

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

3



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

프로그래밍 언어

4

□ 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

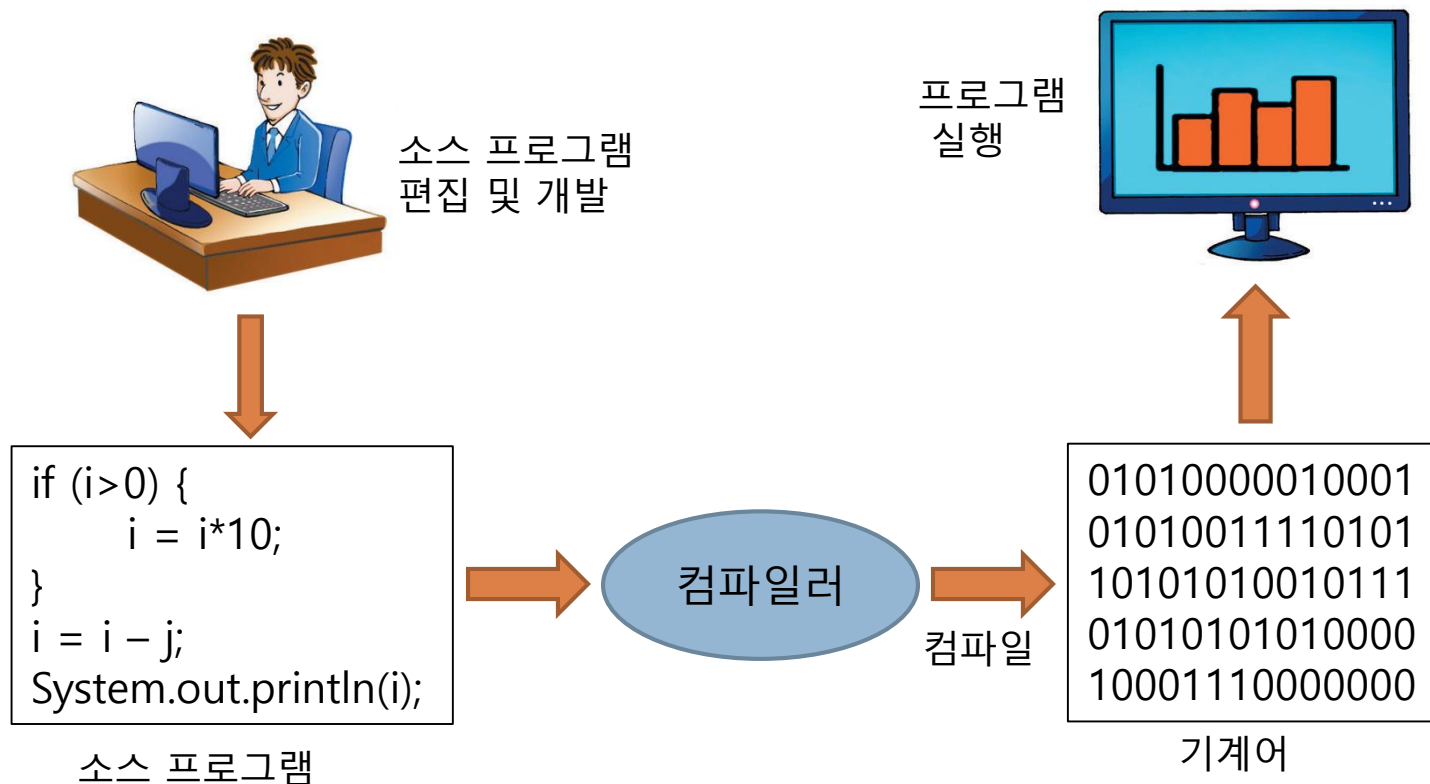
▣ 고급언어

- 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

프로그래밍과 컴파일

5

- 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



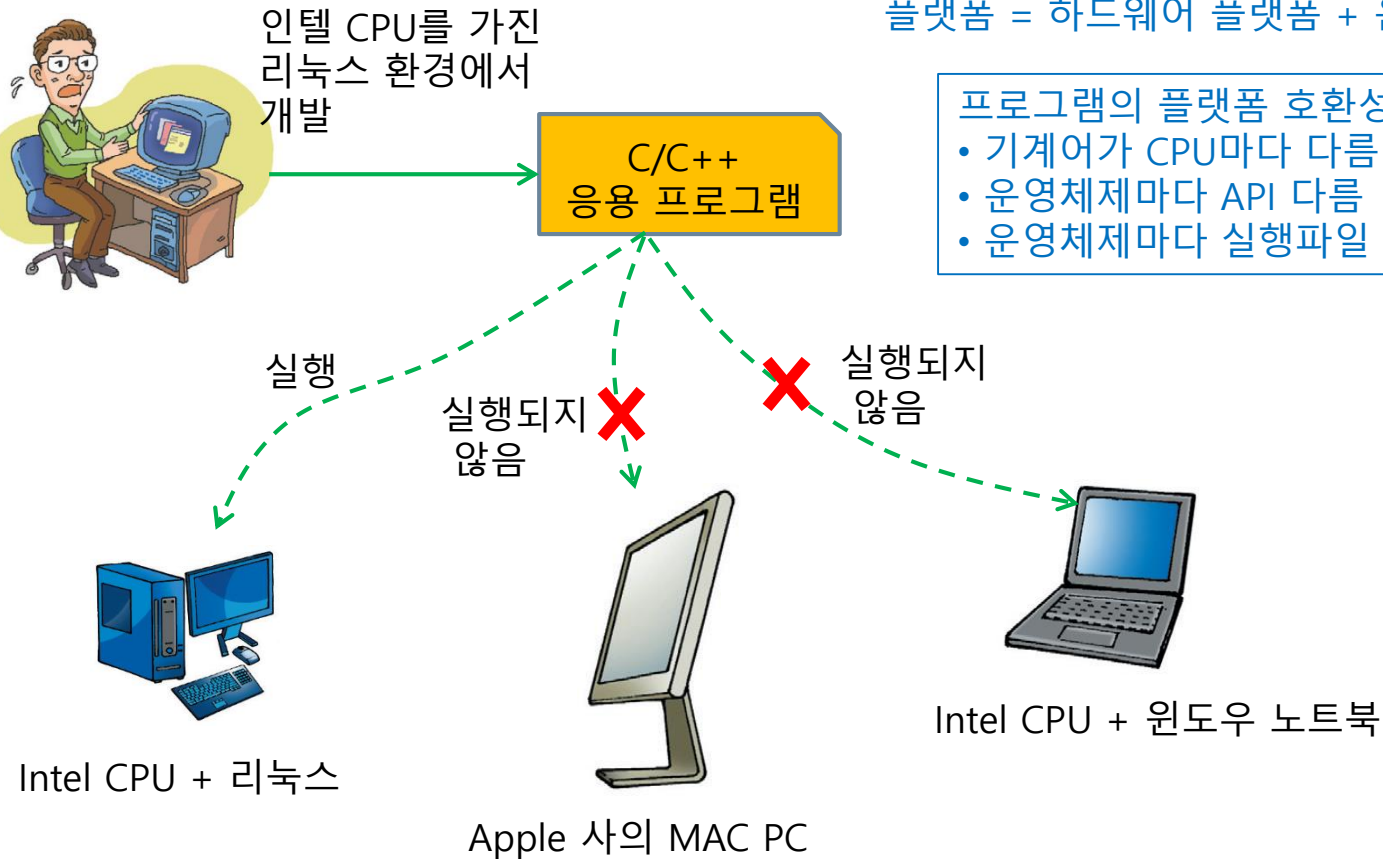
자바의 태동

6

- 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

7



플랫폼 = 하드웨어 플랫폼 + 운영체제 플랫폼

프로그램의 플랫폼 호환성 없는 이유

- 기계어가 CPU마다 다름
- 운영체제마다 API 다름
- 운영체제마다 실행파일 형식 다름

자바의 플랫폼 독립성, WORA

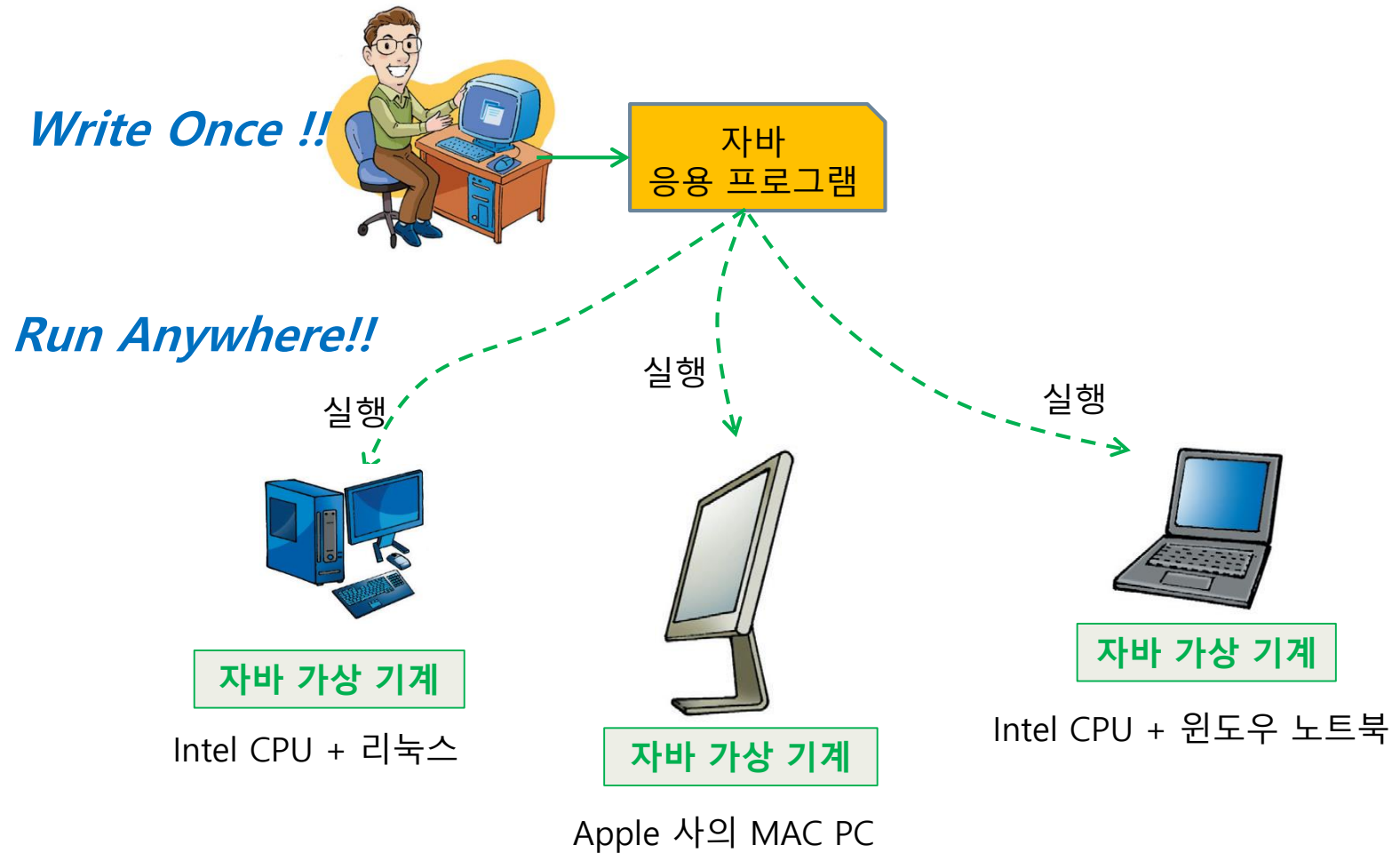
8

- 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

9



자바 가상 기계와 자바 실행 환경

10

- 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 응용프로그램의 실행

11

* 자바는 링크 과정 없음



자바 프로그래밍

Draw.java

Hello.java

Shape.java

(소스 코드)

자바 컴파일러

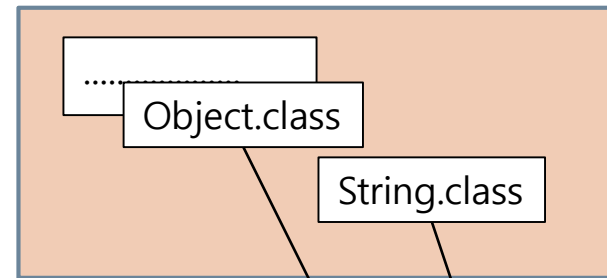
Draw.class

Hello.class

Shape.class

(바이트 코드)

실행에 필요한 자바 클래스 라이브러리(JDK APIs)



