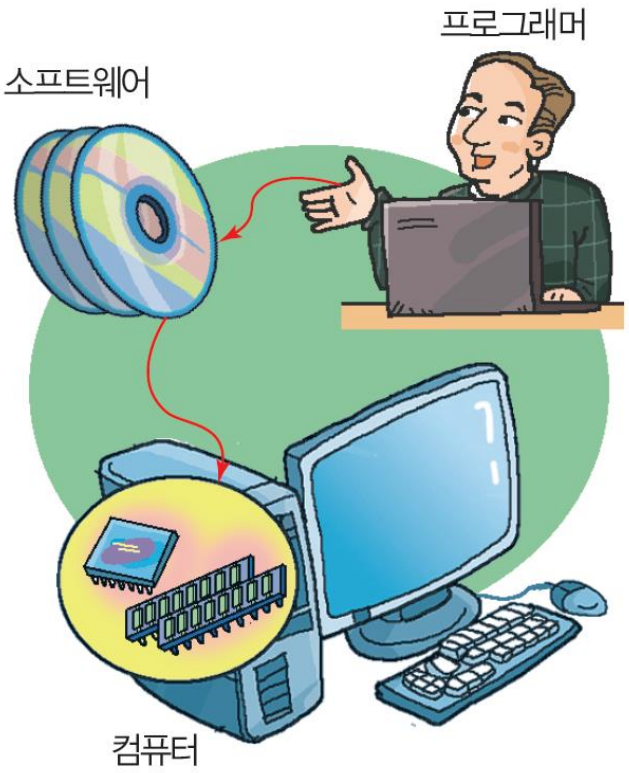
01

자바 시작

학습 목표

1. 컴퓨터가 소프트웨어를 실행하는 범용 계산기 임을 이해
2. 자바의 출현 배경과 플랫폼 독립성, WORA의 개념 이해
3. 자바 가상 기계와 자바의 실행 환경 이해
4. JDK와 JRE 등 자바 개발 환경 이해
5. 이클립스를 이용한 자바 프로그램 작성
6. 자바 응용프로그램의 종류와 특징 이해
7. 자바 언어와 자바 플랫폼의 특징 이해

컴퓨터와 소프트웨어

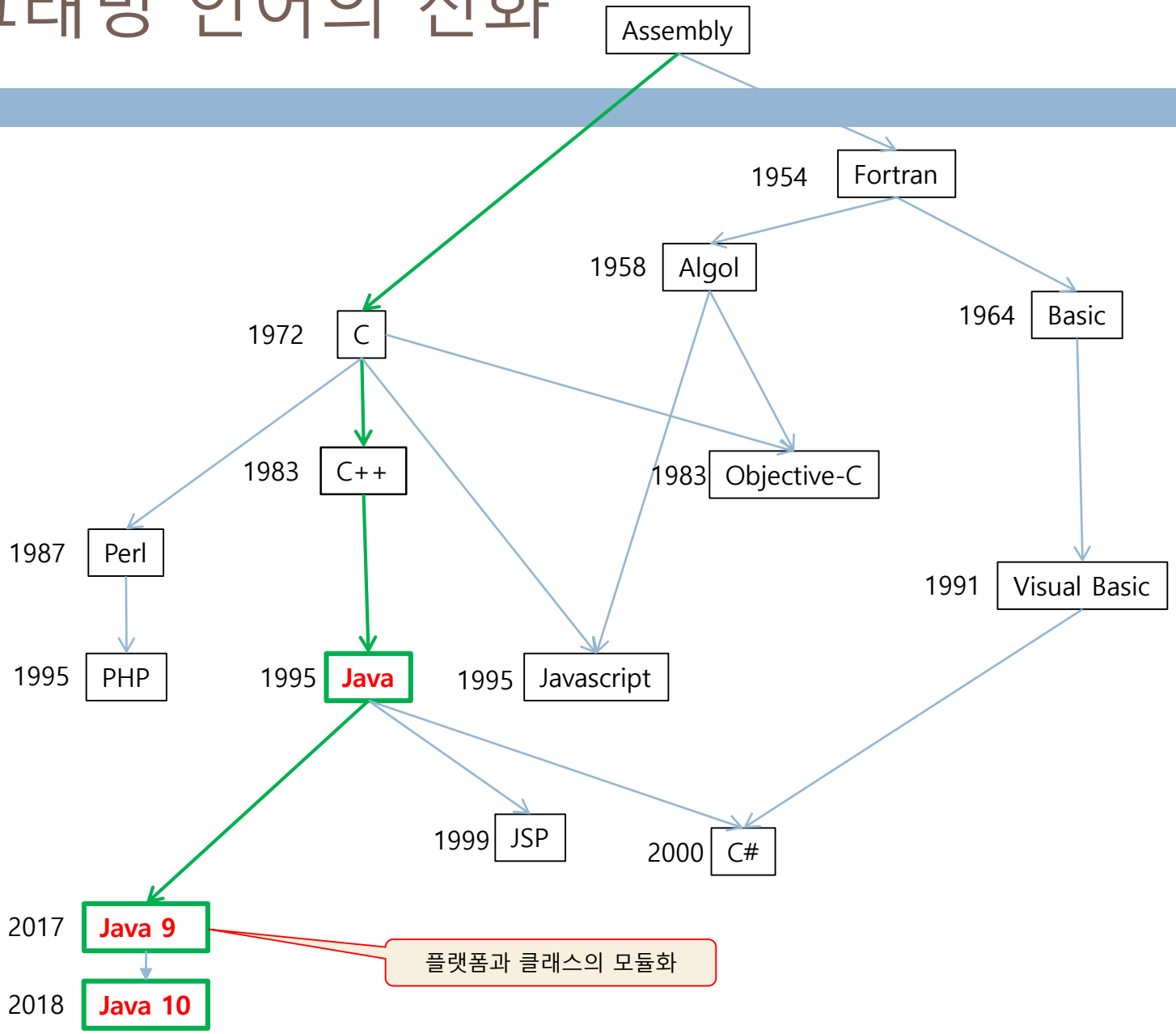


컴퓨터와 프로그래머, 소프트웨어의 관계는
만능 요리 기계, 요리설계사와, 요리순서와 같다.

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 언어로 나눌 수 있음

프로그래밍 언어의 진화


Assembly
1954
Fortran
1958
Algol
1964
Basic
1972
C
1983
C++
1983
Objective-C
1987
Perl
1991
Visual Basic
1995
PHP
1995
Java
1995
Javascript
1999
JSP
2000
C#
2017
Java 9
2018
Java 10
플랫폼과 클래스의 모듈화

프로그래밍과 컴파일

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

소스 프로그램
편집 및 개발

프로그램
실행

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

소스 프로그램

컴파일러

컴파일

```
01010000010001
01010011110101
10101010010111
01010101010000
10001110000000
```

기계어

자바의 태동

- 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 오라클이 인수

기존 언어의 플랫폼 종속성

인텔 CPU를 가진
리눅스 환경에서
개발

C/C++
응용 프로그램

실행

실행되지
않음

실행되지
않음

Intel CPU + 리눅스

Apple 사의 MAC PC

Intel CPU + 윈도우 노트북

플랫폼 = 하드웨어 플랫폼 + 운영체제 플랫폼

프로그램의 플랫폼 호환성 없는 이유
- 기계어가 CPU마다 다름
- 운영체제마다 API 다름
- 운영체제마다 실행파일 형식 다름

자바의 플랫폼 독립성, WORA

- 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)

자바의 플랫폼 독립성

Write Once !!

자바
응용 프로그램

Run Anywhere!!

실행

실행

실행

자바 가상 기계

Intel CPU + 리눅스

자바 가상 기계

Apple 사의 MAC PC

자바 가상 기계

Intel CPU + 윈도우 노트북

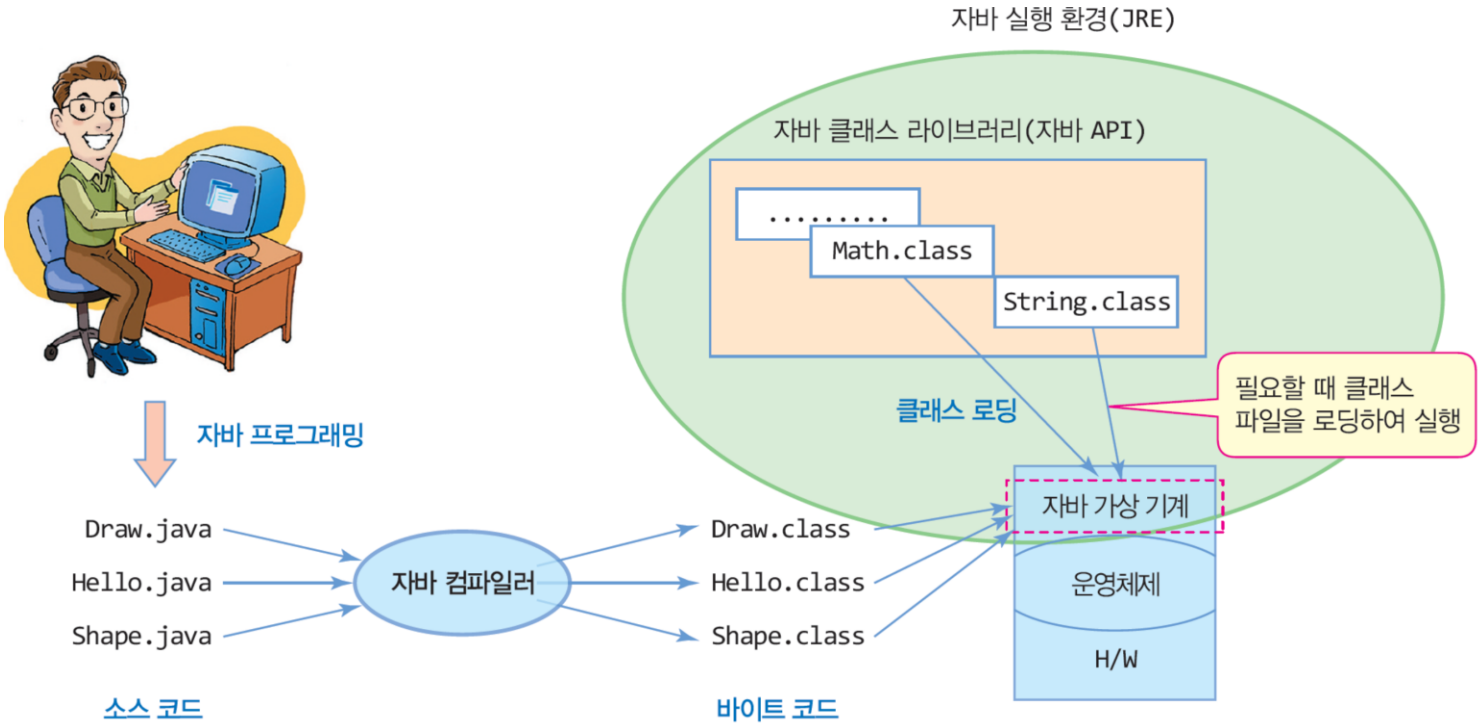
자바 가상 기계와 자바 실행 환경

- 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)
  - 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

