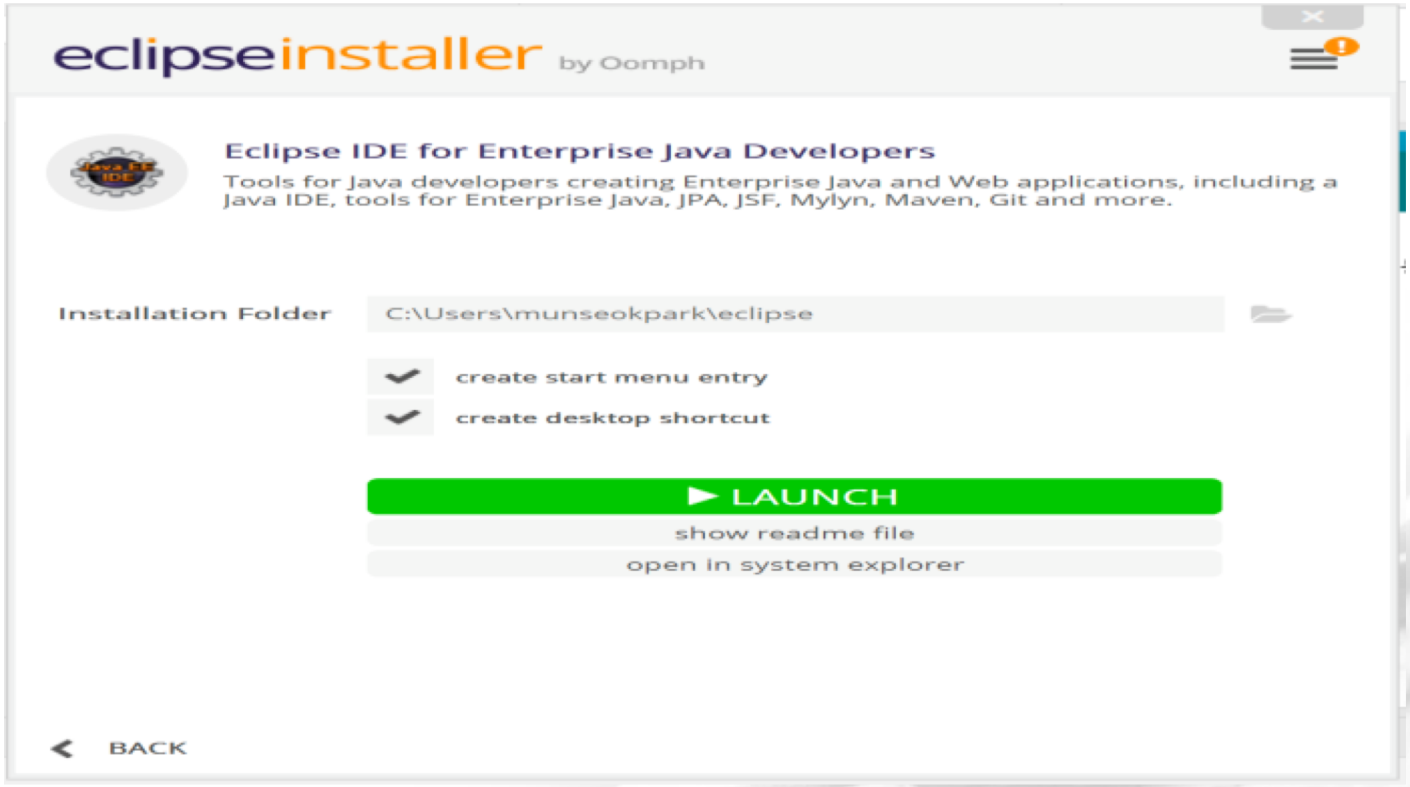
IDE

- Eclipse
  - ✓ Eclipse 설치



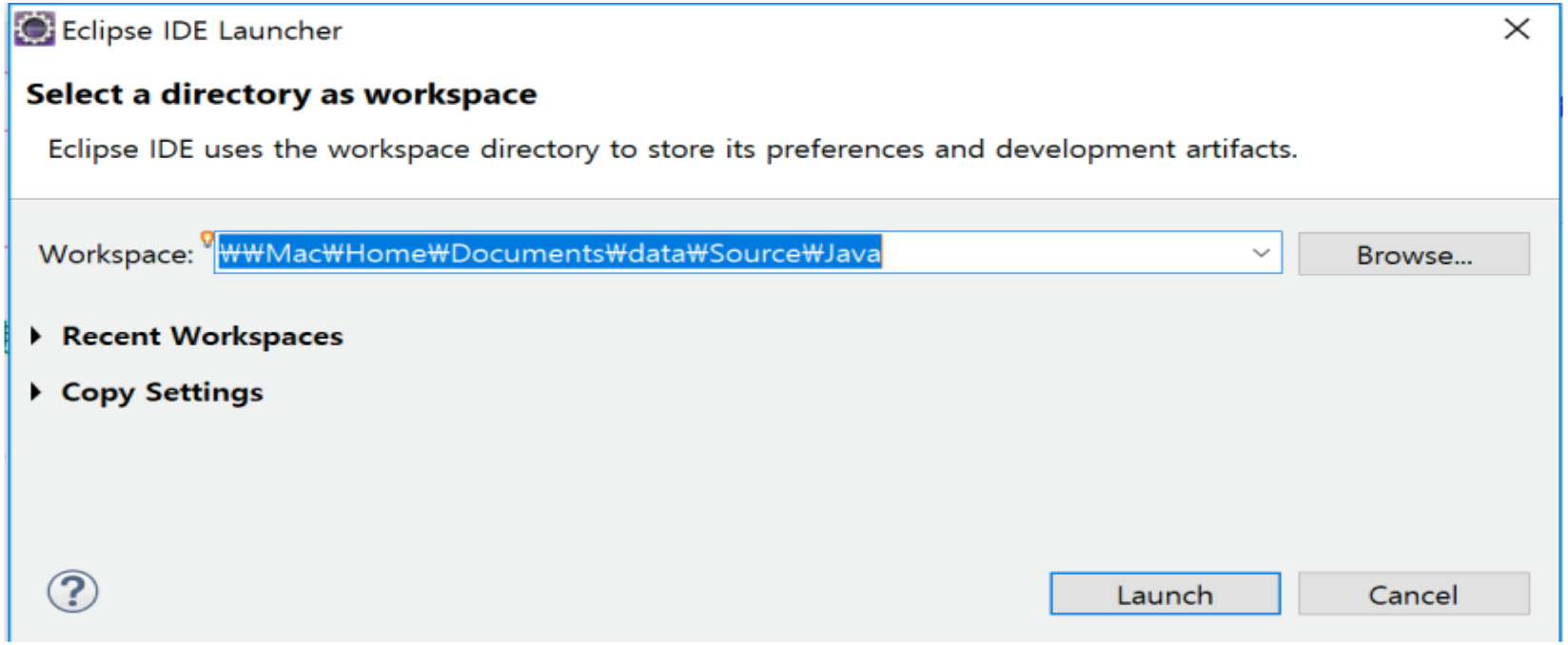
IDE

- Eclipse
  - ✓ Eclipse 설치



IDE

- Eclipse
  - ✓ Eclipse 실행
    - 단축 아이콘이나 Eclipse 디렉토리에서 실행 파일을 클릭
    - 워크스페이스 설정



