

1

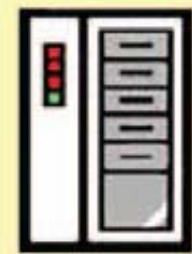
자바 살펴보기

1

자바 시작

컴퓨터의 하드웨어와 소프트웨어

2



메인 프레임



태블릿



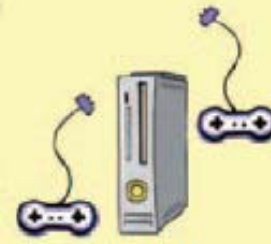
PC



스마트폰



장난감



게임기



앱



앱



소프트웨어 혹은 앱

프로그래밍 언어

3

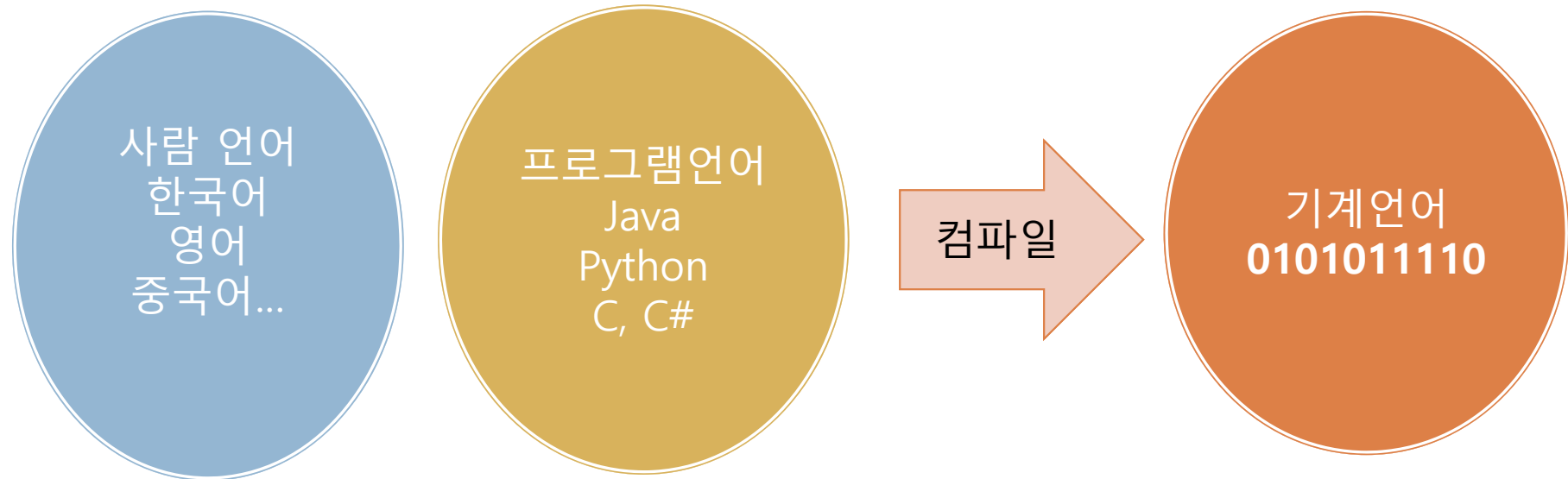
□ 프로그래밍 언어

- ▣ 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 언어

4

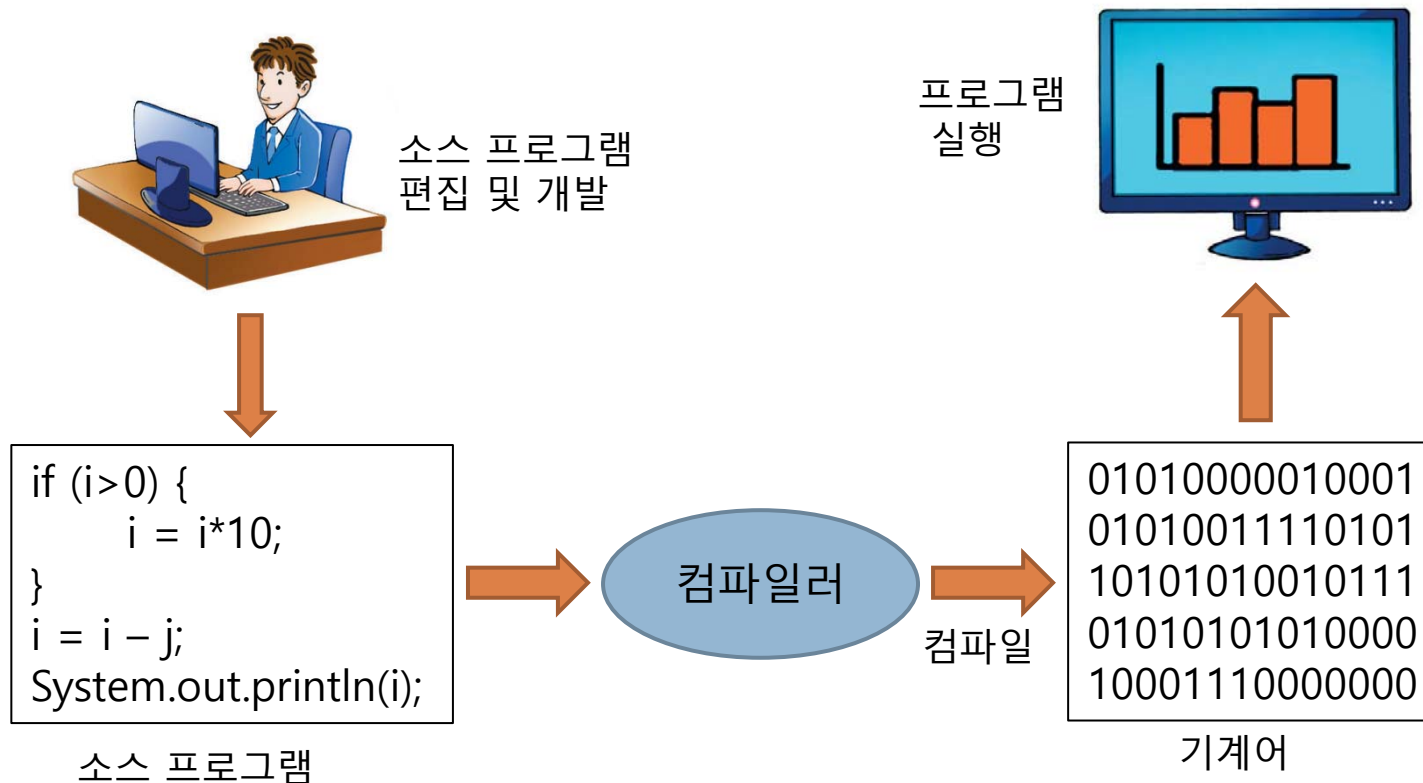
□ 프로그래밍 언어



컴파일

5

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



자바의 태동

6

- **1991**년 그린 프로젝트(Green Project)
 - **선마이크로시스템즈**의 제임스 고슬링(James Gosling)에 의해 시작
 - 가전 제품에 들어갈 소프트웨어를 위해 개발
 - **1995**년에 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
- **2009**년에 선마이크로시스템즈를 **오라클에서 인수**

WORA

7

□ WORA(Write Once Run Anywhere)

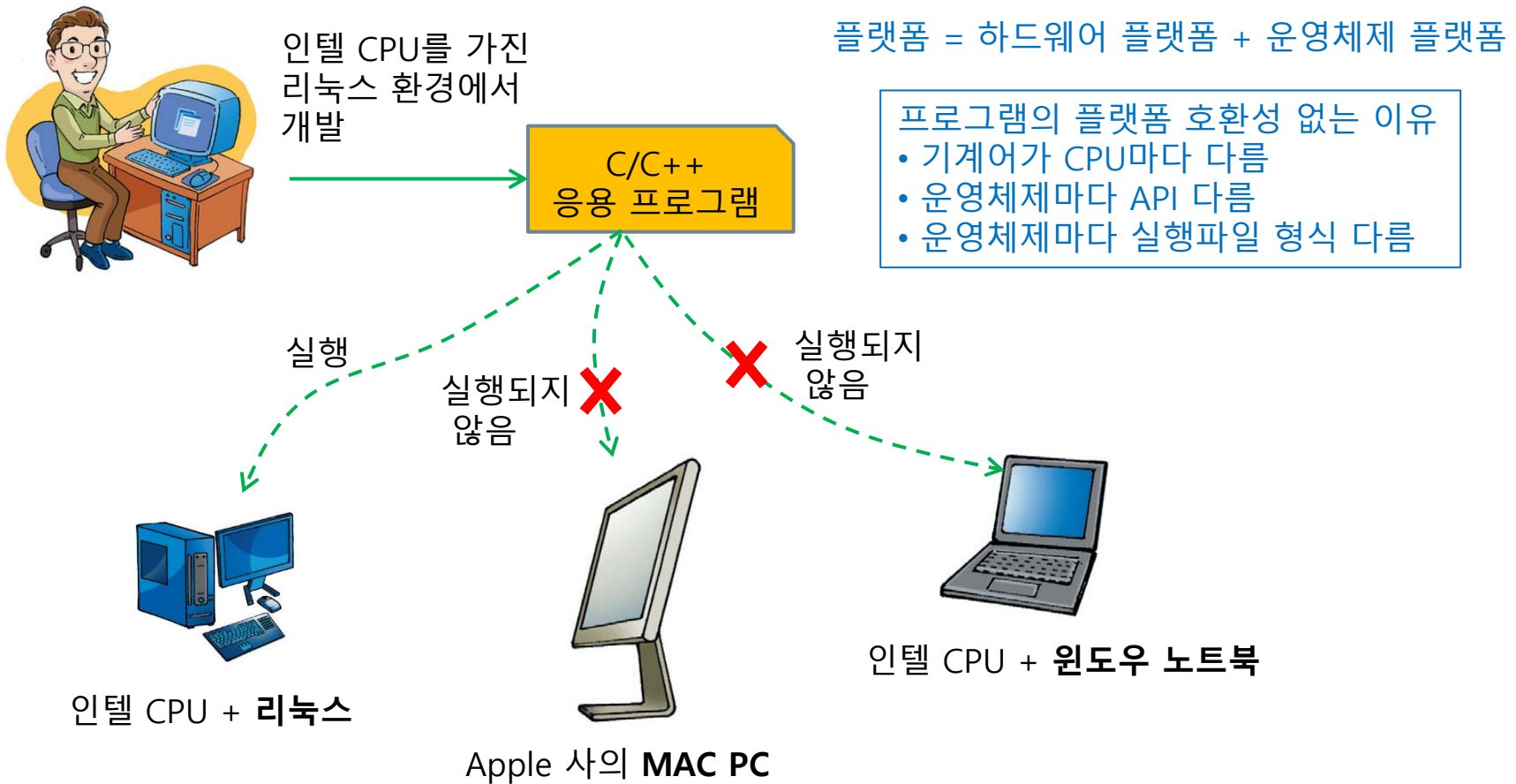
- ▣ 한번 작성된 코드는 모든 플랫폼에서 바로 실행되는 자바의 특징
- ▣ C/C++ 등 기존 언어가 가진 플랫폼 종속성 극복
 - OS, H/W에 상관없이 자바 프로그램이 동일하게 실행
- ▣ 네트워크에 연결된 어느 클라이언트에서나 실행
 - 웹 브라우저, 분산 환경 지원

□ WORA를 가능하게 하는 자바의 특징

- ▣ 바이트 코드(byte code)
 - 자바 소스를 컴파일한 목적 코드
 - CPU에 종속적이지 않은 독립적인 코드
 - JVM에 의해 해석되고 실행됨
- ▣ JVM(Java Virtual Machine)
 - 자바 바이트 코드를 실행하는 자바 가상 기계(소프트웨어)

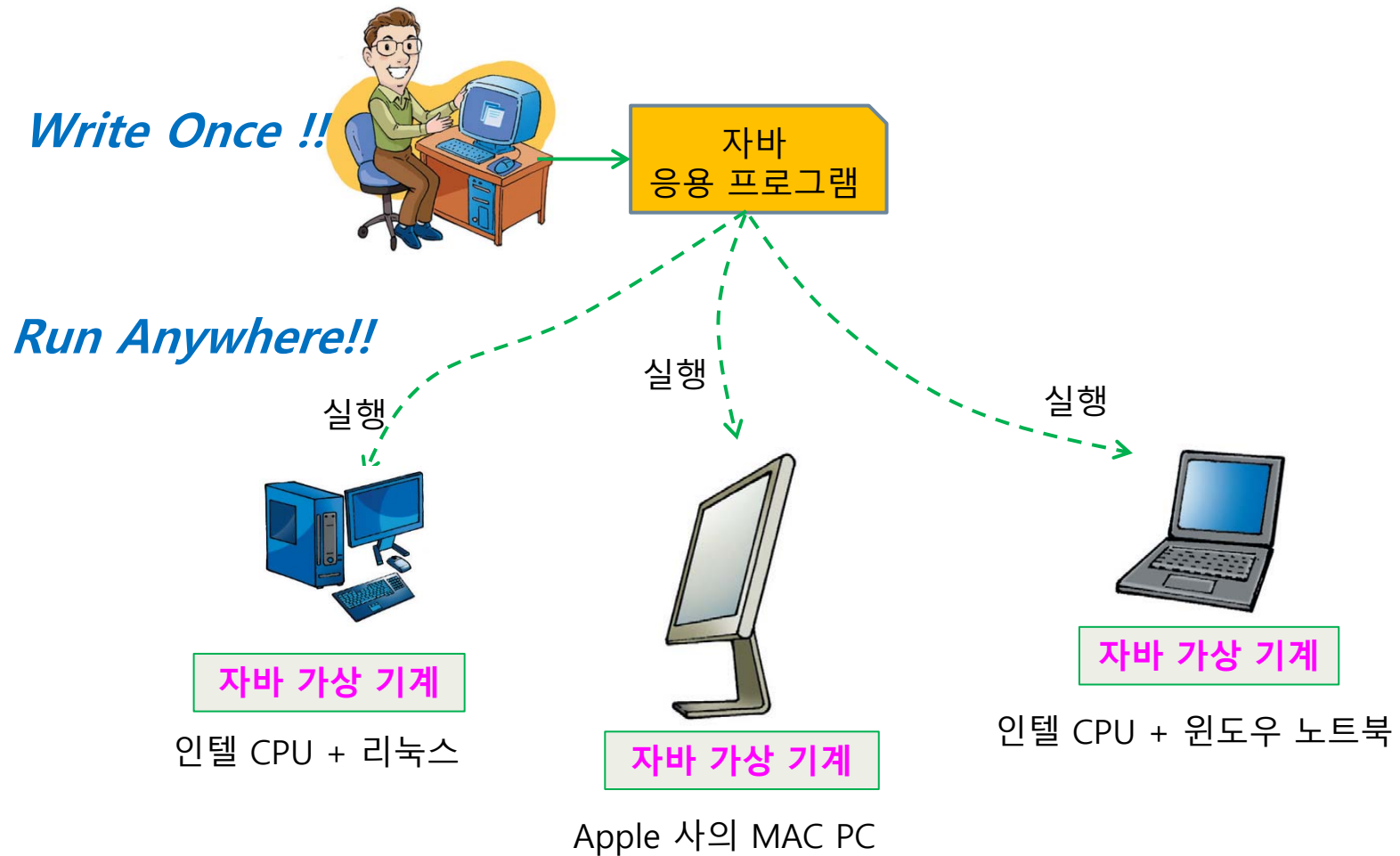
플랫폼 종속성(platform dependency)