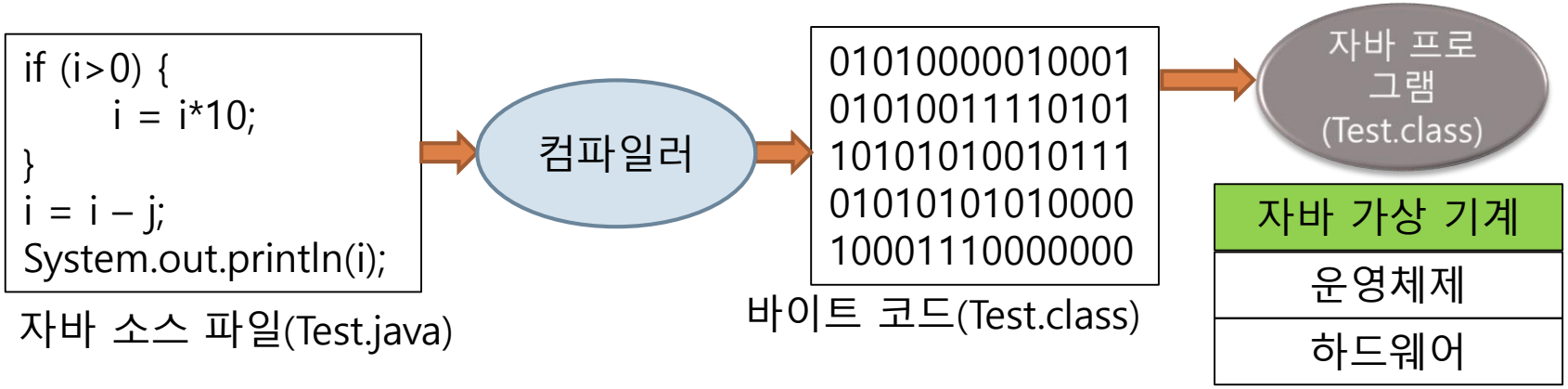
자바 응용프로그램 실행 환경

- 실행 환경
  - 자바 가상 기계 + 자바 플랫폼의 다양한 클래스 라이브러리(자바 API)
- 응용프로그램 실행
  - main() 메소드를 가진 클래스의 main()에서 실행 시작
  - 자바 가상 기계는, 필요할 때 클래스 파일 로딩, 적은 메모리로 실행 가능

