

참고하세요 – 2022년 3월 인쇄본의 수정 사항입니다.

1

□ 이클립스의 최신 2022-12월 버전의 화면으로 갱신

- 1.5절의 이클립스를 사용한 프로그램 개발 과정
- 6.2절의 패키지 만들기
- 기존 화면과 거의 차이 없음

□ 실습 때 Open JDK 사용 권함

- 2018년도부터 오라클의 JDK 유료화 정책으로 대학에서 라이선스 없이 Oracle JDK를 사용하는 것이 불법
- 오라클에서는 Oracle JDK 뿐 아니라, 무료로 사용할 수 있는 Open JDK 소스 코드 공개
- Open JDK는 몇 가지 기능을 제외하고 Oracle JDK와 완전 동일
- 개발자 입장에서 Open JDK는 기존 JDK와 별 차이를 느낄 수 없음

■ 생능출판사 홈페이지에서 "OpenJDK와 이클립스 설치.pdf" 제공

- Microsoft 사에서 OpenJDK 16을 컴파일한 바이너리를 다운받고 활용하는 방법 설명

■ 1장과 6장 등에서 JDK를 설명할 때 Open JDK를 함께 설명하며, 몇몇 그림에는 Microsoft의 Open JDK 16을 설치한 모양을 캡처하여 삽입하였음

□ 갱신된 그림 리스트

- 그림 1-2, 1-9, 1-10, 1-11, 1-15~22, 6-3, 6-5, 6-15, 6-16, 7-3, 7-10

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

4



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

프로그래밍 언어

5

□ 프로그래밍 언어

▣ 프로그램 작성 언어

▣ 기계어(machine language)

- 0, 1의 이진수로 구성된 언어
- 컴퓨터의 CPU는 기계어만 이해하고 처리가능

▣ 어셈블리어

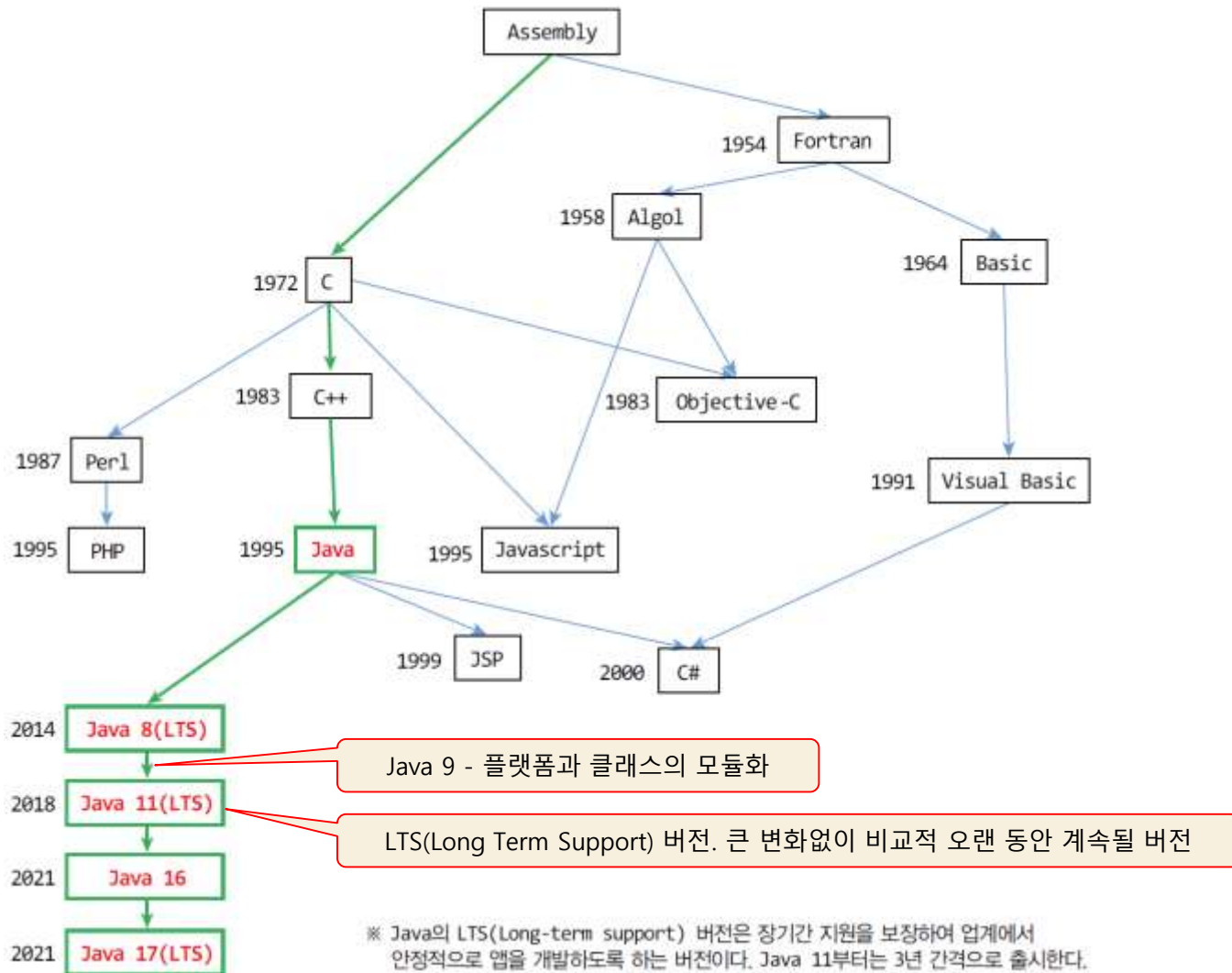
- 기계어 명령을 ADD, SUB, MOVE 등과 같은 표현하기 쉬운 상징적인 단어인 니모닉 기호(mnemonic symbol)로 일대일 대응시킨 언어