8



자바의 플랫폼 독립성, **WORA**

9



자바의 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 가상 기계와 자바 응용프로그램의 실행

11

* 자바는 링크 과정 없음



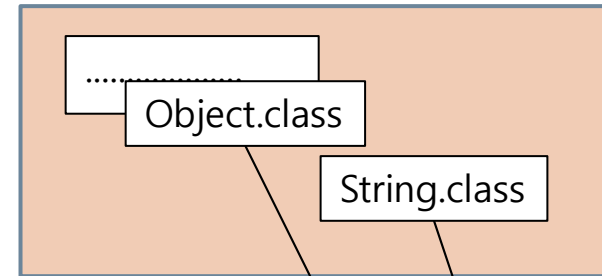
자바 프로그래밍

Draw.java
Hello.java
Shape.java
(소스 코드)

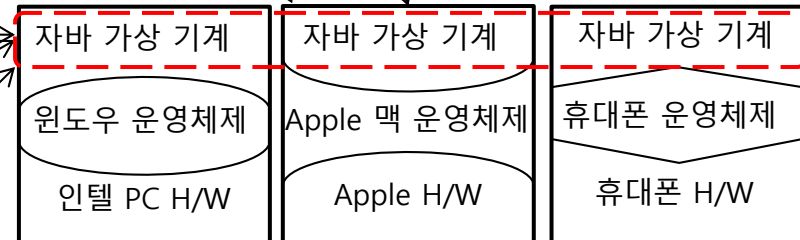
자바 컴파일러

Draw.class
Hello.class
Shape.class
(바이트 코드)

실행에 필요한 자바 클래스 라이브러리(JDK APIs)



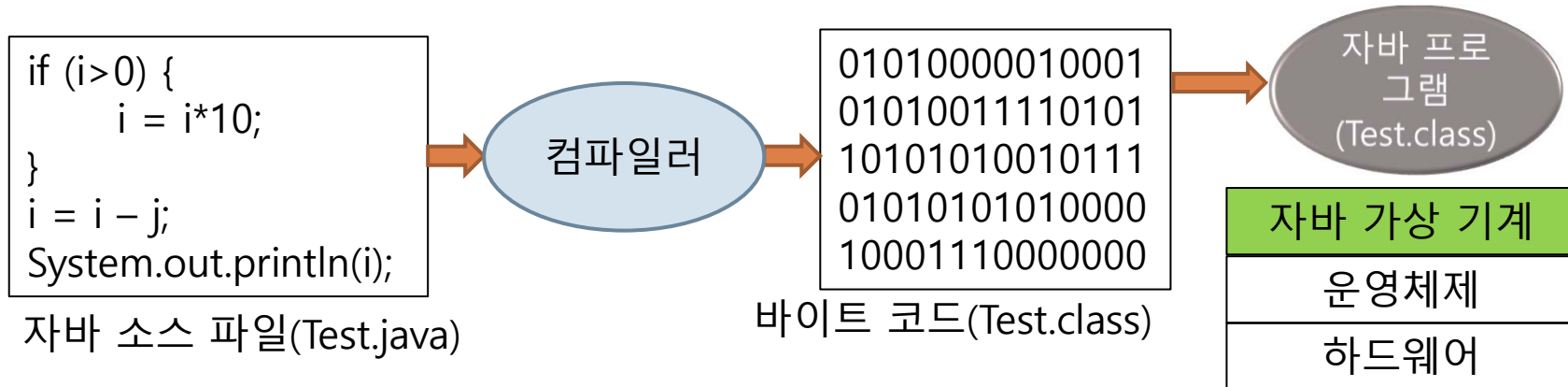
클래스 로딩



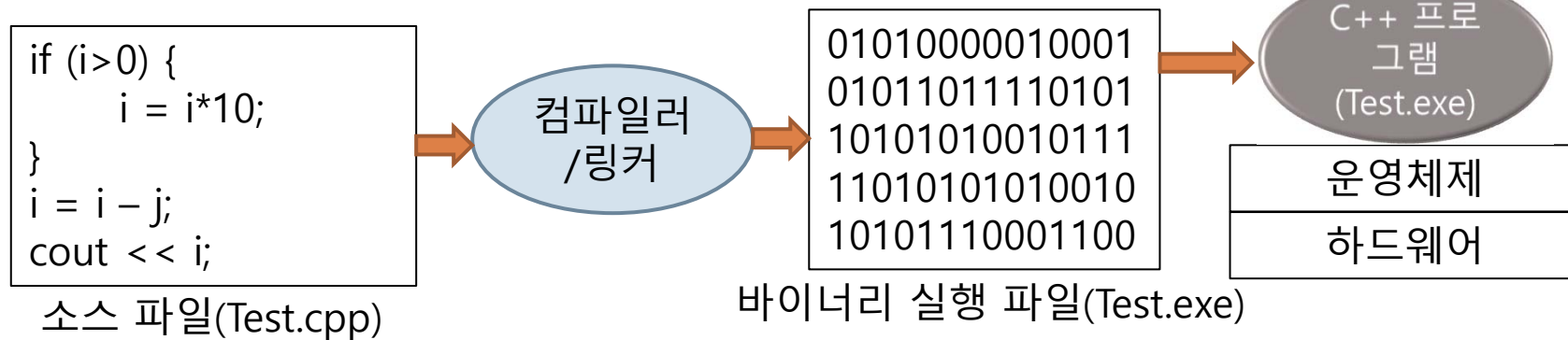
자바와 C/C++의 실행 환경 차이

12

□ 자바



□ C/C++



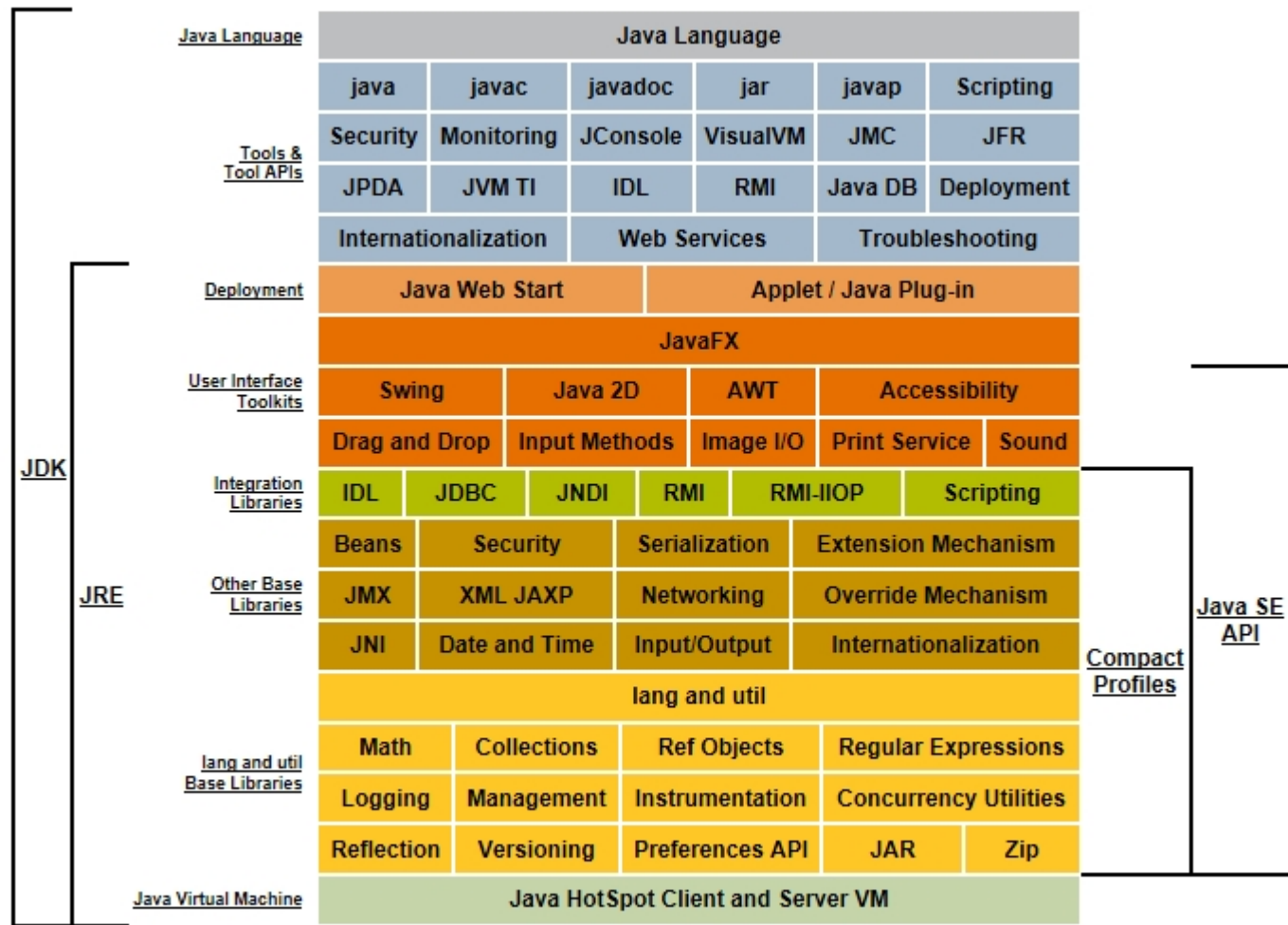
자바의 배포판 종류

13

- 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

Java SE 구성

14



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

JDK와 JRE

15

□ 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

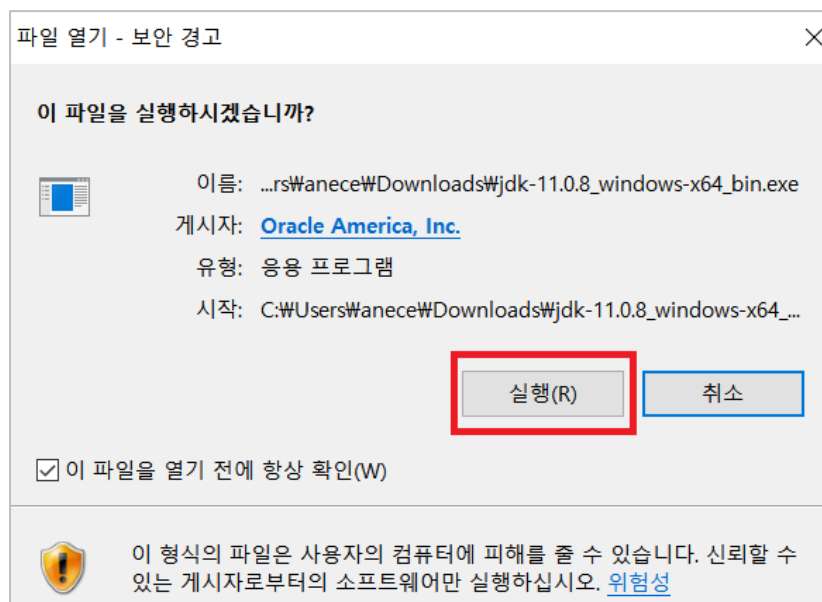
- javac - 자바 소스를 바이트 코드로 변환하는 컴파일러
- java - jre의 bin 디렉터리에 있는 자바 응용프로그램 실행기
- jar - 자바 아카이브 파일 (JAR)의 생성 및 관리하는 유틸리티
- jdb - 자바 디버거
- appletviewer - 웹 브라우저 없이 애플릿을 실행하는 유틸리티