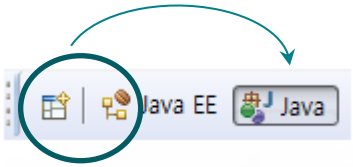
IDE

- Eclipse
  - ✓ 워크스페이스(Workspace)
    - Eclipse에서 생성한 프로젝트가 기본적으로 저장되는 디렉토리
    - 최초 실행 시 워크스페이스 런처(Workspace Launcher)에서 설정
    - .metadata 디렉토리
      - 자동 생성되며 Eclipse 실행 시 필요한 메타데이터 저장
      - 이 디렉토리를 삭제하고 Eclipse 실행 - 초기 상태로 다시 실행
      - .이 붙으면 리눅스나 Mac에서는 숨김 디렉토리 – 숨김 디렉토리 보기 옵션을 실행해야만 보임

IDE

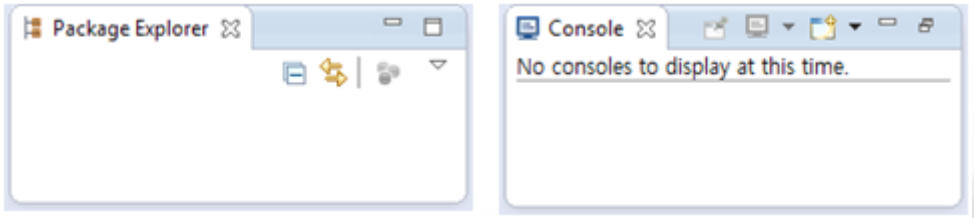
- ❖ Eclipse
  - ✓ Perspective
    - ● 개발 프로젝트 종류별로 유용한 View들을 묶어놓은 것

    - ● Eclipse IDE for Java EE Developers
      - ◆ 기본적으로 Java EE Perspective
      - ◆ Java Application 작업을 할 때는 Java Perspective로 변경해 사용



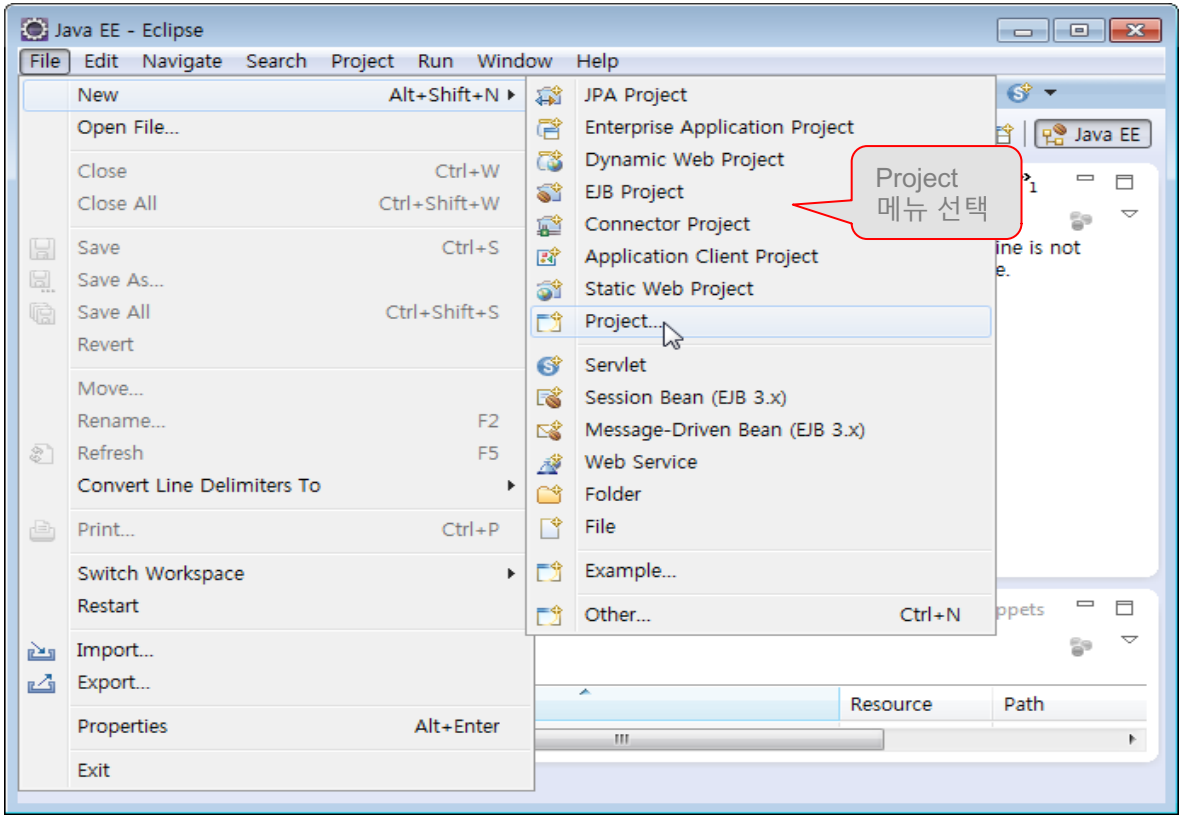
IDE

- Eclipse
  - ✓ 뷰(View)
    - 퍼스펙티브를 구성하는 작은 창으로 여러 가지 목적에 맞게 내용 보여줌
    - 자유롭게 제거 하거나 추가 가능
    - 자주 사용하는 뷰 들
      - Package Explorer
      - Project Explorer
      - Console