▣ 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

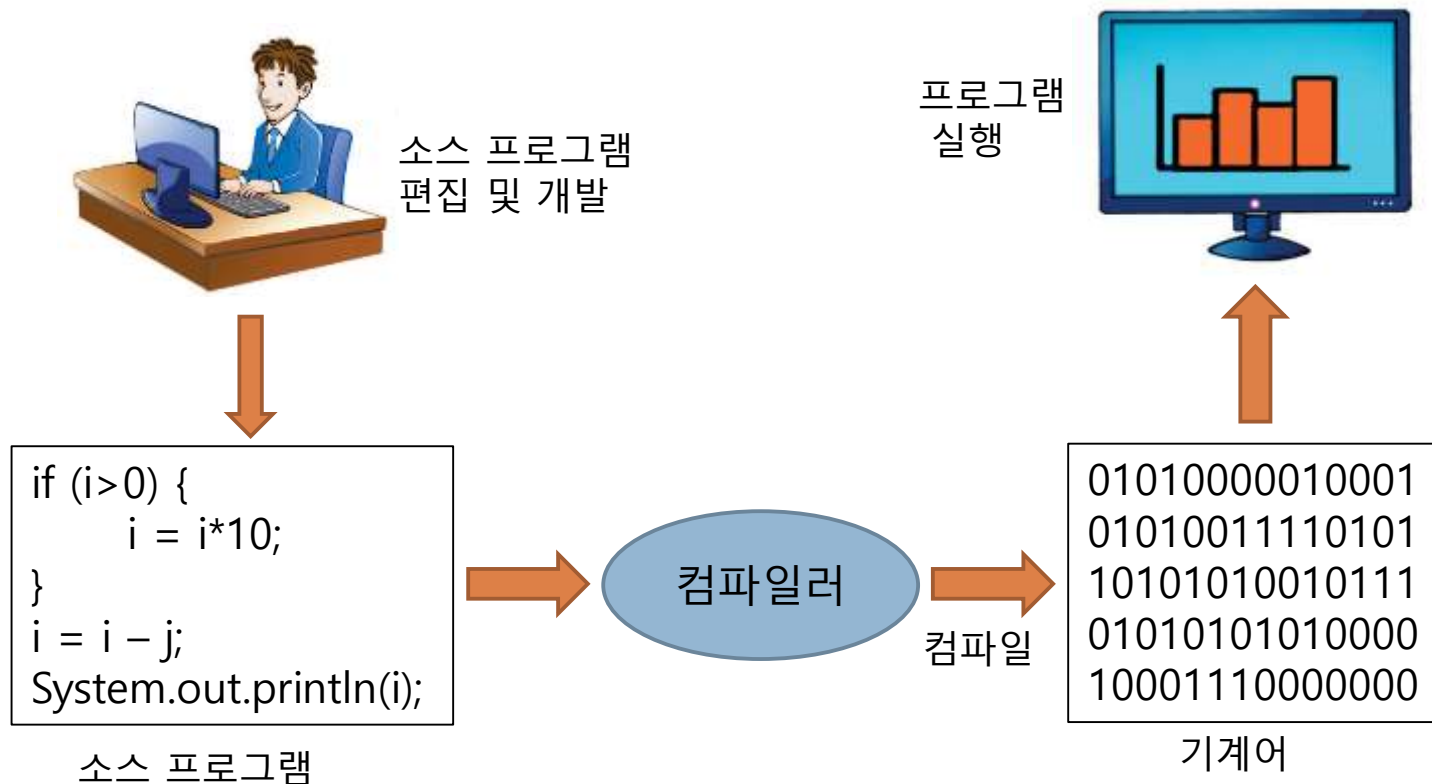
6



프로그래밍과 컴파일

7

- 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**



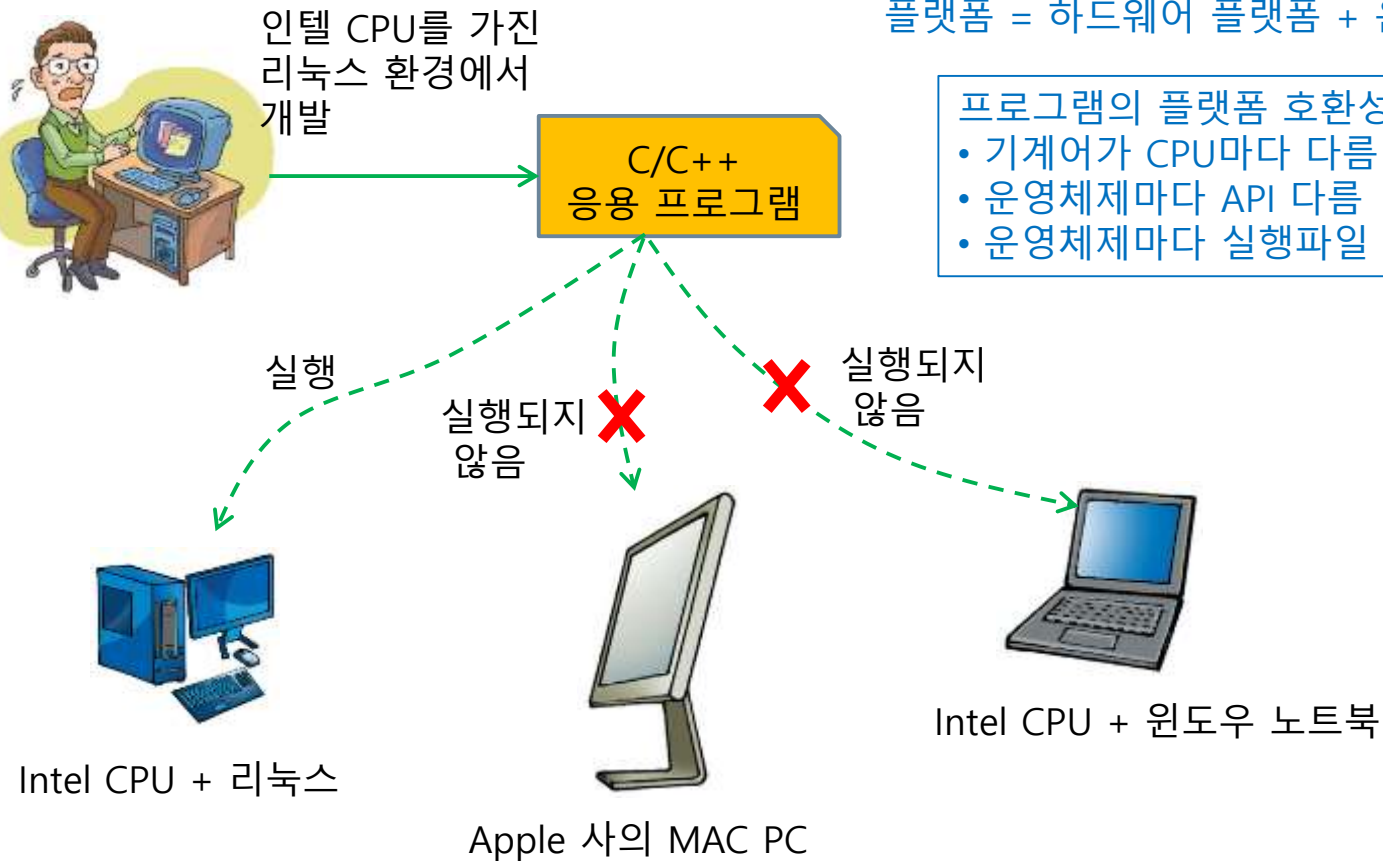
자바의 태동

8

- 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

9



자바의 플랫폼 독립성, WORA

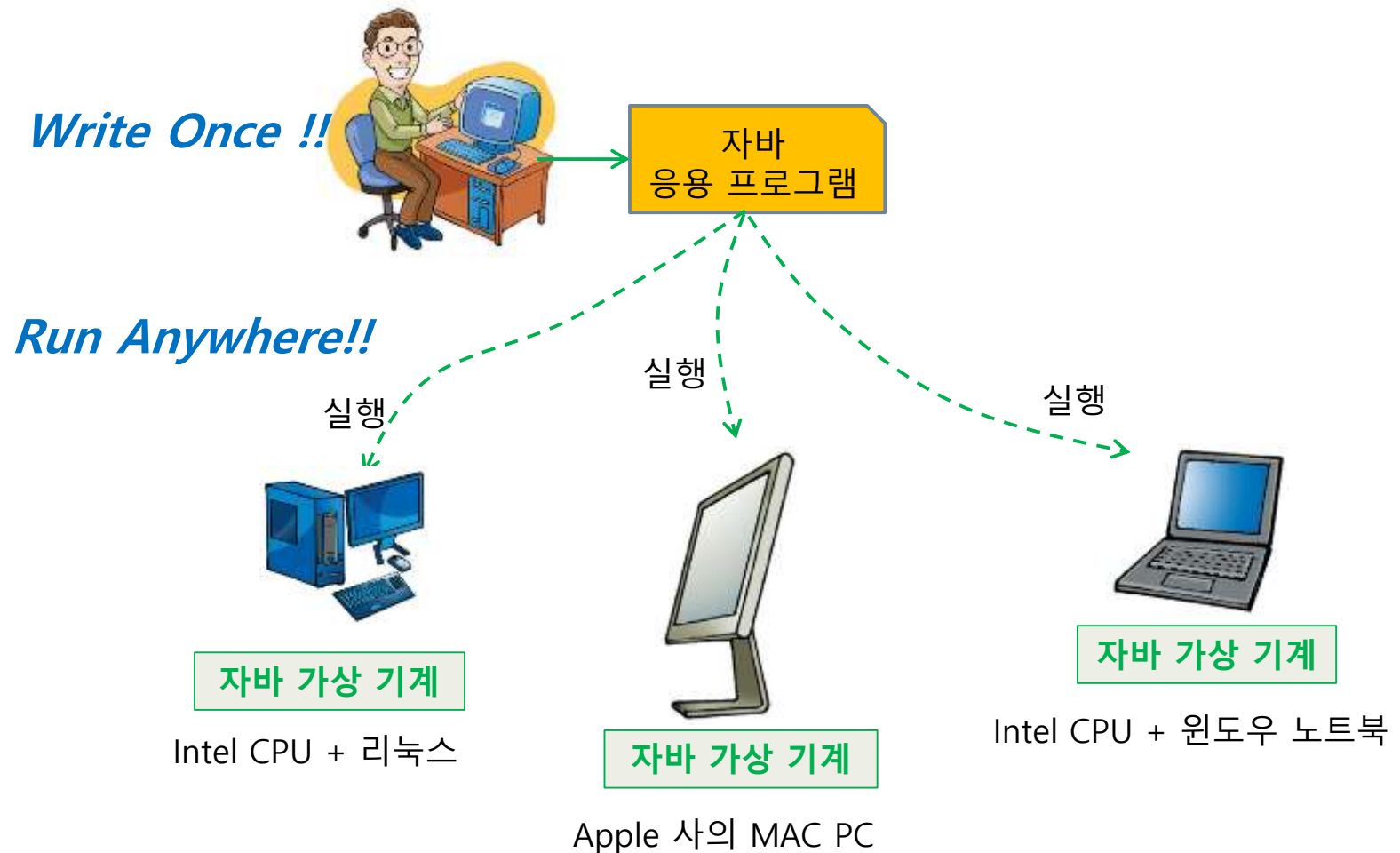
10

- 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

11



자바 가상 기계와 자바 실행 환경

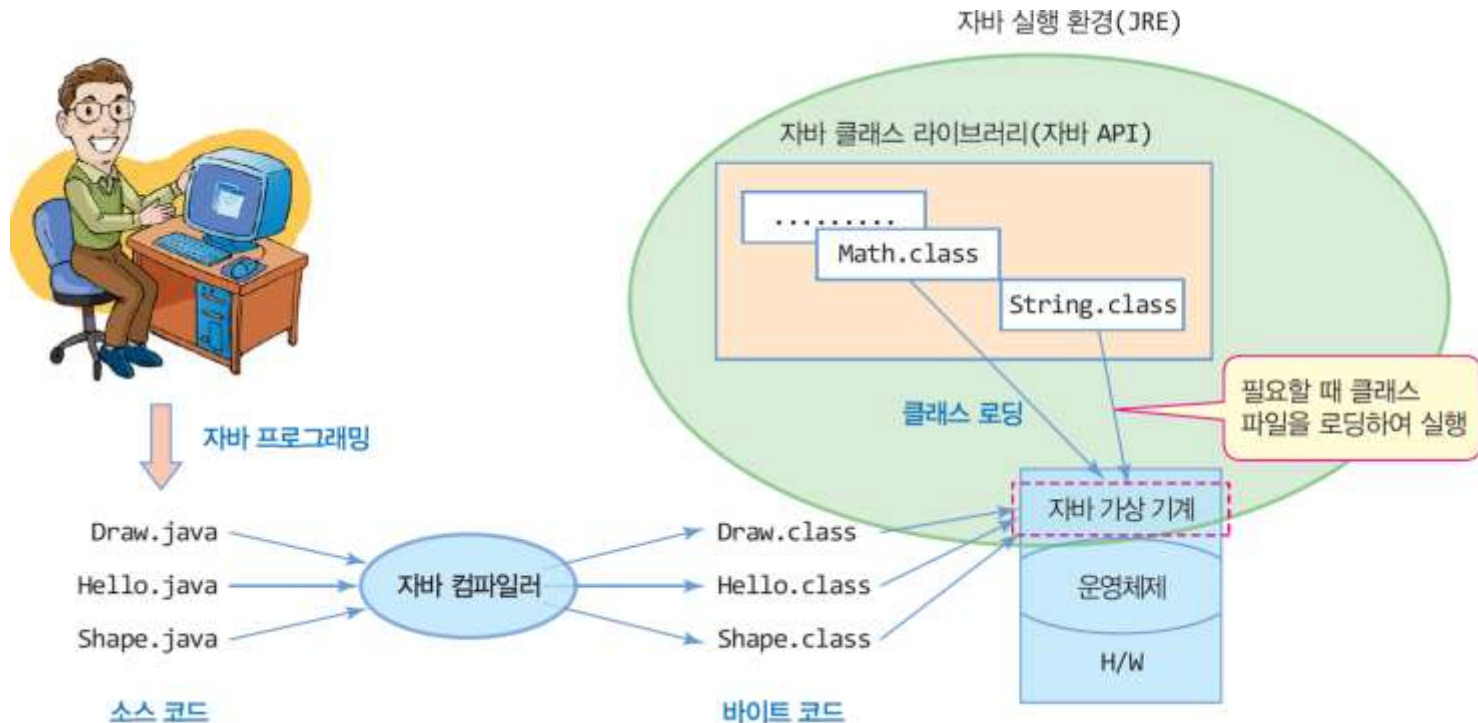
12

- 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

13

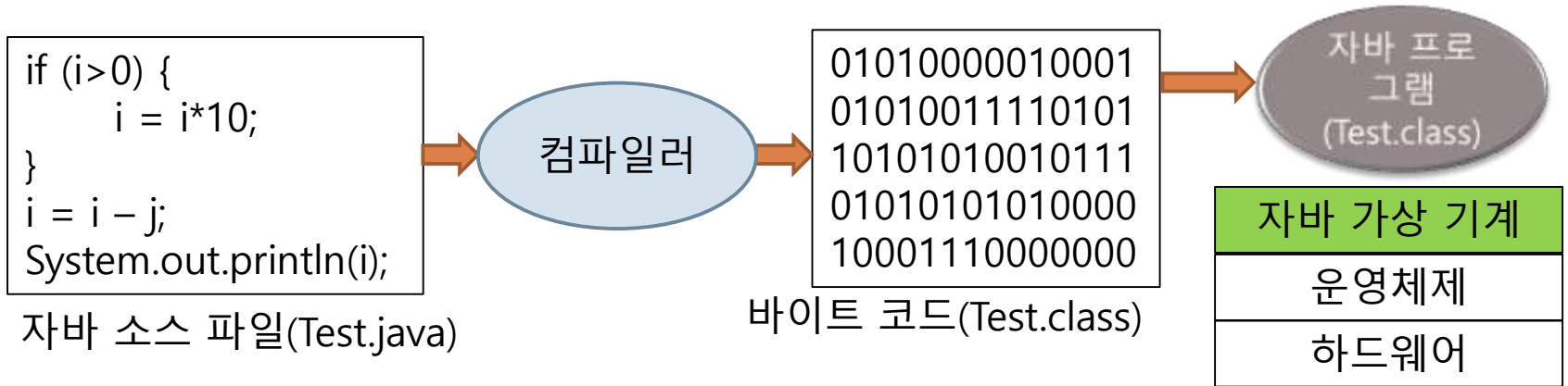
- 실행 환경
 - ▣ 자바 가상 기계 + 자바 플랫폼의 다양한 클래스 라이브러리(자바 API)
- 응용프로그램 실행
 - ▣ main() 메소드를 가진 클래스의 main()에서 실행 시작
 - ▣ 자바 가상 기계는, 필요할 때 클래스 파일 로딩, 적은 메모리로 실행 가능



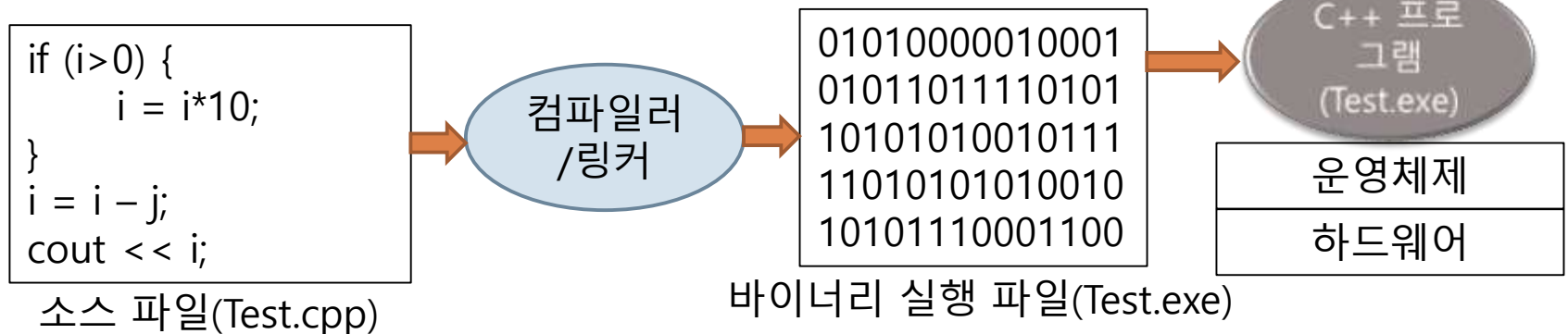
자바와 타언어(C/C++)의 실행 차이