JDK와 JRE

16

설치

jdk-11.0.8_windows-x64_bin.exe 더블클릭

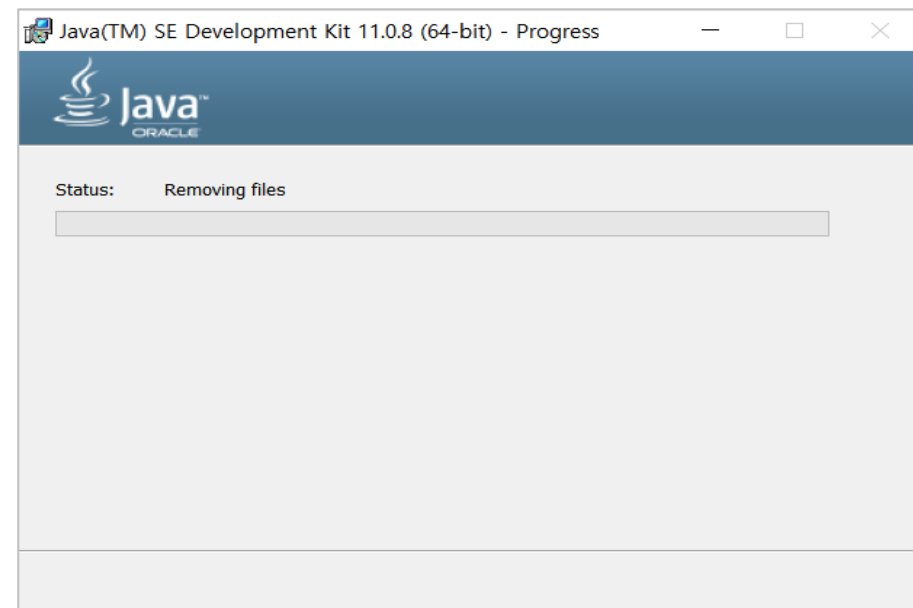
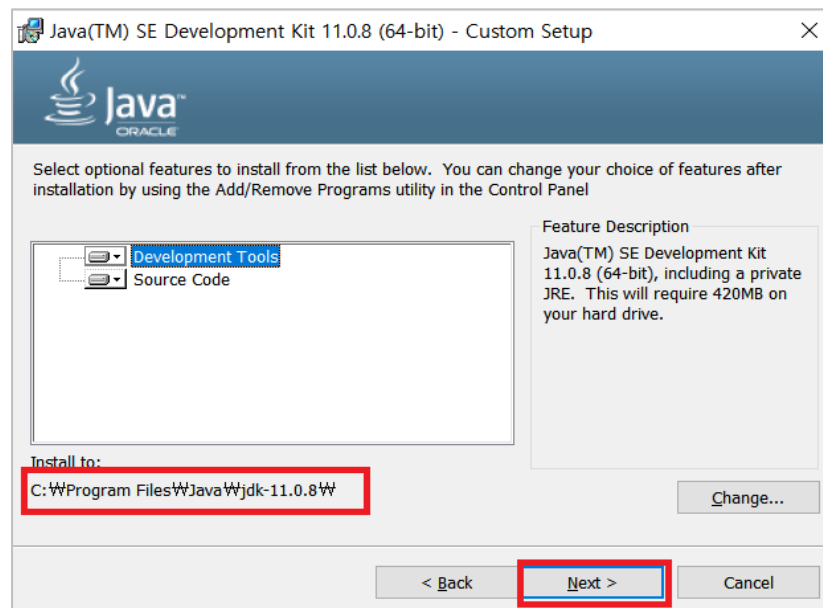


JDK와 JRE

17

설치

설치경로 설정 & 설치 진행

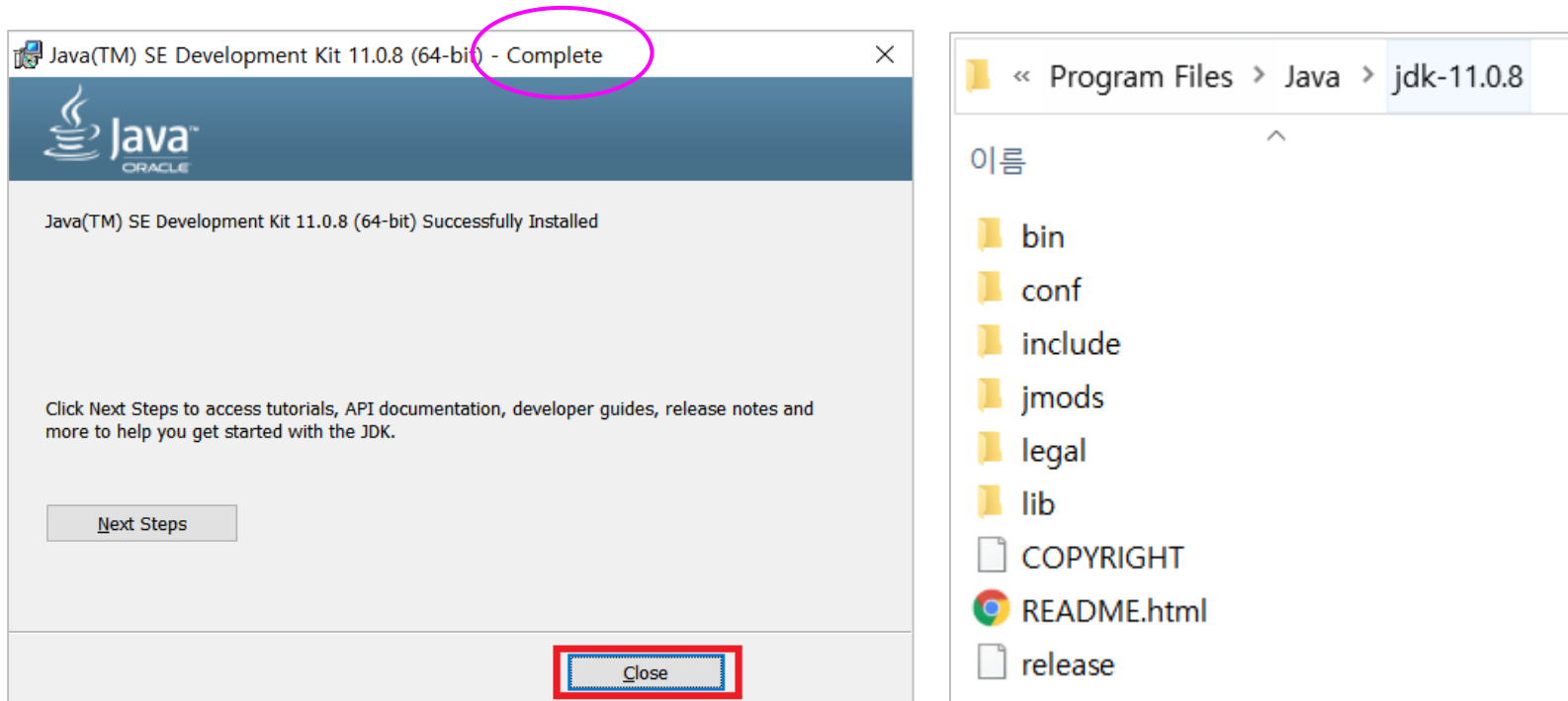


JDK와 JRE

18

설치

설치 완료 & 설치 폴더 구조

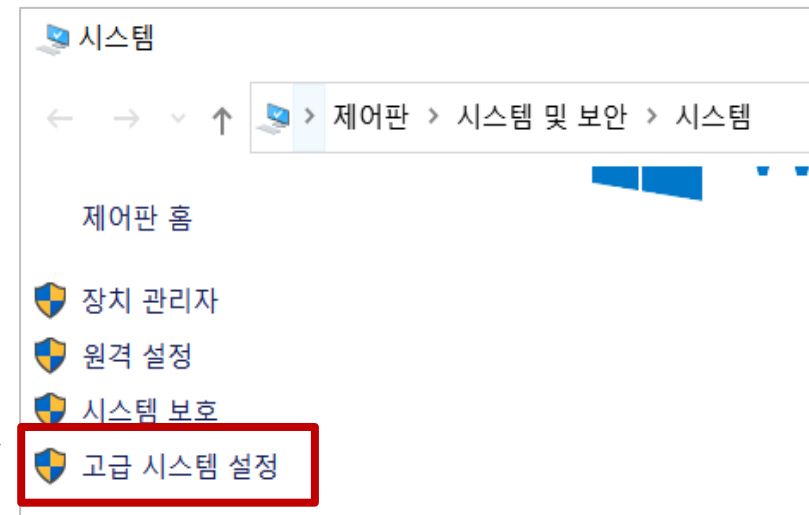
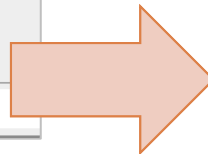
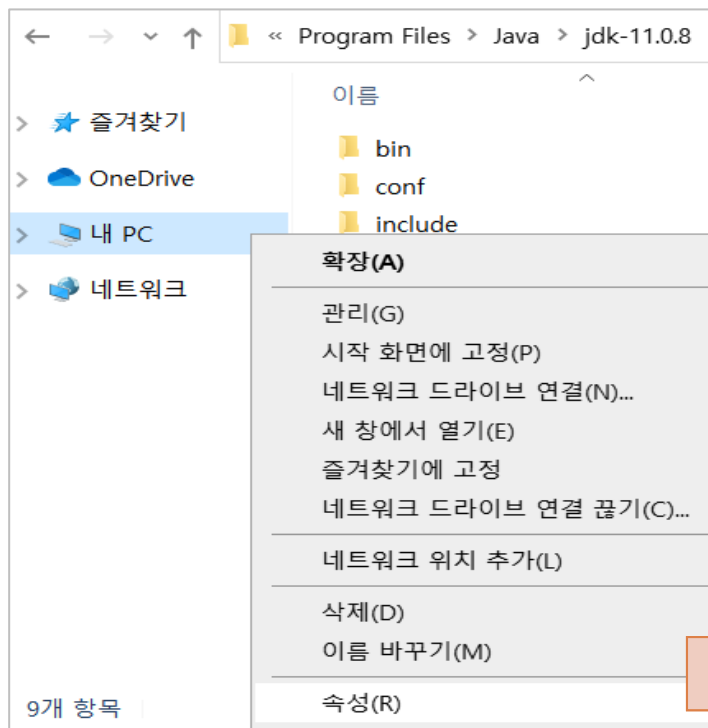


JDK와 JRE

19

□ 환경변수설정

▣ JAVA 명령어 사용 위한 설정

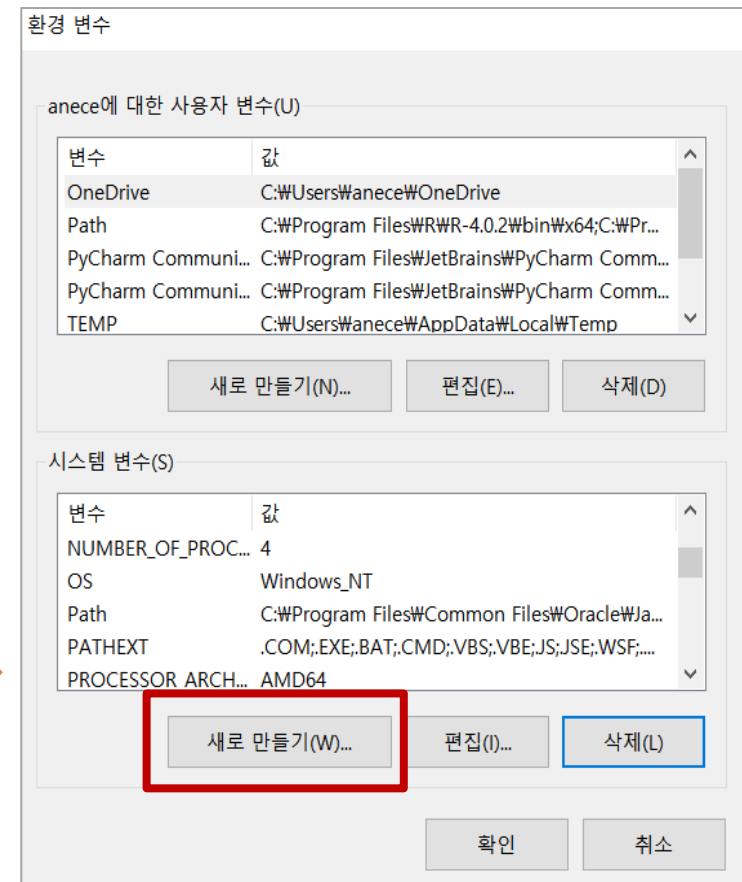
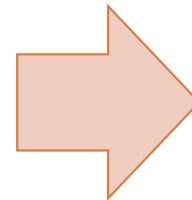
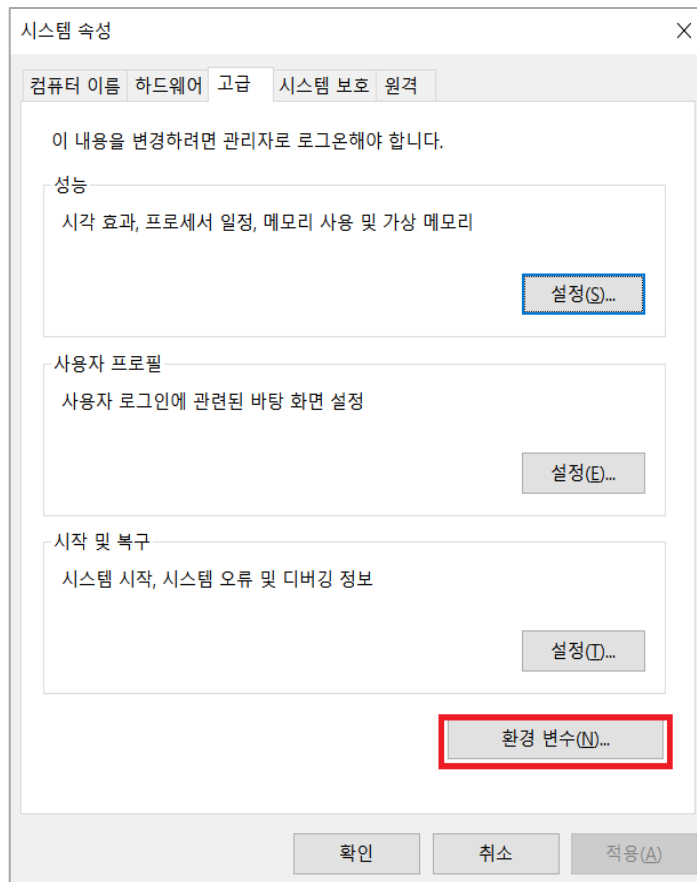