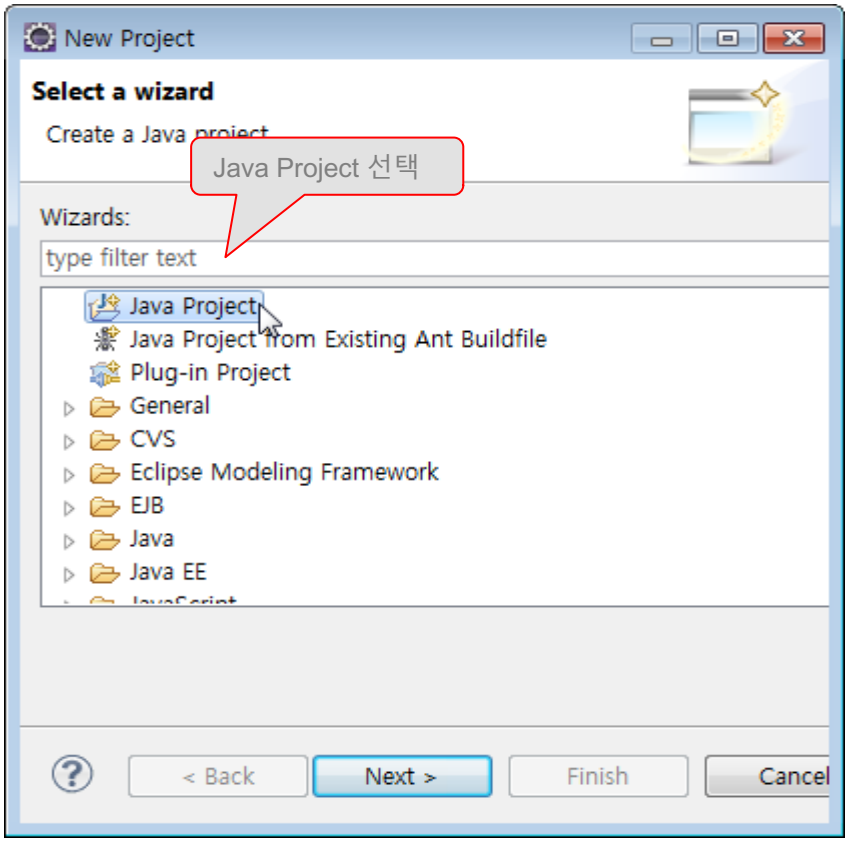
IDE

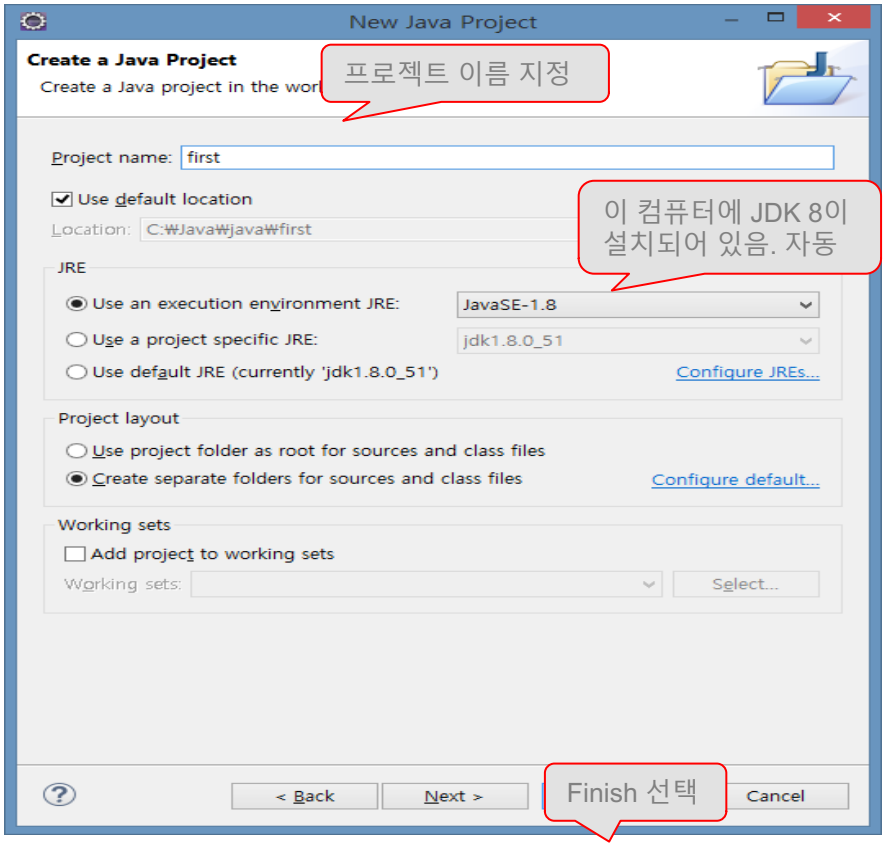
- Eclipse
  - ✓ 프로젝트 생성
    - [File] – [New] – [Project]



IDE

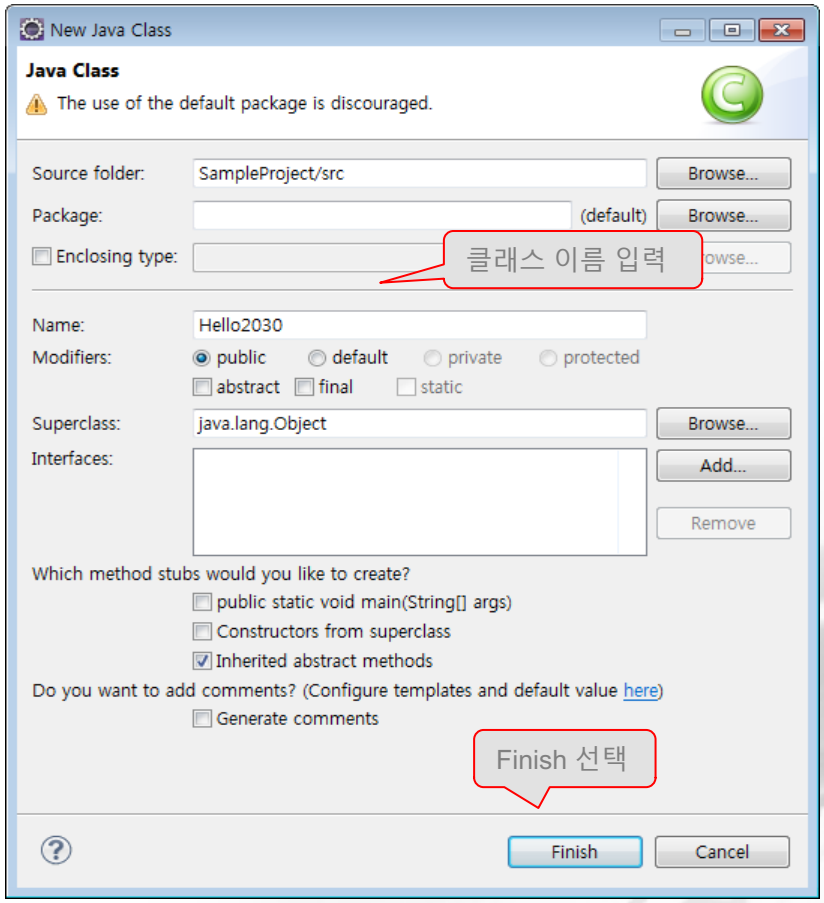
- Eclipse
  - ✓ 프로젝트 생성
    - Java Project를 선택
    - Project name 입력





