

Chapter

01

자바 시작하기



01-1. 프로그래밍 언어와 자바

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- 자바 소개
- 자바 개발 도구 설치
- 환경 변수 설정
- 확인문제



시작하기 전에

핵심 키워드

기계어

프로그래밍 언어

소스 파일

컴파일

JDK

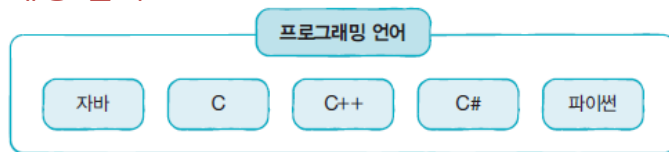
환경 변수

컴퓨터에서 실행하는 프로그램(program)은 특정 목적을 수행하도록 프로그래밍 언어로 작성된 소스를 기계어로 컴파일한 것입니다. 이번 절에서는 프로그래밍 언어의 역할에 대해 알아보고, 자바 언어로 프로그램을 개발할 수 있는 환경을 만들어보겠습니다.

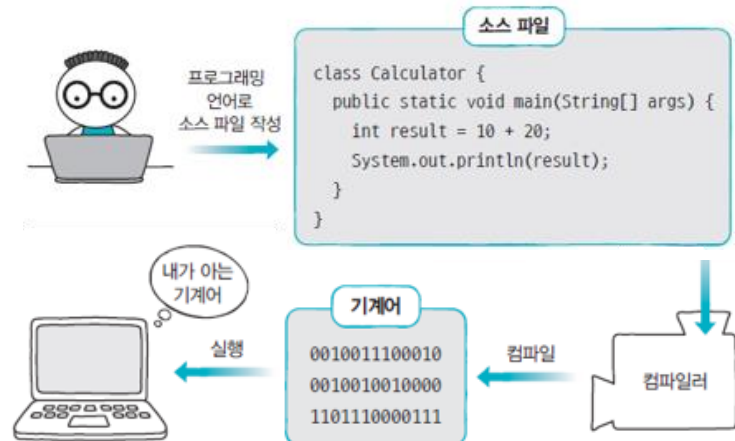
❖ 기계어 (machine language)

- 0과 1로 이루어진 코드를 사용
- 사람과 기계어 사이 다리 역할을 하는 프로그래밍 언어 필요
 - **소스** (source) **파일** : 프로그래밍 언어로 작성한 파일
 - **컴파일** (compile) : 소스 파일을 기계어 파일로 번역

❖ 프로그래밍 언어



- **자바(Java)**: 임베디드, 웹, 안드로이드 소프트웨어(SW) 개발 분야
- **C / C++**: 하드웨어(HW) 장치 제어 및 임베디드 SW 분야
- **C++ / C#**: 데스크탑 앱 또는 ASP.NET 기반 웹 SW 분야
- **파이썬(Python)**: 빅데이터 분석 및 머신러닝 SW 분야



자바 소개

❖ 자바 (Java)

- 1995년 마이크로시스템즈(Sun Microsystems)에서 발표
- 현재 웹사이트 및 다양한 애플리케이션 개발의 핵심 언어



- 오라클 (<http://www.oracle.com>) 라이선스
 - 자바 개발 도구의 배포
- 특징
 - 모든 운영체제에서 실행 가능
 - 객체 지향 프로그래밍 (OOP : Object-Oriented Programming)
 - 메모리 자동 정리
 - 풍부한 무료 라이브러리



자바 개발 도구 설치

❖ 자바 개발 도구 (JDK : Java Development Kit)

■ JDK 역할

- 자바 언어로 소프트웨어를 개발할 때 필요한 환경 및 도구를 제공하는 역할

■ JDK 종류

- Open JDK: <https://openjdk.java.net>
 - 개발, 학습용 및 상업용 모두 무료로 사용
- Oracle JDK: <https://www.oracle.com>
 - 개발, 학습용은 무료로 사용
 - 상업용 목적으로 사용할 경우 연간 사용료 지불
 - 장기 기술지원(LTS: Long Term Support) 및 업데이트 제공으로 안정적

■ 학습용 JDK 선택

- 학습용은 모두 무료이므로 안정적인 **Oracle JDK**를 사용하는 것이 좋음



자바 개발 도구 설치

❖ JDK 버전 체계

Java SE 11. 0. 2 (LTS)

주 버전 개선 버전 업데이트 버전 장기 지원 서비스 버전

JDK 버전	설명
Java SE 12	주 버전이 12
Java SE 11.0.2 (LTS)	주 버전이 11이고, 수정이 2번 되었음 장기 지원 서비스를 제공받을 수 있는 버전
Java SE 8u202	주 버전이 8이고, 수정이 202번 되었음

❖ 설치하기: <https://www.oracle.com>

- [Downloads] 클릭
- [Developer Downloads] > Java
- [Java (JDK) for Developers]



자바 개발 도구 설치

■ 설치할 PC 사양에 맞는 설치 파일을 다운로드

- 64비트 윈도우 운영체제의 경우:

- **JDK 11** 이후 버전:

jdk-xx.x.x_windows-x64_bin.exe

- **JDK 8** 버전:

jdk-8uxxx-windows-x64.exe

- 32비트 윈도우 운영체제의 경우:

- **JDK 8** 버전:

jdk-8uxxx-windows-i586.exe

■ 설치 방법:

- 기본 설치

■ 설치된 경로

- JDK 11 이후 버전:

C:\Program Files\Java\jdk -xx.x.x

- JDK 8 버전:

C:\Program Files\Java\jdk1.8.0_xxx



❖ JAVA_HOME

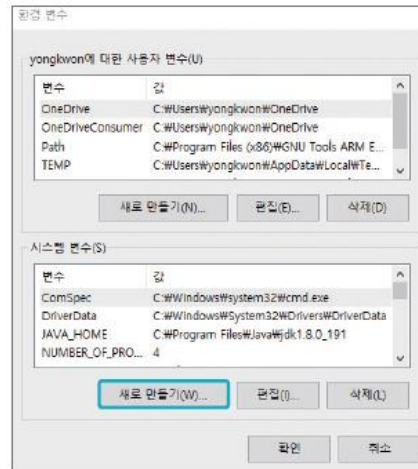
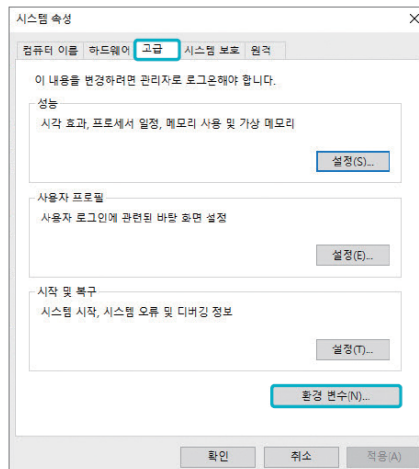
- JDK가 설치된 폴더
- JDK 위치를 찾을 때 JAVA_HOME 환경 변수 이용 필요한 경우 있음
- JAVA_HOME 환경 변수 만들고 JDK 설치 폴더 등록
 - 윈도우 [시스템 속성] 대화상자 이용

윈도우 버전	[시스템 속성] 대화상자 실행 방법
Windows 7	시작 → 제어판 → 시스템 및 보안 → 시스템 → 고급 시스템 설정
Windows 8	화면 오른쪽 아래로 마우스 포인터 옮김 → 설정 → 제어판 → Windows 7과 동일
Windows 10	검색 → '제어판' 입력 후 선택 → Windows 7과 동일

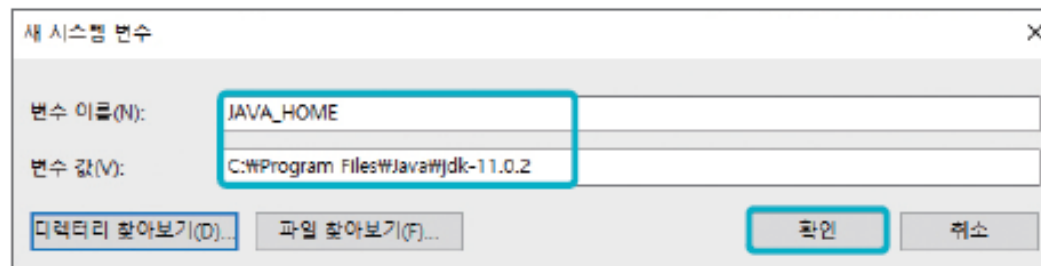


환경 변수 설정

- [시스템 속성] - [고급] - [환경 변수] - [시스템 변수] - [새로 만들기]



- [새 시스템 변수] 대화상자
 - [변수 이름] : JAVA_HOME
 - [변수 값] : JDK 설치 경로 입력
 - [확인]



환경 변수 설정

■ Path 환경 변수 수정

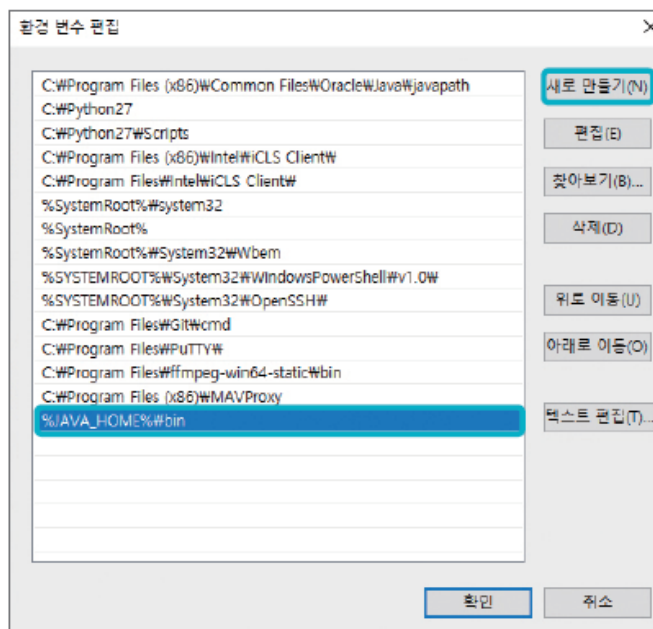
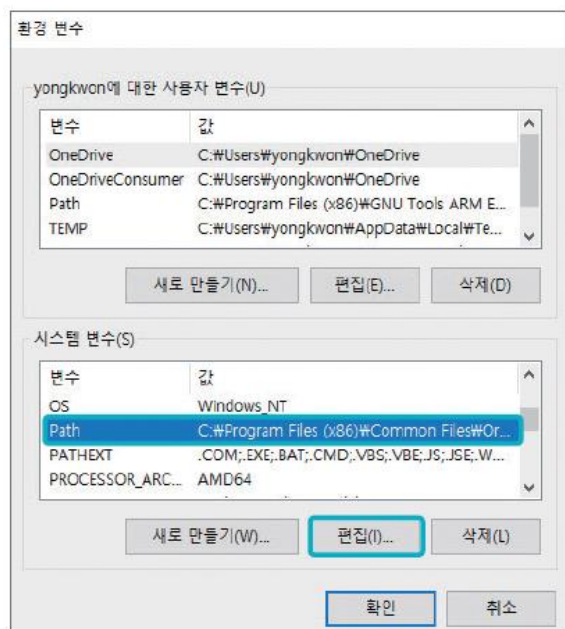
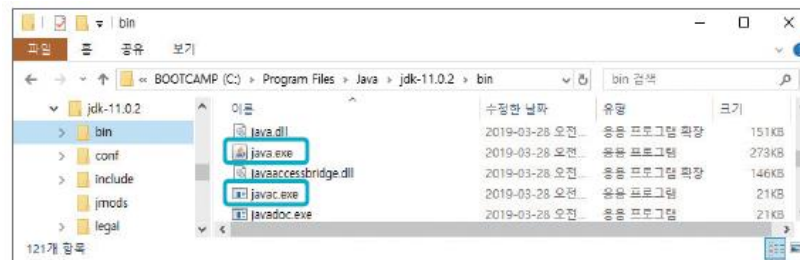
- javac 및 java 명령어를 다른 폴더에서 사용하려면 Path에 bin 폴더 등록