JDK와 JRE

20

□ 환경변수설정

▣ JAVA 명령어 사용 위한 설정

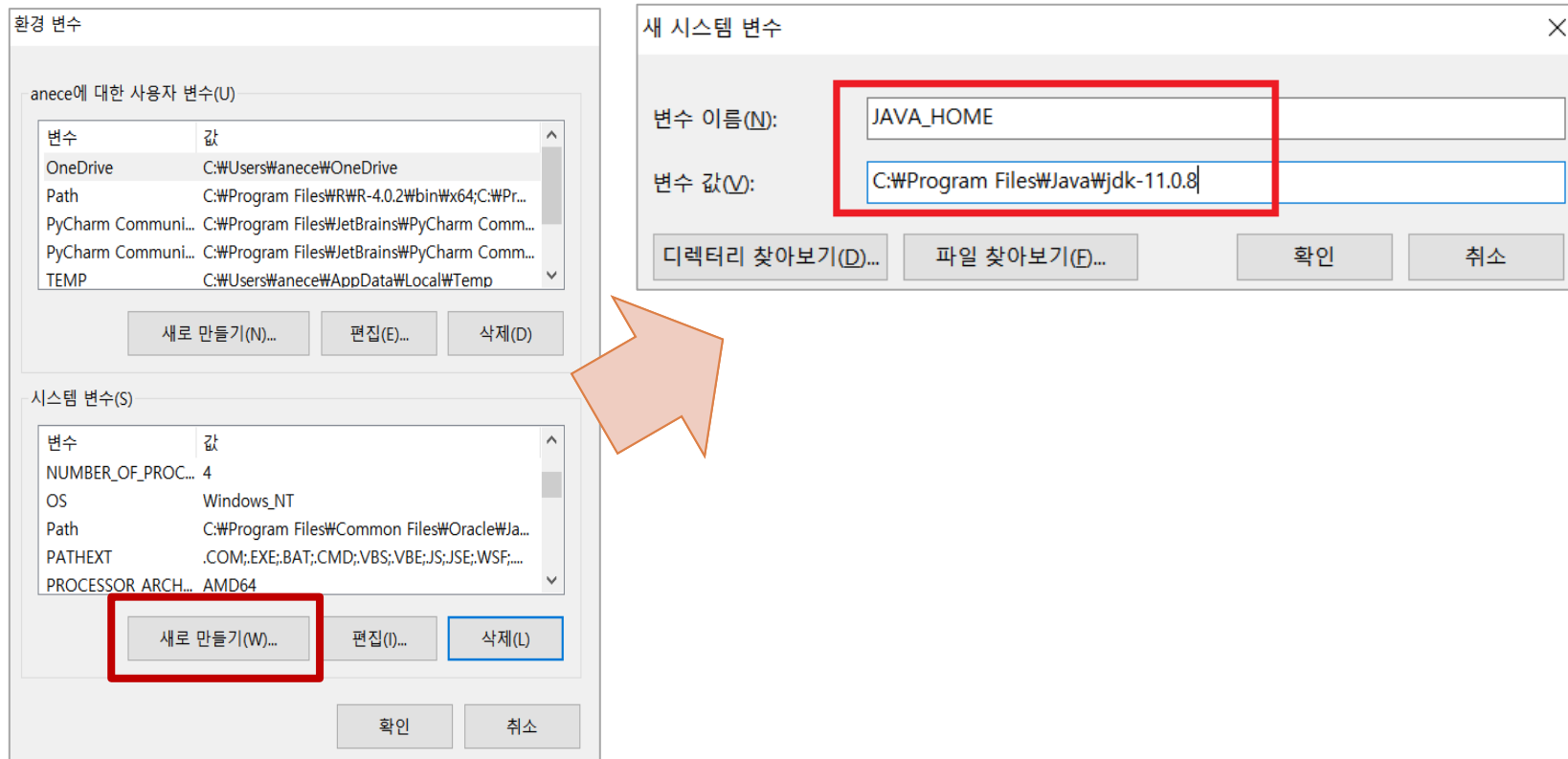


JDK와 JRE

21

□ 환경변수설정

▣ JAVA 명령어 사용 위한 설정

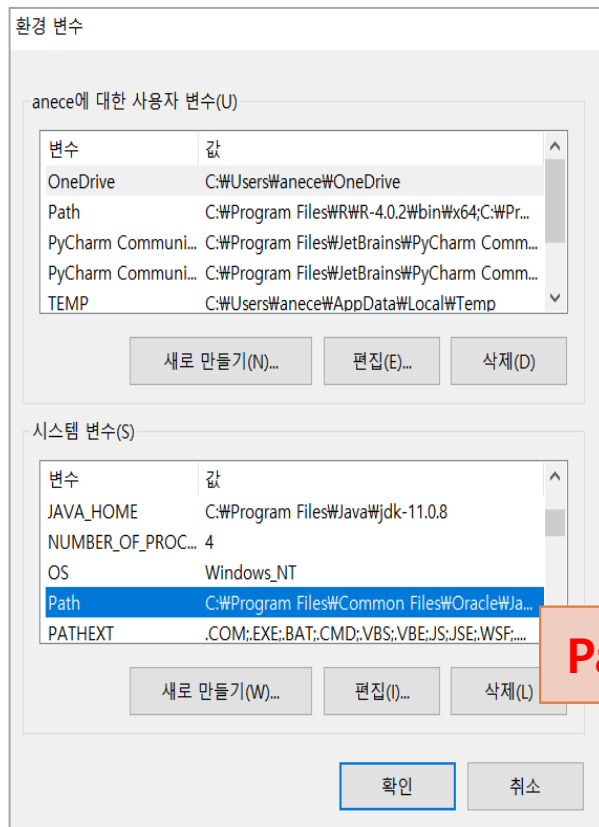


JDK와 JRE

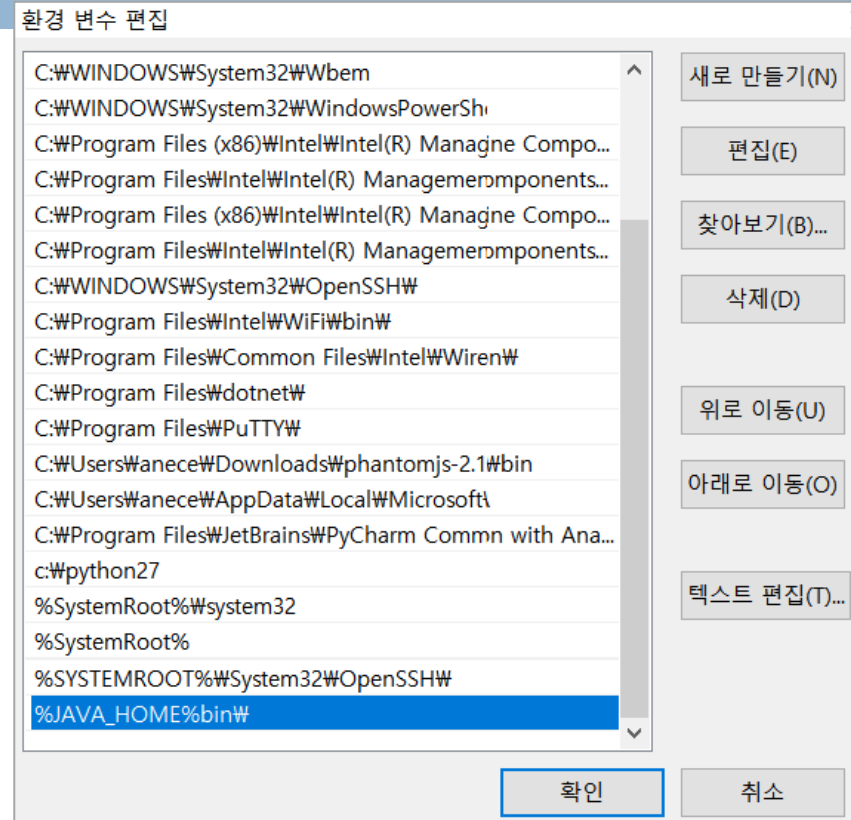
22

□ 환경변수설정

▣ JAVA 명령어 사용 위한 설정



Path

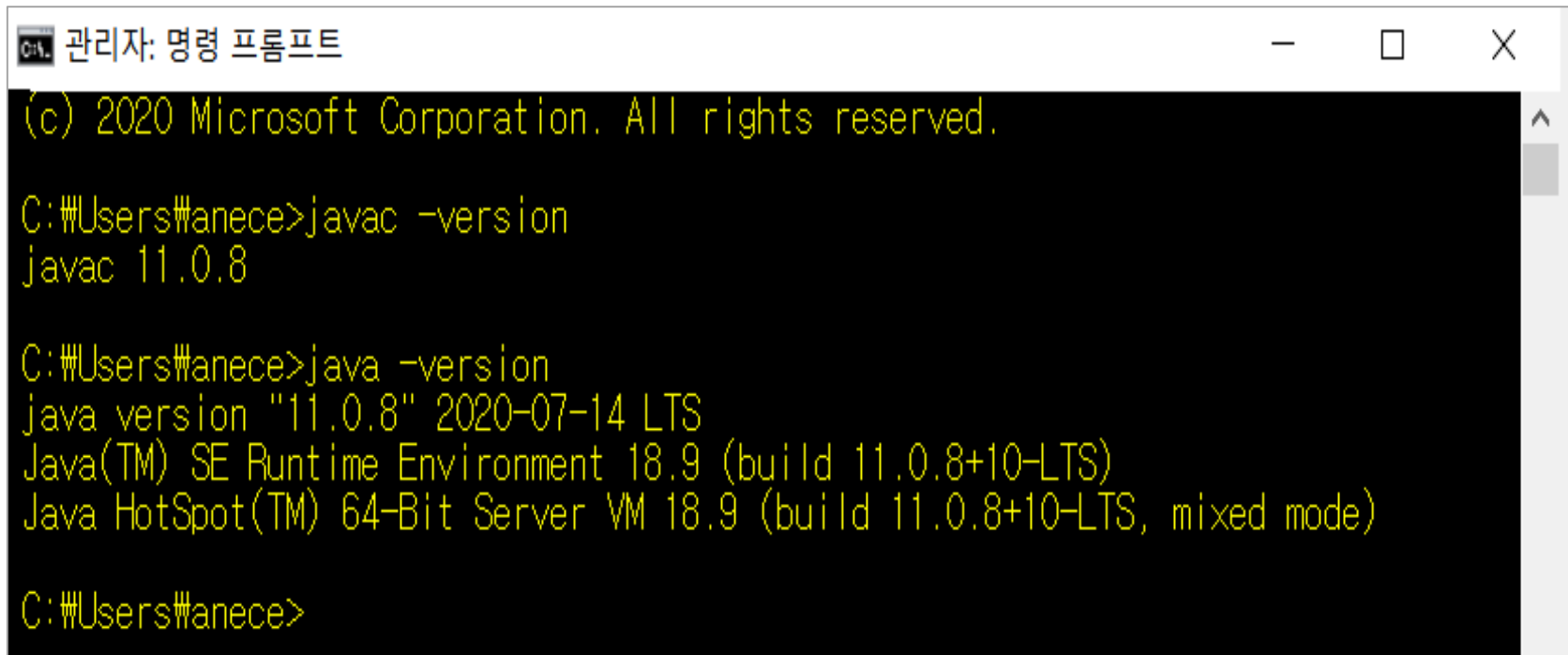


C:\Program Files\Java\jdk-11.0.8\bin
또는
%JAVA_HOME%\bin\

JDK와 JRE

23

- 환경변수설정
 - ▣ JAVA 명령어 사용 위한 설정



```
관리자: 명령 프롬프트
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\anece>javac -version
javac 11.0.8

C:\Users\anece>java -version
java version "11.0.8" 2020-07-14 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.8+10-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.8+10-LTS, mixed mode)

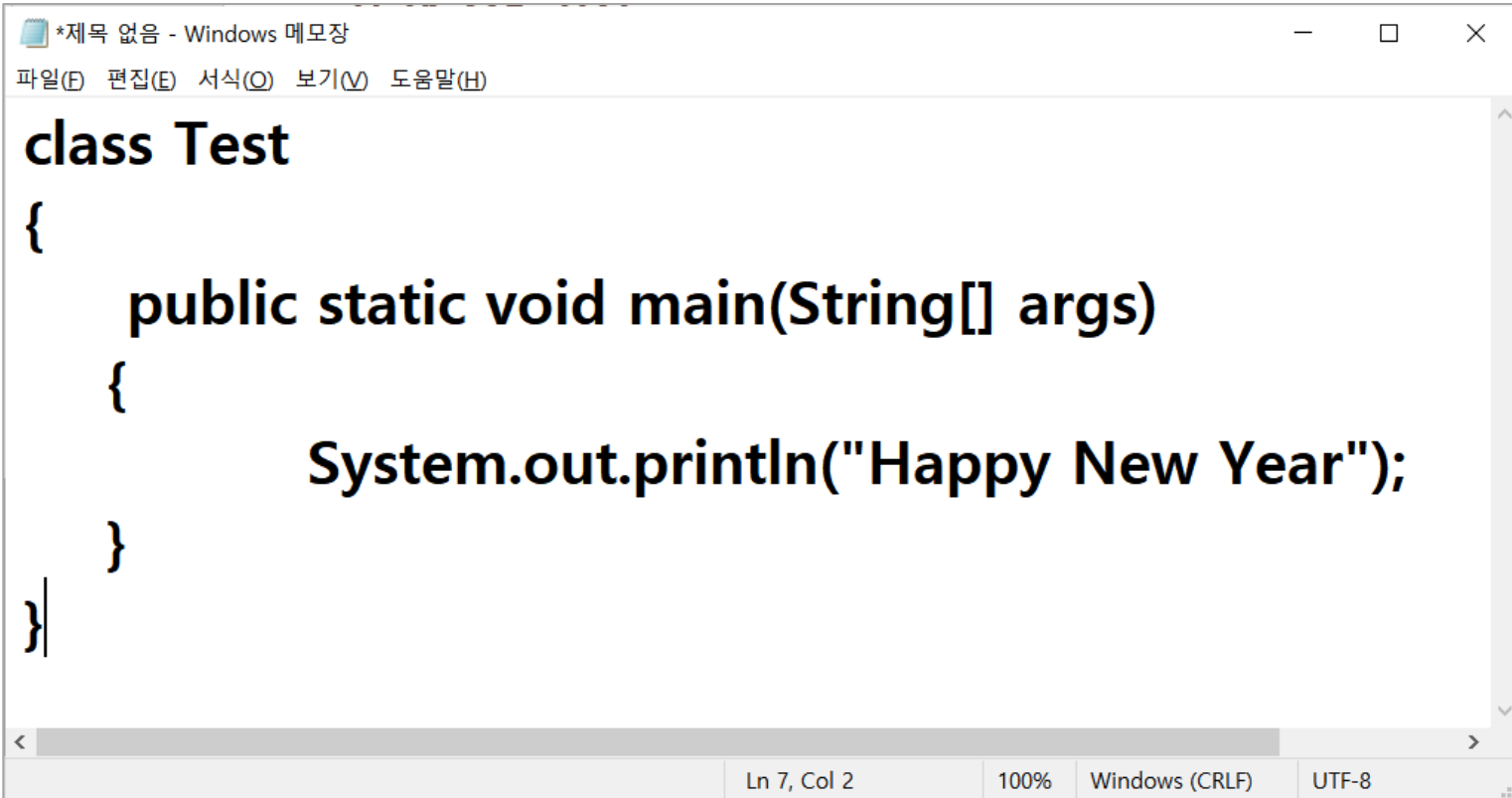
C:\Users\anece>
```