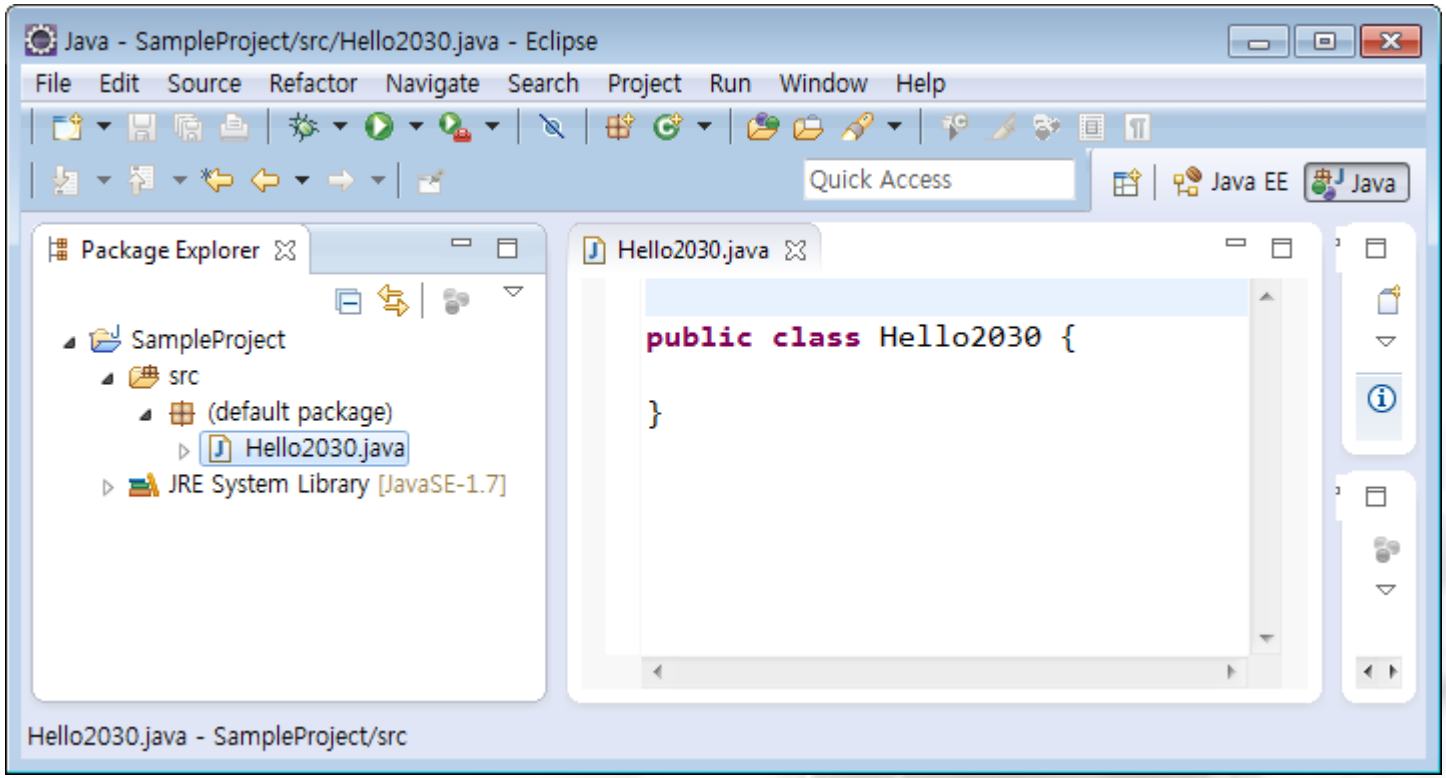
IDE

- Eclipse
  - ✓ 프로젝트 생성
    - 클래스 생성: File->New->Class 메뉴 선택



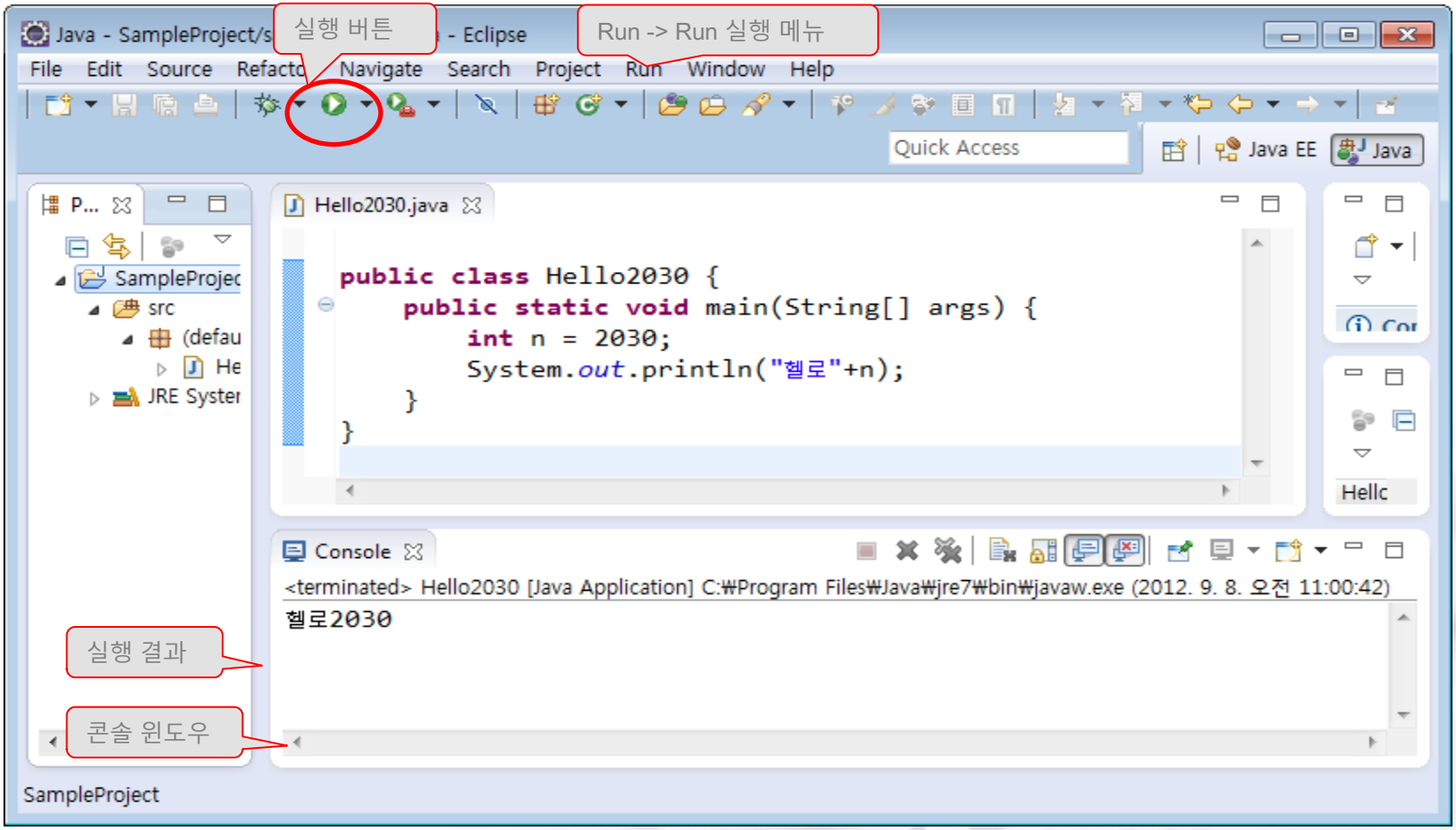
IDE

- Eclipse
  - ✓ 프로젝트 생성



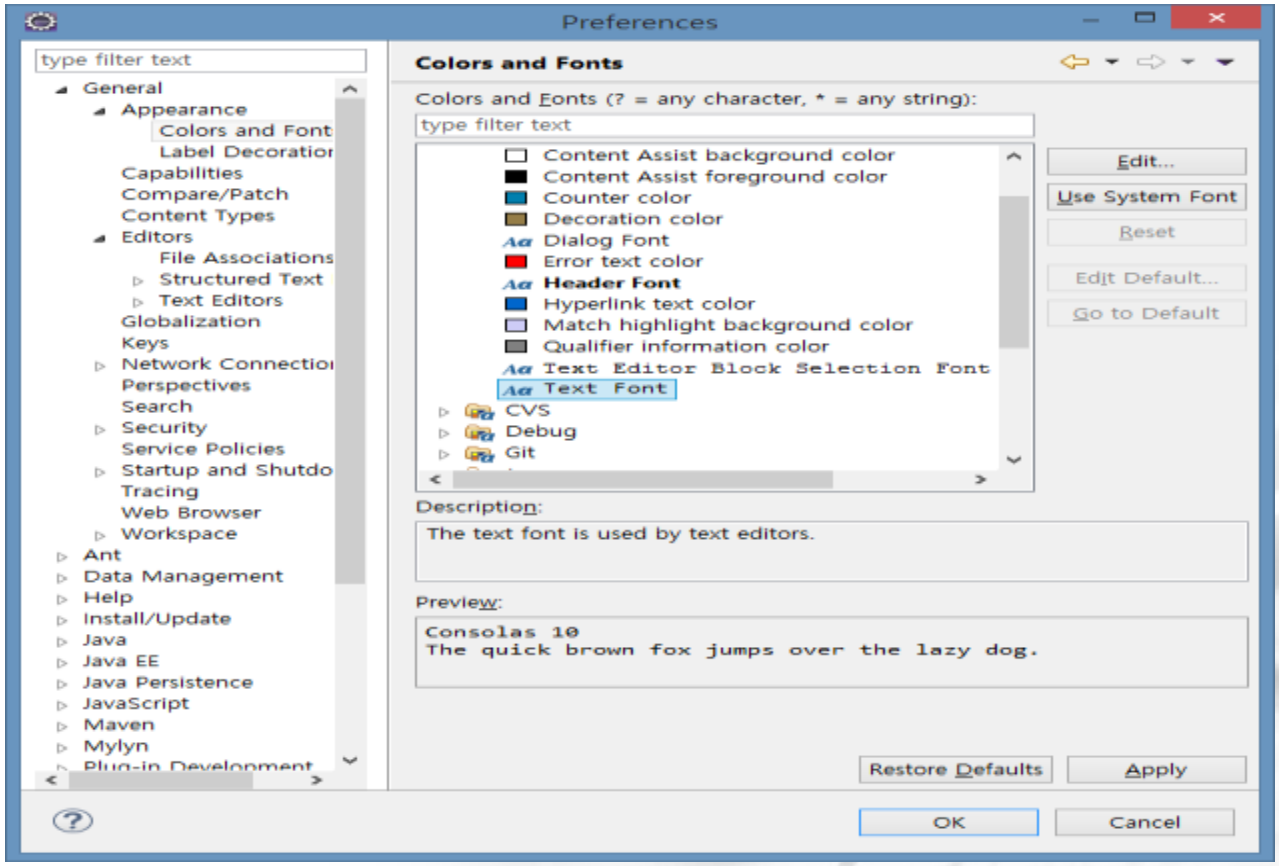