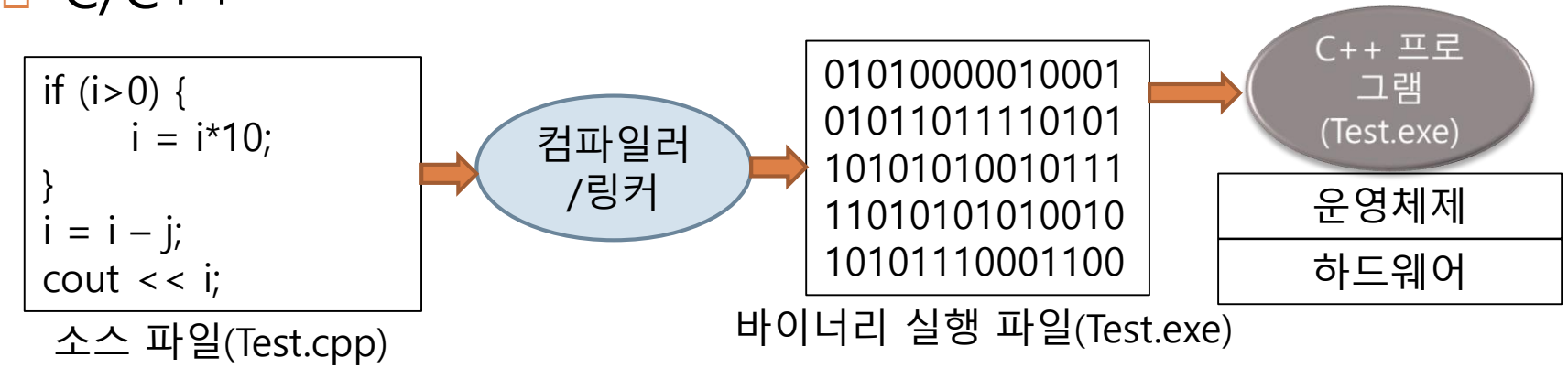


자바와 타언어(C/C++)의 실행 차이

자바



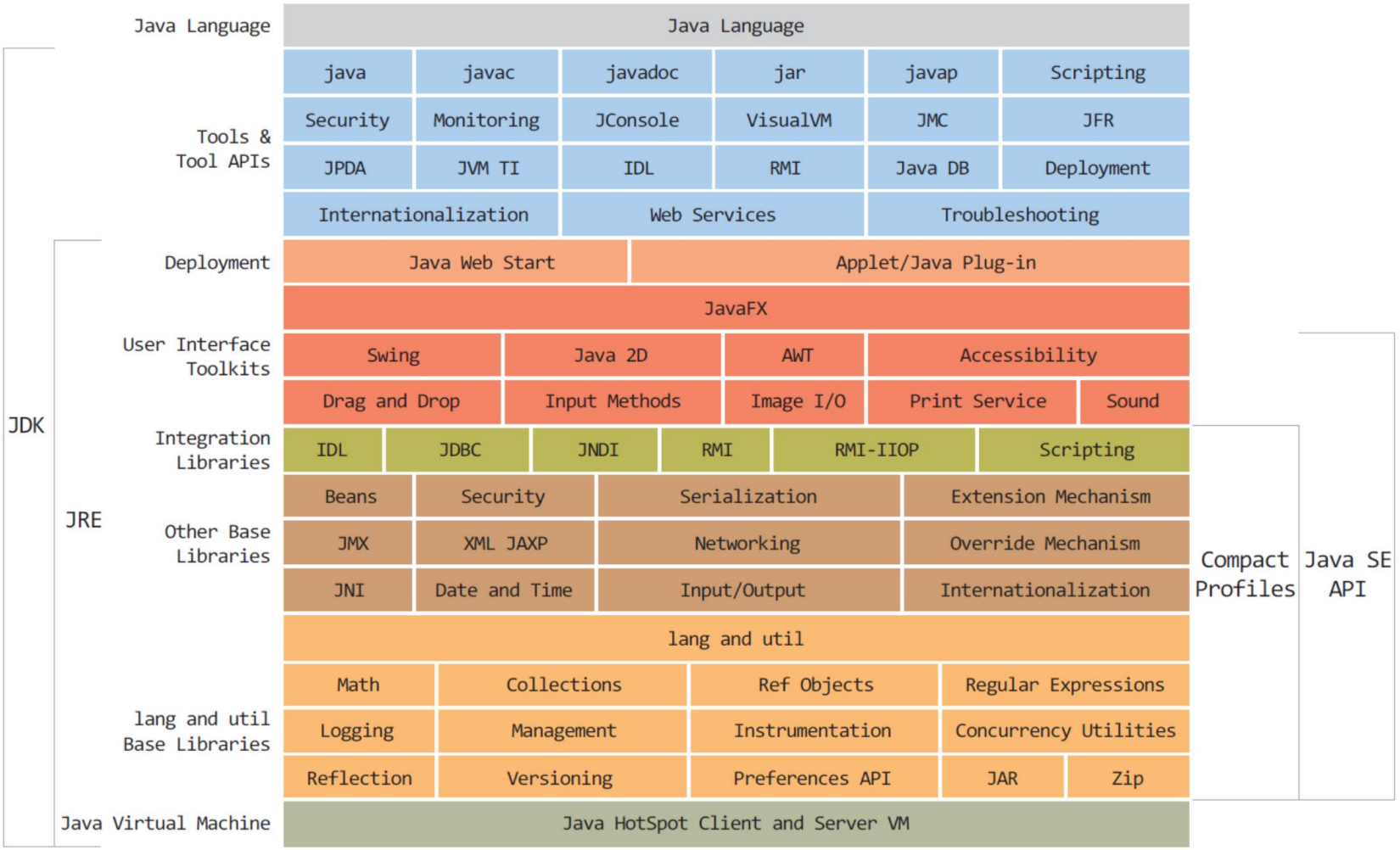
C/C++



JDK와 JRE

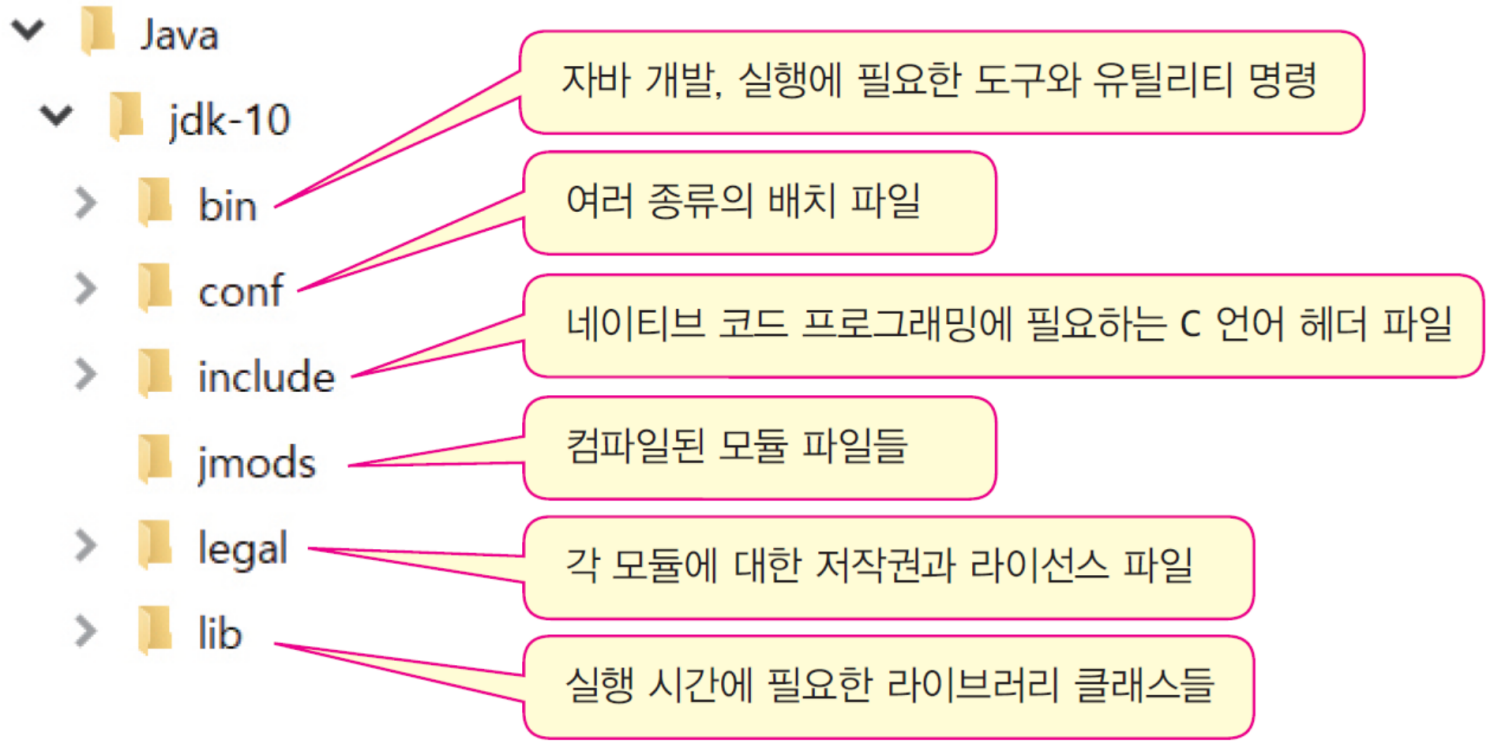
- JDK(Java Development Kit)
  - 자바 응용 개발 환경. 개발에 필요한 도구 포함
    - 컴파일러, 컴파일된 자바 API 클래스들이 들어 있는 모듈 파일들, 샘플 등 포함
- JRE(Java Runtime Environment)
  - 자바 실행 환경. JVM 포함
  - 컴파일된 자바 API 들이 들어 있는 모듈 파일
  - 개발자가 아닌 경우 JRE만 따로 다운 가능
- JDK와 JRE의 개발 및 배포
  - 오라클의 Technology Network의 자바 사이트에서 다운로드
    - http://www.oracle.com/technetwork/java/index.html
- 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



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

나는 누구?

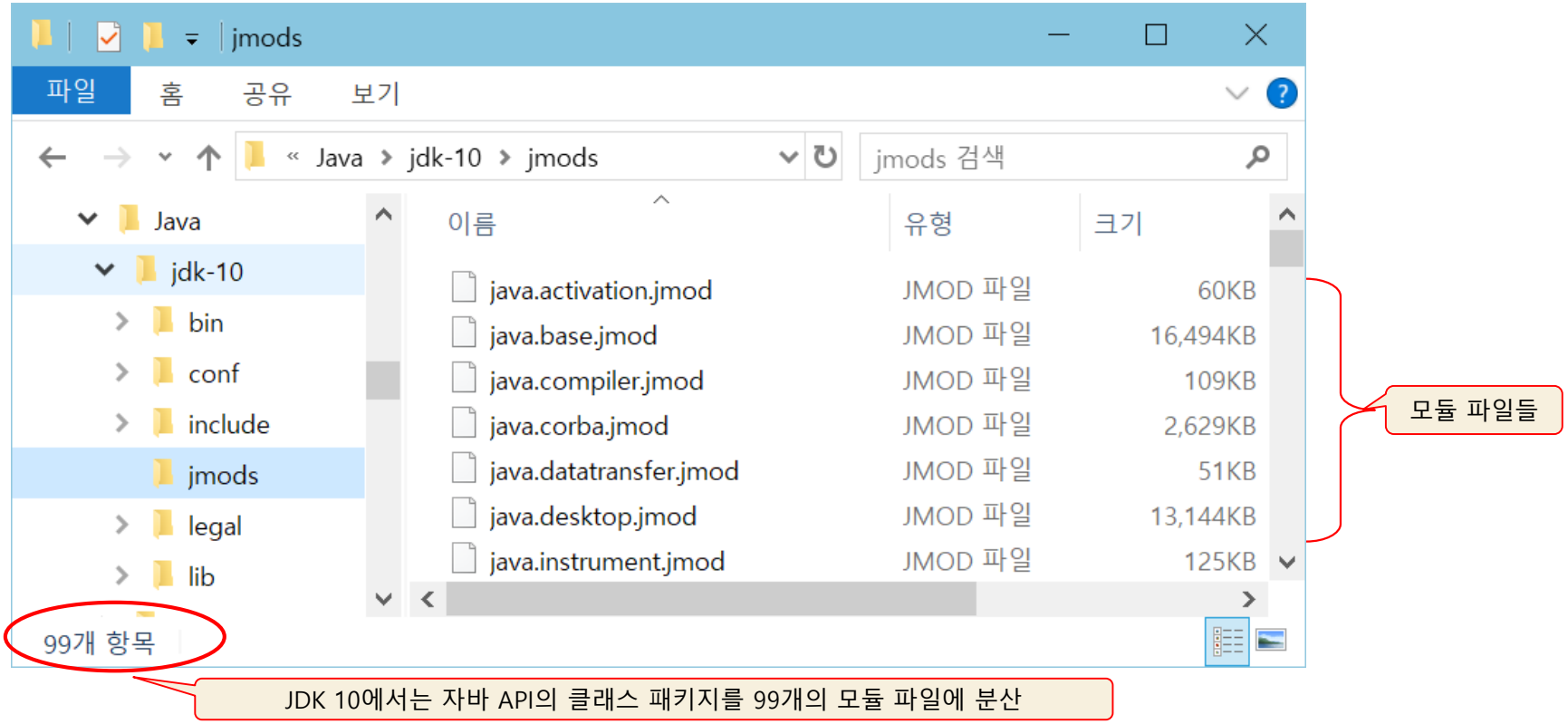
(사진 출처 : 위키 백과)

Java 9부터 시작된 모듈 프로그래밍

- 모듈화(modularity)
  - Java 9에서 정의된 새로운 기능, 2017년 9월 21일 출시
  - 모듈
    - 자바 패키지들과 이미지, XML 파일 등의 자원들을 묶은 단위
  - 모듈 프로그래밍
    - 자바 응용프로그램을 마치 직소 퍼즐(jigsaw)을 연결하듯이 필요한 모듈을 연결하는 방식으로 작성
- 자바 플랫폼의 모듈화
  - 실행 시간에 사용되는 자바 API의 모든 클래스들을 모듈들로 분할
  - 모듈화의 목적
    - 세밀한 모듈화, 자바 응용프로그램이 실행되는데 필요없는 모듈 배제
    - 작은 크기의 실행 환경 구성
    - 하드웨어가 열악한 소형 IoT 장치 지원
- 모듈 방식이 아닌, 기존 방식으로 자바 프로그래밍 해도 무관
  - 자바 플랫폼이 모듈 방식으로 바뀌었지만,
  - 굳이 응용프로그램을 모듈 방식으로 작성할 필요 없음
    - 모듈 설계자들도 이런 사실 강조

자바에서 제공하는 전체 모듈 리스트(Java SE)

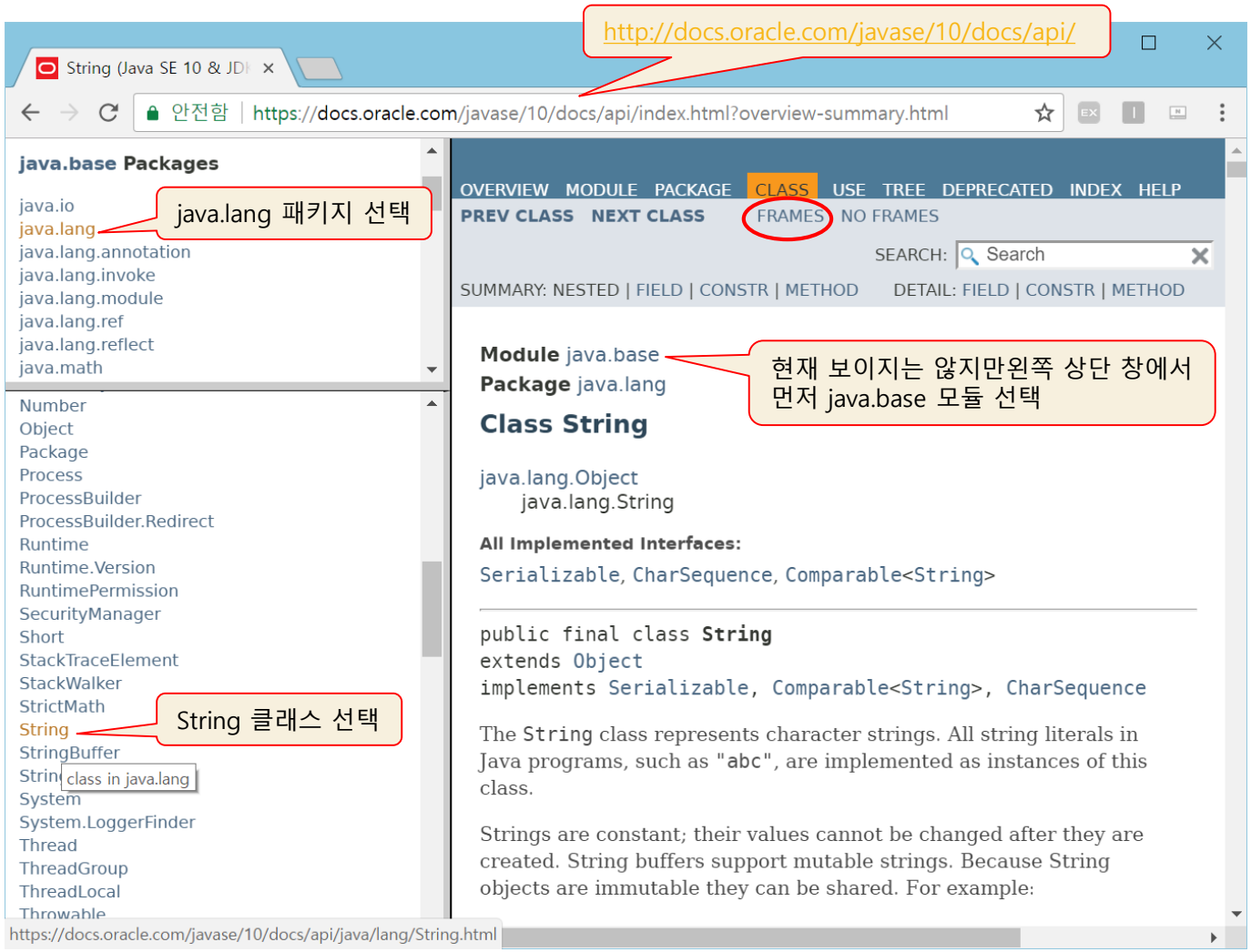
- Java 9부터 플랫폼을 모듈화함
  - Java SE의 모든 클래스들을 모듈들로 재구성
  - JDK의 설치 디렉터리 밑의 jmods 디렉터리에 있음