14

□ 자바



□ C/C++



JDK와 JRE

15

- 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(JVM과 컴파일된 자바 API 들이 들어 있는 모듈 파일들)
 - ▣ 개발자가 아닌 경우 JRE만 따로 다운 가능
- JDK와 JRE의 개발 및 배포
 - ▣ 오라클의 Technology Network의 자바 사이트에서 다운로드
 - <http://www.oracle.com/java/technologies/>
- 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

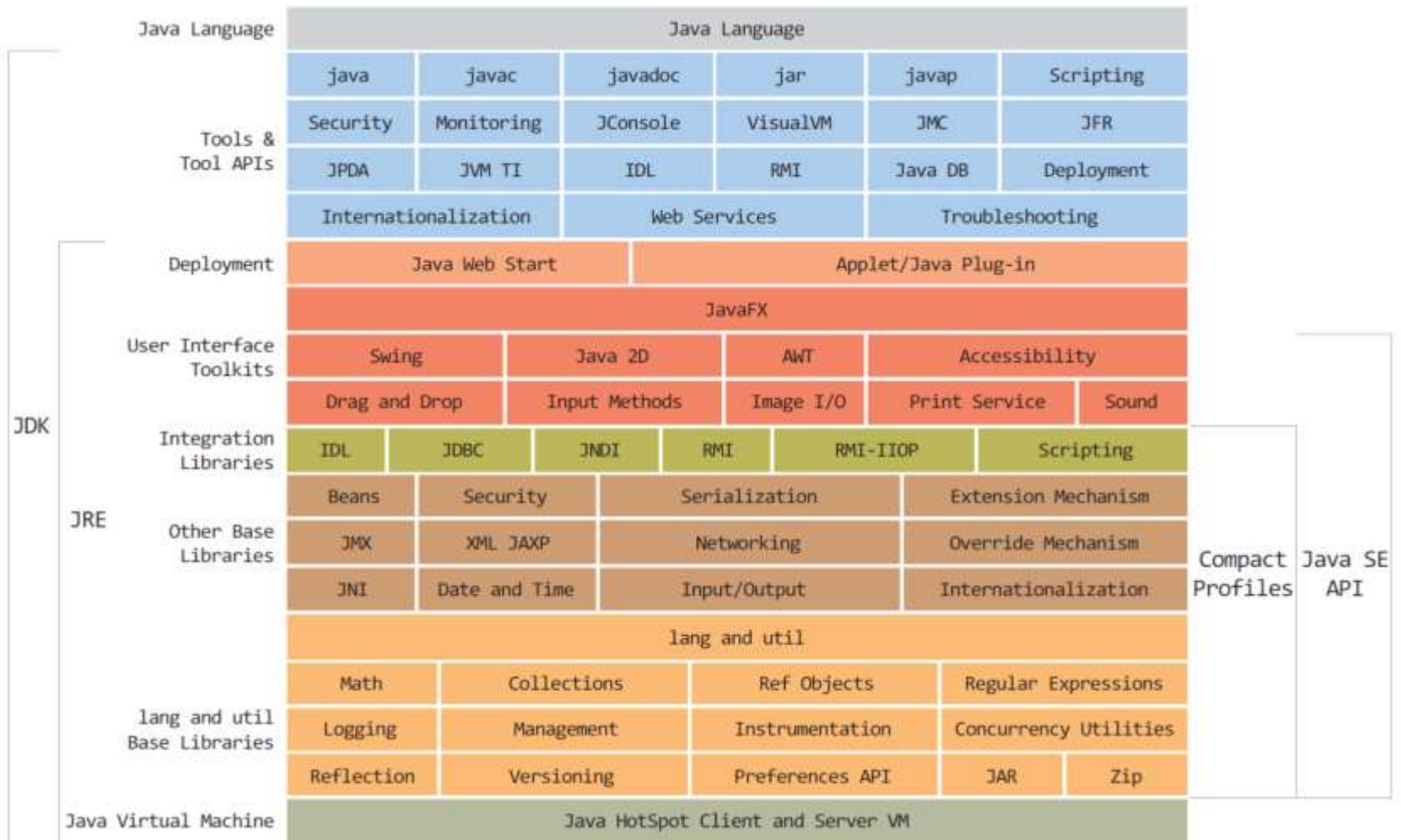
JDK의 유료 버전과 무료 버전

16

- JDK는 유료 버전과 무료 버전으로 분리
- Oracle JDK – 유료
 - ▣ 2018년 7월부터 유료화(개인 사용자의 경우 무료)
 - ▣ 학교 등 기관에서는 라이선스 없이 사용불가
- **Open JDK – 무료**
 - ▣ 학교 강의에서 활용 가능
 - ▣ 오라클에서 제공하며 Oracle JDK와 동일한 소스 코드
 - Oracle JDK와 사용 방법, GUI 등 모든 것이 동일
 - 다만, Oracle JDK에서 몇몇 기능 빠짐
 - ▣ Open JDK를 다운받을 수 있는 사이트 많음
 - ▣ 이 책에서는 Microsoft가 제공하는 OpenJDK 활용
 - 생능출판사의 홈페이지에서 Microsoft의 OpenJDK를 다운/설치 방법이 담긴 문서를 다운받을 수 있음(OpenJDK와 이클립스 설치.pdf)