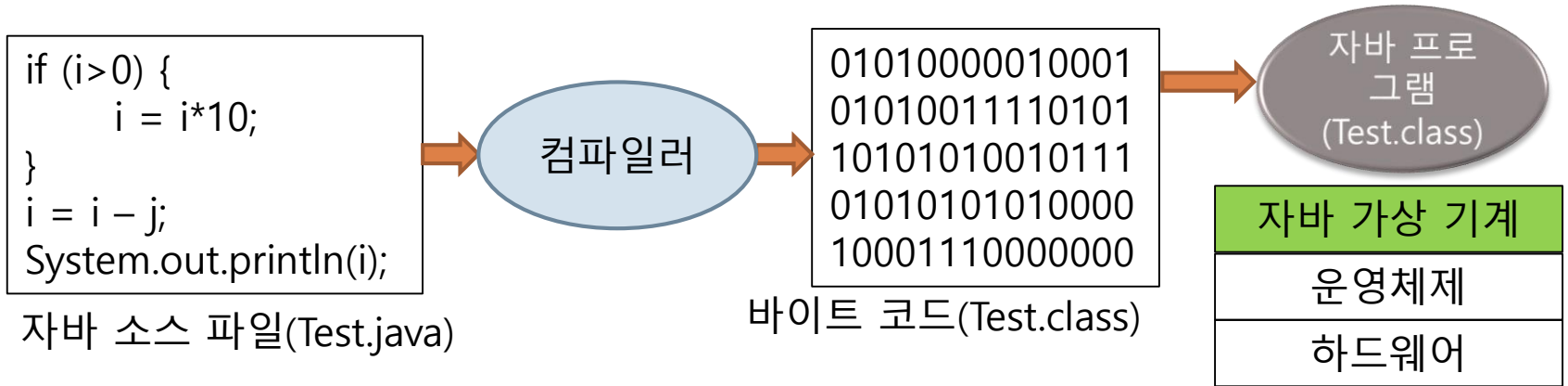
클래스 로딩



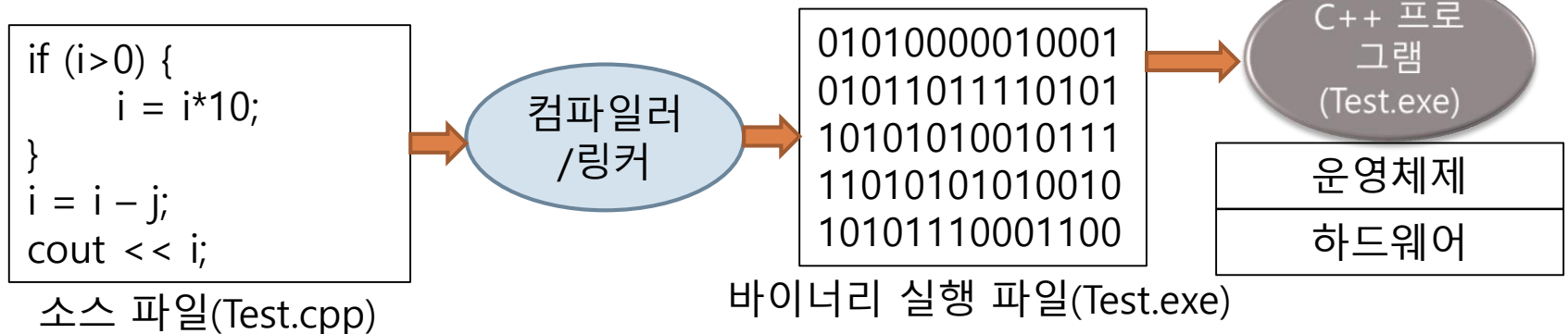
자바와 타언어(C/C++)의 실행 차이

12

□ 자바



□ C/C++



자바의 배포판

13

- 오라클은 개발 환경에 따라 다양한 자바 개발 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

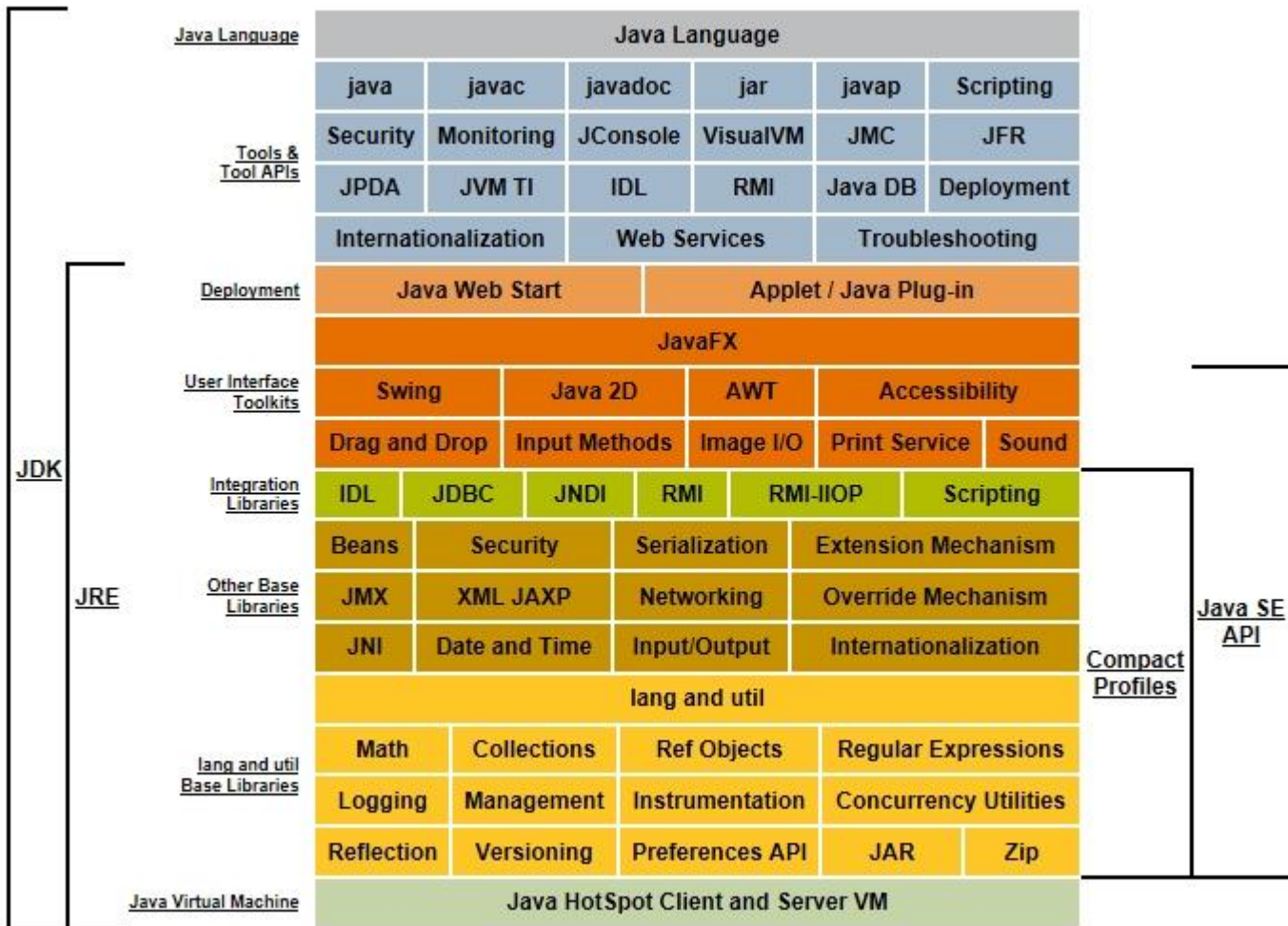
자바와 오픈 소스

14

- 오픈 소스란?
 - ▣ 소프트웨어 제작자의 권리를 보존
 - ▣ 누구나 액세스할 수 있도록 소스 코드를 무상 공개한 소프트웨어
- 오픈 소스의 장점
 - ▣ 공개된 소스 코드를 참조함으로써 개발 시간 및 비용 단축
 - ▣ 공개된 소프트웨어를 다수의 인원이 참여 개량, 우수한 품질의 소프트웨어 개발
- 오픈 소스의 단점
 - ▣ 무단으로 상용 소프트웨어에 사용할 경우 저작권 침해 발생
 - ▣ 다양한 개량 버전의 소프트웨어로 인한 호환성 문제
- 오픈 소스 소프트웨어 사례
 - ▣ Linux, OpenOffice, Open Solaris, Mozilla, Apache, GNU, WebKit 등
 - ▣ 2006년 11월, 선마이크로시스템즈는 자바를 GPL 라이선스로 소스 오픈
 - ▣ <http://sourceforge.net> : 오픈 소스 사이트

Java SE 구성

15



출처: <http://download.oracle.com/javase/8/docs/>

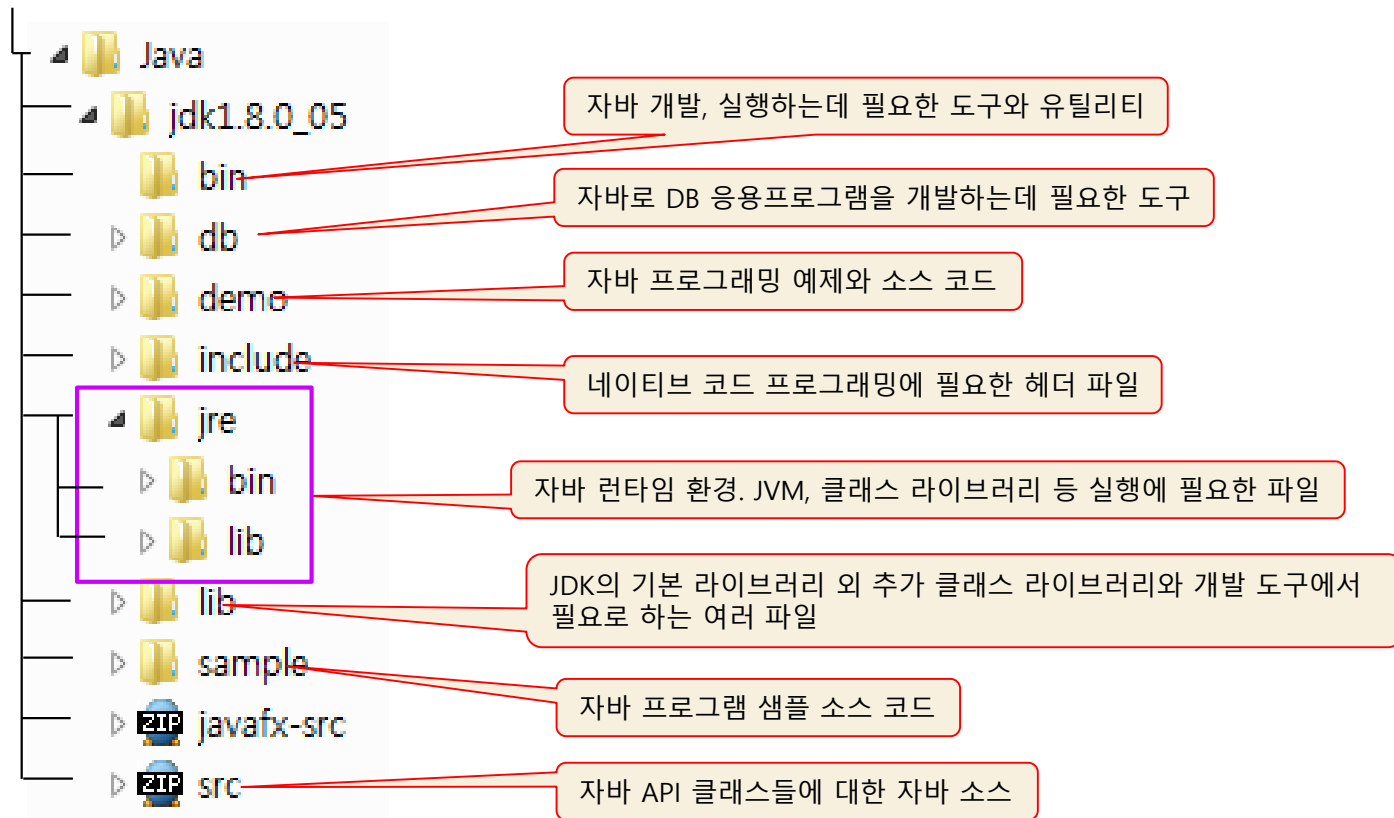
JDK와 JRE

16

- 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)을 생성 및 관리하는 유틸리티
 - ▣ jdb - 자바 디버거
 - ▣ appletviewer - 웹 브라우저 없이 애플릿을 실행하는 유틸리티

JDK 설치 후 디렉터리 구조

17



나는 누구?

18



(사진 출처 : 위키 백과)