자바 API

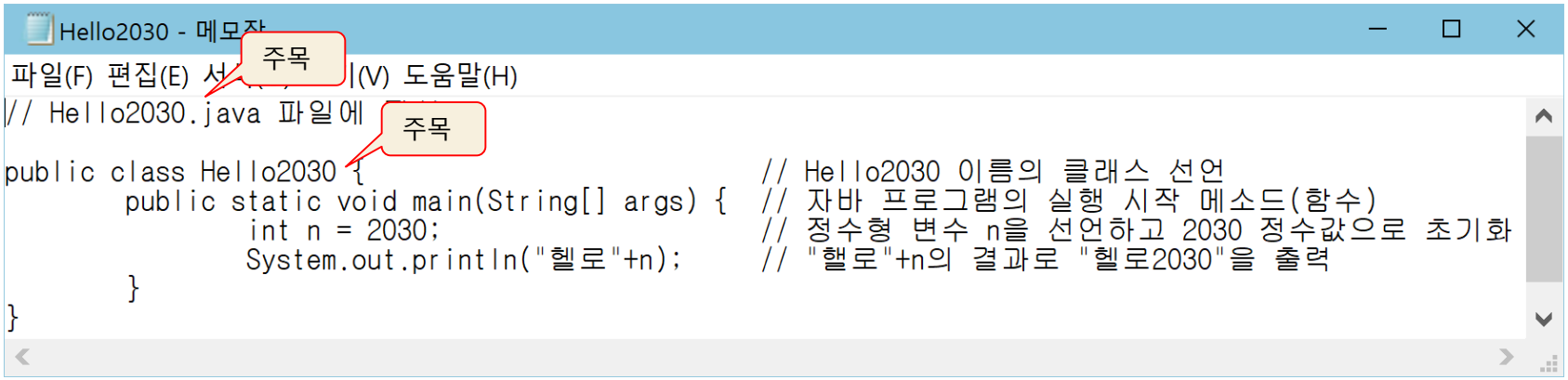
- 자바 API(Application Programming Interface)란?
  - JDK에 포함된 클래스 라이브러리
    - 주요한 기능들을 미리 구현한 클래스 라이브러리의 집합
  - 개발자는 API를 이용하여 쉽고 빠르게 자바 프로그램 개발
    - API에서 정의한 규격에 따라 클래스 사용
- 자바 패키지(package)
  - 서로 관련된 클래스들을 분류하여 묶어 놓은 것
  - 계층구조로 되어 있음
    - 클래스의 이름에 패키지 이름도 포함
    - 다른 패키지에 동일한 이름의 클래스 존재 가능
  - 자바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

자바 온라인 API 문서


http://docs.oracle.com/javase/10/docs/api/
String (Java SE 10 & JD
안전함 | https://docs.oracle.com/javase/10/docs/api/index.html?overview-summary.html
java.base Packages
java.io
java.lang
java.lang.annotation
java.lang.invoke
java.lang.module
java.lang.ref
java.lang.reflect
java.math
java.lang 패키지 선택
Number
Object
Package
Process
ProcessBuilder
ProcessBuilder.Redirect
Runtime
Runtime.Version
RuntimePermission
SecurityManager
Short
StackTraceElement
StackWalker
StrictMath
String
StringBuffer
Strin
class in java.lang
System
System.LoggerFinder
Thread
ThreadGroup
ThreadLocal
Throwable
String 클래스 선택
OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SEARCH: Search
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
Module java.base
Package java.lang
현재 보이지는 않지만왼쪽 상단 창에서
먼저 java.base 모듈 선택
Class String
java.lang.Object
java.lang.String
All Implemented Interfaces:
Serializable, CharSequence, Comparable<String>
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.
Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:
https://docs.oracle.com/javase/10/docs/api/java/lang/String.html

자바 프로그램 개발 : (1) 자바 소스 편집

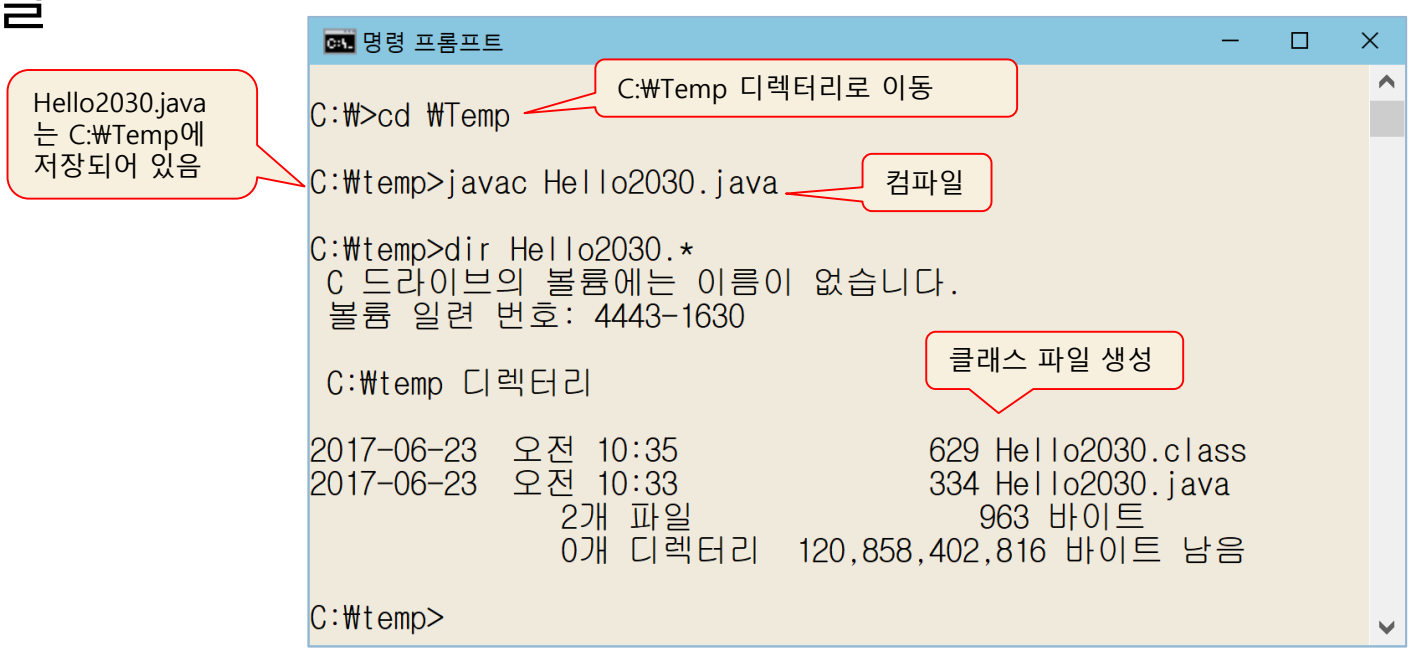
- 어떤 편집기를 사용해도 무관
  - 메모장으로 작성한 샘플



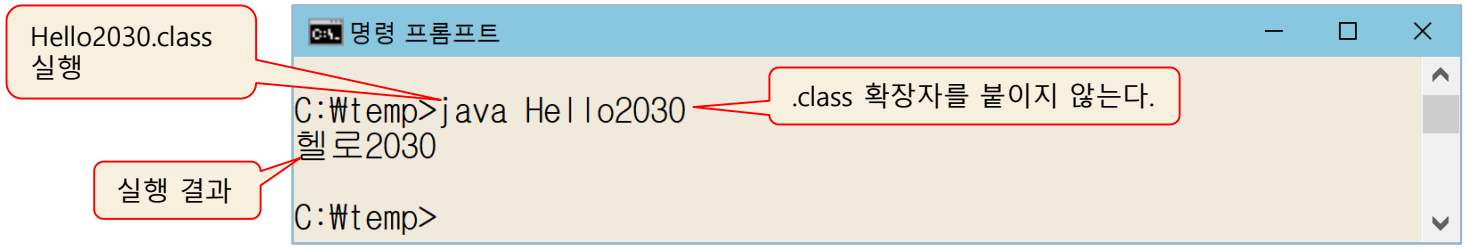
- 작성 후 Hello2030.java로 저장
  - 반드시 클래스와 동일한 이름으로 파일 저장
    - C:₩Temp에 저장
  - 확장자 .java