Java SE 구성

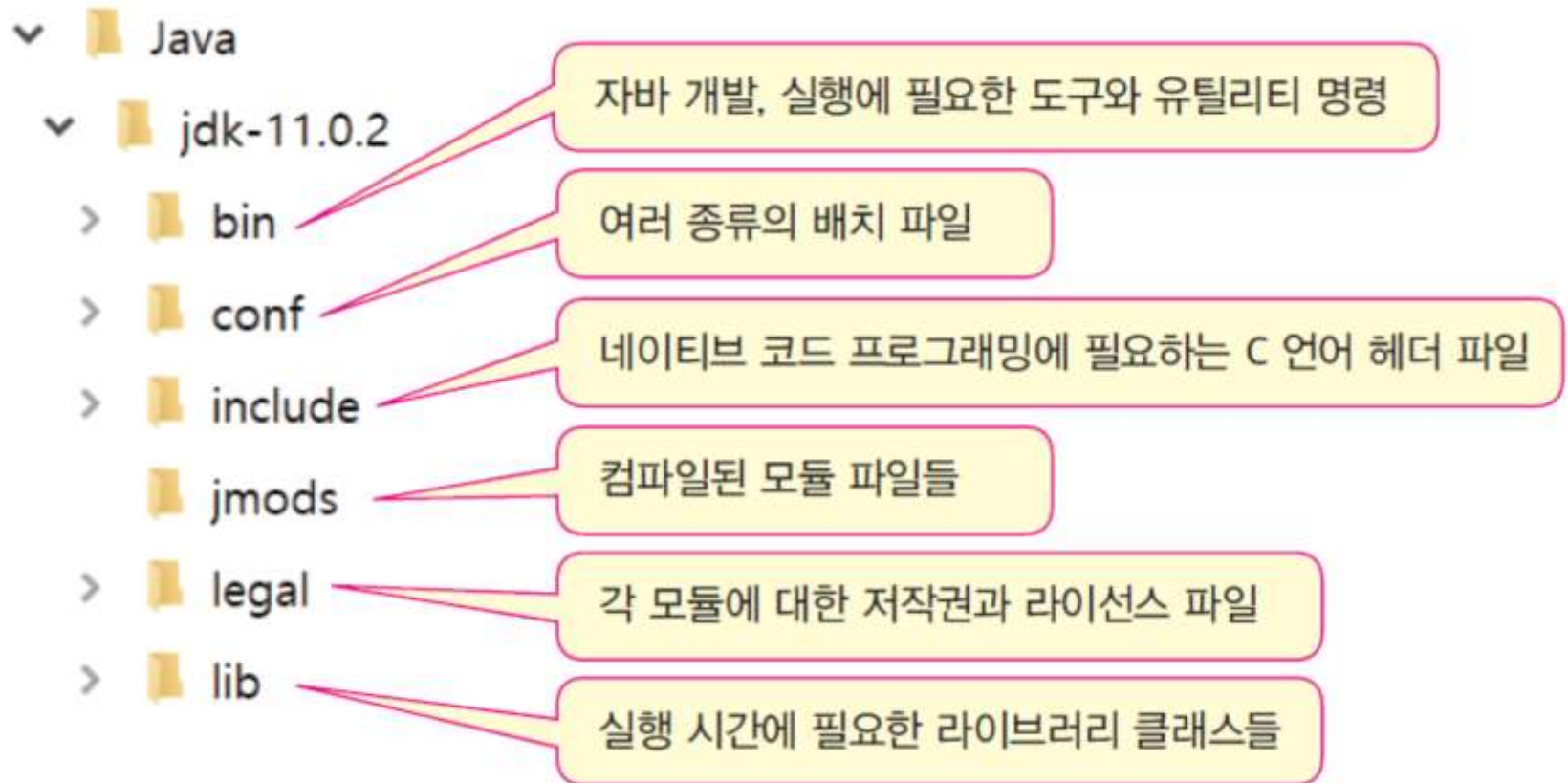
17



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

Oracle JDK 설치(JDK11) 후 디렉터리 구조

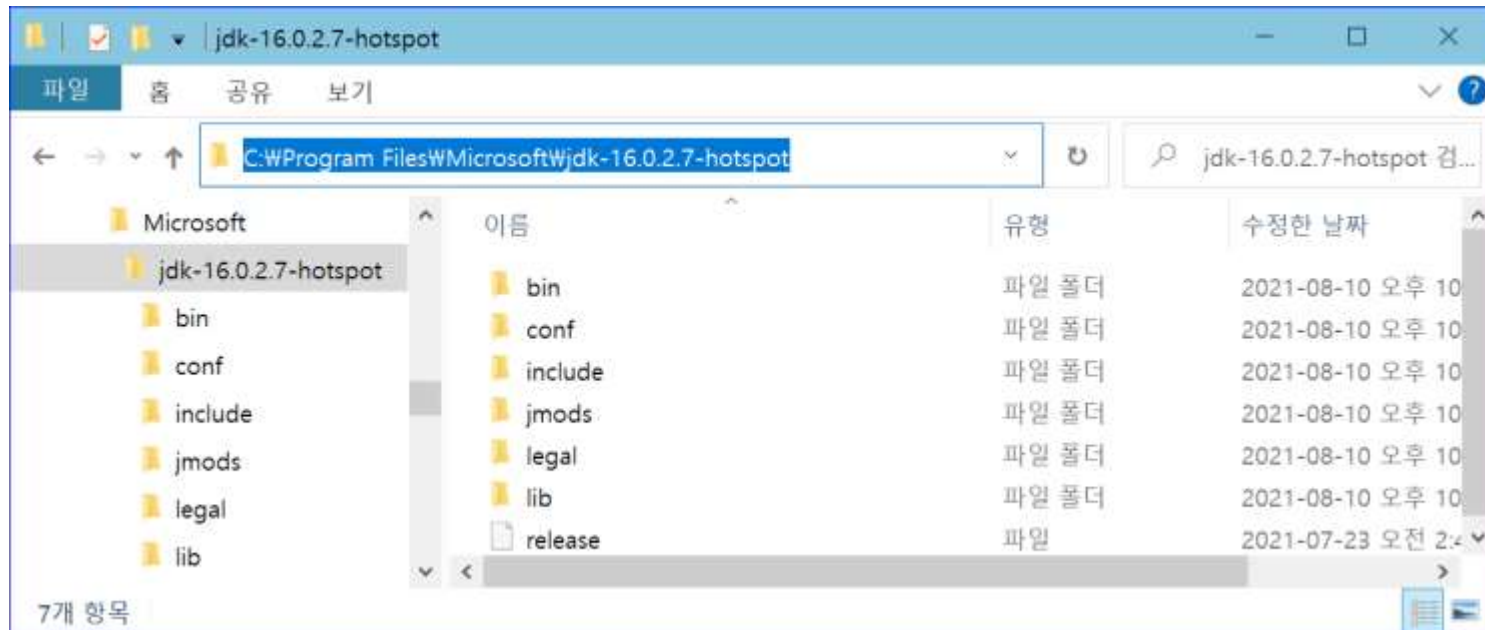
18



Microsoft 사의 OpenJDK 설치 후 디렉터리

19

- Microsoft사의 OpenJDK JavaSE16을 설치한 상태
 - ▣ Microsoft사의 OpenJDK 이름은 jdk-16.0.2.7-hotspot



Oracle JDK를 설치한 상태와 동일함

자바의 배포판 종류

20

- 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?

21



(사진 출처 : 위키 백과)

Java 9부터 시작된 모듈 프로그래밍

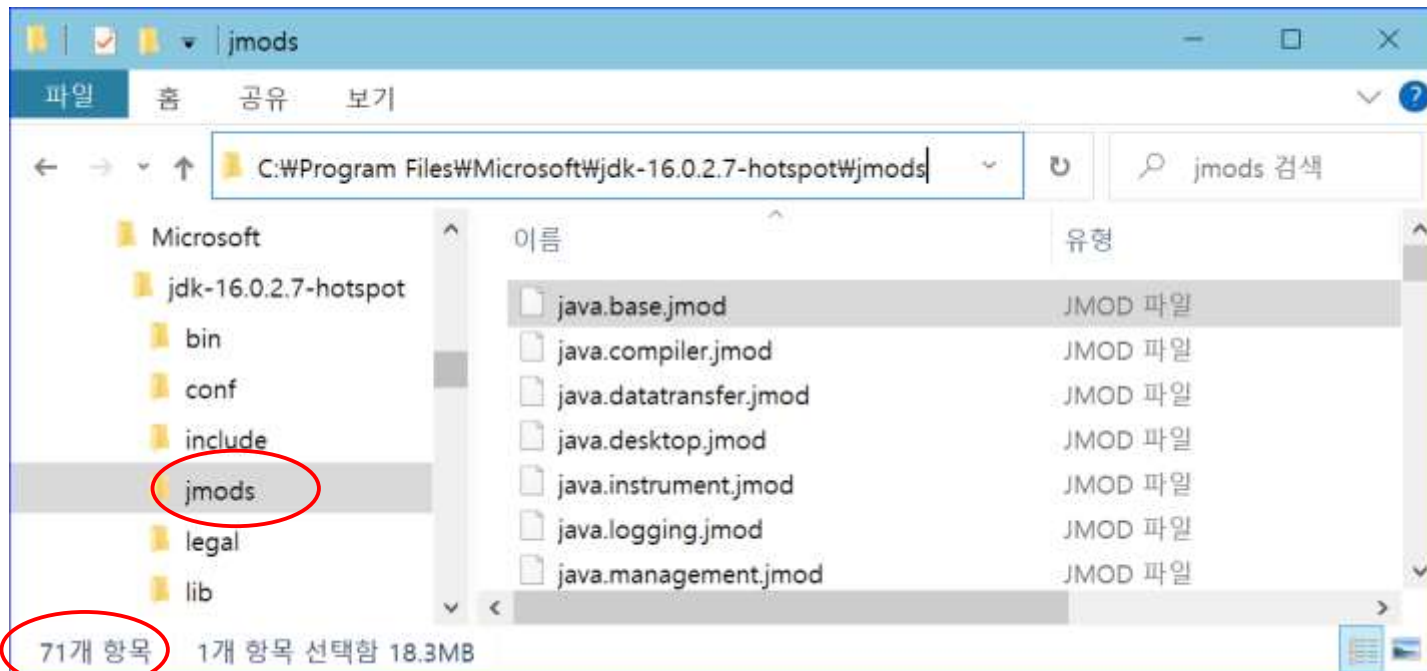
22

- 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)