IDE

- Eclipse
  - ✓ 프로젝트 생성



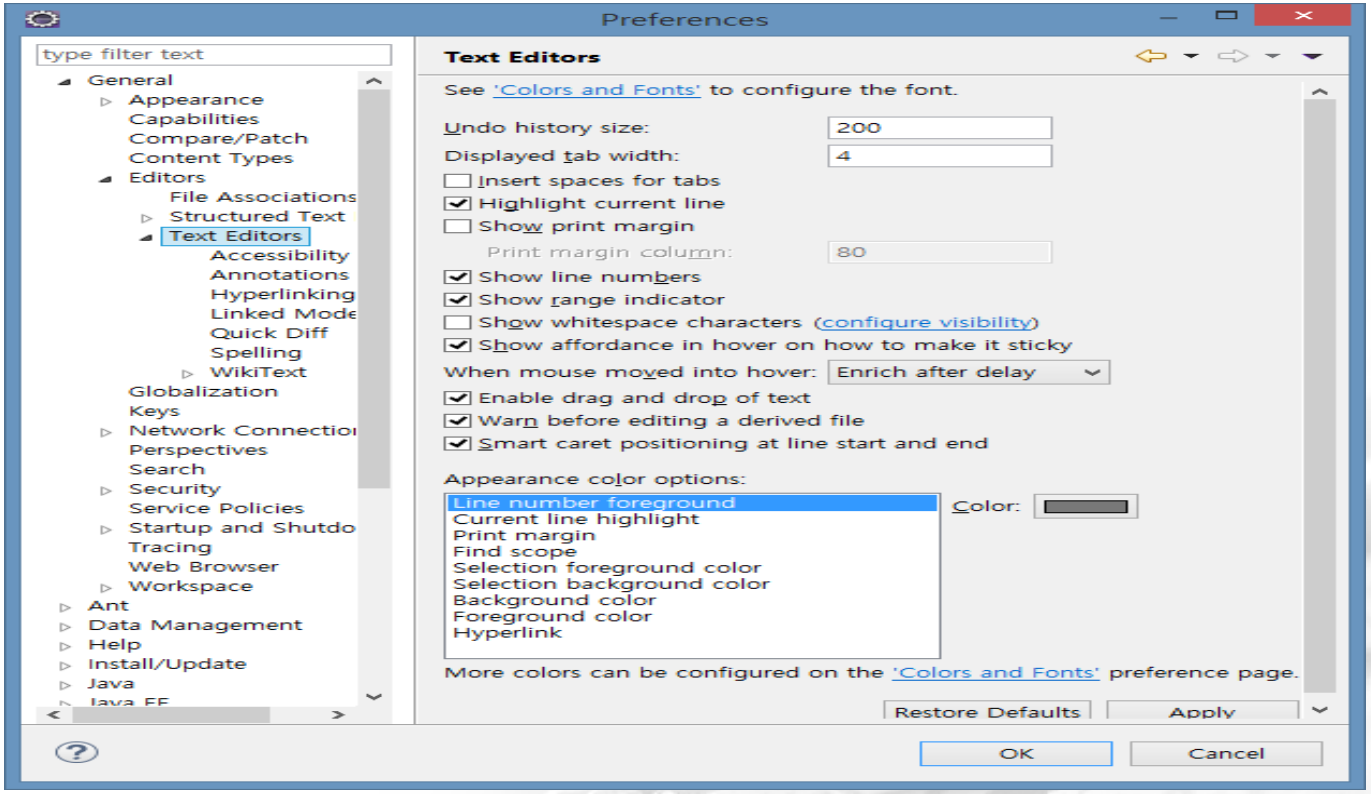
IDE

- Eclipse
  - ✓ Eclipse 설정: [Window] – [Preferences] - Workspace 별로 설정
    - 에디터의 텍스트 폰트 설정: [General] – [Appearance] – [Colors and Fonts]