- [환경 변수] 대화상자

- [시스템 변수]에서 Path 환경변수 선택 후 [편집]

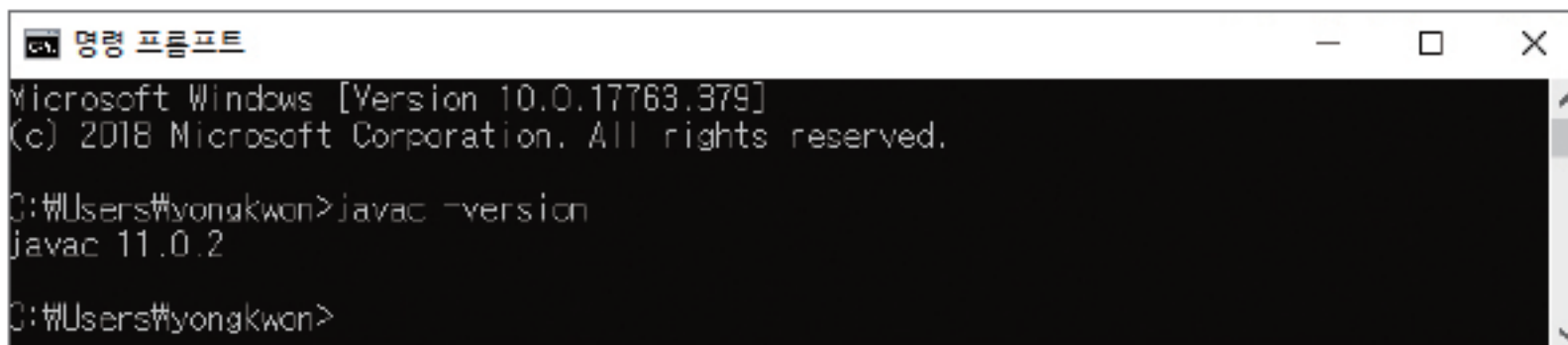
- [환경 변수 편집] 대화상자

- [새로 만들기] - %JAVA_HOME%\bin 입력



환경 변수 설정

- 명령 프롬프트 실행
 - javac -version 입력 후 키보드 [Enter]
 - 그림과 같이 출력

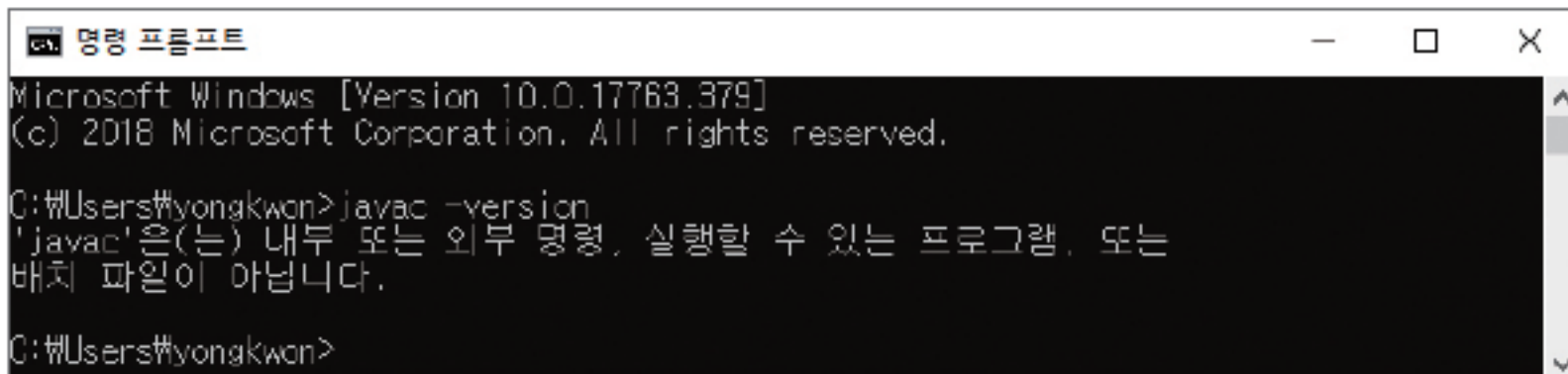


```
명령 프롬프트
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\myongkwon>javac -version
javac 11.0.2

C:\Users\myongkwon>
```