JDK와 JRE

24

□ 설정 확인하기

▣ 메모장 실행 >> 파일 작성 => **Test.java**



```
*제목 없음 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

class Test
{
    public static void main(String[] args)
    {
        System.out.println("Happy New Year");
    }
}
```

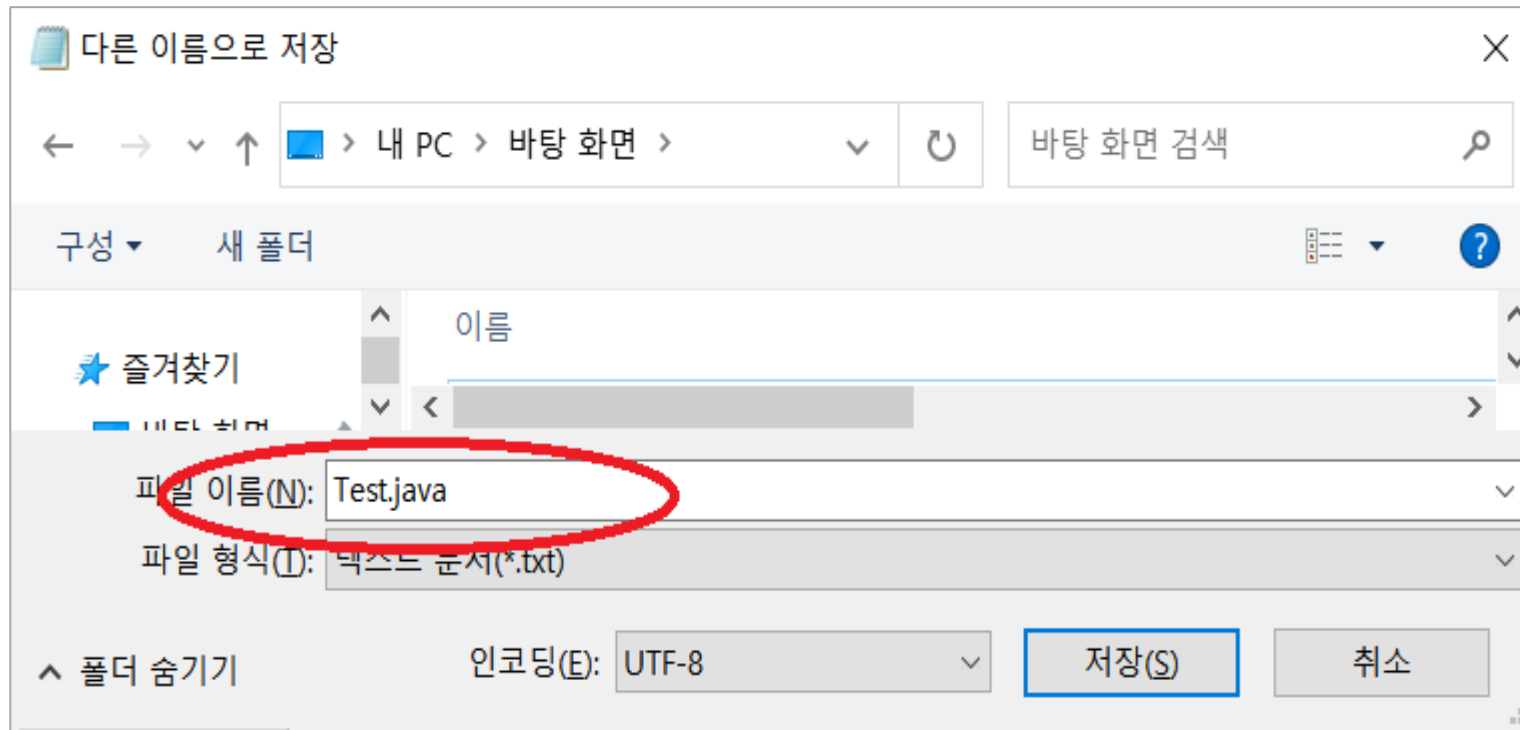
Ln 7, Col 2 100% Windows (CRLF) UTF-8

JDK와 JRE

25

□ 설정 확인하기

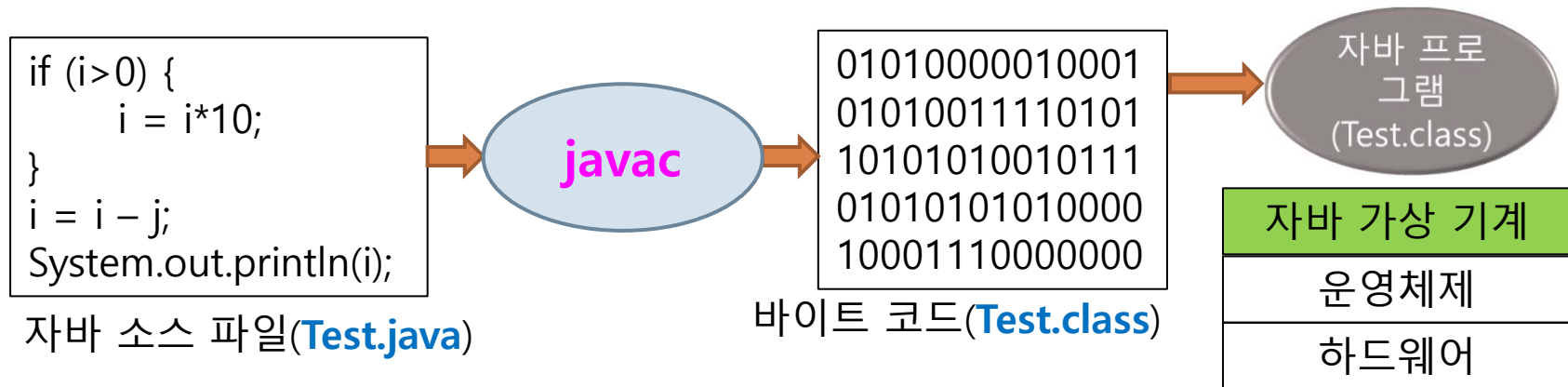
▣ 파일 저장 >> Test.java



자바와 C/C++의 실행 환경 차이

26

□ 자바

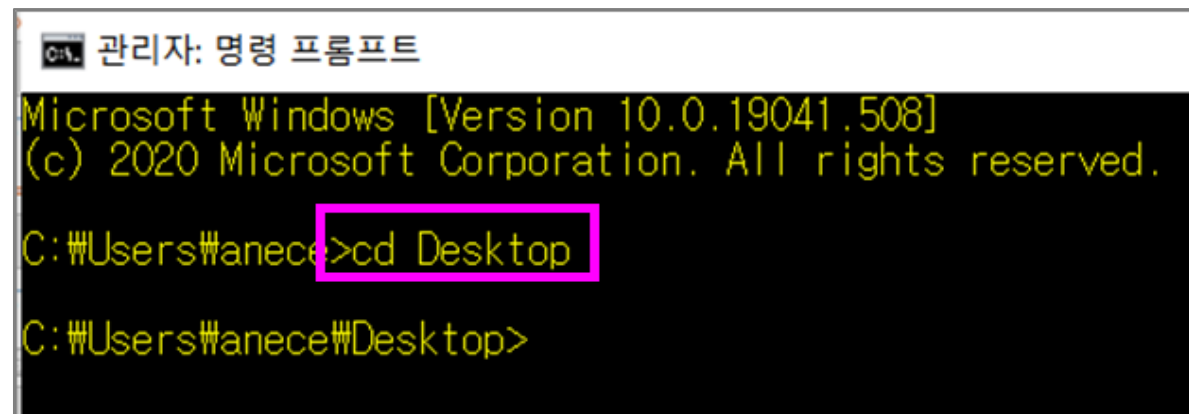
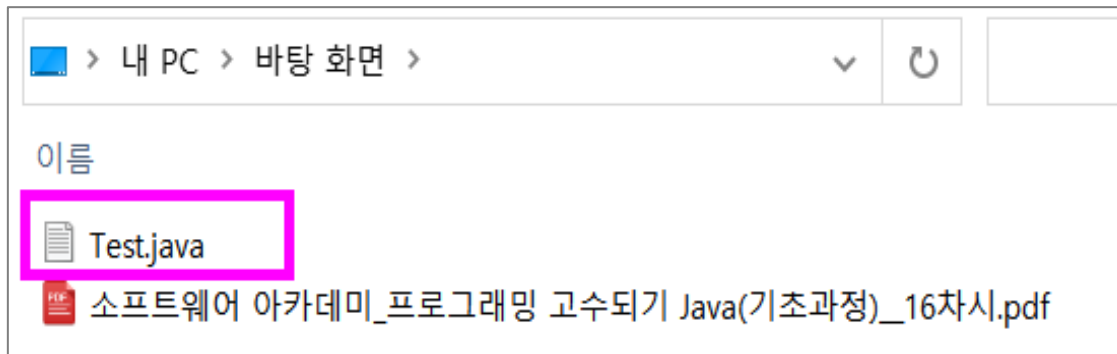