23

- Java 9부터 플랫폼을 모듈화함
 - ▣ Java SE의 모든 클래스들을 모듈들로 재구성
 - ▣ JDK의 설치 디렉터리 밑의 jmods 디렉터리에 있음
 - 다음은 Microsoft사 OpenJDK, jdk-16.0.2.7-hotspot에 담긴 모듈 사례



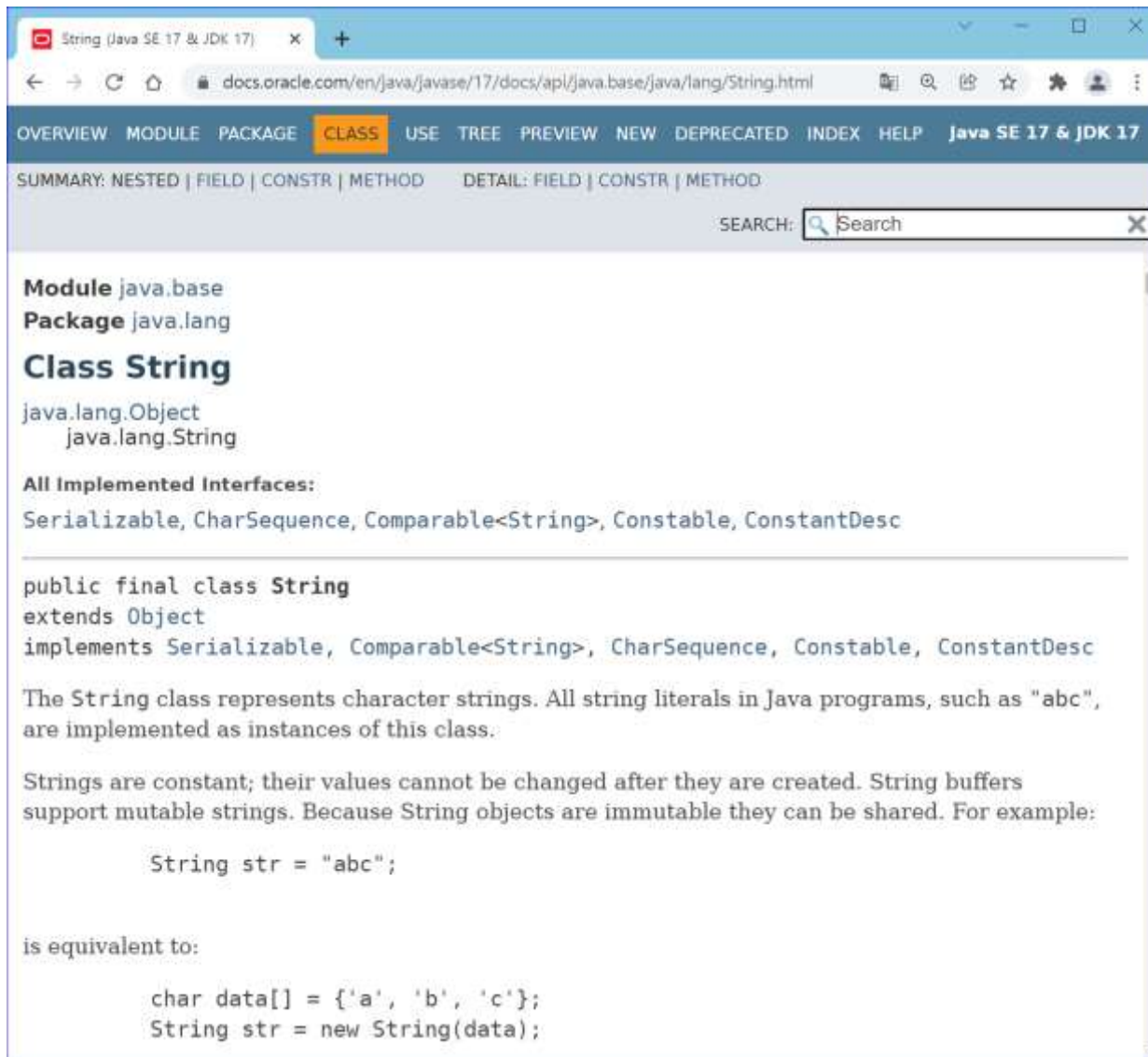
모듈 파일들

OpenJDK 16에서 자바 API의 클래스 패키지를 71개의 모듈 파일에 분산

- 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

25



The screenshot shows the Oracle Java SE 17 & JDK 17 API documentation for the `String` class. The browser address bar shows the URL `docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html`. The navigation bar includes tabs for OVERVIEW, MODULE, PACKAGE, CLASS (selected), USE, TREE, PREVIEW, NEW, DEPRECATED, INDEX, and HELP. The page title is "String (Java SE 17 & JDK 17)".

Module `java.base`
Package `java.lang`
Class `String`

`java.lang.Object`
`java.lang.String`

All Implemented Interfaces:
`Serializable`, `CharSequence`, `Comparable<String>`, `Constable`, `ConstantDesc`

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence, Constable, ConstantDesc
```

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:

```
String str = "abc";
```

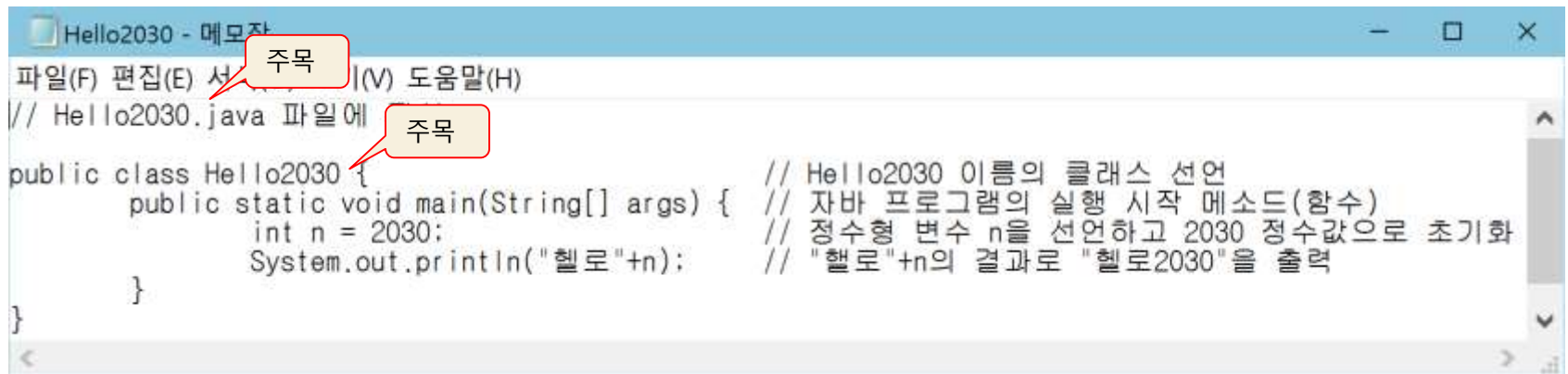
is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

자바 프로그램 개발 : (1) 자바 소스 편집

26

- 어떤 편집기를 사용해도 무관
 - ▣ 메모장으로 작성한 샘플



```
파일(F) 편집(E) 서... (V) 도움말(H)
// Hello2030.java 파일에
public class Hello2030 {
    public static void main(String[] args) {
        int n = 2030;
        System.out.println("헬로"+n);
    }
}

// Hello2030 이름의 클래스 선언
// 자바 프로그램의 실행 시작 메소드(함수)
// 정수형 변수 n을 선언하고 2030 정수값으로 초기화
// "헬로"+n의 결과로 "헬로2030"을 출력
```

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

자바 프로그램 개발 : (2) 컴파일 및 실행

27

□ 컴파일

Hello2030.java
는 C:\WTemp에
저장되어 있음

```
명령 프롬프트
C:\W>cd WTemp
C:\Wtemp>javac Hello2030.java
C:\Wtemp>dir Hello2030.*
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 4443-1630

C:\Wtemp 디렉터리

2017-06-23 오전 10:35        629 Hello2030.class
2017-06-23 오전 10:33        334 Hello2030.java
                        2개 파일           963 바이트
                        0개 디렉터리 120,858,402,816 바이트 남음