- 아래와 같은 경우 환경변수 설정 오류



```
명령 프롬프트
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\myongkwon>javac -version
'javac'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:\Users\myongkwon>
```

키워드로 끝내는 핵심 포인트

- **기계어**: 컴퓨터(운영체제)가 이해하고 실행할 수 있는 0과 1로 이루어진 코드를 말합니다.
- **프로그래밍 언어**: 사람이 기계어를 이해하는 것은 매우 어렵기 때문에 사람의 언어와 기계어의 다리 역할을 합니다. 종류로는 C, C++, 자바^{Java}, 파이썬^{Python} 등이 있습니다.
- **소스 파일**: 프로그래밍 언어로 작성된 파일을 말합니다.
- **컴파일**: 소스 파일을 기계어로 번역하는 것을 말합니다. 이 역할을 담당하는 소프트웨어를 컴파일러라고 합니다.
- **JDK**: 자바 개발 도구^{Java Development Kit}의 줄임말로, 자바로 프로그램을 개발할 수 있는 실행 환경(JVM)과 개발 도구(컴파일러) 등을 제공합니다.
- **환경 변수**: 운영체제가 실행하는 데 필요한 정보를 제공해주는 변수를 말합니다. JDK를 설치한 후 명령 라인(명령 프롬프트, 터미널)에서 컴파일러(javac)와 실행(java) 명령어를 사용하려면 JAVA_HOME 환경 변수를 등록하고 Path 환경 변수를 수정하는 것이 좋습니다.



Chapter

01

자바 시작하기



01-2. 이클립스 개발 환경 구축

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- 이클립스 설치
- 워크스페이스
- 퍼스펙티브와 뷰
- 키워드로 끝내는 핵심 포인트



시작하기 전에

핵심 키워드

이클립스

워크스페이스

뷰

퍼스펙티브

복잡한 프로그램을 개발할 경우에는 개발자의 코딩 실수를 줄이기 위해 키워드의 색상 구분, 자동 코드 완성 및 디버깅(debugging: 모의 실행을 해서 코드의 오류를 찾는 것) 기능을 갖춘 소스 편집 툴을 사용하는 것이 좋습니다.
이번 절에서는 기업체에서 가장 선호하는 개발 전문 툴인 이클립스의 사용 방법에 대해 알아보겠습니다.



좋은 편집 툴이란
개발자에게 보다
정확한 코딩을 유
도하도록 도와주
는 것!

❖ 이클립스 (eclipse) : <http://www.eclipse.org>

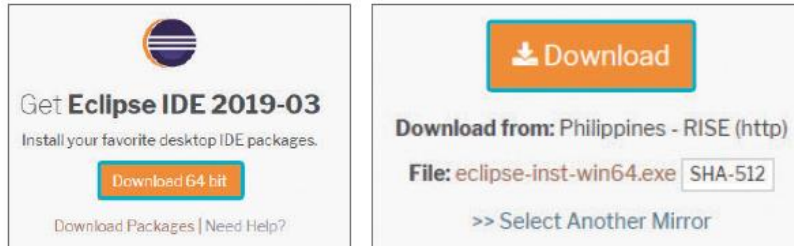
- 무료 오픈 소스 통합 개발 환경(IDE: Integrated Development Environment)
 - IDE: 프로젝트 생성, 자동 코드 완성, 디버깅 등과 같이 개발에 필요한 여러 가지 기능을 통합적으로 제공해주는 툴
- 기본적으로 자바 프로그램을 개발할 수 있도록 구성되어 있음
 - 플러그인(plugin)을 설치하면 웹 애플리케이션 개발, C, C++ 애플리케이션 개발 등 다양한 개발 환경을 구축할 수 있음
- 학습자 뿐만 아니라 고급 개발자에 이르기까지 광범위하게 사용



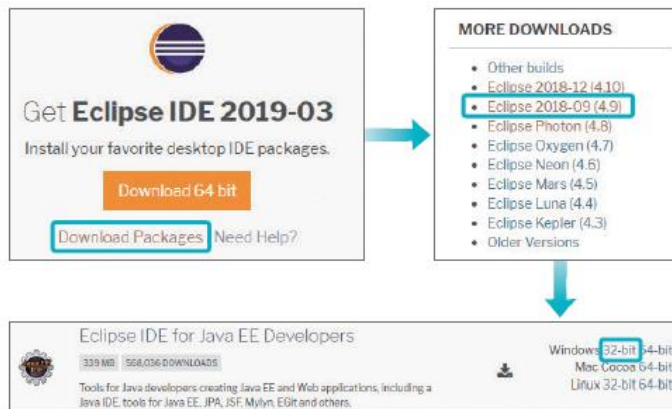
이클립스 설치

❖ 설치 파일 다운로드(실습)