자바 API

19

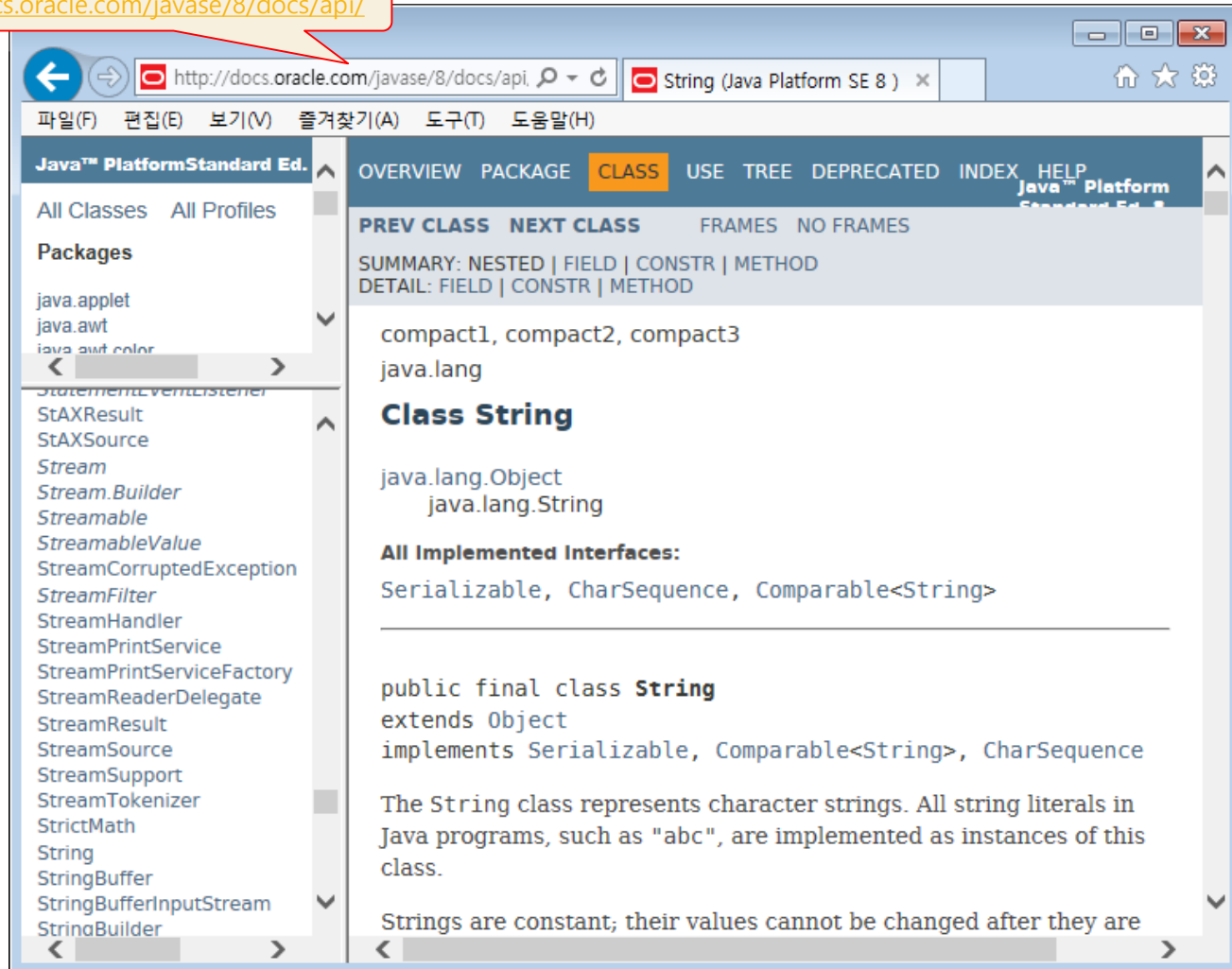
- 자바 패키지(package)
 - ▣ 서로 관련된 클래스들을 분류하여 묶어 놓은 것
 - ▣ 계층구조로 되어 있음
 - ▣ 개발자 자신의 패키지 생성 가능

- 자바 API(Application Programming Interface)
 - ▣ 개발자가 이용하여 쉽고 빠르게 자바 프로그램을 개발할 수 있는 자바 라이브러리
 - JDK에 클래스 라이브러리로 제공
 - 패키지 형태로 제공

자바 온라인 API 문서

20

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



자바 프로그램 개발

21

1. 자바 소스 편집

```

// Hello2030.java 파일에 작성

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 정수 값으로 초기화
// "헬로"+n의 결과로 "헬로2030"을 출력

```

2. 자바 소스 컴파일

```

C:\> 관리자: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\WKitae>cd WTemp
C:\Temp>javac Hello2030.java
C:\Temp>dir Hello2030.*
C 드라이브의 볼륨: BOOTCAMP
볼륨 일련 번호: 18A9-67A9

C:\Temp 디렉터리

2014-08-08 오후 04:44                629 Hello2030.class
2014-03-18 오후 07:24                329 Hello2030.java
                2개 파일                958 바이트
                0개 디렉터리 37,879,521,280 바이트 남음

C:\Temp>

```

3. 자바 응용프로그램 실행

```

C:\Temp>java Hello2030
헬로2030

C:\Temp>

```

자바 통합 개발 환경-이클립스(Eclipse)

22

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

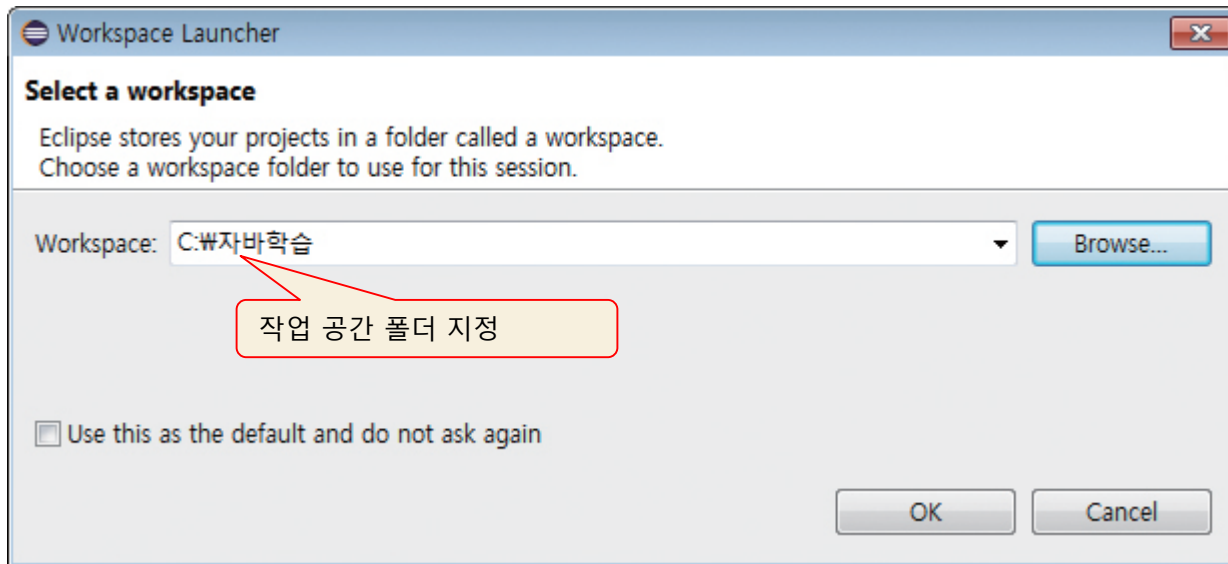
- 이클립스(Eclipse)
 - ▣ 자바 응용 프로그램 개발을 위한 통합 개발 환경
 - ▣ IBM에 의해 개발된 오픈 소스 프로젝트
 - ▣ <http://www.eclipse.org/downloads/> 에서 다운로드