자바 프로그램 개발 : (2) 컴파일 및 실행

- 컴파일



- 실행



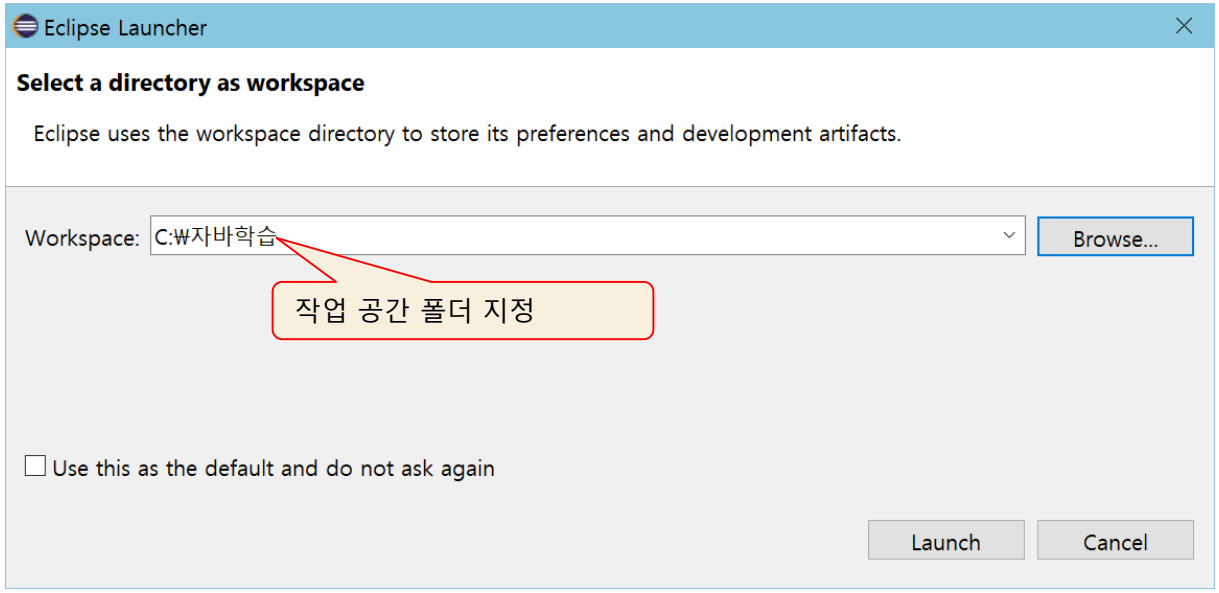
자바 통합 개발 환경–이클립스(Eclipse)

- IDE(Integrated Development Environment)란?
  - 통합 개발 환경
  - 편집, 컴파일, 디버깅을 한번에 할 수 있는 통합된 개발 환경

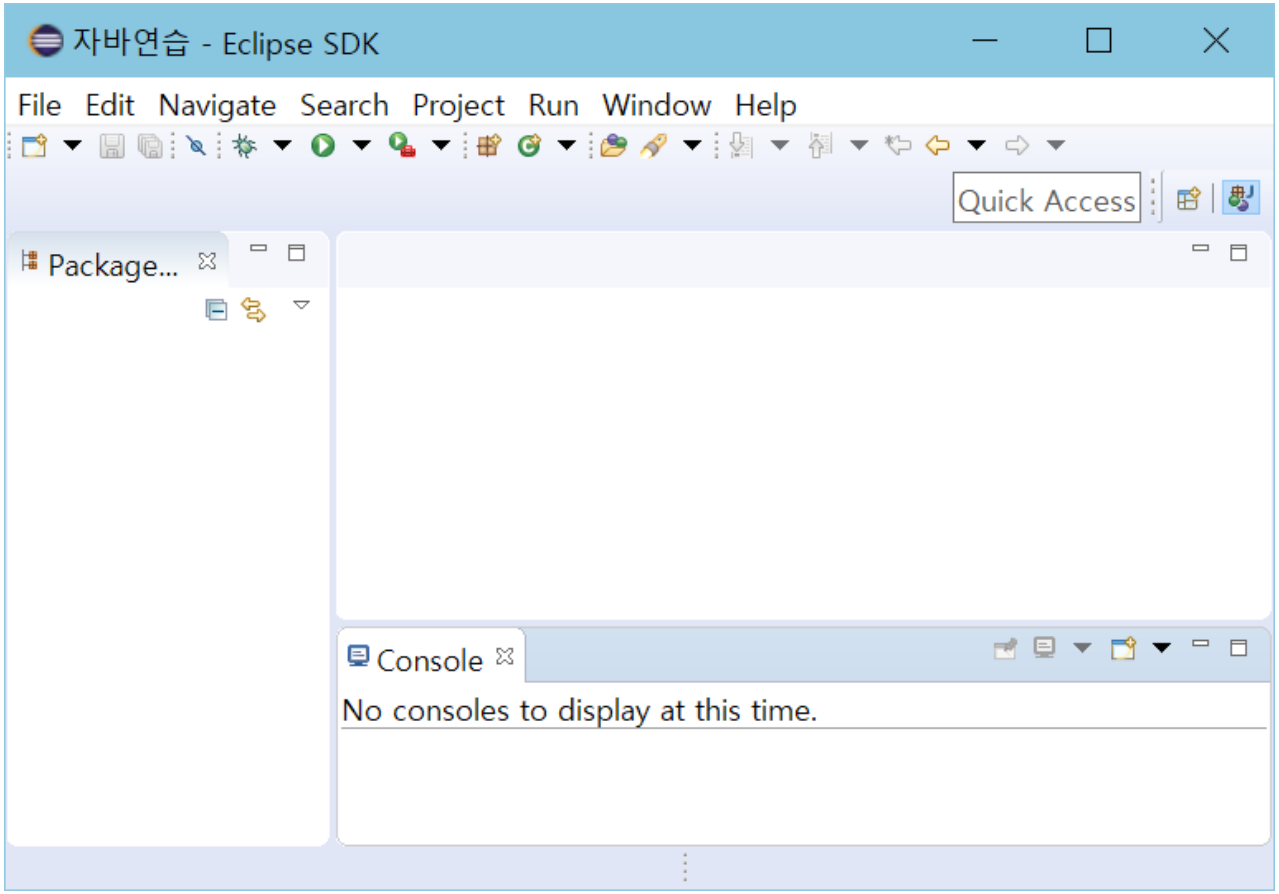
- 이클립스(Eclipse)
  - 자바 응용 프로그램 개발을 위한 통합 개발 환경
  - IBM에 의해 개발된 오픈 소스 프로젝트
  - http://www.eclipse.org/downloads/ 에서 다운로드