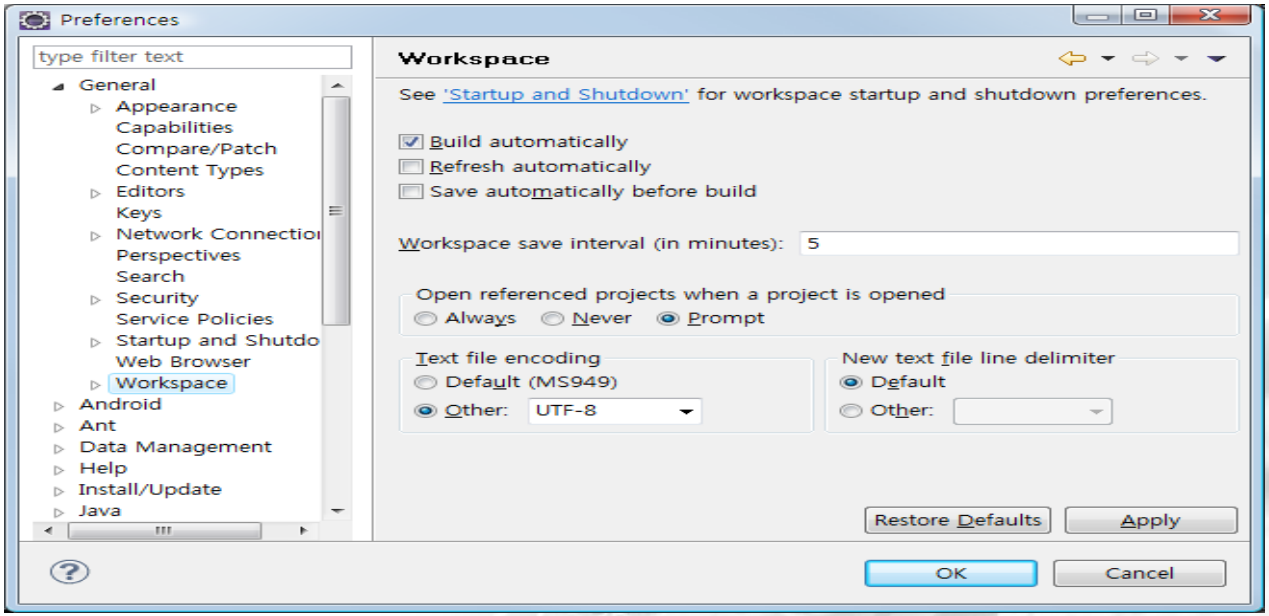
IDE

- Eclipse
  - ✓ Eclipse 설정: [Window] – [Preferences] - Workspace 별로 설정
    - 라인 번호 출력



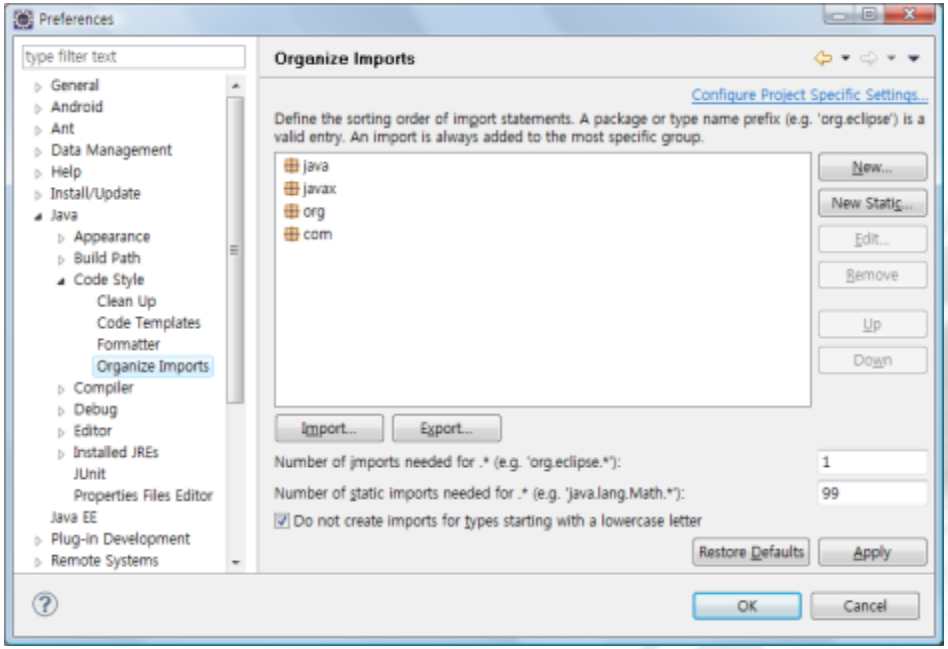
IDE

- Eclipse
  - ✓ Eclipse 설정: [Window] – [Preferences] - Workspace 별로 설정
    - 문자 인코딩 설정
      - WebApplication의 경우에는 워크스페이스의 Text file encoding 의 문자 셋을 UTF-8로 설정
      - 인코딩 설정이 다르면 영문 과 숫자를 제외한 문자가 제대로 출력되지 않음