JDK와 JRE

27

□ 설정 확인하기

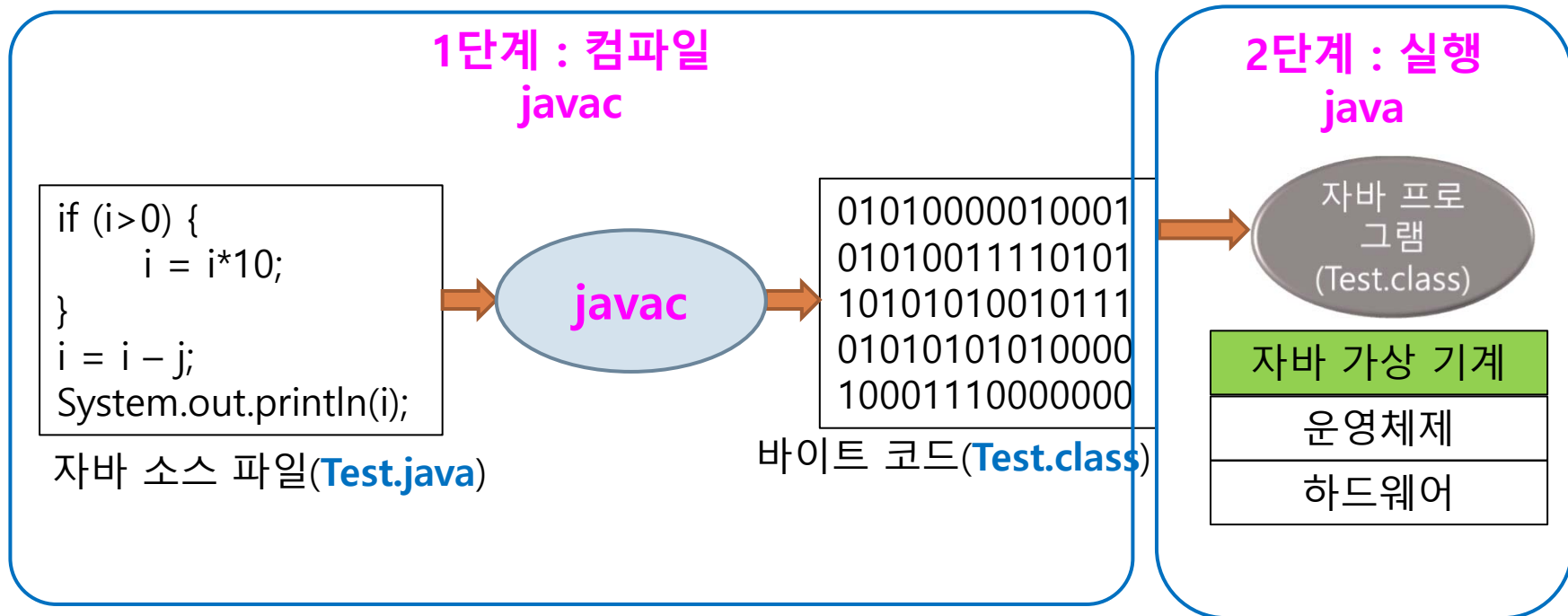
▣ Java 실행 >> cmd 실행



자바와 C/C++의 실행 환경 차이

28

□ 자바

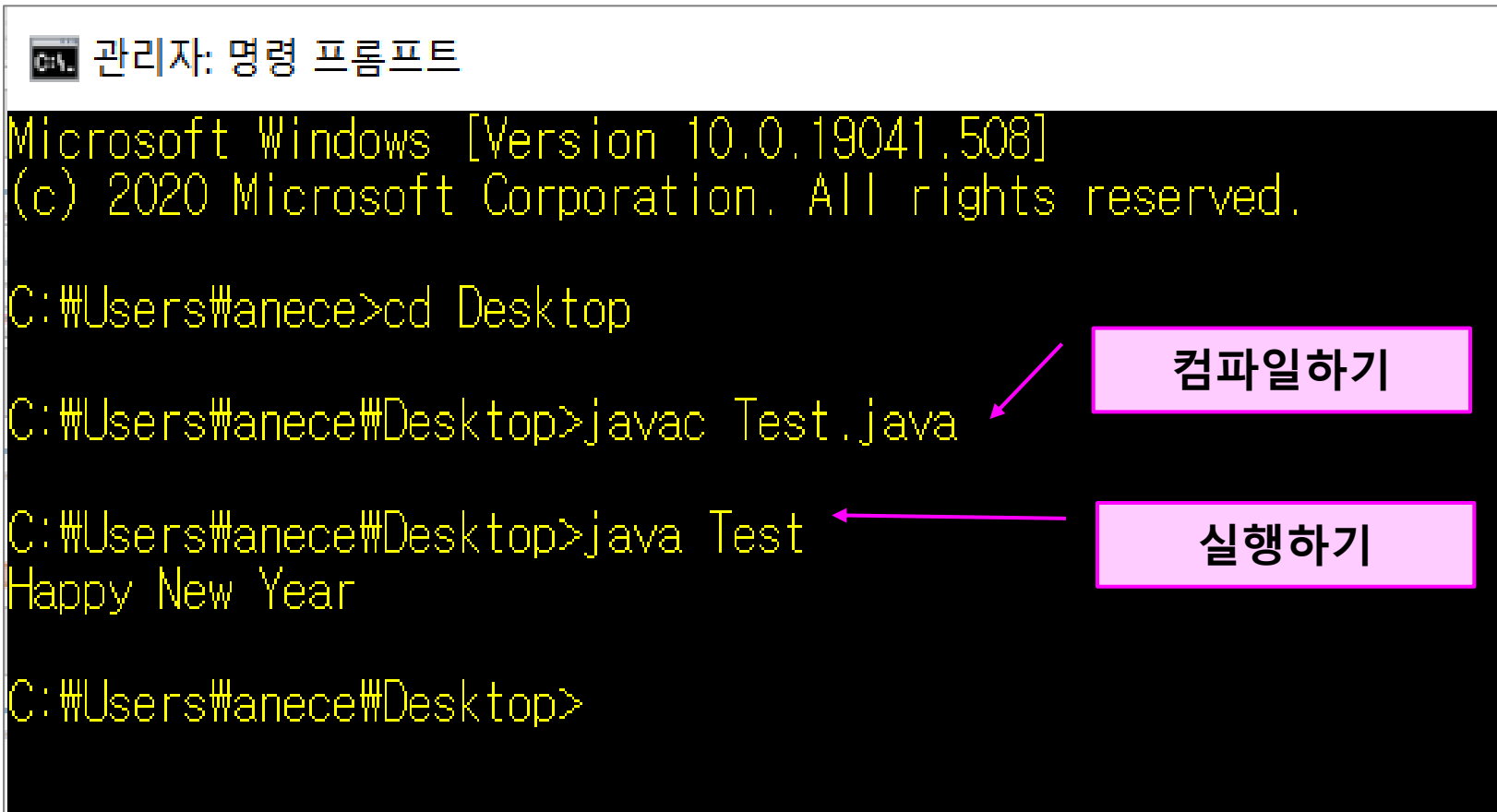


JDK와 JRE

29

□ 설정 확인하기

▣ Java 실행 >> cmd 실행



The screenshot shows a Windows Command Prompt window titled "관리자: 명령 프롬프트". The text inside the window is as follows:

```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\#anece>cd Desktop
C:\Users\#anece\Desktop>javac Test.java
C:\Users\#anece\Desktop>java Test
Happy New Year
C:\Users\#anece\Desktop>
```

Two pink boxes with arrows point to specific lines in the command prompt:

- A pink box labeled "컴파일하기" (Compile) has an arrow pointing to the line `C:\Users\#anece\Desktop>javac Test.java`.
- A pink box labeled "실행하기" (Execute) has an arrow pointing to the line `C:\Users\#anece\Desktop>java Test`.

자바 통합 개발 환경-이클립스(Eclipse)

30

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

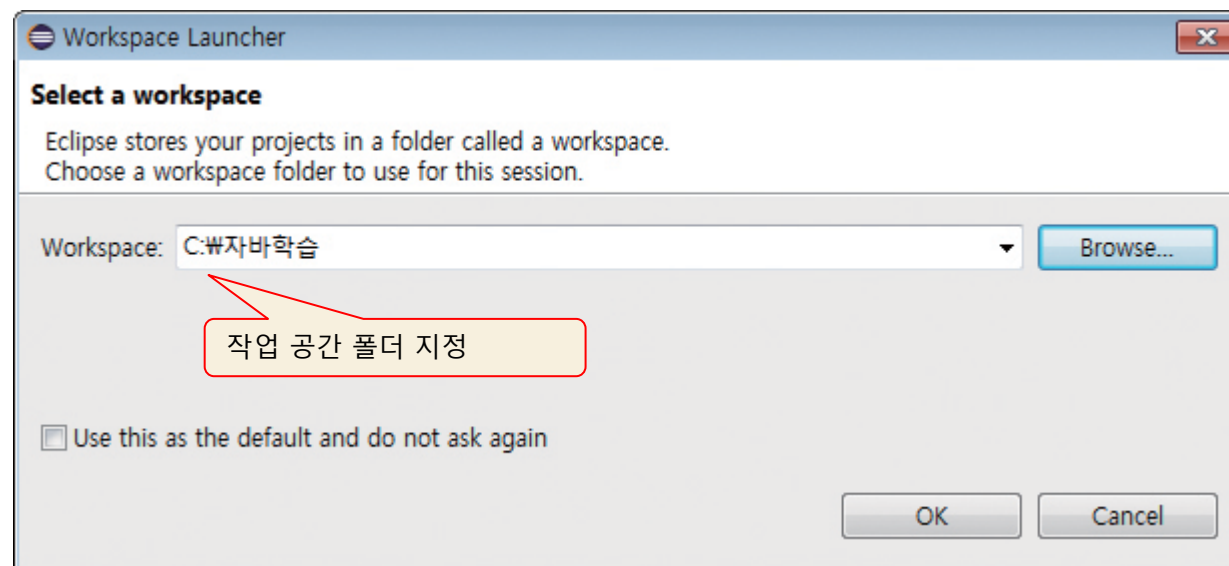
- 이클립스(Eclipse)
 - ▣ 자바 응용 프로그램 개발을 위한 통합 개발 환경
 - ▣ IBM에 의해 개발된 오픈 소스 프로젝트
 - ▣ <http://www.eclipse.org/downloads/> 에서 다운로드

이클립스 실행

31



이클립스 Luna 배포판

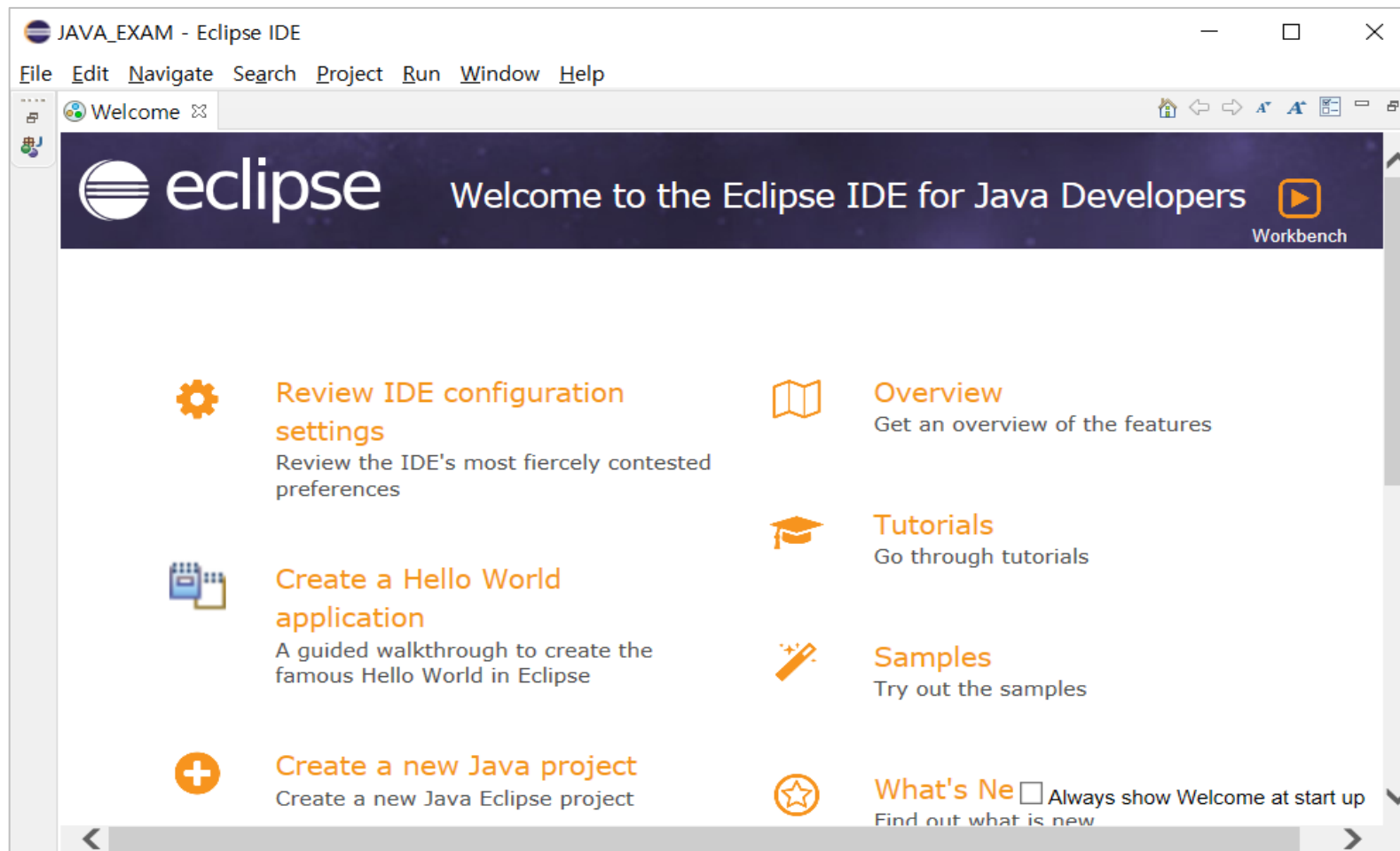


작업 공간 폴더 지정

자바 통합 개발 환경-이클립스(Eclipse)

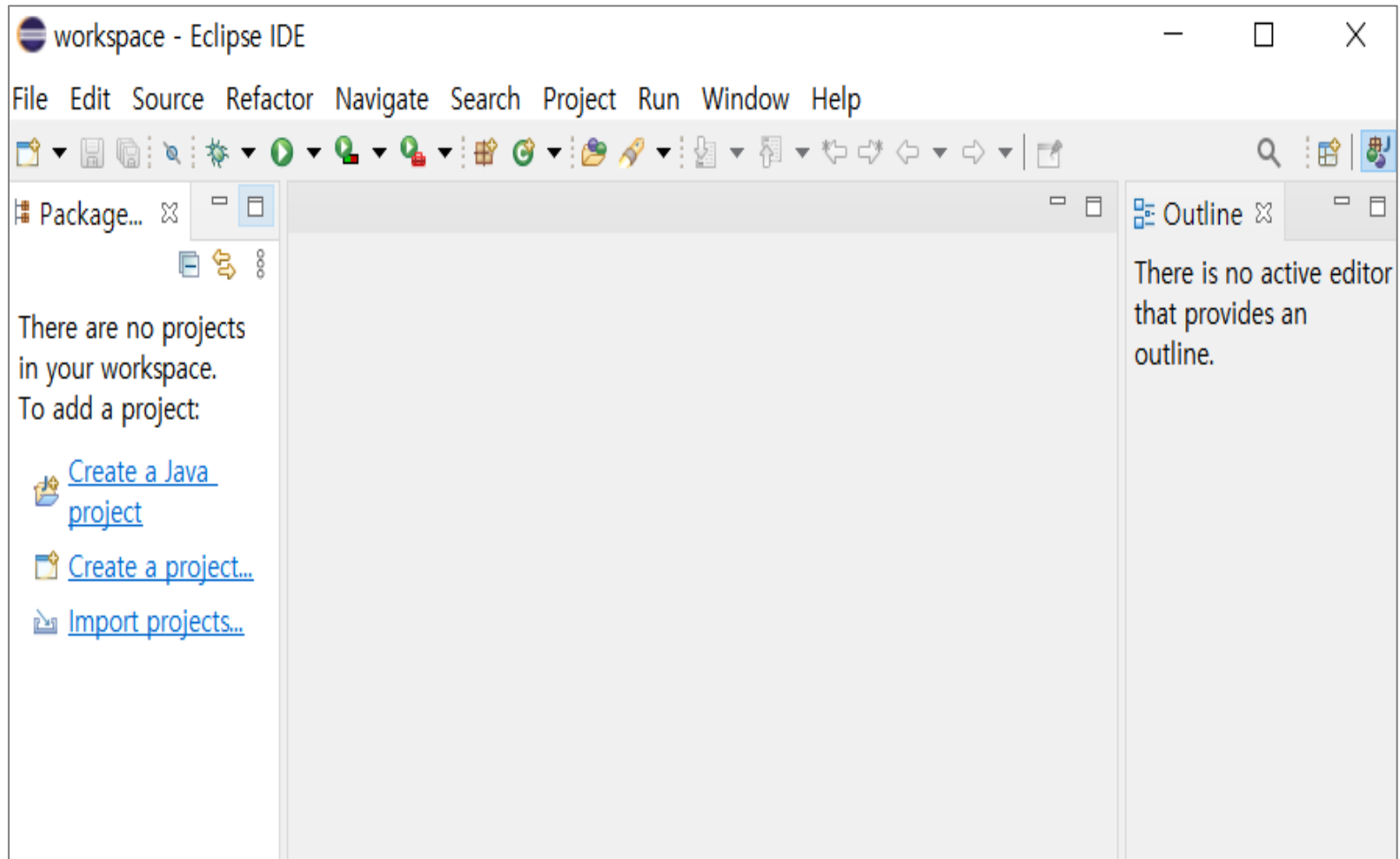
32

□ IDE(Integrated Development Environment)