- Eclipse IDE 2019-03 버전 이후부터는 기본적으로 64비트 설치용 인스톨러 파일 제공



- ZIP 또는 32비트 버전도 다운로드 가능



❖ 설치(실습)



워크스페이스, 퍼스펙티브, 뷰

❖ 워크스페이스(workspace)

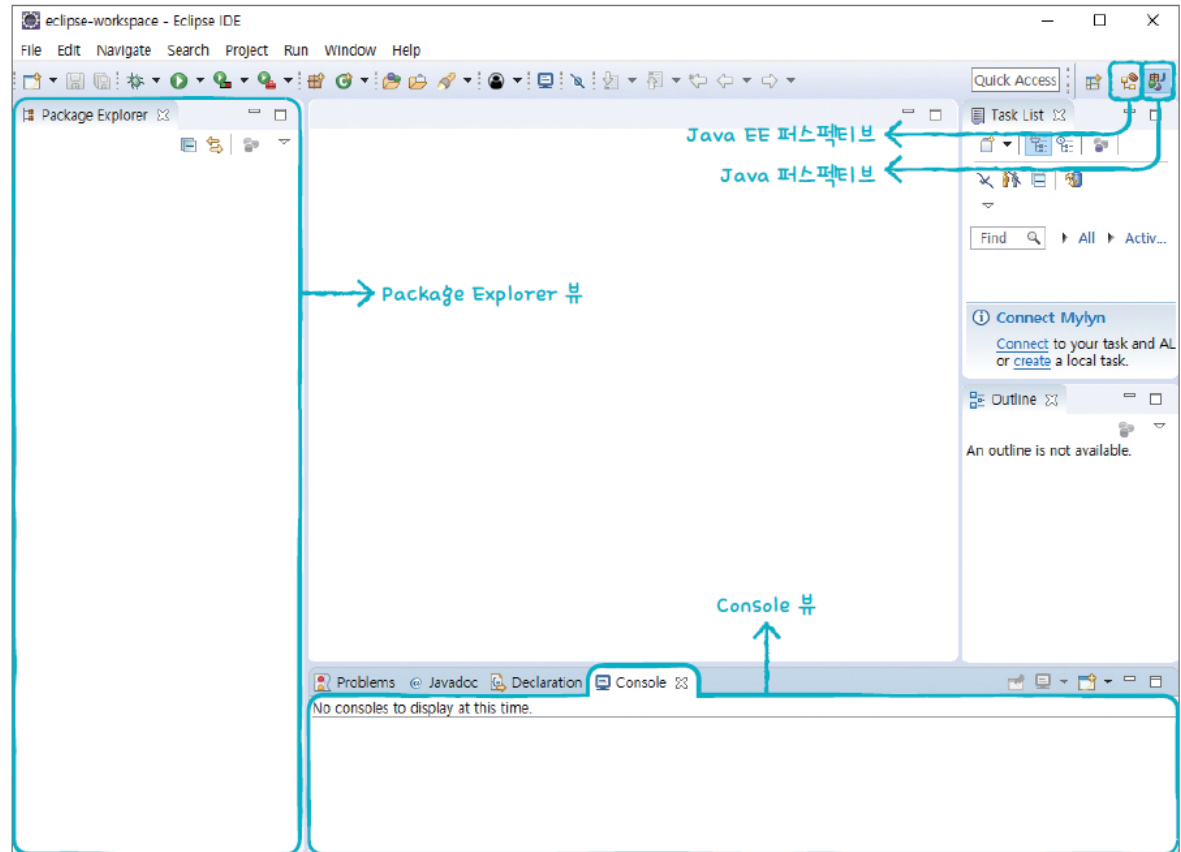
- 프로젝트 폴더가 저장
- 개발 환경 정보와 관련된 메타 데이터가 저장된 폴더(.metadata)가 저장

❖ 퍼스펙티브(perspective)

- 프로젝트를 개발할 때 유용하게 사용할 수 있는 뷰View들을 미리 묶어 이름을 붙여놓은 것

❖ 뷰(view)

- 이클립스 내부에서 사용되는 작은 창



키워드로 끝내는 핵심 포인트

- **이클립스**: 무료로 사용할 수 있는 오픈 소스 통합 개발 환경IDE: Integrated Development Environment입니다. IDE란 프로젝트 생성, 자동 코드 완성, 디버깅 등과 같이 개발에 필요한 여러 가지 기능을 통합적으로 제공해주는 툴을 말합니다.
- **워크스페이스**: 이클립스 실행과 관련된 메타 데이터metadata와 프로젝트 폴더가 저장되는 폴더를 말합니다.
- **뷰**: 이클립스 내부에서 사용되는 작은 창을 말합니다.
- **퍼스펙티브**: 프로젝트를 개발할 때 유용하게 사용할 수 있는 뷰view들을 미리 묶어 이름을 붙여 놓은 것을 말합니다.