C:\Wtemp>
```

C:\WTemp 디렉터리로 이동

컴파일

클래스 파일 생성

□ 실행

Hello2030.class
실행

```
명령 프롬프트
C:\Wtemp>java Hello2030
헬로2030
C:\Wtemp>
```

.class 확장자를 붙이지 않는다.

실행 결과

자바 통합 개발 환경-이클립스(Eclipse)

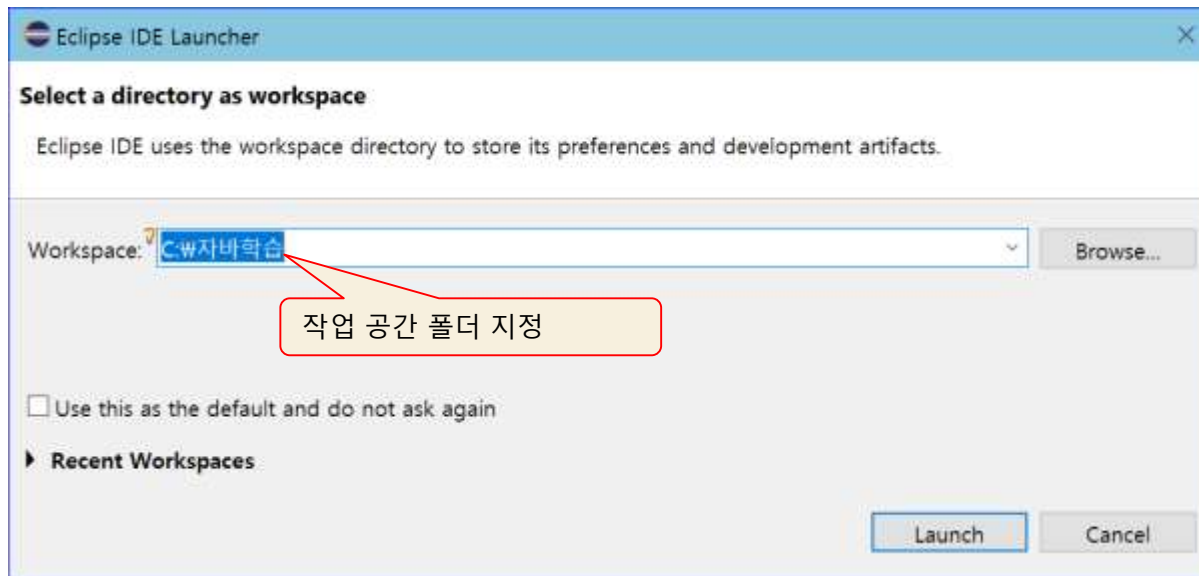
28

- 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

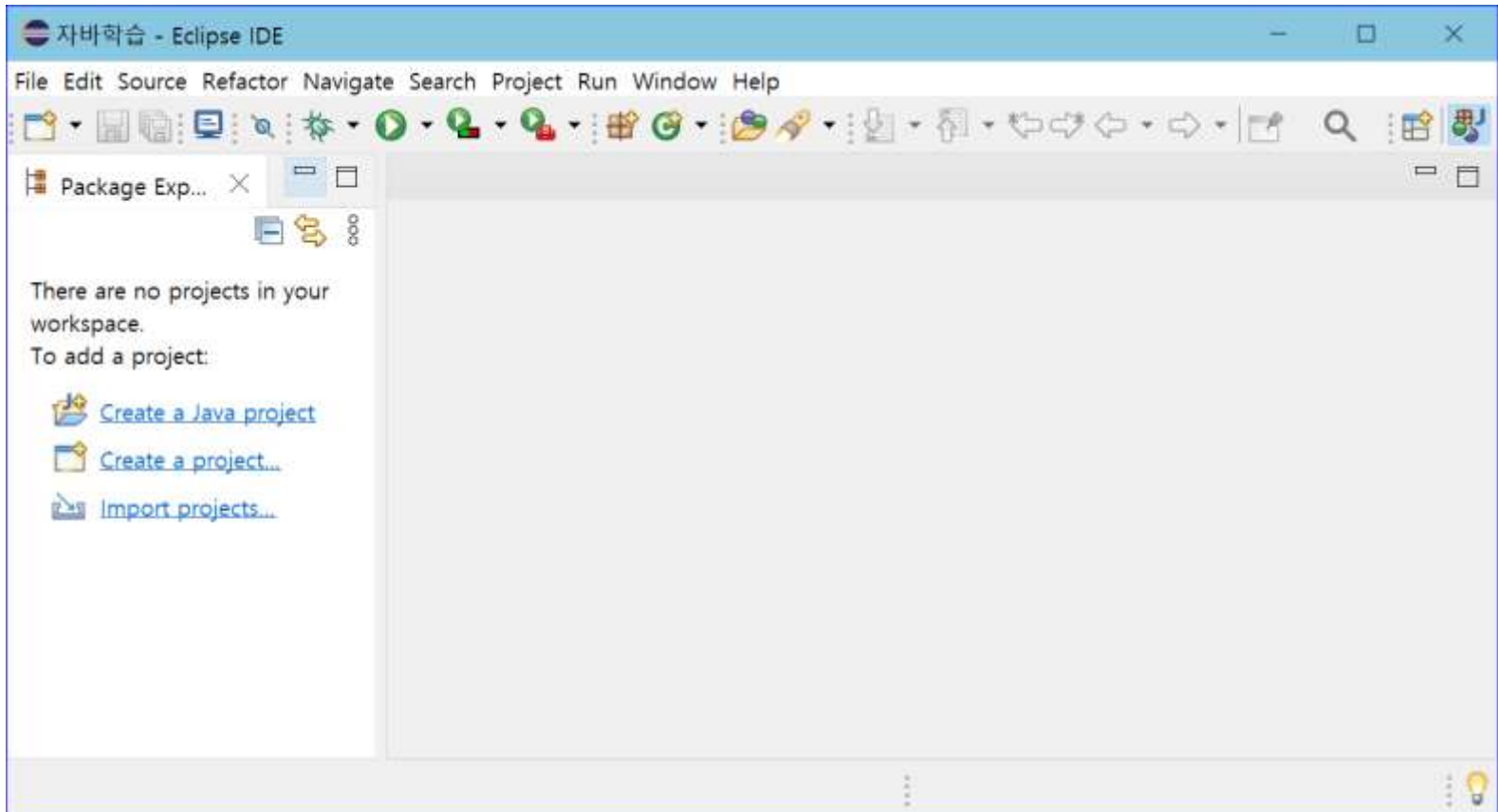
이클립스 실행

29



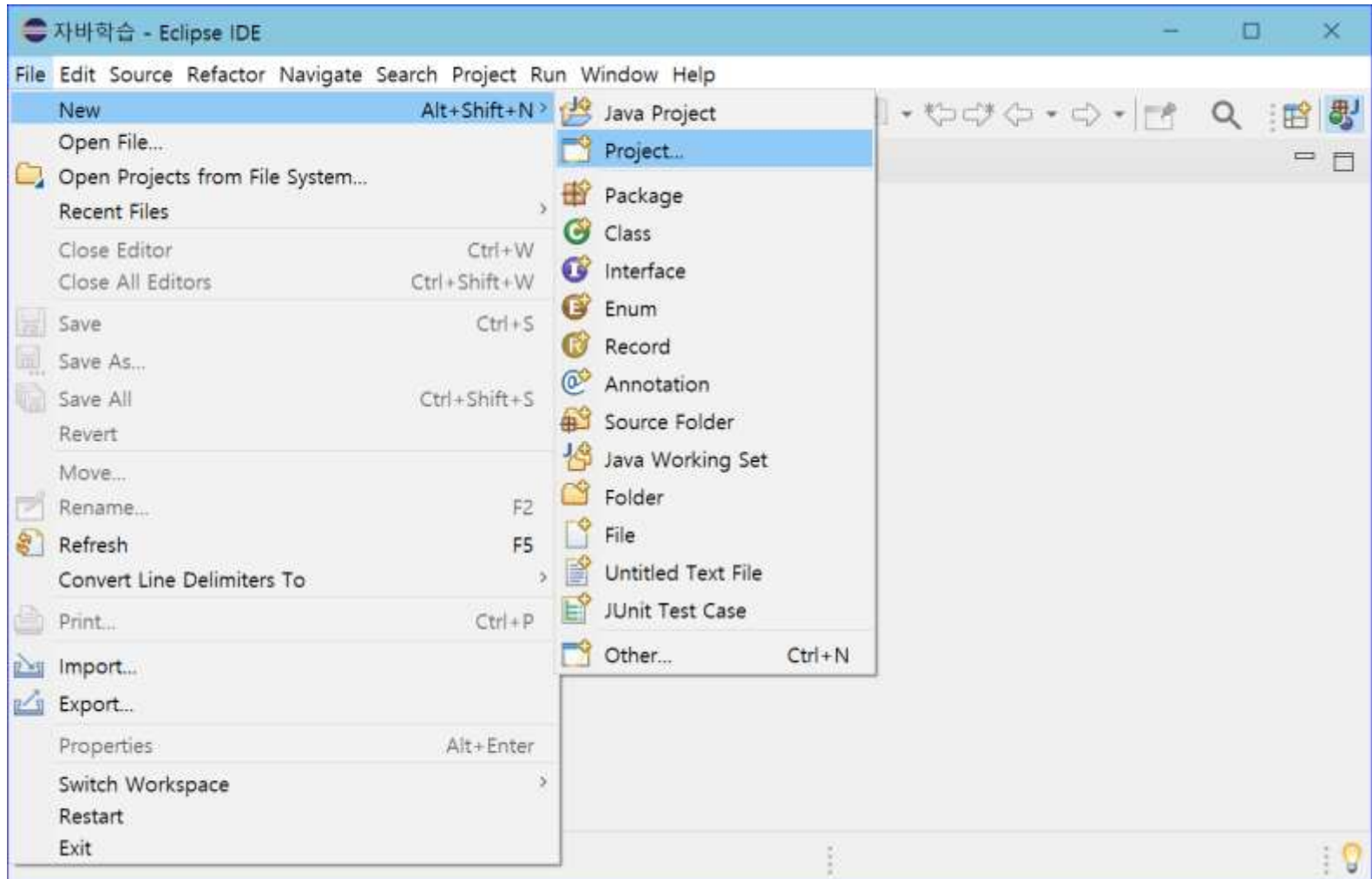
이클립스의 사용자 인터페이스

30



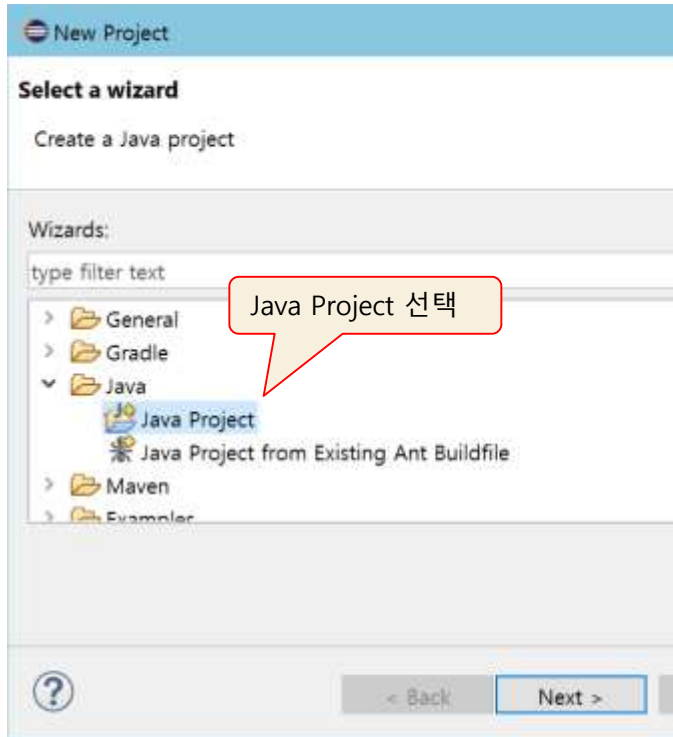
프로젝트 생성 메뉴

31

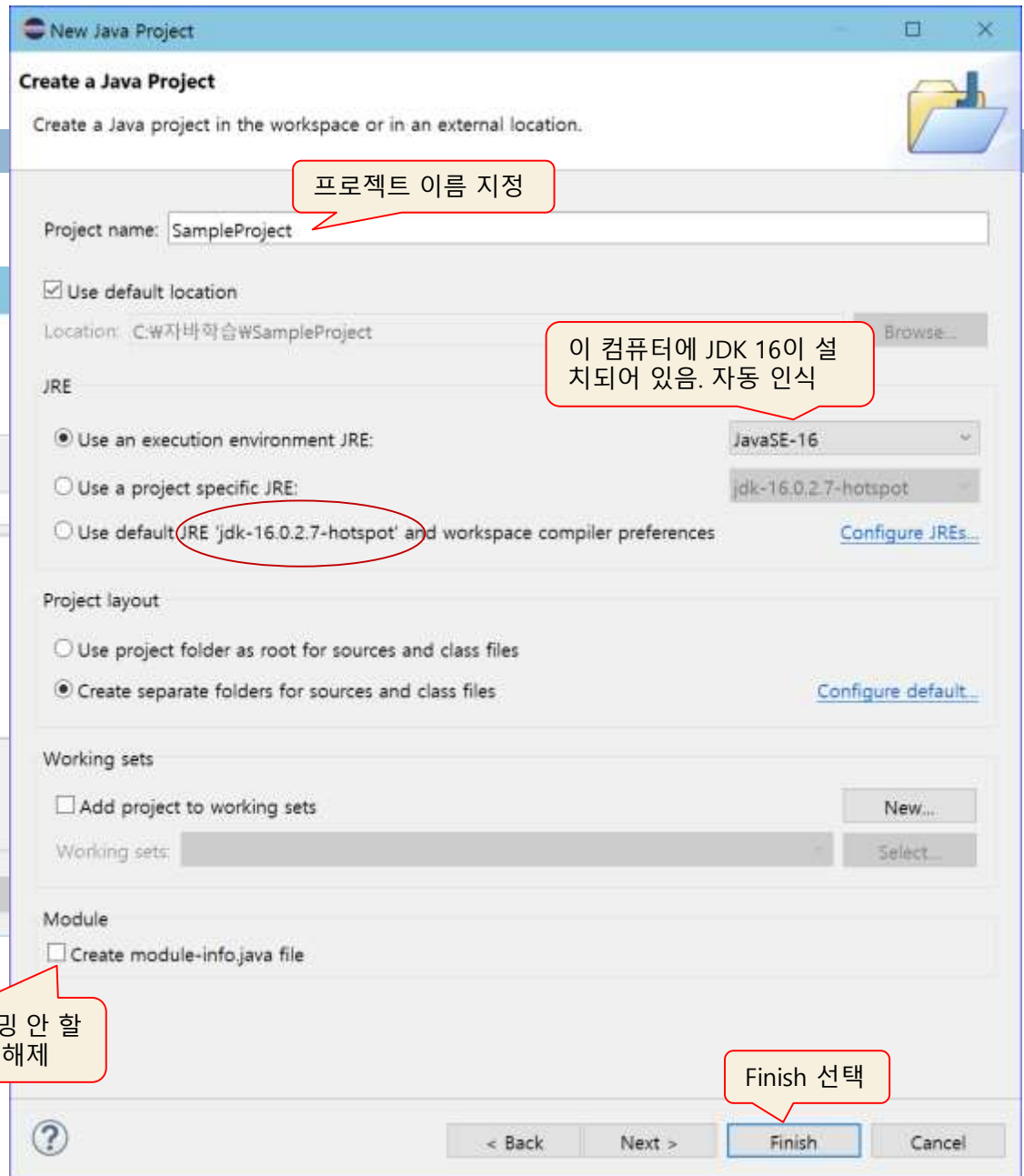


프로젝트 생성

32



Java Project 선택



프로젝트 이름 지정

이 컴퓨터에 JDK 16이 설치되어 있음. 자동 인식