이클립스 실행

23



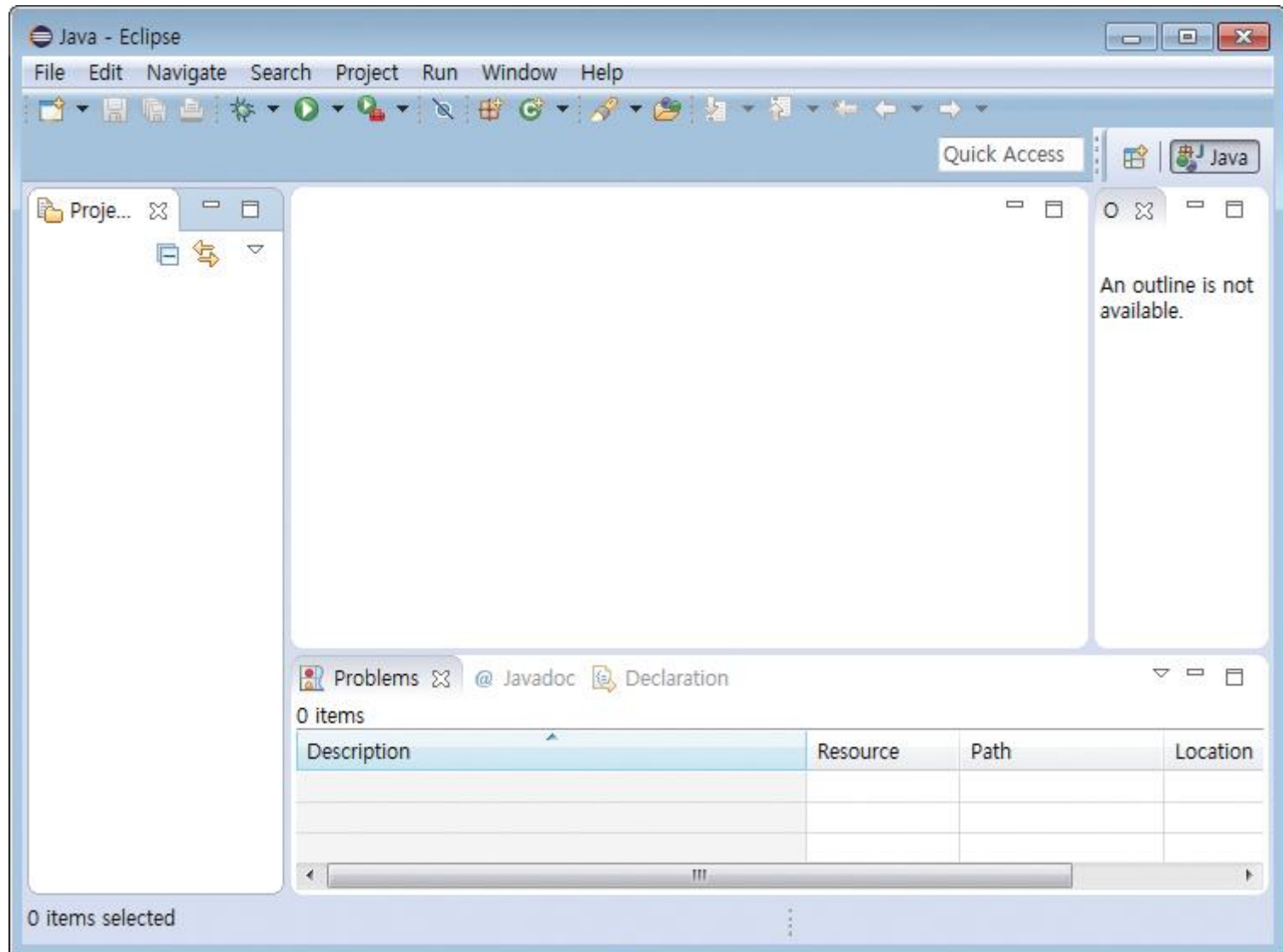
이클립스 Luna 배포판



작업 공간 폴더 지정

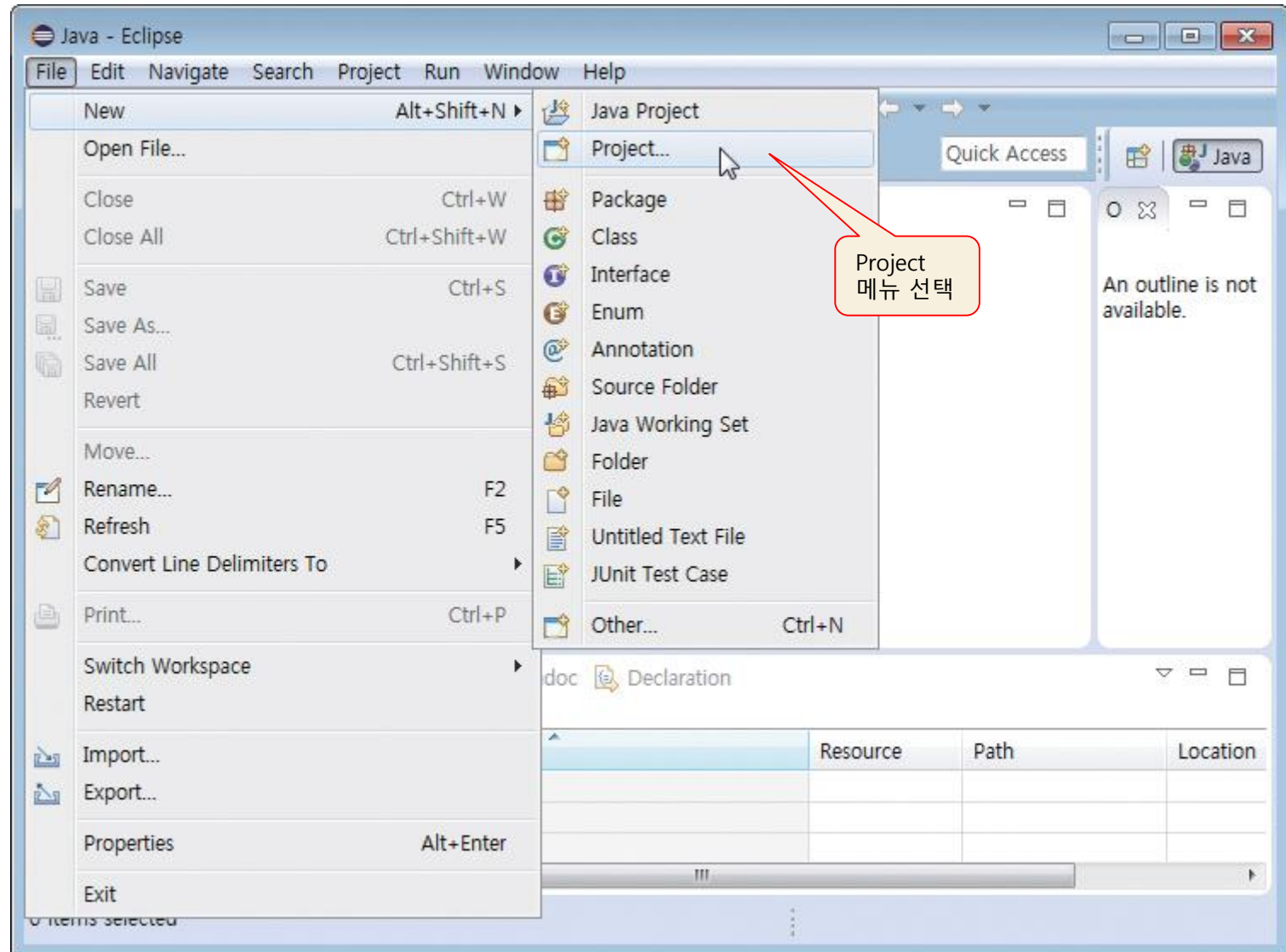