Chapter

01

자바 시작하기



01-3. 자바 프로그램 개발 과정

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- 바이트 코드 파일과 자바 가상 기계
- 프로젝트 생성부터 실행까지
- 프로그램 소스 분석
- 주석 사용하기
- 실행문과 세미콜론(;)
- 좀 더 알아보기
- 키워드로 끝내는 핵심 포인트



시작하기 전에

핵심 키워드

바이트 코드 파일

JVM

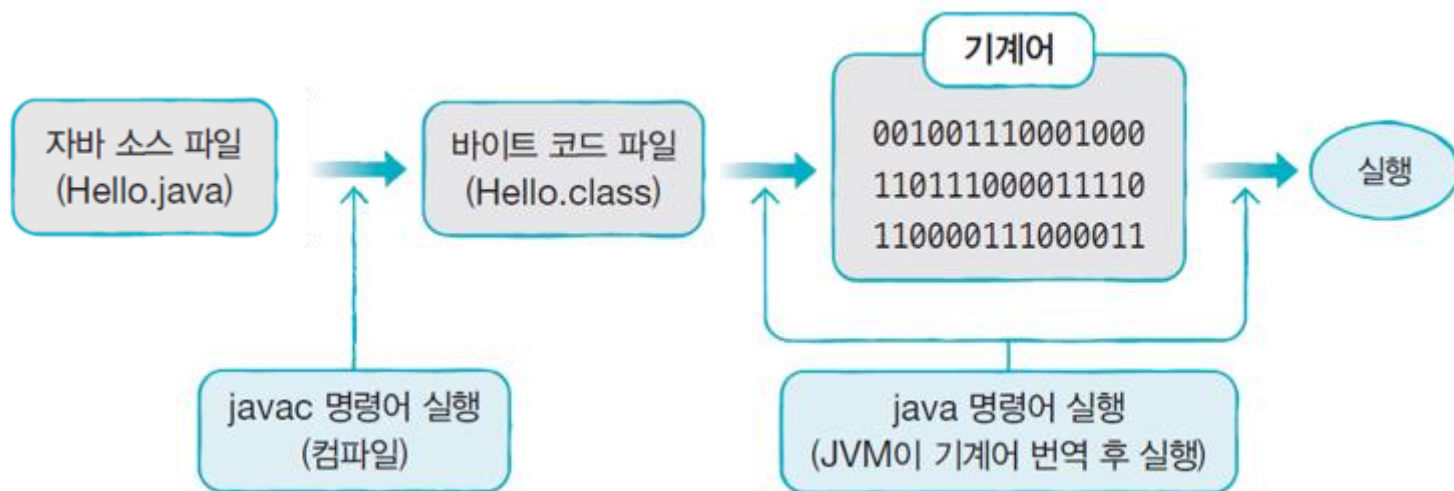
클래스 선언

main() 메소드

주석

실행문

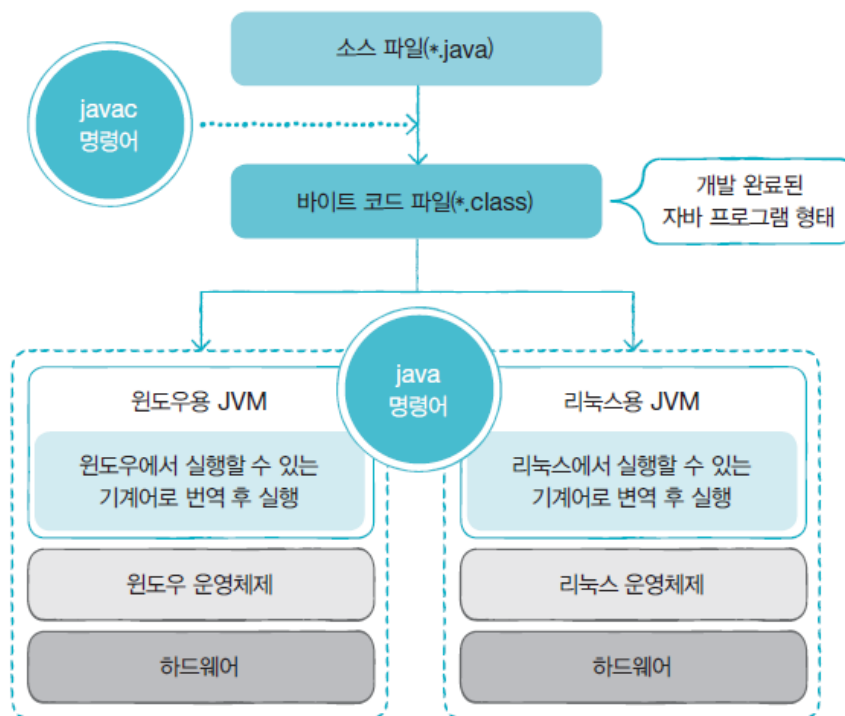
이번 절에서는 소스 파일 작성부터 실행까지 실습을 진행하면서 자바 프로그램 개발 과정을 이해해보겠습니다. 그리고 소스 파일의 구조에 대해 알아보겠습니다.