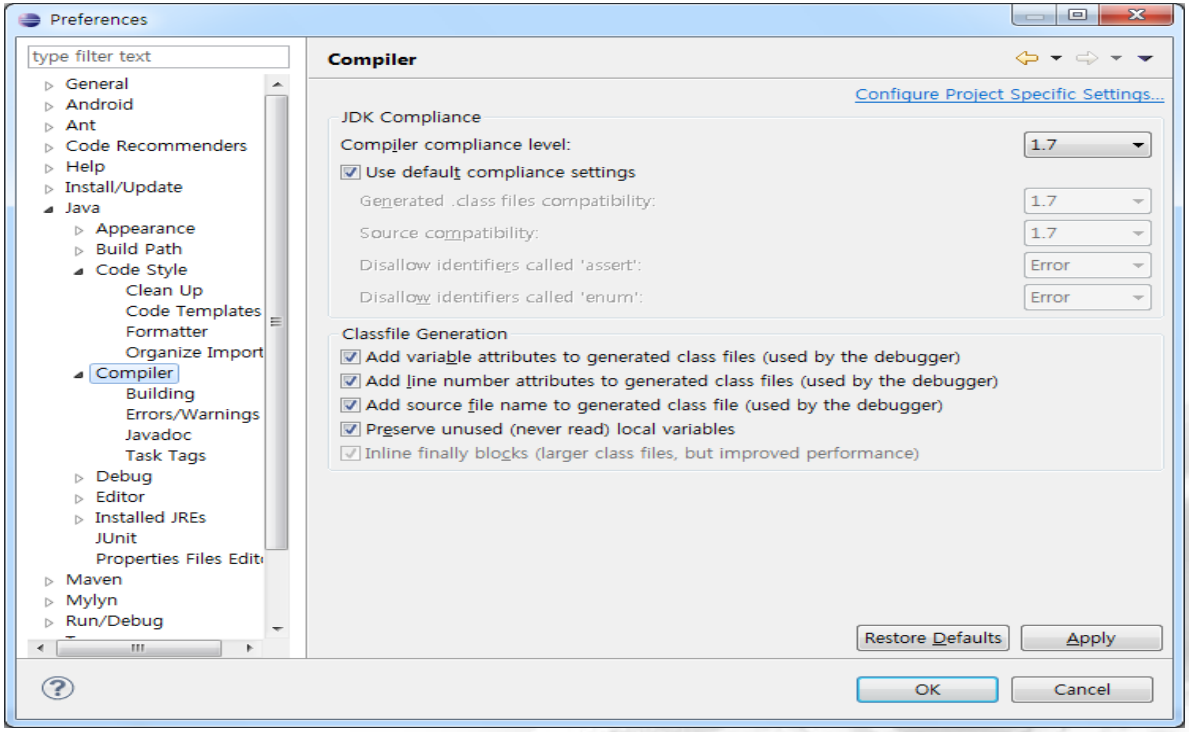
IDE

- Eclipse
  - ✓ Eclipse 설정: [Window] – [Preferences] - Workspace 별로 설정
    - 옵션 설정: [Java-Code Style-Organize Imports] 페이지의 Number of imports needed for .* 옵션을 1로 변경
      - Eclipse는 import문을 자동으로 정리해 주는 편리한 기능을 제공하는데 매 클래스마다 import문이 작성되면 소스가 너무 길어짐
      - 이 값을 1로 바꿔 놓으면 패키지의 모든 클래스를 한꺼번에 import 해 주므로 실습할 때 편리하고 결과 소스가 짧아지는 효과



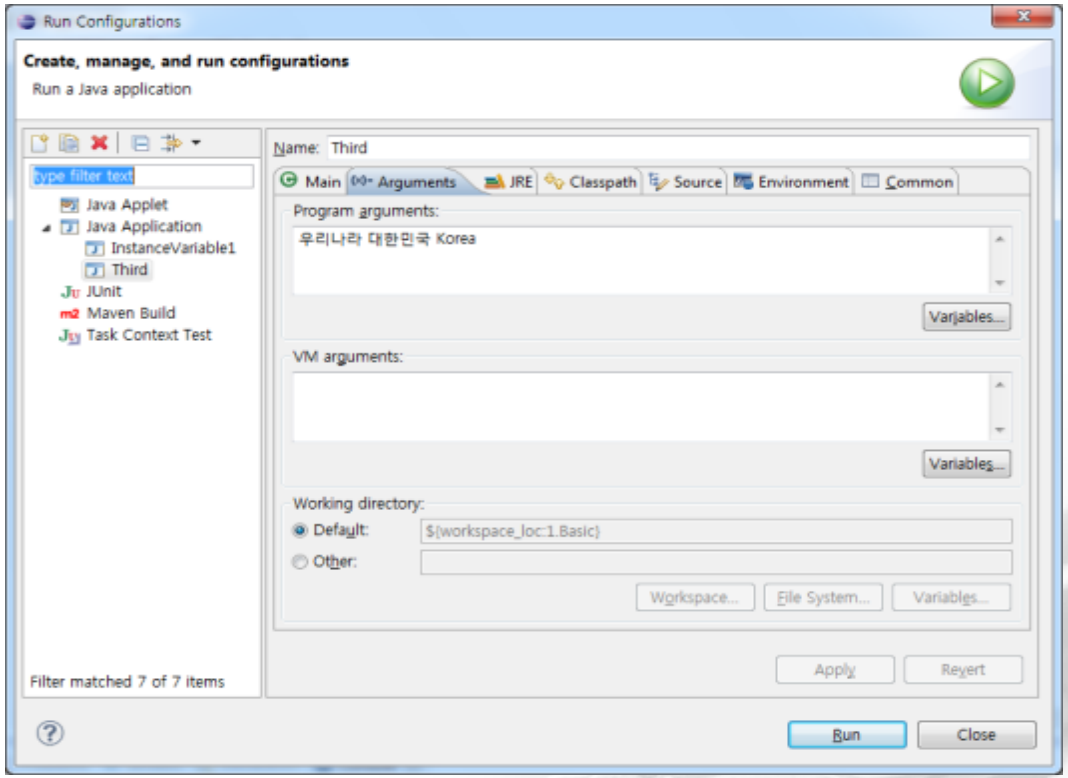
IDE

- Eclipse
  - ✓ Eclipse 설정: [Window] – [Preferences] - Workspace 별로 설정
    - 옵션 설정 – 컴파일 버전 설정



IDE

- Eclipse
  - ✓ Eclipse에서 매개변수를 입력해서 실행
    - [Run] – [Run Configuration] – Arguments 탭에서 입력