이클립스의 사용자 인터페이스

24



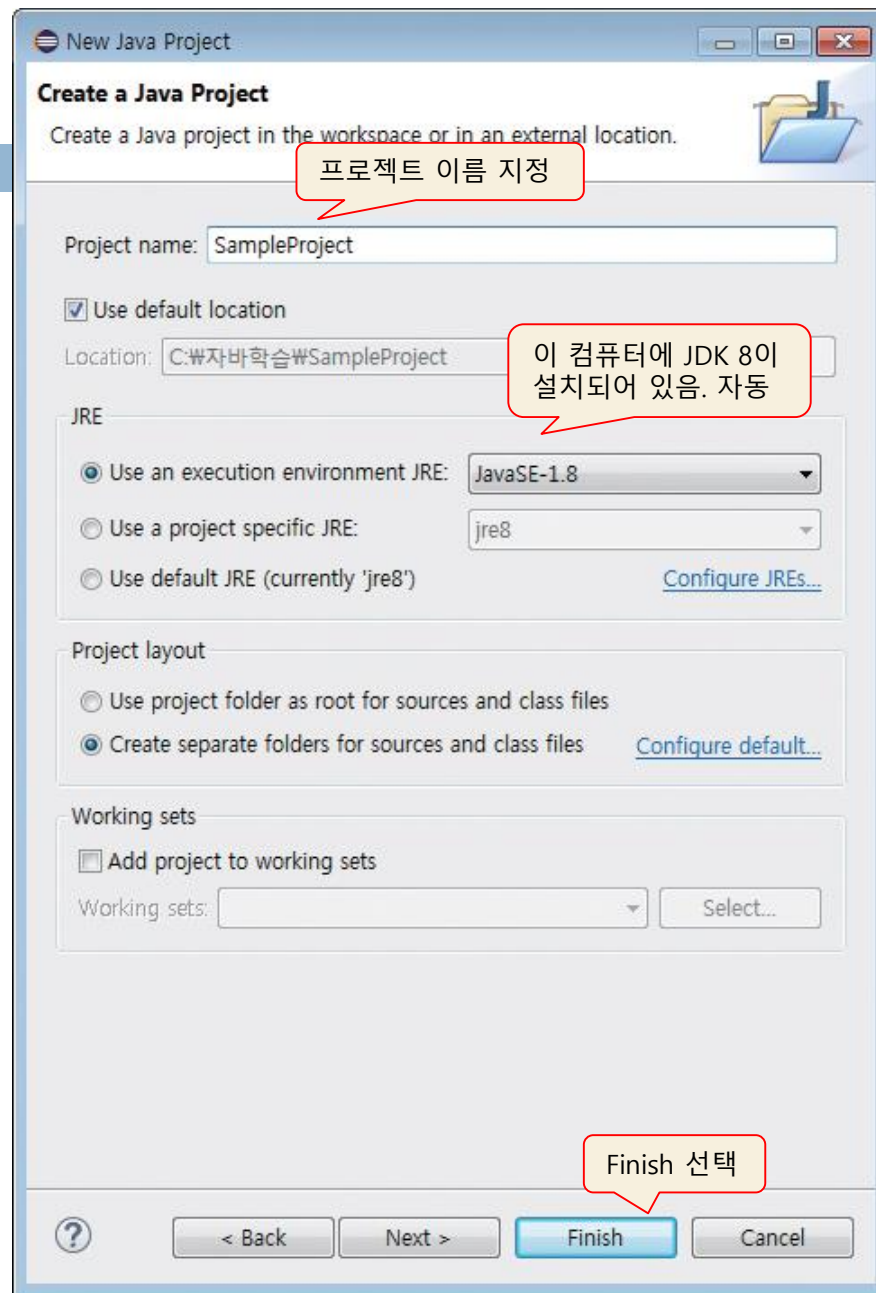
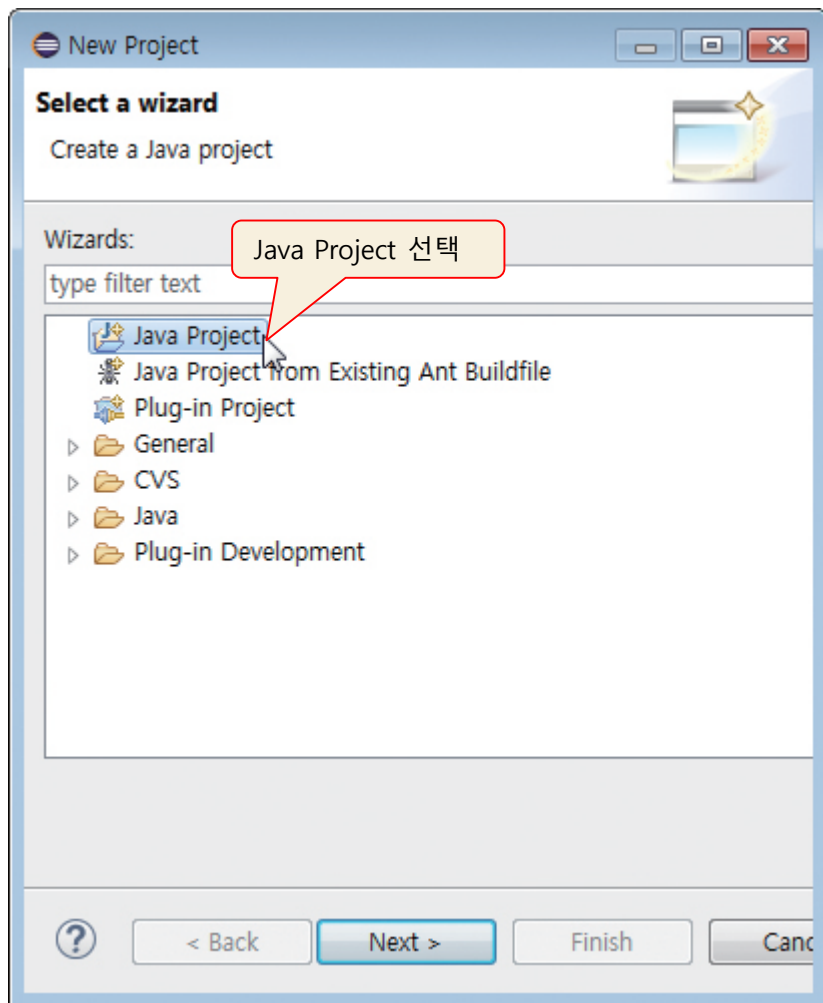
프로젝트 생성