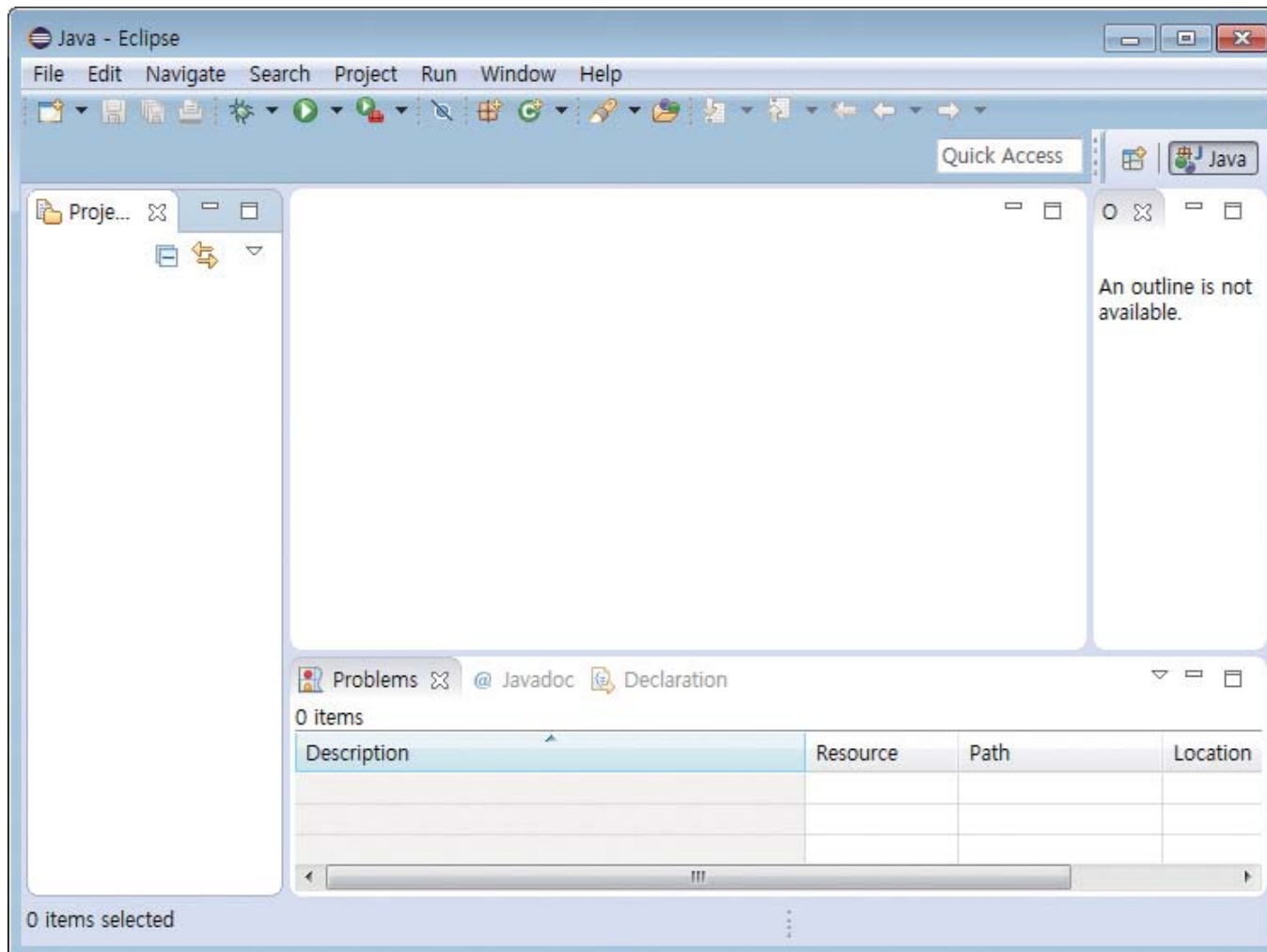
이클립스의 사용자 인터페이스

33



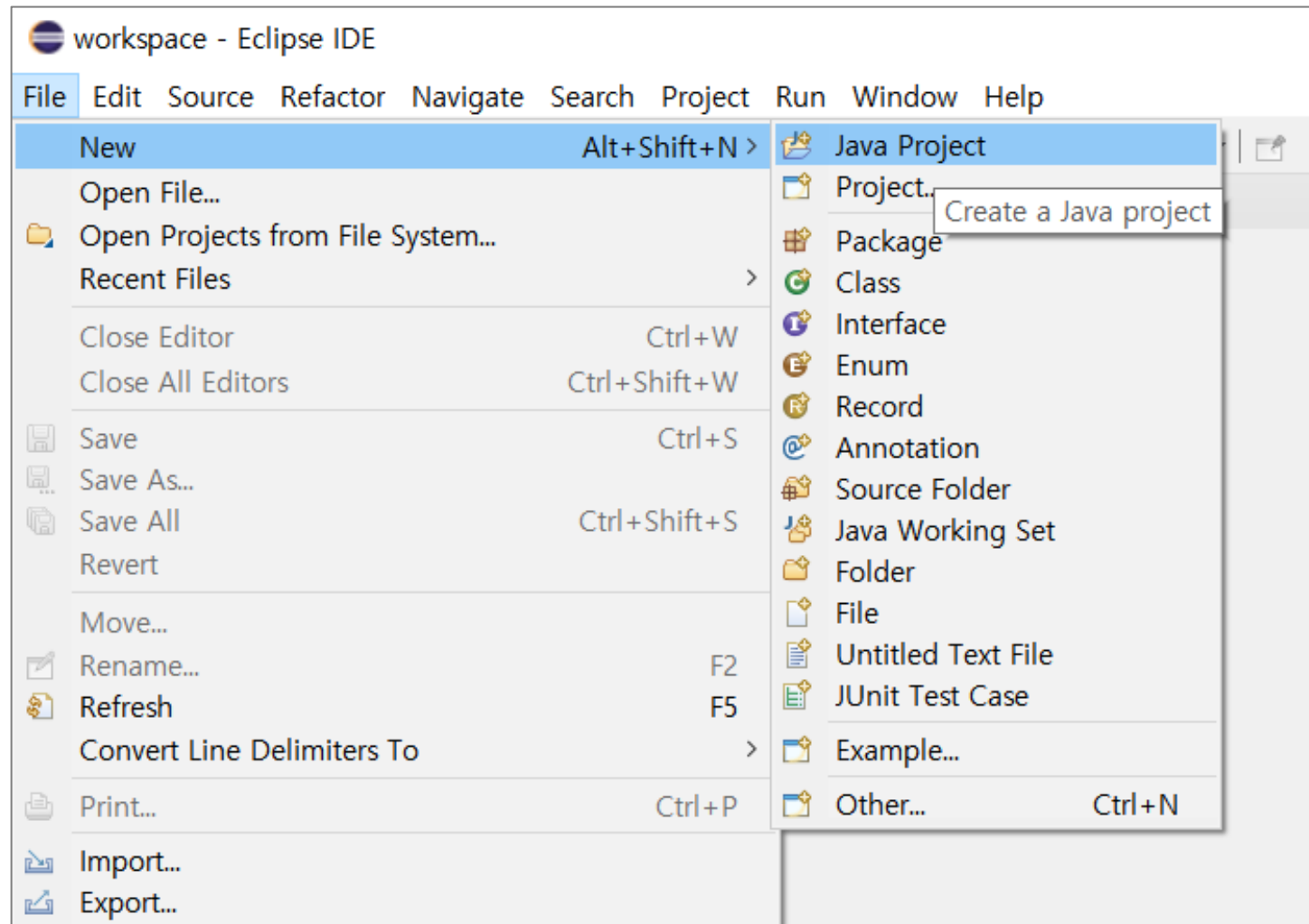
이클립스의 사용자 인터페이스

34



프로젝트 생성

35



프로젝트 생성

36

New Java Project

Create a Java Project

Create a Java project in the workspace location.

Project name: Hello

☒ Use default location

Location: C:\Users\Wanece\Desktop\workspace\Hello [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-11

☐ Use a project specific JRE: jdk-11.0.8

☐ Use default JRE 'jdk-11.0.8' and workspace compiler preferences [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

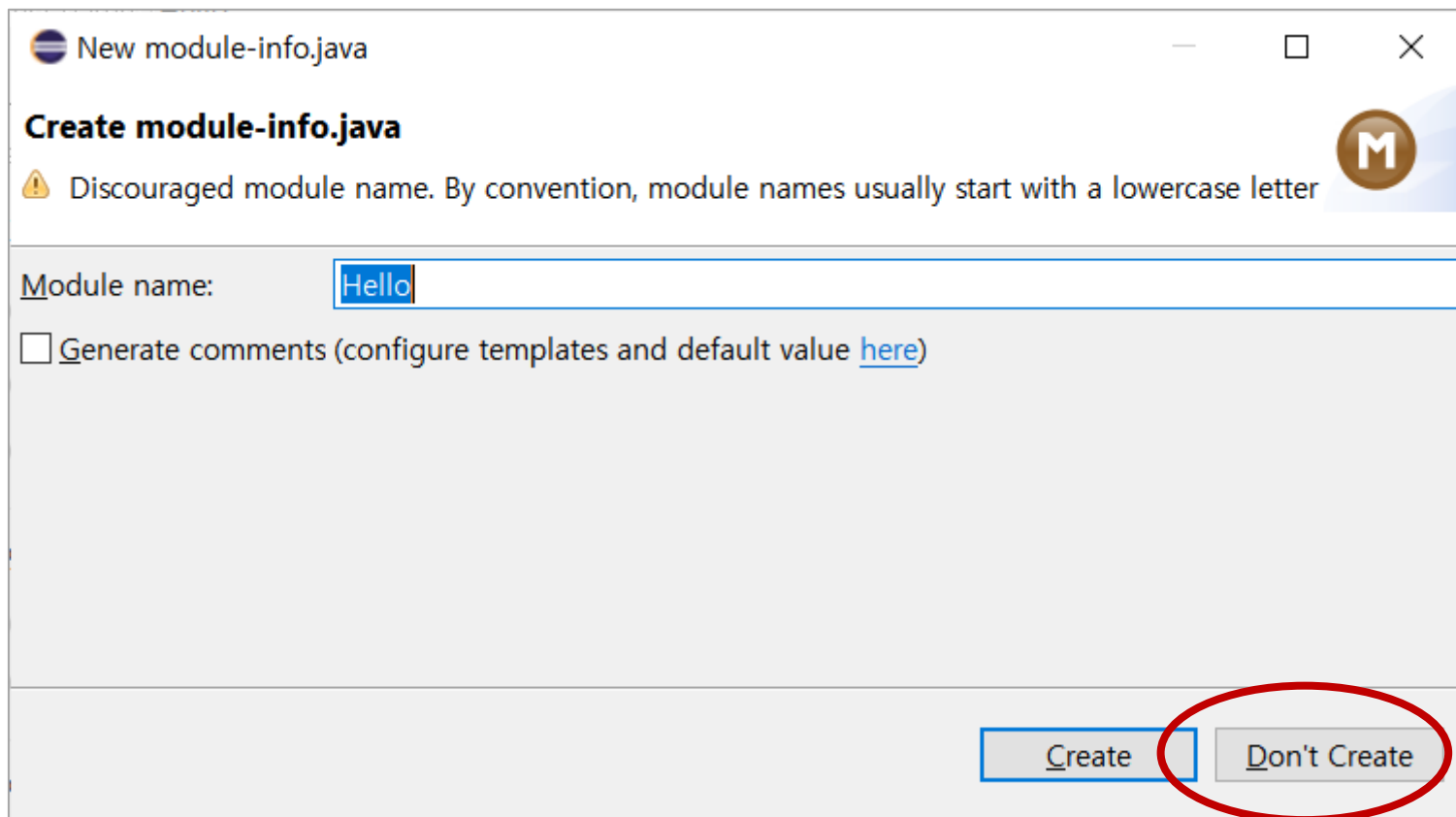
☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[? < Back](#) [Next >](#) **Finish** [Cancel](#)

프로젝트 생성

37



클래스 생성

38

File->New->Class 메뉴 선택

New Java Class

Java Class

The use of the default package is discouraged.

Source folder: Hello/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: HelloJava 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