IDE

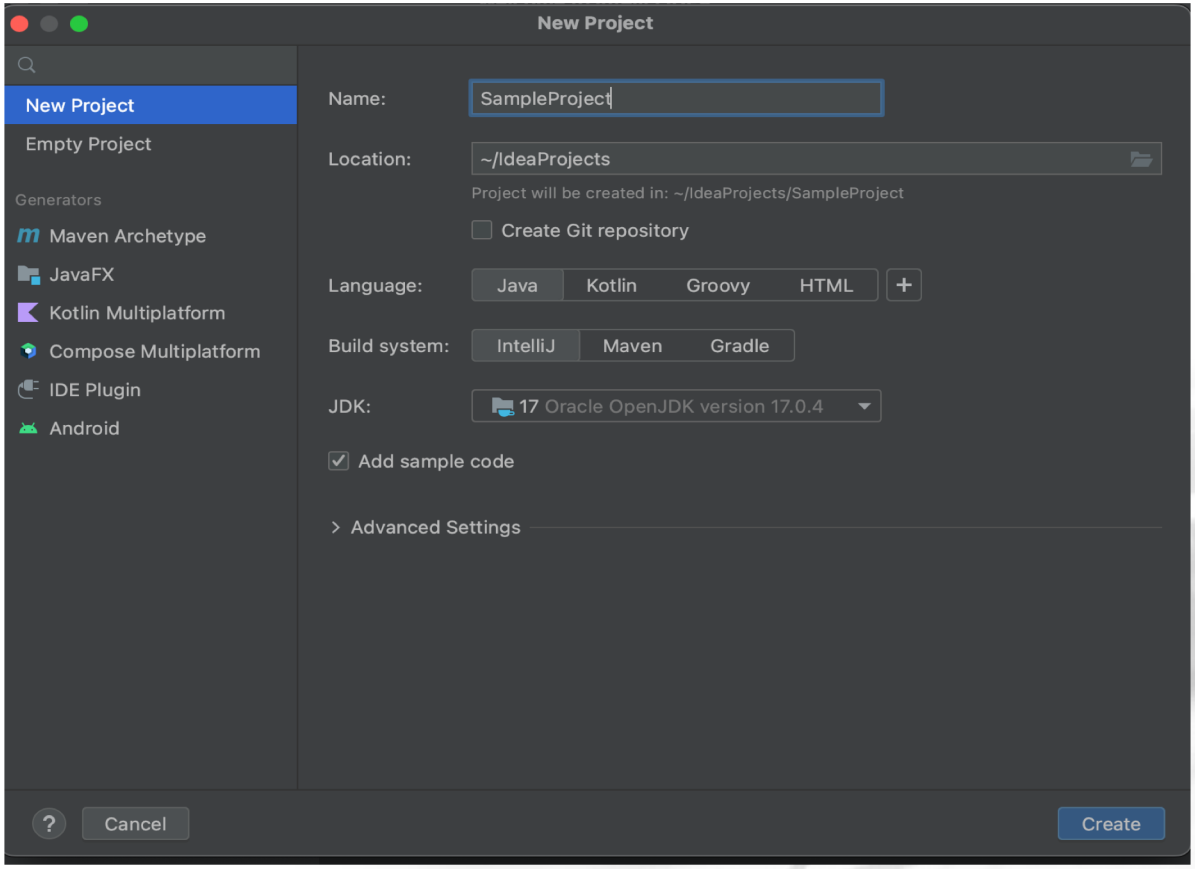
- Eclipse
  - ✓ 자주 사용하는 Eclipse 단축키
    - /* */ : 주석처리 -> 블럭 지정한 후에 ctrl + alt + /
    - 주석 해제는 ctrl + alt + ₩
    - // : 한 줄 주석처리 -> ctrl+/
    - 자동 완성 기능 : ctrl + spacebar
    - Error Fix: 에러가 발생한 부분에서 ctrl+1 ->에러가 발생한 곳에 대해 해결 방법 제시
    - Undo/Redo : ctrl + Z / ctrl + Y
    - System.out.println(); 생성 : sysout 입력하고 ctrl + spacebar
    - 들여쓰기 자동 수정 : ctrl + I -> 커서가 있는 줄의 들여쓰기를 자동으로 맞춤
    - 블록을 지정하고 실행 시 블록 내에서 자동 들여쓰기
    - shift + alt + s r : getter/setter 자동 생성
    - F11 : 디버깅 시작
    - F4 : 상속 구조 클래스 보기(Method 등)
    - alt + shift + r : 변수 및 Method 이름 변경
    - ctrl + m : 에디터 화면 넓게/좁게
    - ctrl + shift + O : 자동 import 처리 (사용하지 않는 Class는 삭제)

IDE

- IntelliJ
  - ✓ 설치
    - 다운로드: https://www.jetbrains.com/ko-kr/idea/download/

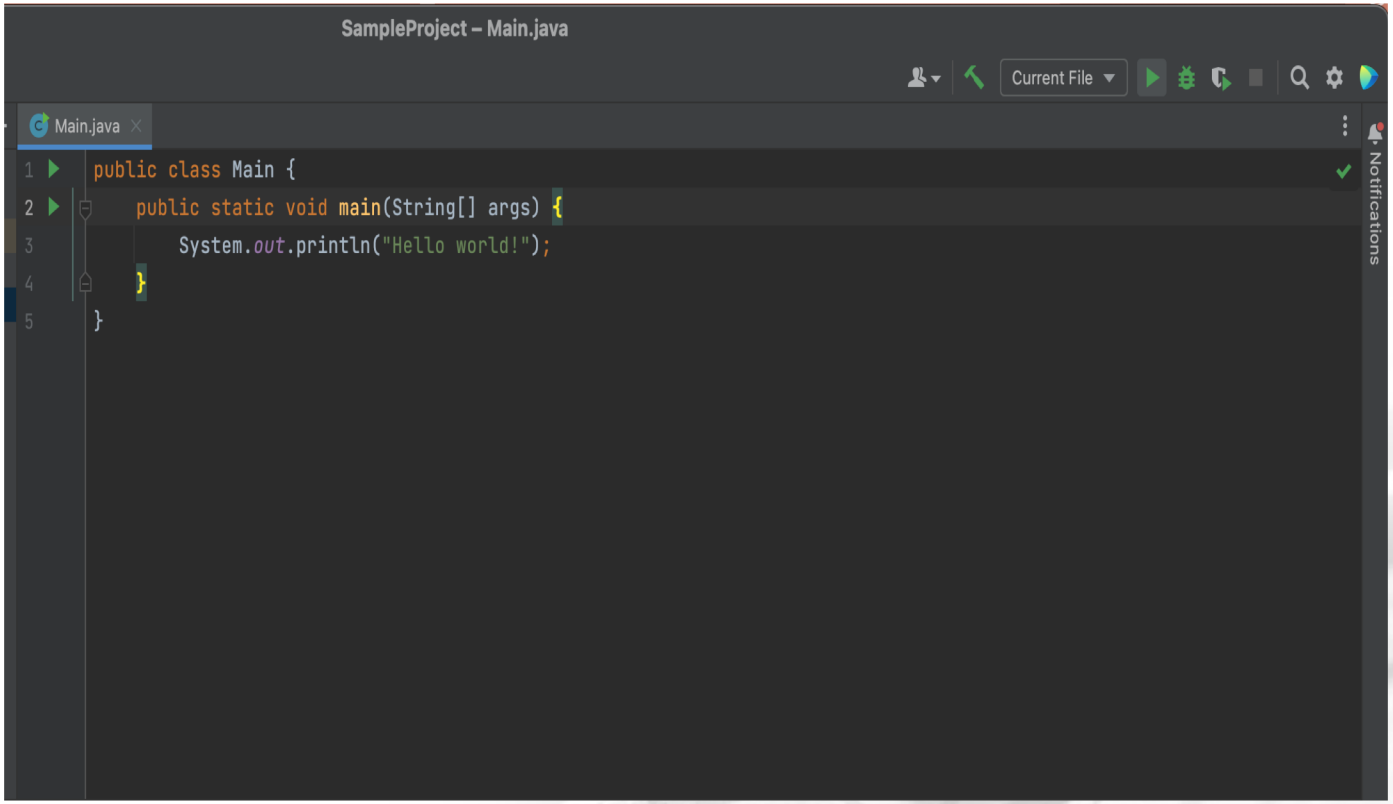
IDE

- IntelliJ
  - ✓ 프로젝트 생성 및 실행
    - [File] – [New] – [Project]



IDE

- IntelliJ
  - ✓ 프로젝트 생성 및 실행
    - 삼각형 버튼을 눌러서 실행



연습문제

- IDE란 무엇이고 현재 자신이 Java Programming을 위해 사용하는 IDE는 ?
- JRE 와 Java API는?
- 자신의 이름과 나이를 출력하는 프로그램을 작성