25



프로젝트 생성

26



클래스 생성

27

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) **main()을 체크하면 자동으로 main() 메소드 생성**

☐ Constructors from superclass

☒ Inherited abstract methods

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

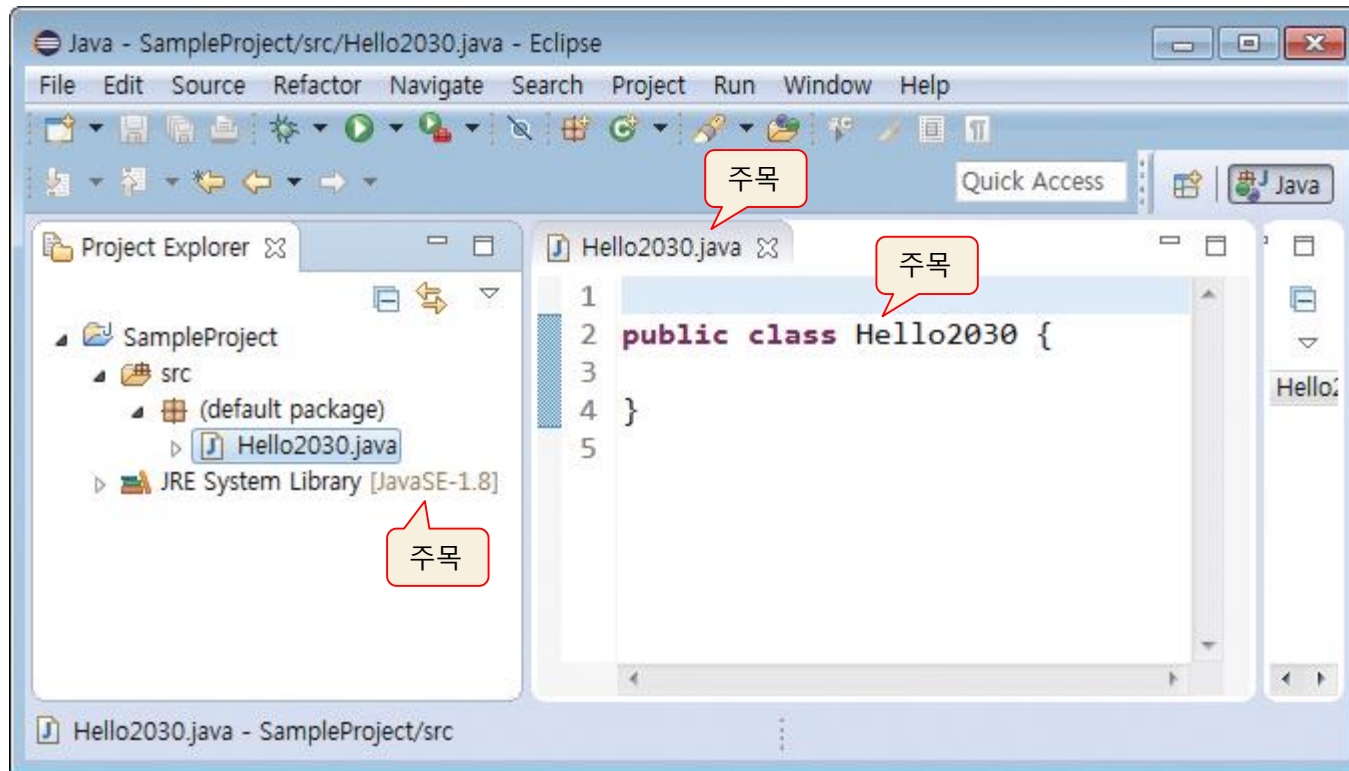
☐ Generate comments

Finish 선택

Finish Cancel

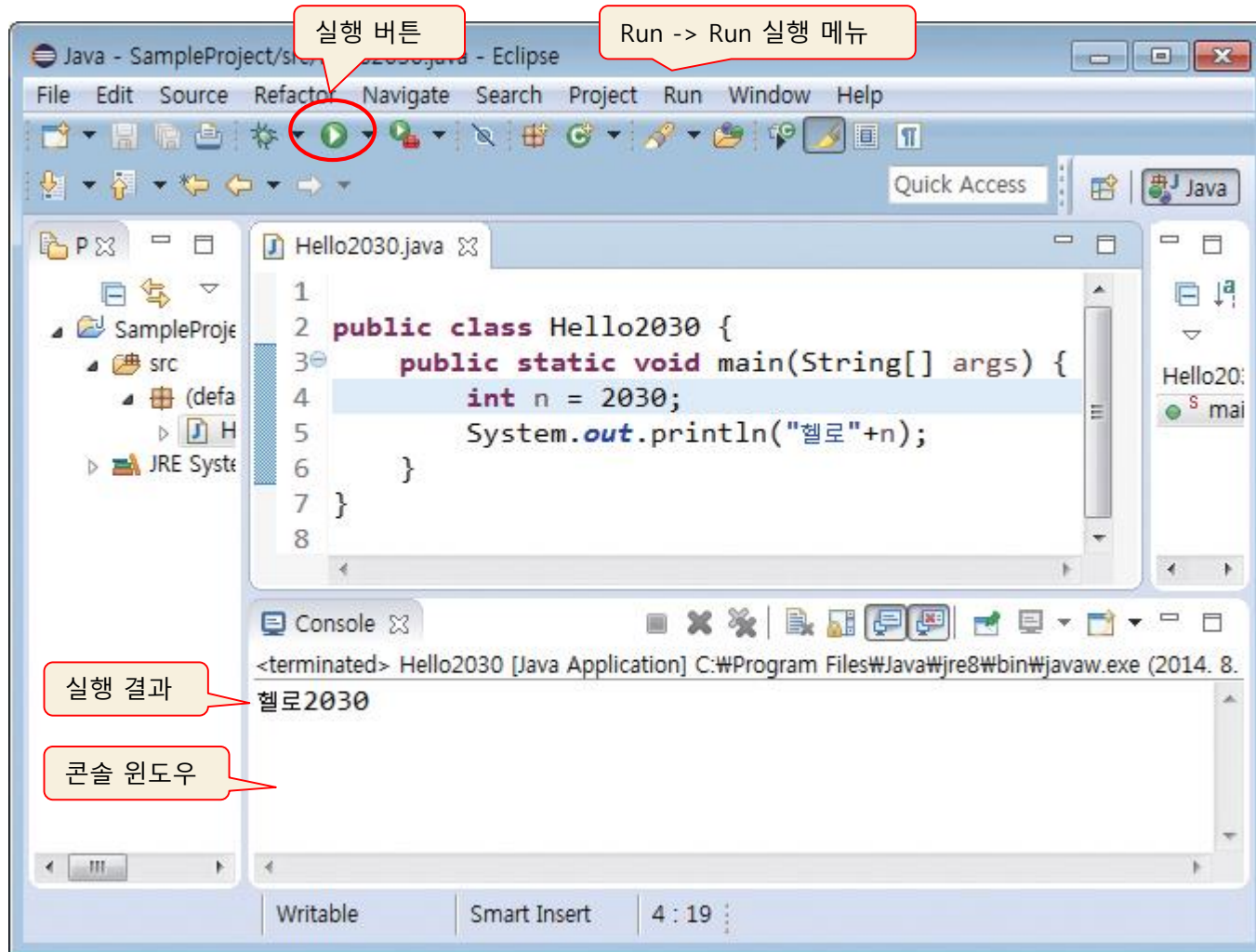
생성된 자바 소스

28



소스 편집과 컴파일 및 실행

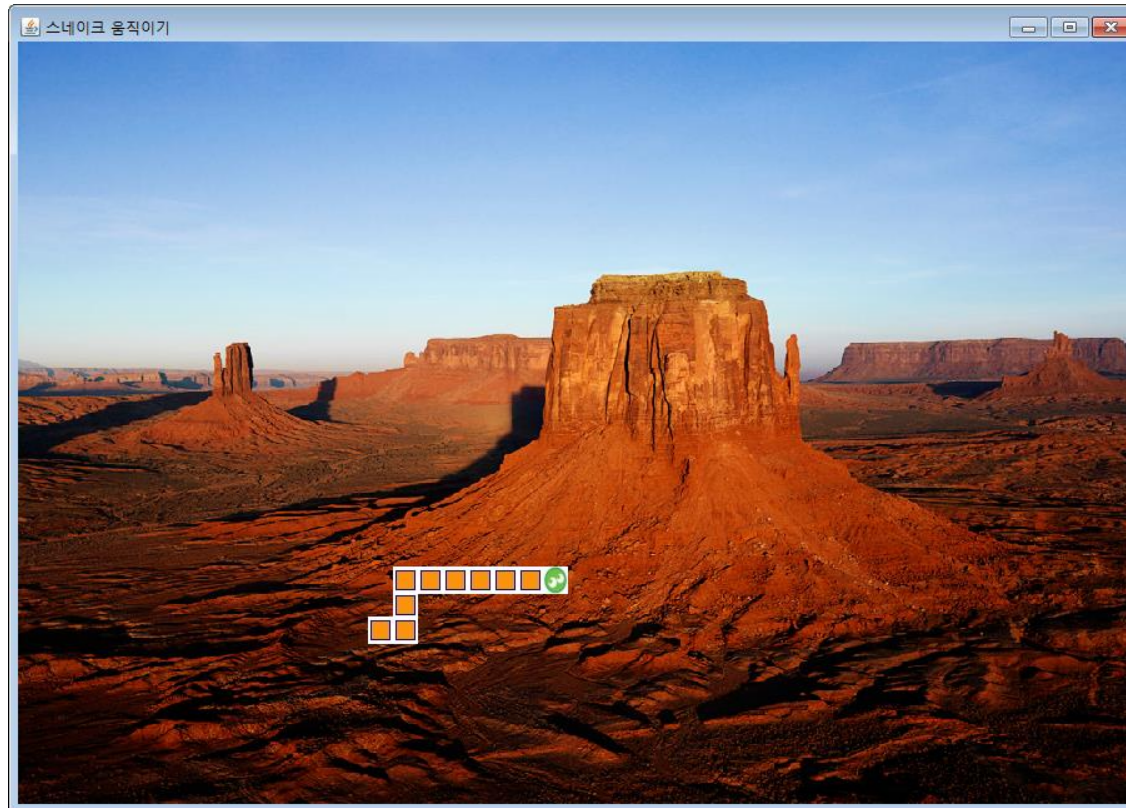
29



자바 응용의 종류 : 데스크톱 응용프로그램

30