이클립스 실행

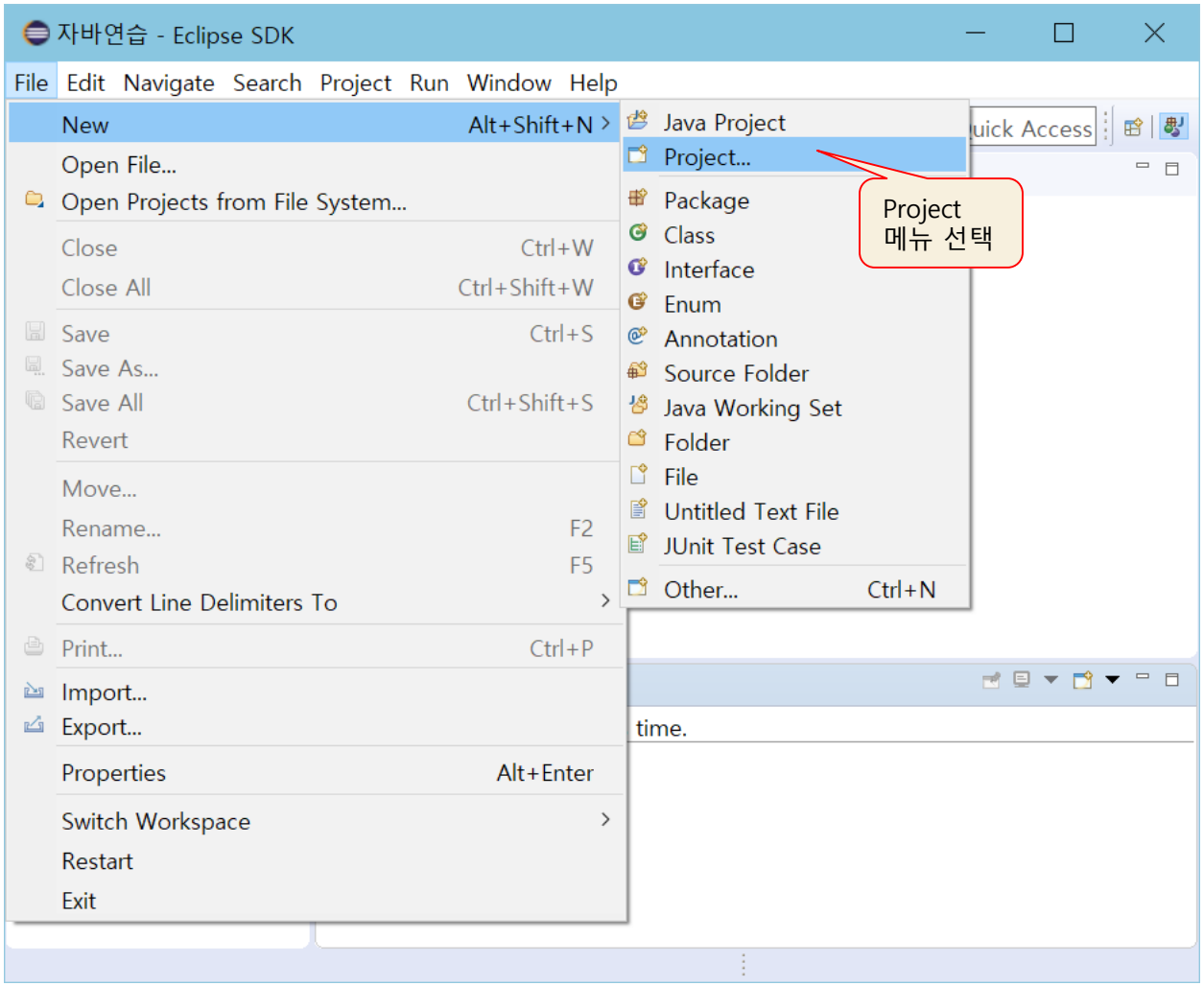
이클립스 Oxygen 배포판



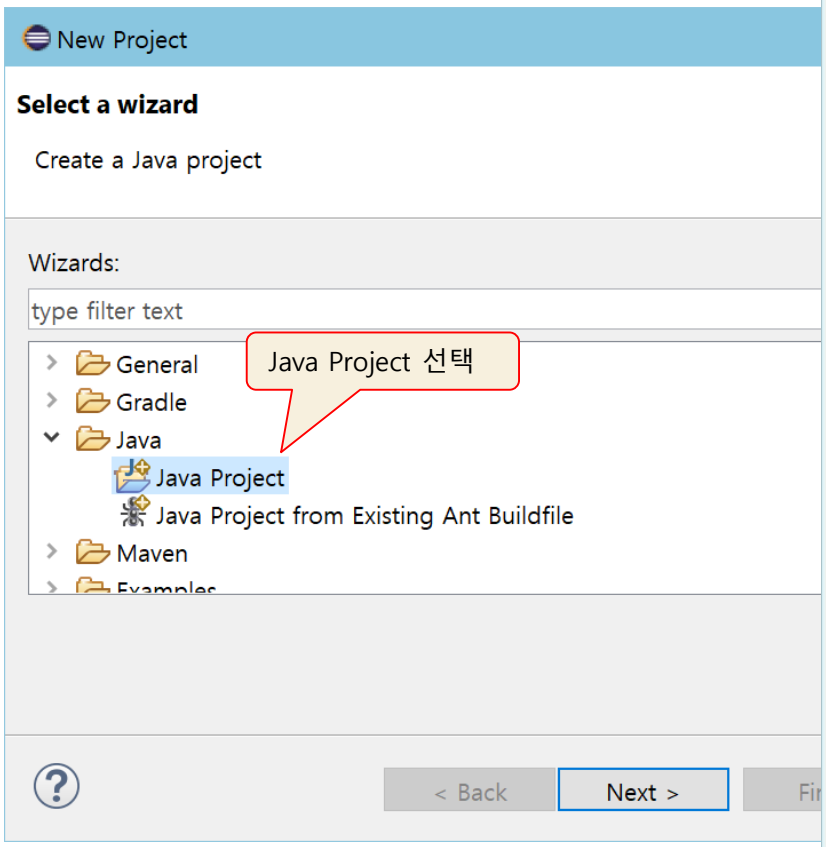
이클립스의 사용자 인터페이스

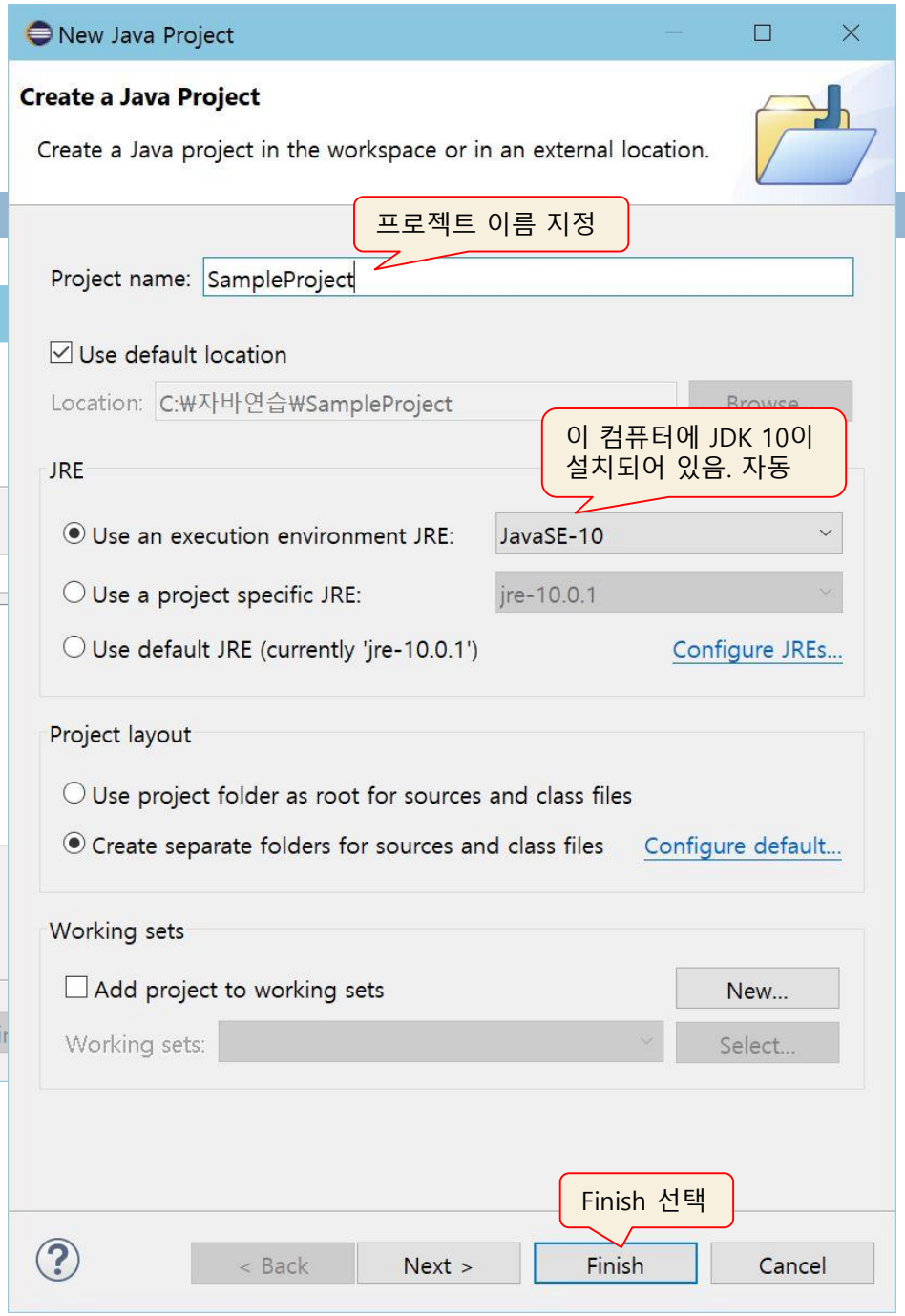


프로젝트 생성 메뉴


자바연습 - Eclipse SDK
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Other... Ctrl+N
Project
메뉴 선택
uick Access
time.

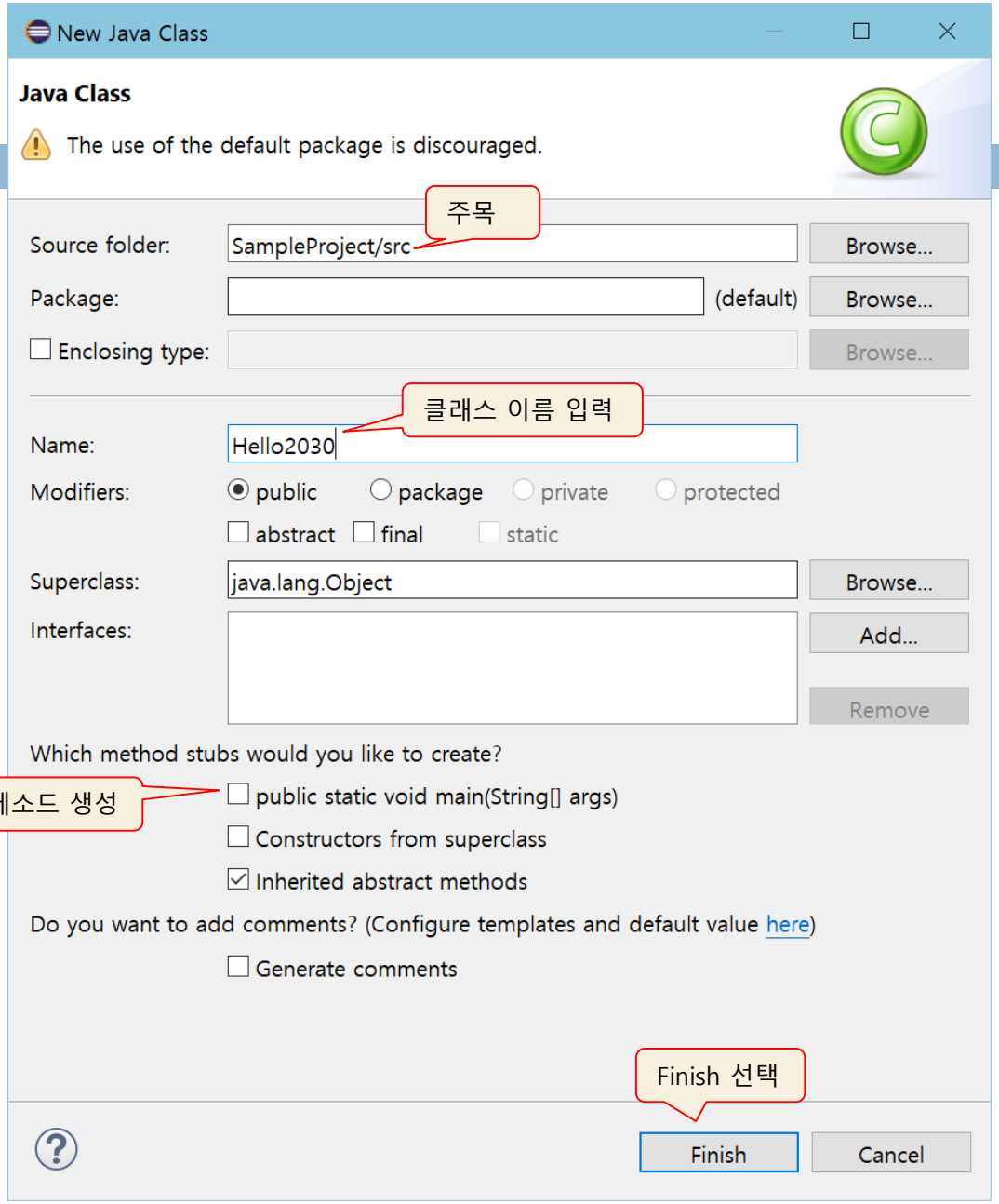
프로젝트 생성


New Project
Select a wizard
Create a Java project
Wizards:
type filter text
General
Gradle
Java
Java Project
Java Project from Existing Ant Buildfile
Maven
Examples
Java Project 선택
< Back
Next >