바이트 코드 파일과 자바 가상 기계

❖ 바이트 코드 파일과 자바 가상 기계

- 자바 프로그램은 완전한 기계어가 아닌, 바이트 코드(byte code) 파일(.class)로 구성
- 바이트 코드 파일은 운영체제에서 바로 실행할 수 없음
- 자바 가상 기계(JVM: Java Virtual Machine)가 완전한 기계어로 번역하고 실행



프로젝트 생성부터 실행까지

❖ 이클립스에서 실습

- [1단계] 프로젝트 생성 -> [2단계] 소스 파일 생성과 작성 -> [3단계] 바이트 코드 실행

❖ 명령 라인에서 실습

- 컴파일

JDK 8 이전 버전

컴파일	<code>javac -d [바이트 코드 파일 저장 위치] [소스 경로/*.java]</code>
	<code>javac -d bin src/sec03/exam01/*.java</code>

JDK 11 이후 버전

컴파일	<code>javac -d [바이트 코드 파일 저장 위치] [소스 경로/module-info.java 소스 경로/*.java]</code>
	<code>javac -d bin src/module-info.java src/sec03/exam01/*.java</code>

- 실행

JDK 8 이전 버전

실행	<code>java -cp [바이트 코드 파일 저장 위치] [패키지이름...클래스이름]</code>
	<code>java -cp bin sec03.exam01.Hello</code>

JDK 11 이후 버전

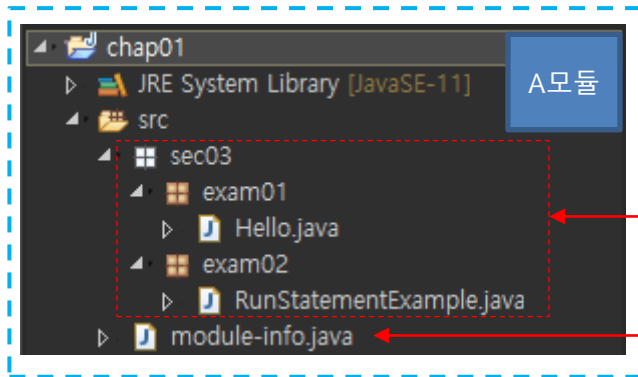
실행	<code>java -p [바이트 코드 파일 저장 위치] -m 모듈/패키지이름...클래스이름</code>
	<code>java -p bin -m chap01/sec03.exam01.Hello</code>



모듈 기술자(Java 11 버전 이후)

❖ 모듈(Module)이란

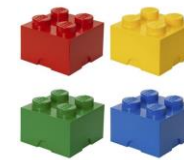
- 외부에서 재사용할 수 있는 패키지들을 묶은 것
- 이클립스의 프로젝트는 하나의 모듈을 개발하는 것



모듈

패키지

모듈 기술자



작품



프로그램

❖ 모듈 기술자(module-info.java)

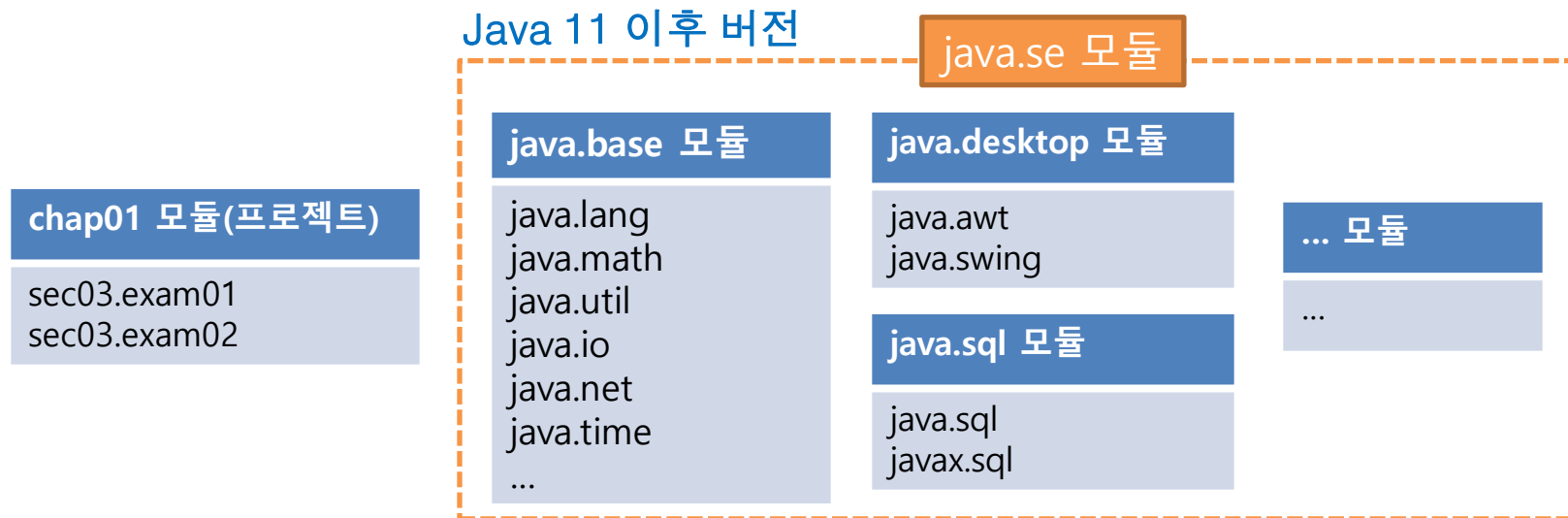
- 해당 **모듈의 이름**이 무엇인지,
- 해당 모듈이 **의존하는 모듈**이 무엇인지
- 해당 모듈을 외부에서 어떻게 사용할 수 있는지



모듈 기술자(Java 11 버전 이후)