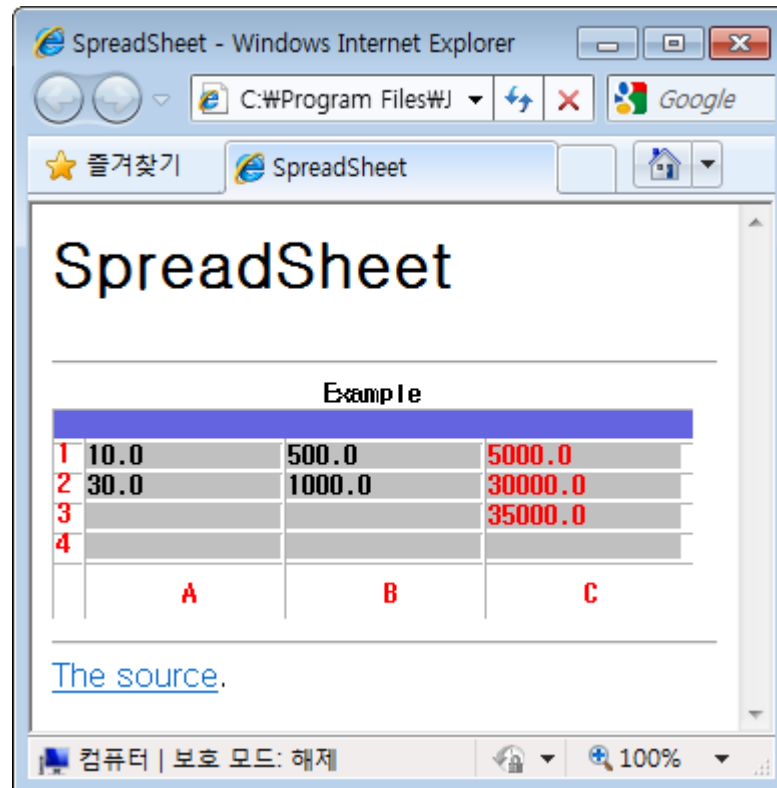
- 가장 전형적인 자바 응용프로그램
 - ▣ PC 등의 데스크톱 컴퓨터에 설치되어 실행
 - ▣ JRE가 설치된 어떤 환경에서도 실행
 - 다른 응용프로그램의 도움이 필요 없이 단독으로 실행



자바 응용의 종류 : 애플릿 응용프로그램

31

- 애플릿(applet)
 - ▣ 웹 브라우저에 의해 구동되고 실행이 제어되는 자바 프로그램
 - ▣ 애플릿은 사용할 수 있는 자원 접근에 제약 있음

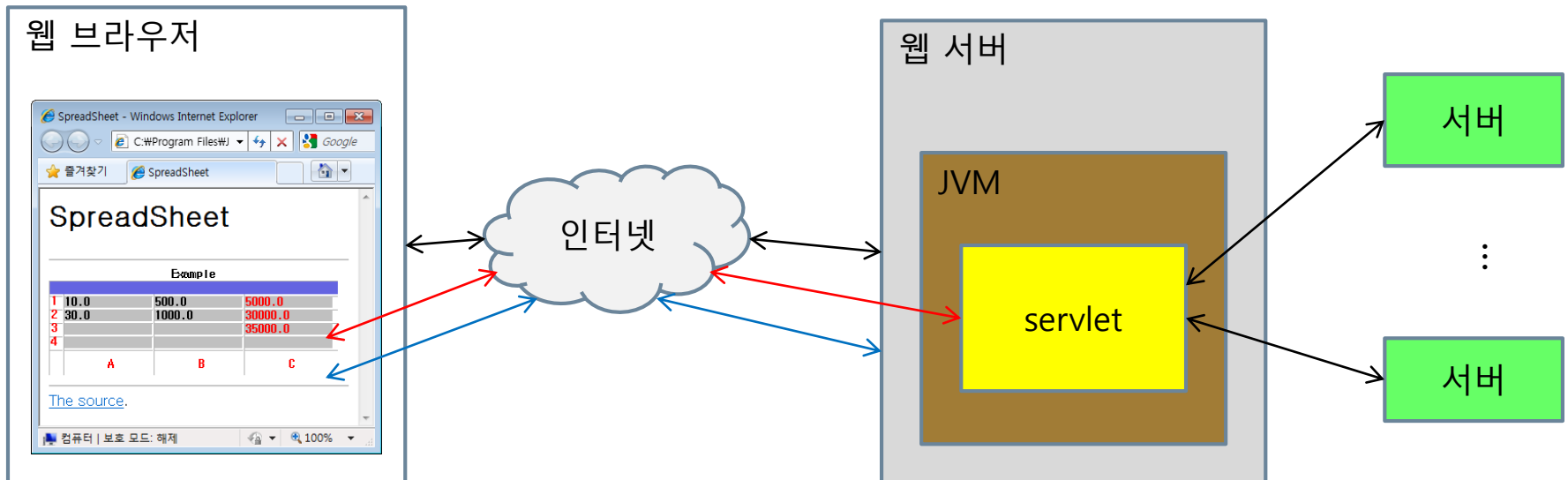


자바 응용의 종류 : 서블릿 응용프로그램

32

□ 서블릿(servlet)

- 애플릿과 반대로 서버에서 실행되는 자바 프로그램
 - 서버 클라이언트 모델에서 서블릿과 애플릿이 각각 통신하면서 실행
- 데이터베이스 서버 및 기타 서버와 연동하는 복잡한 기능 구현 시 사용
- 사용자 인터페이스가 필요 없는 응용
- 웹 서버에 의해 실행 통제 받음