New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
프로젝트 이름 지정
Project name: SampleProject
Use default location
Location: C:₩자바연습₩SampleProject
Browse...
이 컴퓨터에 JDK 10이 설치되어 있음. 자동
JRE
Use an execution environment JRE: JavaSE-10
Use a project specific JRE: jre-10.0.1
Use default JRE (currently 'jre-10.0.1')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
New...
Working sets:
Select...
Finish 선택
< Back
Next >
Finish
Cancel

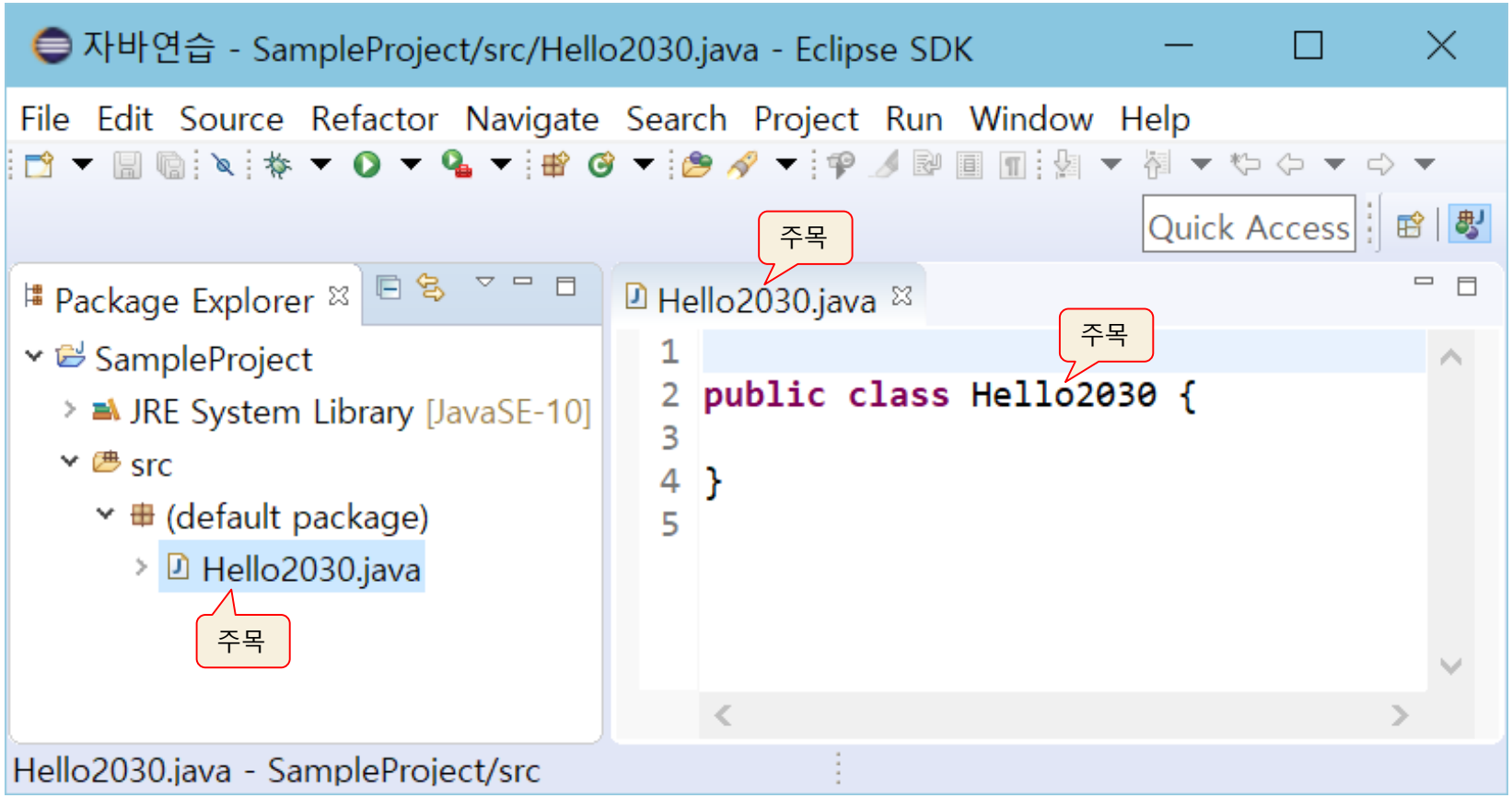
클래스 생성

File->New->Class 메뉴 선택

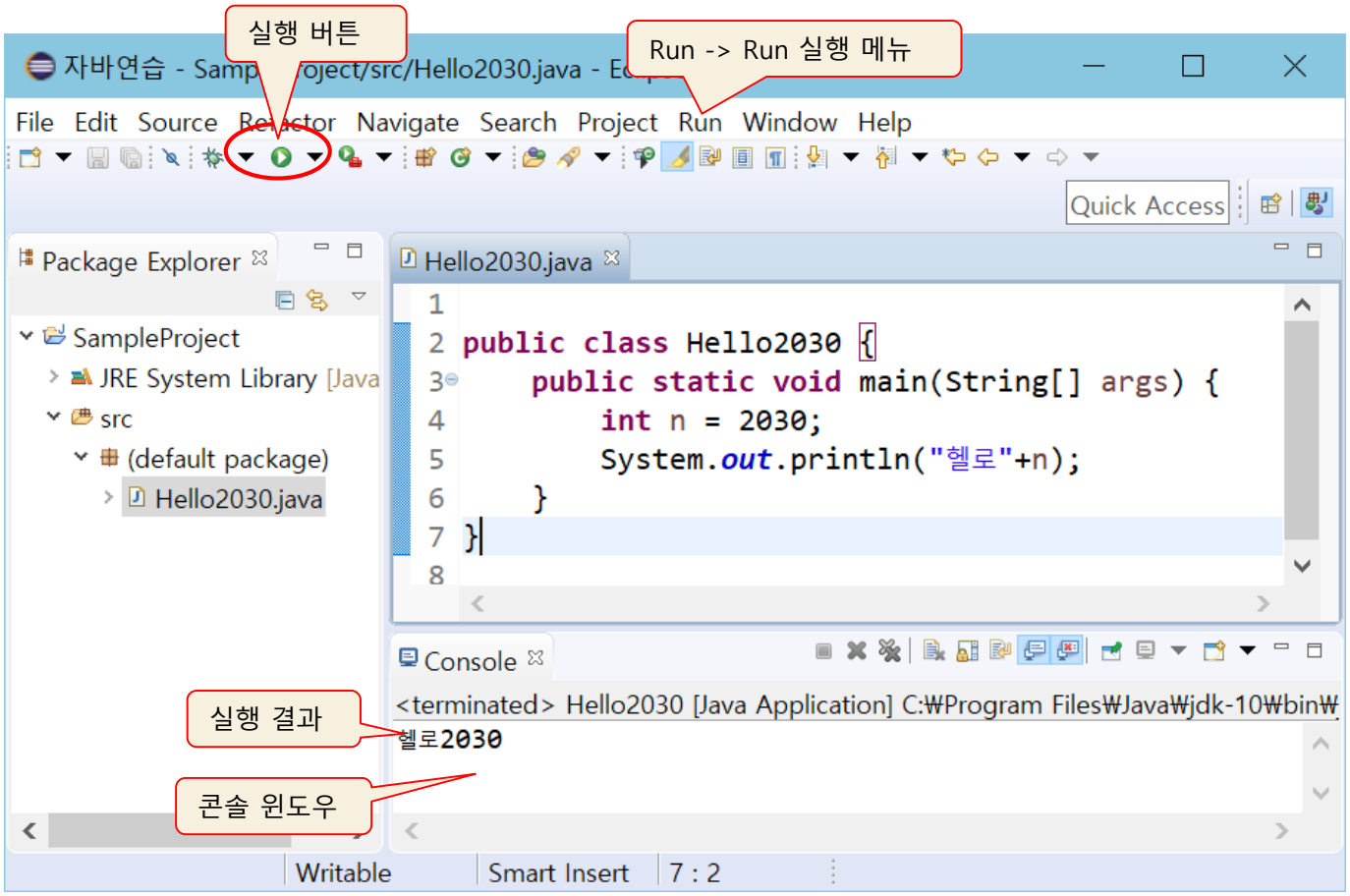


main()을 체크하면 자동으로 main() 메소드 생성

생성된 자바 소스

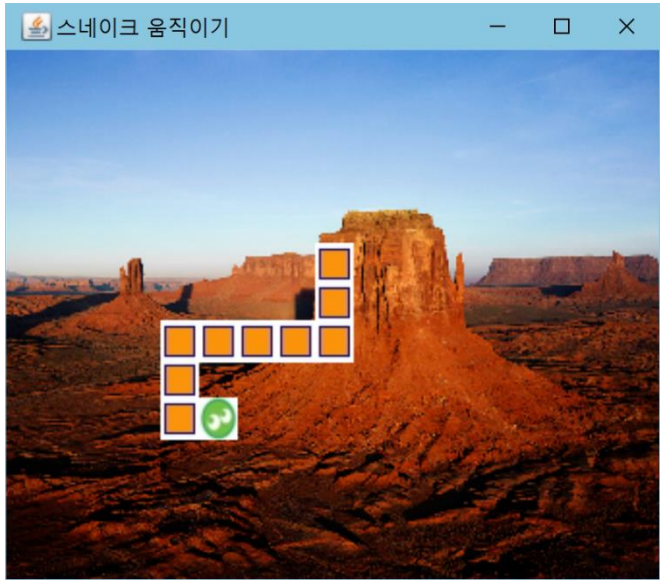

자바연습 - SampleProject/src/Hello2030.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
주목
Package Explorer
SampleProject
JRE System Library [JavaSE-10]
src
(default package)
Hello2030.java
주목
Hello2030.java
주목
1
2 public class Hello2030 {
3
4 }
5
Hello2030.java - SampleProject/src