모듈 프로그래밍 안 할 것이므로 체크 해제

Finish 선택

클래스 생성

33

File->New->Class 메뉴 선택

New Java Class

Java Class

! The use of the default package is discouraged.

Source folder: SampleProject/src **주목** Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: Hello2030 **클래스 이름 입력**

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

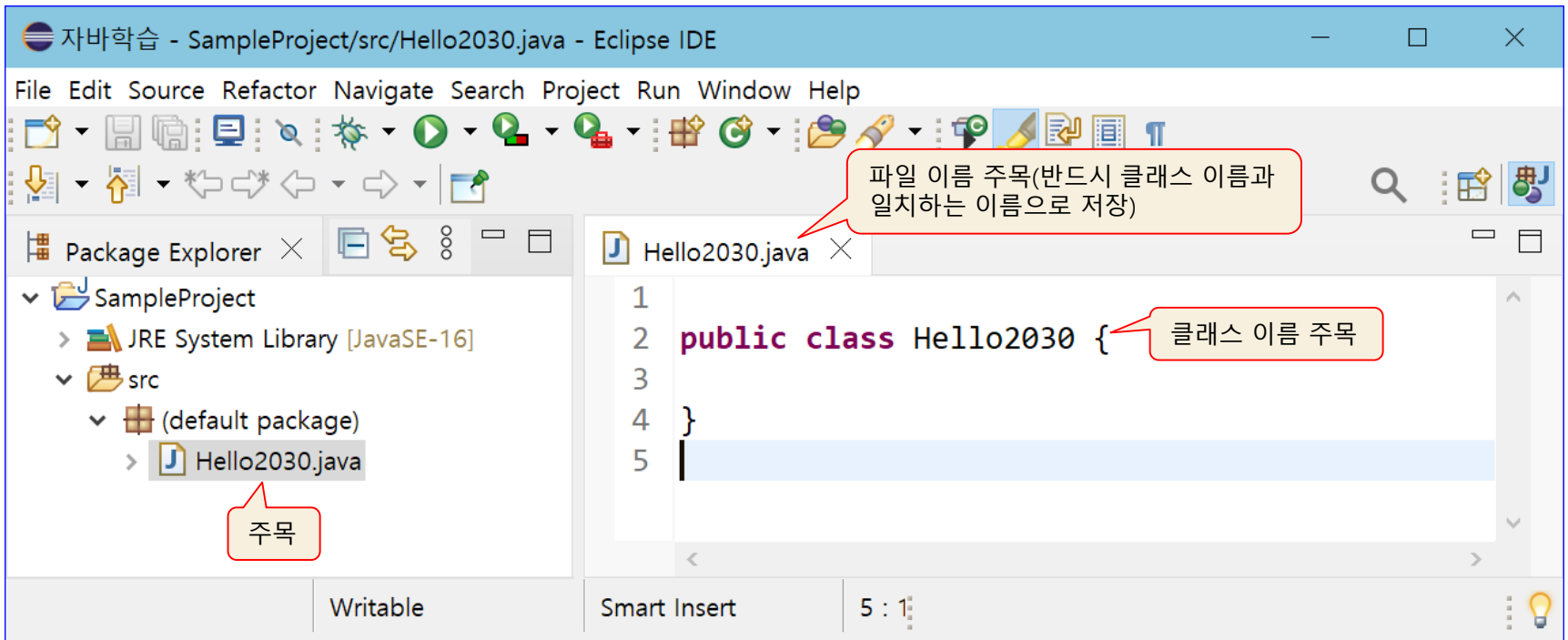
main()을 체크하면 자동으로 main() 메소드 생성

Finish 선택

Finish Cancel

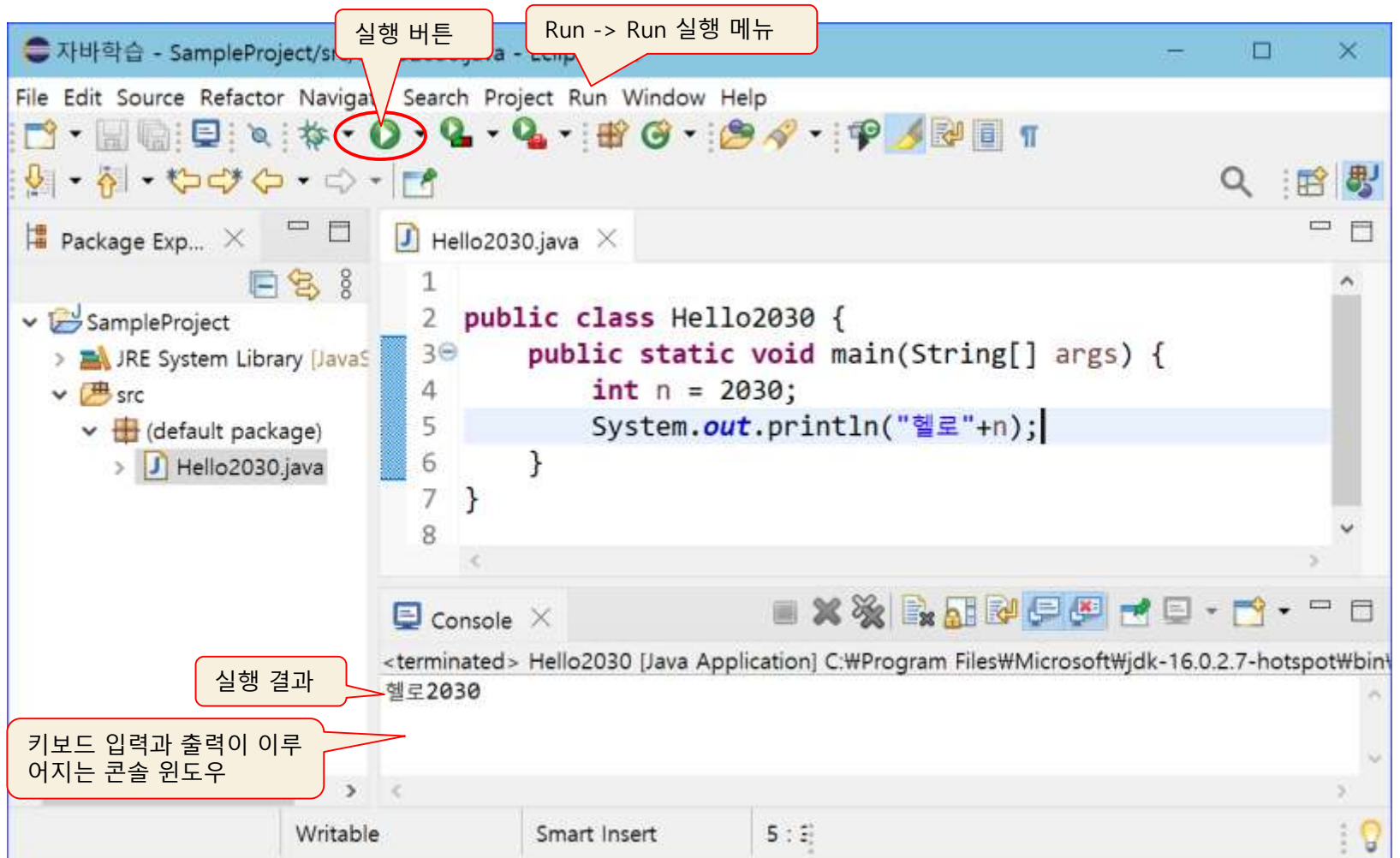
생성된 자바 소스

34



소스 편집과 컴파일 및 실행

35



자바 응용의 종류 : 데스크톱 응용프로그램

36

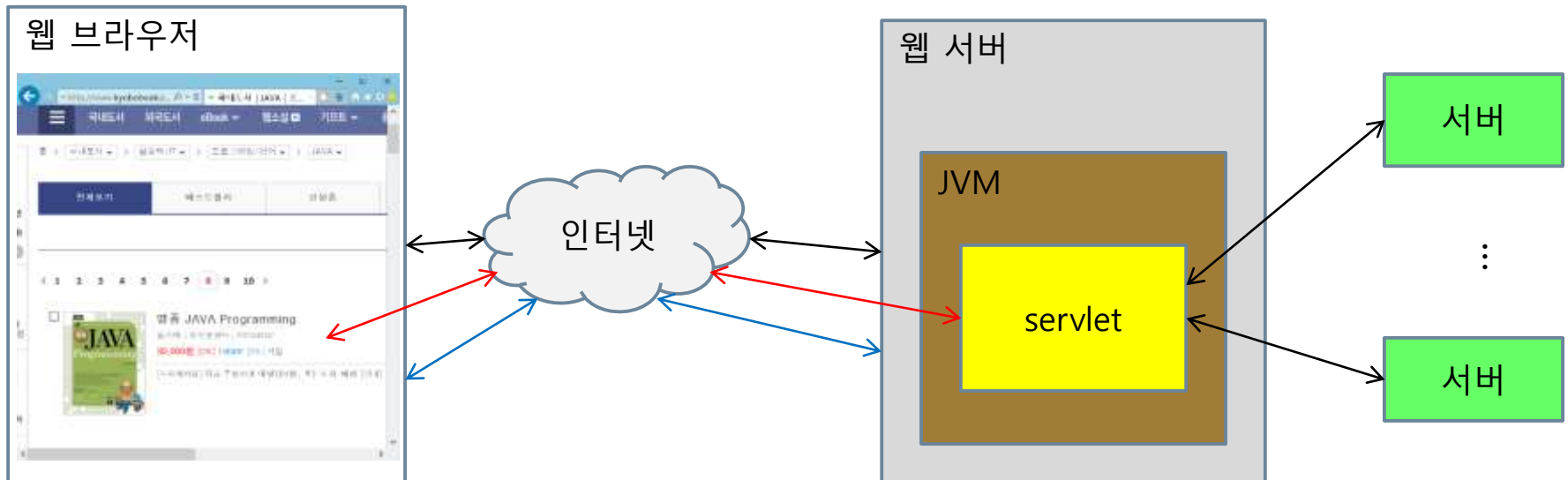