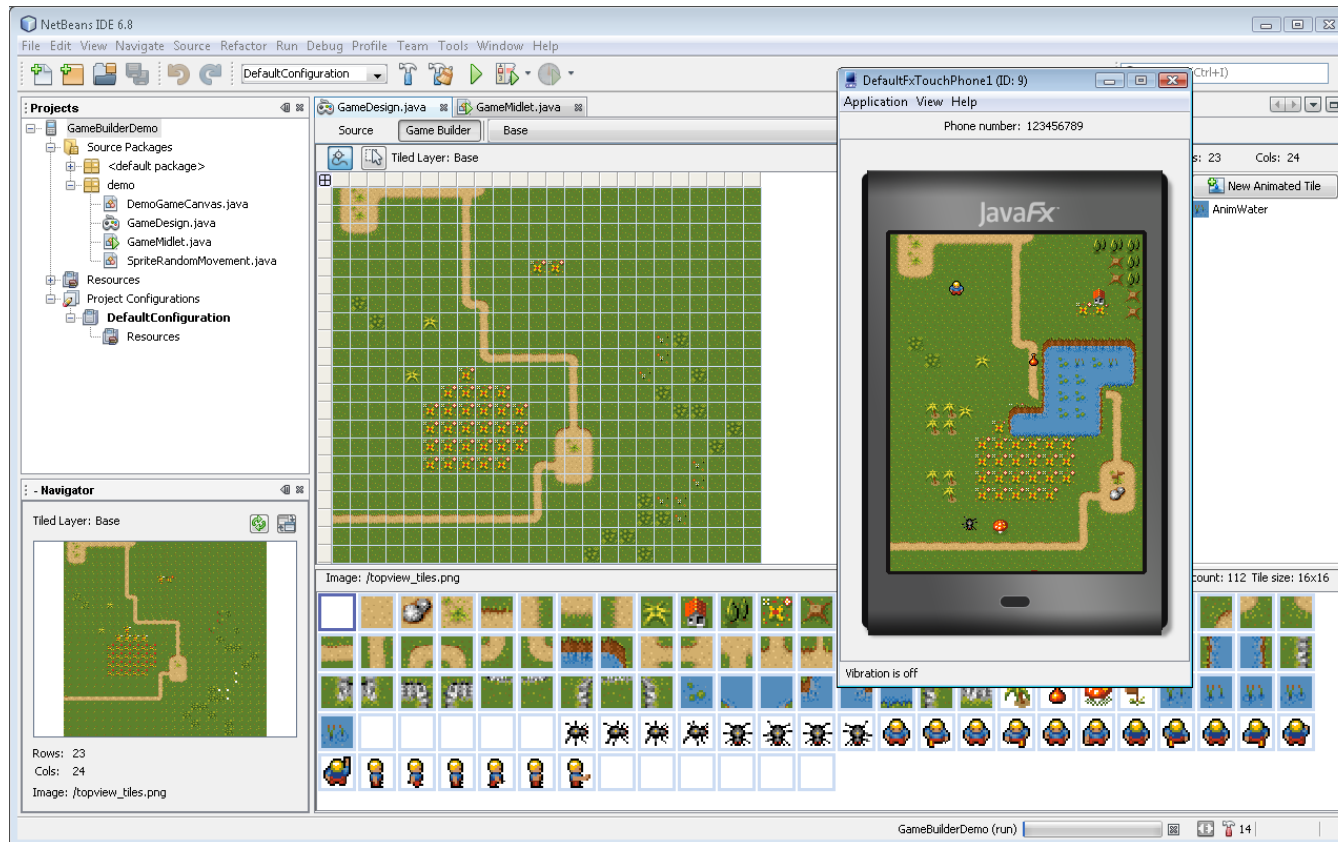
자바 응용의 종류 : 모바일 응용프로그램

33

□ Java ME

▣ 모바일 기기를 위한 자바 배포판

- 유럽, 미국 시장에 출시되는 대부분의 모바일 단말기에 탑재
- 노키아, 삼성, LG, 소니 에릭슨, 모토로라 등 단말기 제조사

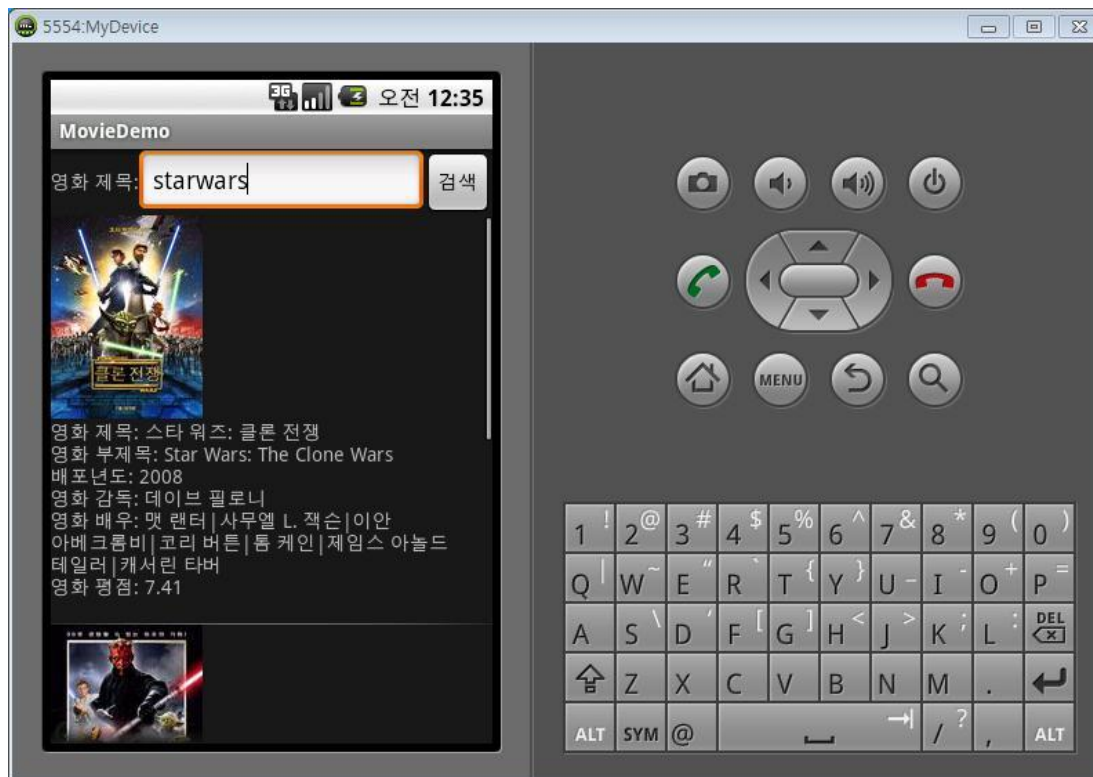


자바 모바일 응용 : 안드로이드 앱

34

□ 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



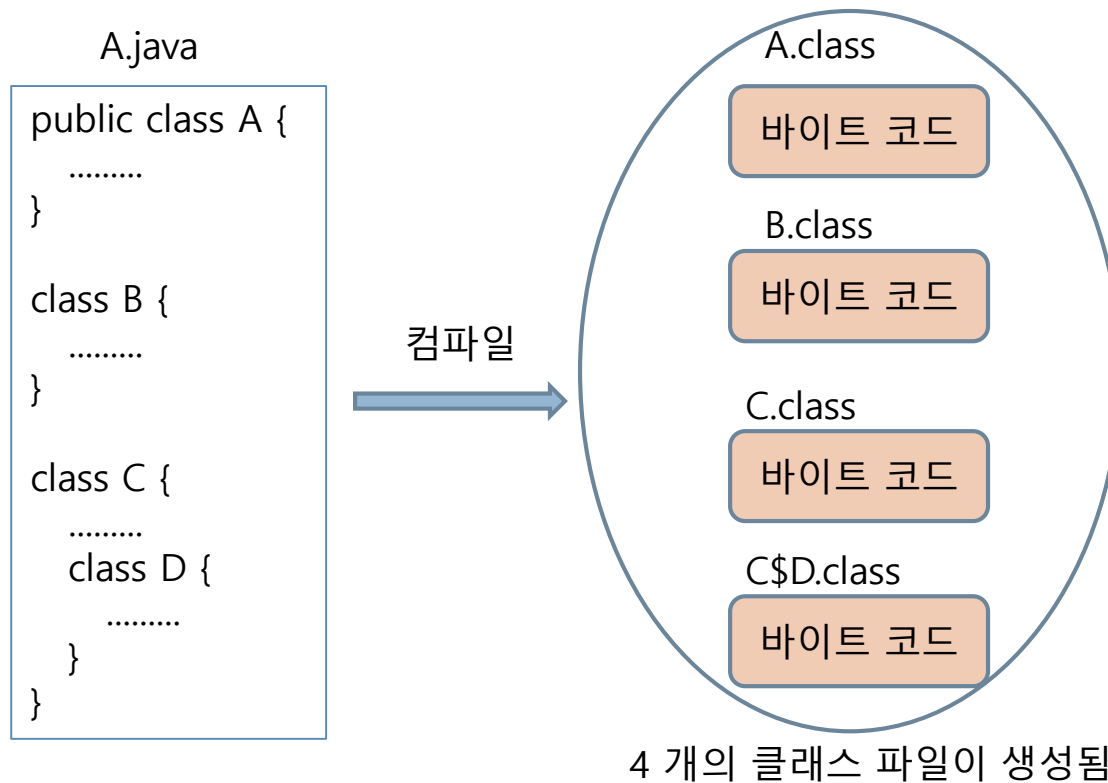
자바의 특성(1)

35

- 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

36



자바의 특징(2)

37

□ 실행 모듈

□ 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 javaWlang 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)

38

- 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