소스 편집과 컴파일 및 실행



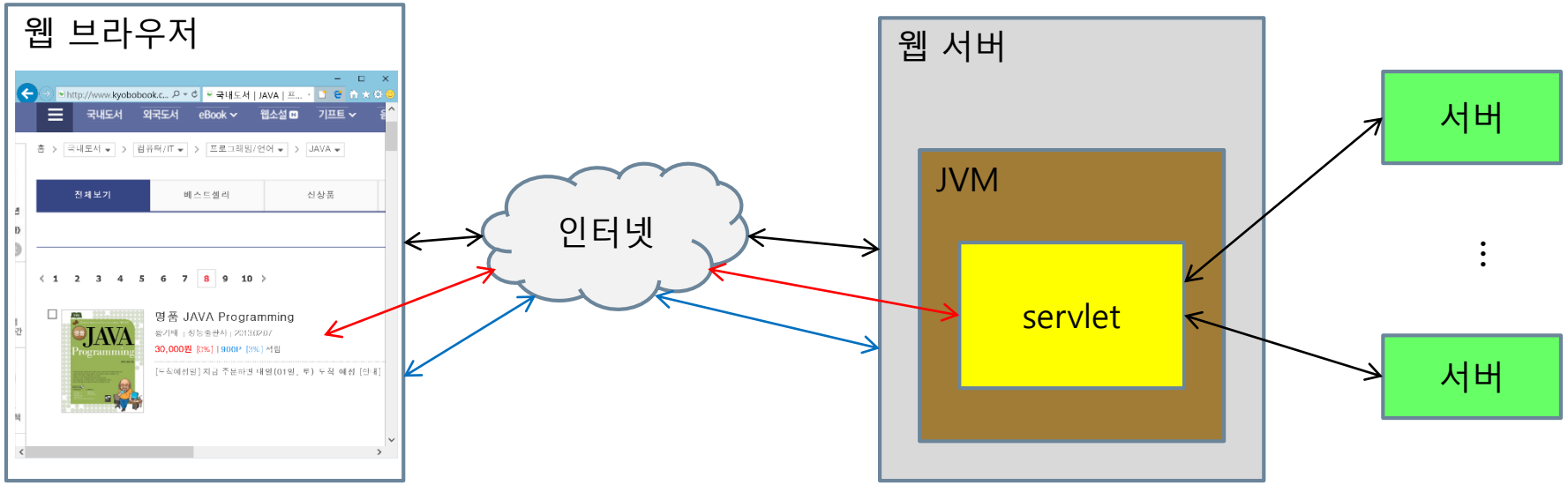
자바 응용의 종류 : 데스크톱 응용프로그램

- 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



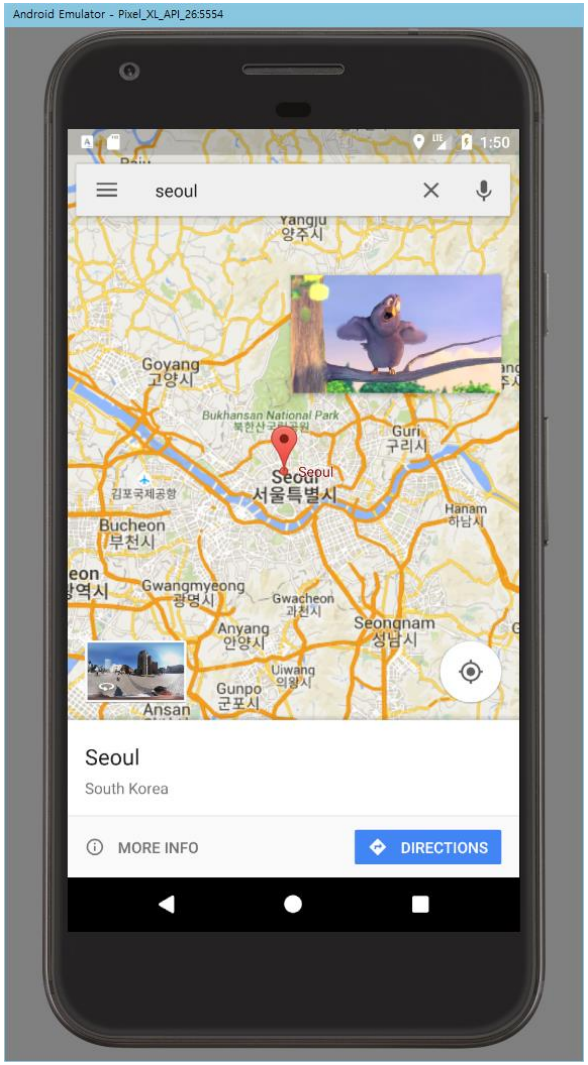
자바 응용의 종류 : 서블릿 응용프로그램

- 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

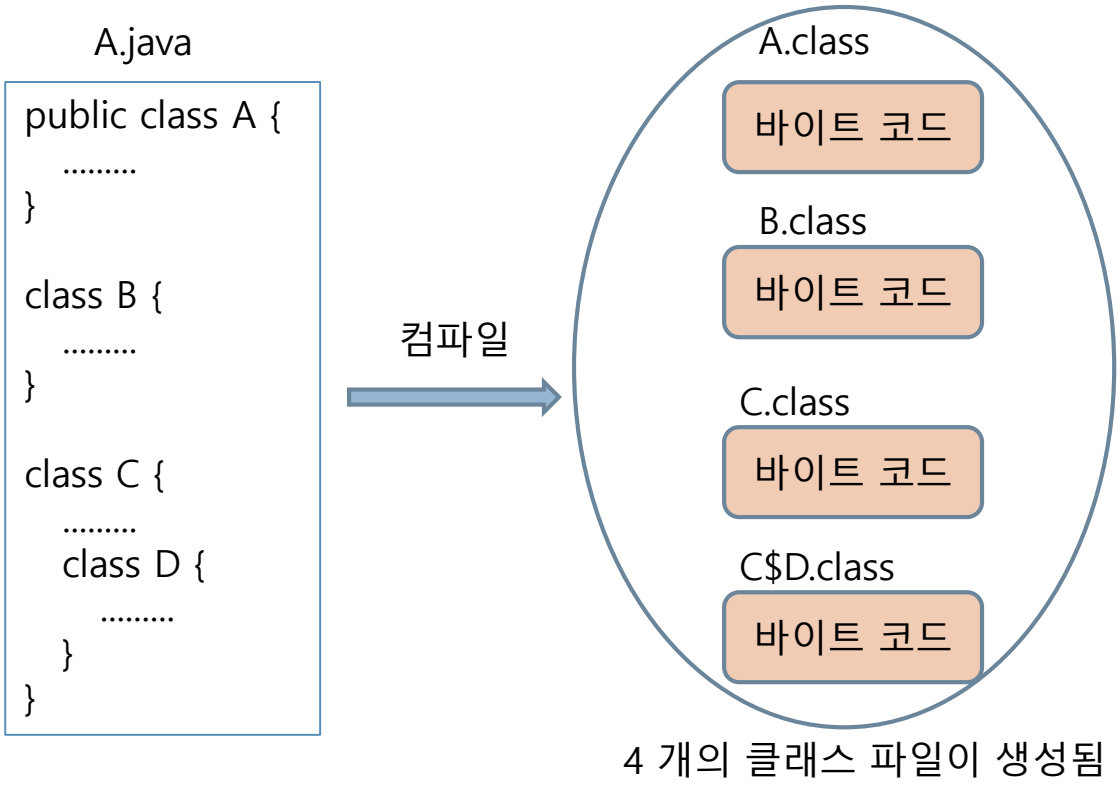
- 안드로이드
  - 구글의 주도로 여러 모바일 회사가 모여 구성한 OHA(Open Handset Alliance)에서 만든 무료 모바일 플랫폼
  - 개발 언어는 자바를 사용하나 JVM에 해당하는 Dalvik은 기존 바이트 코드와 호환성이 없어 변환 필요



자바의 특성(1)

- 플랫폼 독립성
  - 하드웨어, 운영체제에 종속되지 않는 바이트 코드로 플랫폼 독립성
- 객체지향
  - 캡슐화, 상속, 다형성 지원
- 클래스로 캡슐화
  - 자바의 모든 변수나 함수는 클래스 내에 선언
  - 클래스 안에서 클래스(내부 클래스) 작성 가능
- 소스(.java)와 클래스(.class) 파일
  - 하나의 소스 파일에 여러 클래스를 작성 가능
    - public 클래스는 하나만 가능
  - 소스 파일의 이름과 public으로 선언된 클래스 이름은 같아야 함
  - 클래스 파일에는 하나의 클래스만 존재
    - 다수의 클래스를 가진 자바 소스를 컴파일하면 클래스마다 별도 클래스 파일 생성

소스 파일과 클래스, 클래스 파일의 관계



자바의 특징(2)

- 실행 코드 배포
  - 구성
    - 한 개의 class 파일 또는 다수의 class 파일로 구성
    - 여러 폴더에 걸쳐 다수의 클래스 파일로 구성된 경우 : jar 압축 파일로 배포
  - 자바 응용프로그램의 실행은 main() 메소드에서 시작
    - 하나의 클래스 파일에 두 개 이상의 main() 메소드가 있을 수 없음
      - 각 클래스 파일이 main() 메소드를 포함하는 것은 상관없음
- 패키지
  - 서로 관련 있는 여러 클래스를 패키지로 묶어 관리
  - 패키지는 폴더 개념
    - 예) java.lang.System은 java₩lang 디렉터리의 System.class 파일
- 멀티스레드
  - 여러 스레드의 동시 수행 환경 지원
    - 자바는 운영체제의 도움 없이 자체적으로 멀티스레드 지원
    - C/C++ 프로그램은 멀티스레드를 위해 운영체제 API를 호출
- 가비지 컬렉션
  - 자바 언어는 메모리 할당 기능은 있어도 메모리 반환 기능 없음
    - 사용하지 않는 메모리는 자바 가상 기계에 의해 자동 반환 – 가비지 컬렉션

자바의 특징(3)

- 실시간 응용프로그램에 부적합
  - 실행 도중 예측할 수 없는 시점에 가비지 컬렉션 실행 때문
    - 응용프로그램의 일시적 중단 발생
- 자바 프로그램은 안전
  - 타입 체크 엄격
  - 물리적 주소를 사용하는 포인터 개념 없음
- 프로그램 작성 쉬움
  - 포인터 개념이 없음
  - 동적 메모리 반환 하지 않음
  - 다양한 라이브러리 지원
- 실행 속도 개선을 위한 JIT 컴파일러 사용
  - 자바는 바이트 코드를 인터프리터 방식으로 실행
    - 기계어가 실행되는 것보다 느림
  - JIT 컴파일 기법으로 실행 속도 개선
    - JIT 컴파일 – 실행 중에 바이트 코드를 기계어 코드로 컴파일하여 기계어를 실행하는 기법