❖ 의존하는 모듈

- 해당 모듈(프로젝트)가 실행하기 위해서 필요한 외부 모듈을 말함



- 기본적으로 **java.base** 모듈만 사용 가능하므로 다른 모듈을 사용할 경우 의존 모듈로 등록해야 함

```
module-info.java x
1 module chap01 {
2     requires java.desktop;
3     requires java.sql;
4 }
```

```
*module-info.java x
1 module chap01 {
2     requires java.se;
3 }
```



모듈화가 필요한 이유

❖ 모듈화가 필요한 이유

- 패키지 보안
- 작은 최적의 런타임 이미지를 만들기 위해

Java 8

java.lang	java.awt	...
java.math	java.swing	
java.util		
java.io	java.sql	
java.net	javax.sql	
java.time		

자바 실행(런타임) 환경

Java 8

java.lang	java.awt	...
java.math	java.swing	
java.util		
java.io	java.sql	
java.net	javax.sql	
java.time		

250MB



Hello
프로그램

Java 11 이후 버전

java.se 모듈

java.base 모듈

java.lang
java.math
java.util
java.io
java.net
java.time
...

java.desktop 모듈

java.awt
java.swing

java.sql 모듈

java.sql
javax.sql

... 모듈

...

자바 실행(런타임) 환경

java.base 모듈

java.lang
java.math
java.util
java.io
java.net
java.time
...

30MB



Hello
프로그램

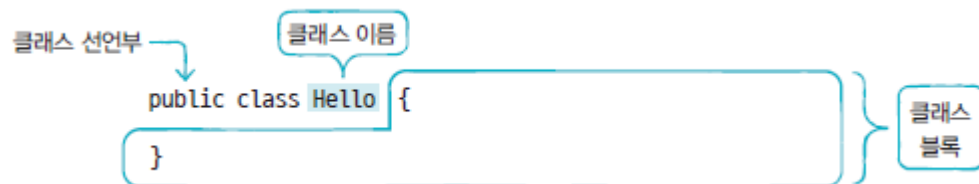


프로그램 소스 분석

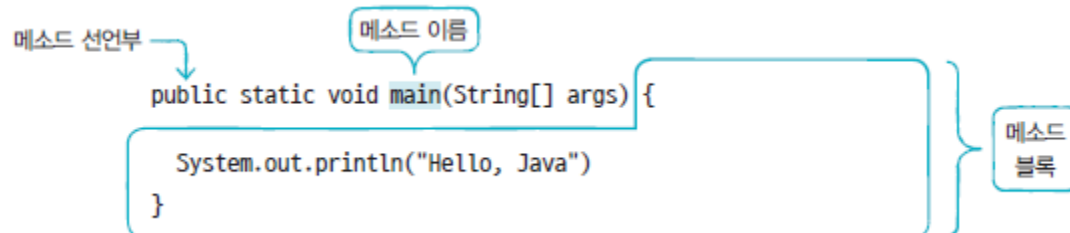
❖ 패키지 선언

```
package sec03.exam01;
```

❖ 클래스 선언



❖ 메소드 선언



주석 사용하기

❖ 주석

- 코드에 설명을 붙여놓은 것
- 컴파일 과정에서 무시되고 실행문만 바이트 코드로 번역

❖ 주석의 종류

구분	주석 기호	설명
라인 주석	// ...	//부터 라인 끝까지 주석으로 처리합니다.
범위 주석	/* ... */	/*와 */ 사이에 있는 내용은 모두 주석으로 처리합니다.
도큐먼트 주석	/** ... */	/**와 */ 사이에 있는 내용은 모두 주석으로 처리합니다. 주로 javadoc 명령어로 API 도큐먼트를 생성하는 데 사용합니다.



실행문과 세미콜론(;)

❖ 실행문

- main() 메소드 블록 내부에는 다양한 실행문들이 작성
- 실행문 끝에는 반드시 세미콜론(;)을 붙여서 실행문이 끝났음을 표시

```
System.out.println("Hello, Java");
```

❖ 실행문의 종류

```
int x;           //변수 x 선언
x = 1;           //변수 x에 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); //println 메소드 호출
```



키워드로 끝내는 핵심 포인트

- **바이트 코드 파일:** 자바 소스 파일을 javac 명령어로 컴파일한 파일을 말합니다.
- **JVM:** 자바 가상 기계^{Java Virtual Machine}는 바이트 코드 파일을 운영체제를 위한 완전한 기계어로 번역하고 실행하는 역할을 합니다. JVM은 java 명령어에 의해 구동됩니다.
- **클래스 선언:** 자바 소스 파일은 클래스 선언부와 클래스 블록으로 구성됩니다. 이렇게 작성하는 것을 클래스 선언이라고 합니다.
- **main() 메소드:** java 명령어로 바이트 코드 파일을 실행하면 제일 먼저 main() 메소드를 찾아 블록 내부를 실행합니다. 그래서 main() 메소드를 프로그램 실행 진입점^{entry point}이라고 부릅니다.
- **주석:** 주석은 프로그램 실행과는 상관없이 코드에 설명을 붙인 것을 말합니다. 주석은 컴파일 과정에서 무시되고 실행문만 바이트 코드로 번역됩니다.
- **실행문:** 변수 선언, 값 저장, 메소드 호출에 해당하는 코드를 말합니다. 실행문 끝에는 세미콜론(;)을 붙여야 합니다.