- 가장 전형적인 자바 응용프로그램
 - ▣ PC 등의 데스크톱 컴퓨터에 설치되어 실행
 - ▣ 자바 실행 환경(JRE)이 설치된 어떤 컴퓨터에서도 실행
 - 다른 응용프로그램의 도움 필요 없이 단독으로 실행



자바 응용의 종류 : 서블릿 응용프로그램

37

- 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

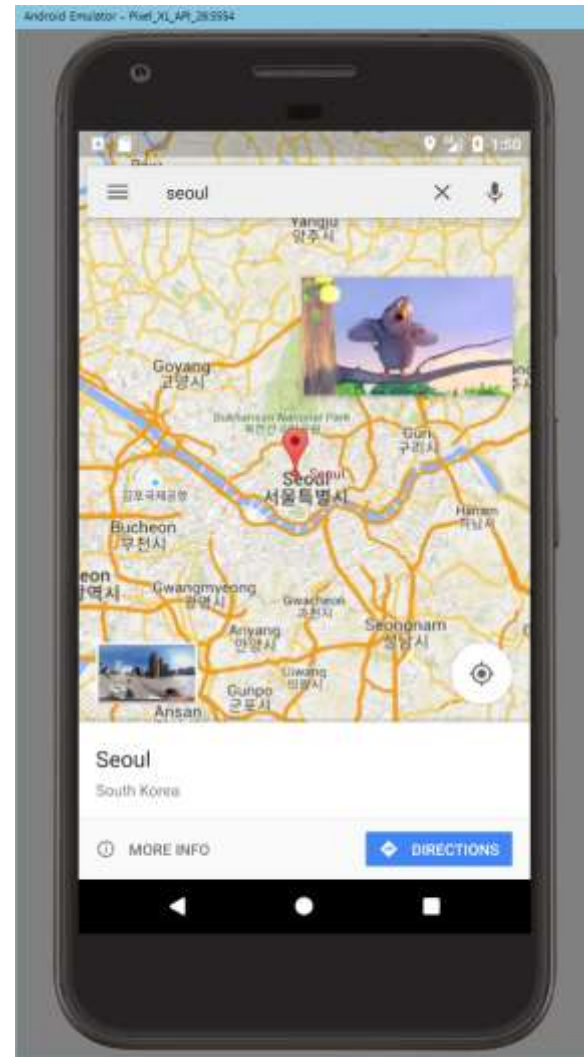


자바 모바일 응용 : 안드로이드 앱

38

□ 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



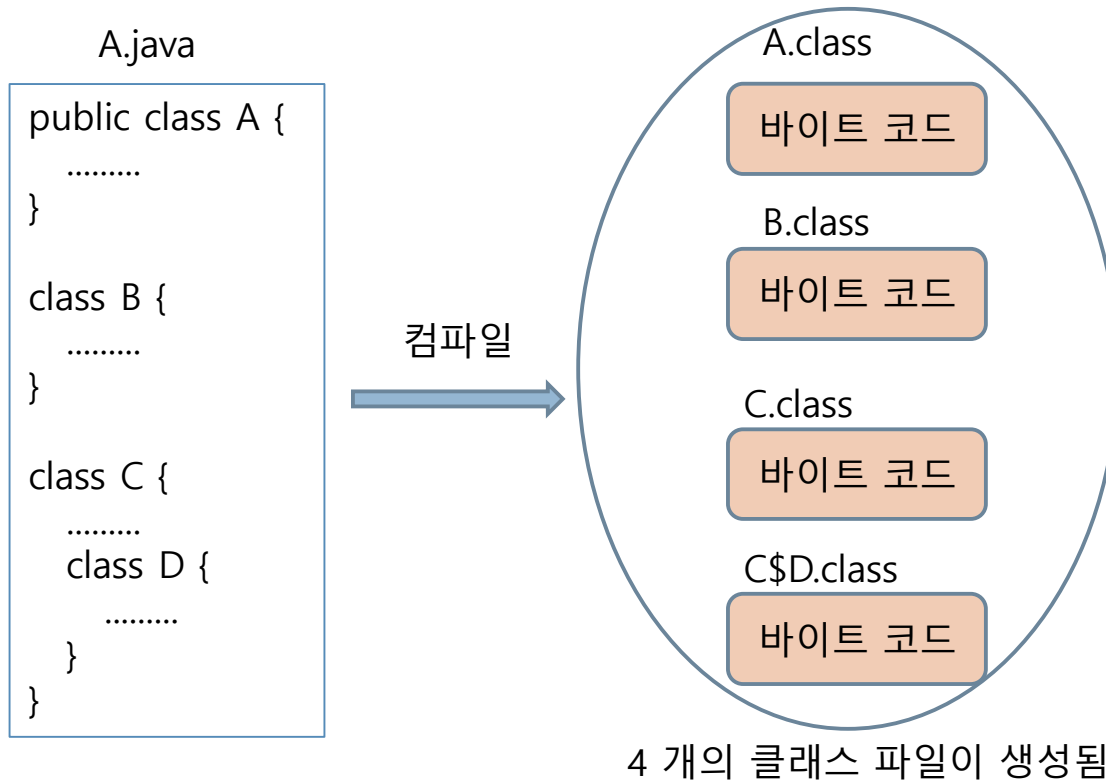
자바의 특성(1)

39

- 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

40



자바의 특징(2)

41

□ 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)

42

- 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