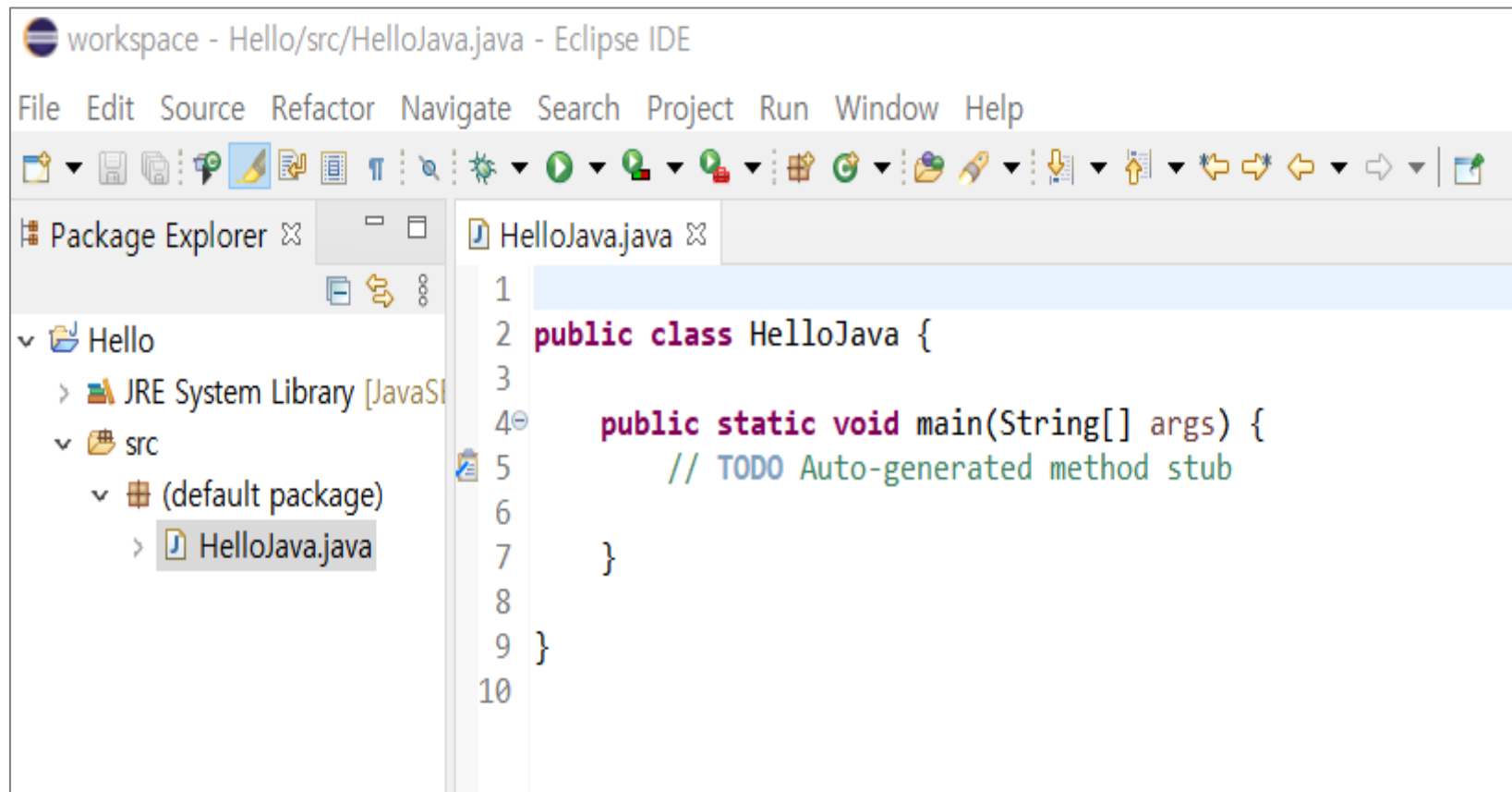
Finish 선택

Finish Cancel

main()을 체크하면 자동으로 main() 메소드 생성

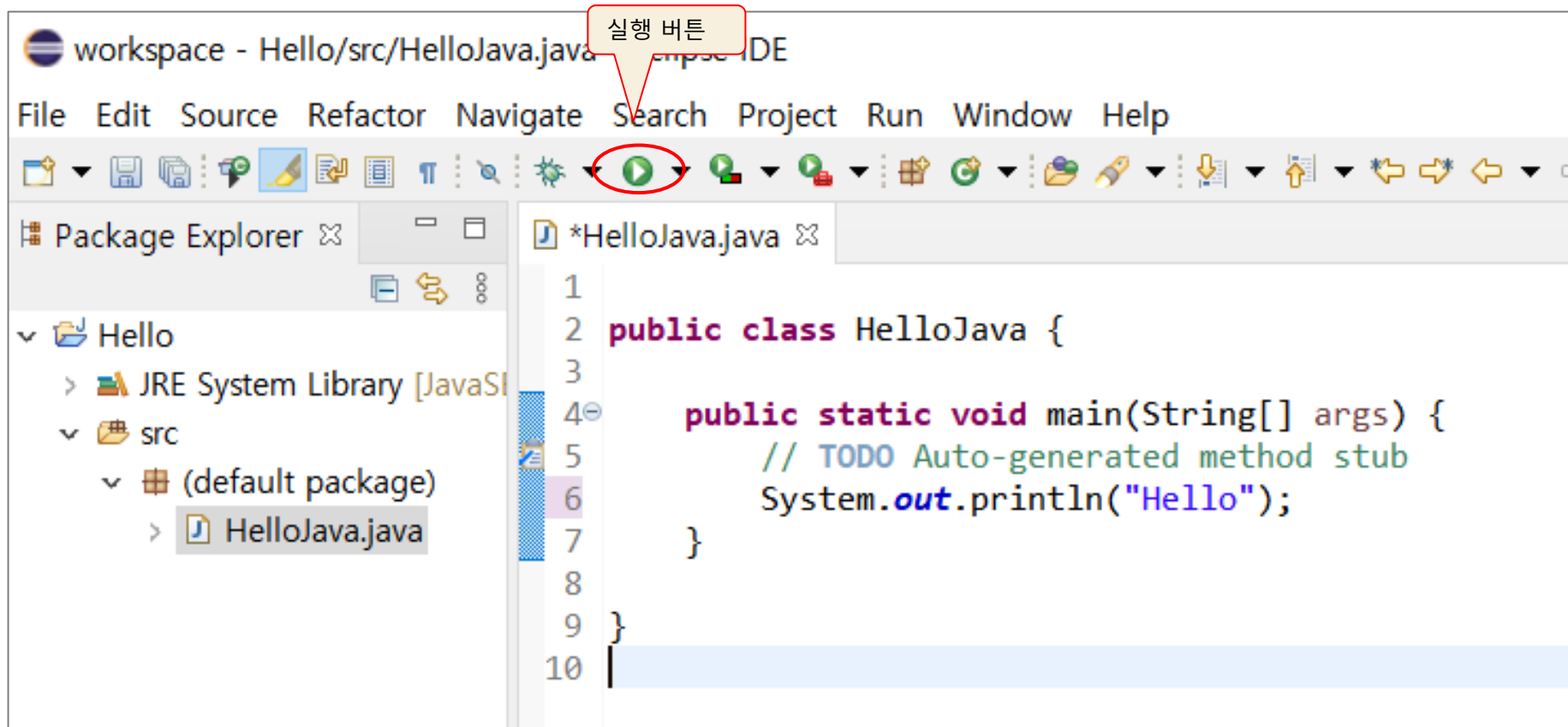
생성된 자바 소스

39



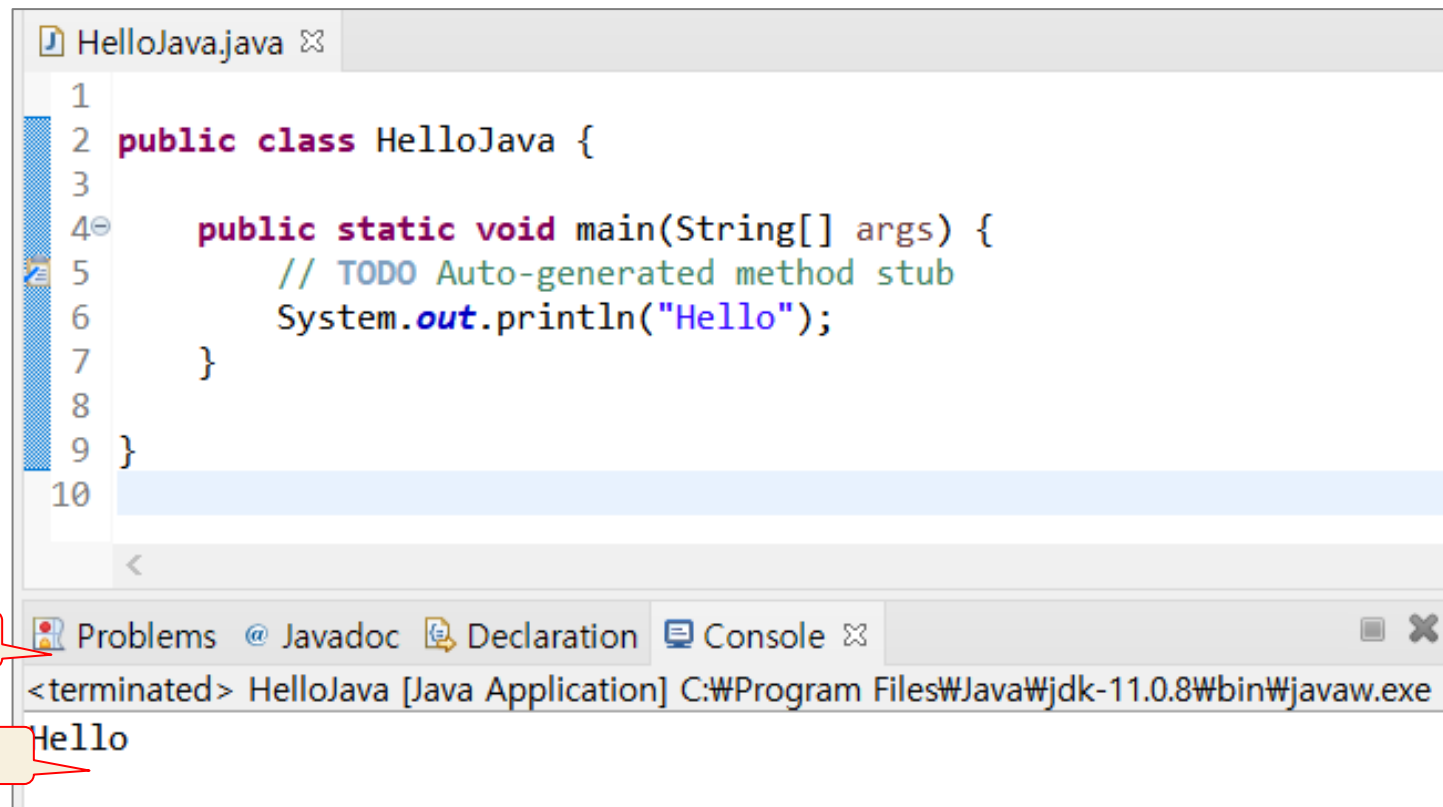
소스 편집과 컴파일 및 실행

40



소스 편집과 컴파일 및 실행

41



객체지향 언어의 목적

42

□ 소프트웨어의 생산성 향상

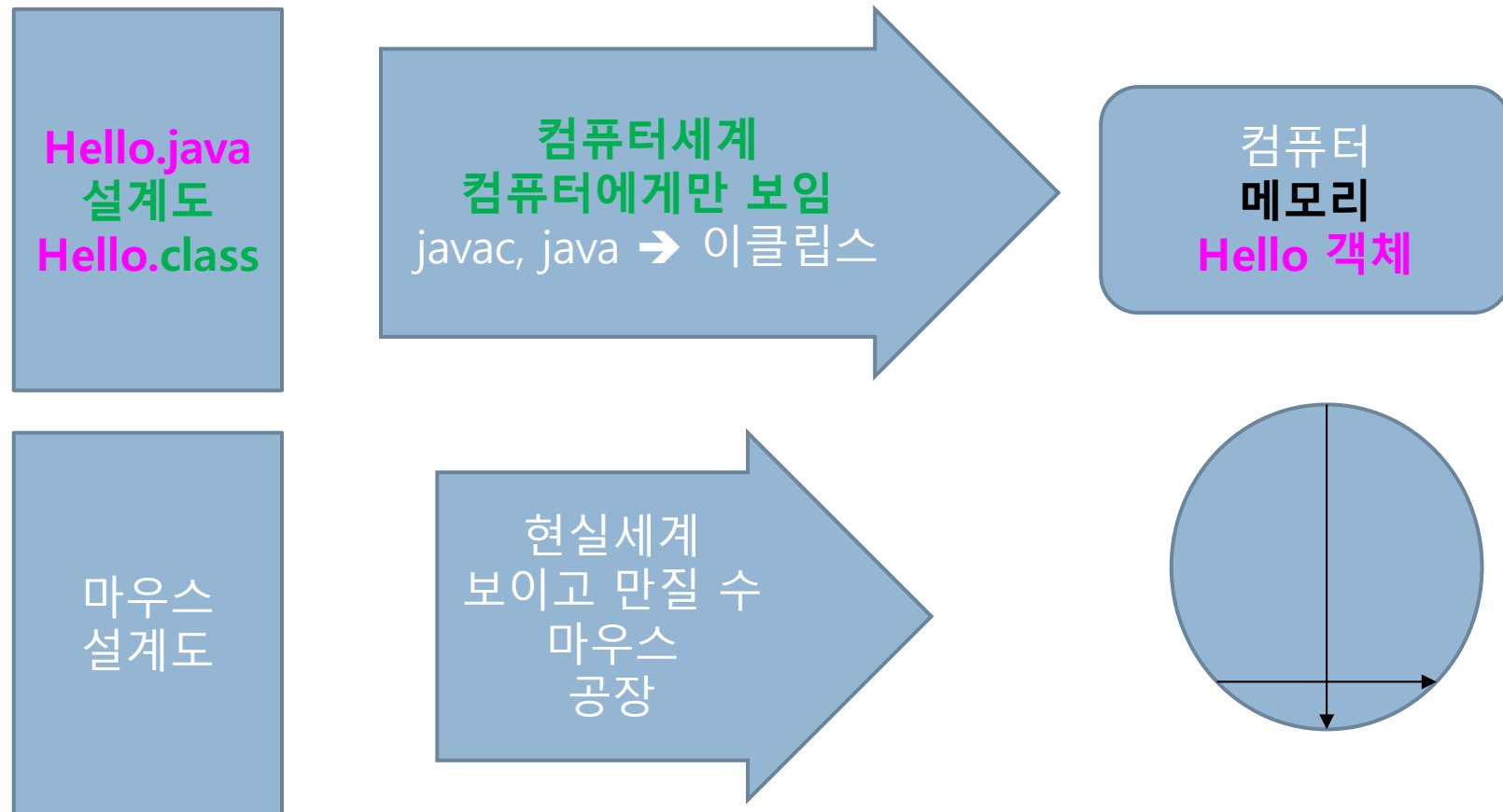
- 컴퓨터 산업 발전에 따라 소프트웨어의 생명 주기(life cycle) 단축
- 객체 지향 언어는 상속, 다형성, 객체, 캡슐화 등 소프트웨어 재사용을 위한 여러 장치 내장
 - 소프트웨어의 재사용과 부분 수정을 통해 소프트웨어를 다시 만드는 부담을 대폭 줄임으로써 소프트웨어의 생산성이 향상

□ 실세계에 대한 쉬운 모델링

- 과거
 - 수학 계산/통계 처리를 하는 등의 처리 과정, 계산 절차가 중요
- 현재
 - 컴퓨터가 산업 전반에 활용
 - 실세계에서 발생하는 일을 프로그래밍
 - 실세계에서는 절차나 과정보다 일과 관련된 물체(객체)들의 상호 작용으로 묘사하는 것이 용이
- 실세계의 일을 보다 쉽게 프로그래밍하기 위한 객체 중심의 객체 지향 언어 탄생

객체지향 언어의 목적

43



절차 지향 프로그래밍과 객체 지향 프로그래밍

44

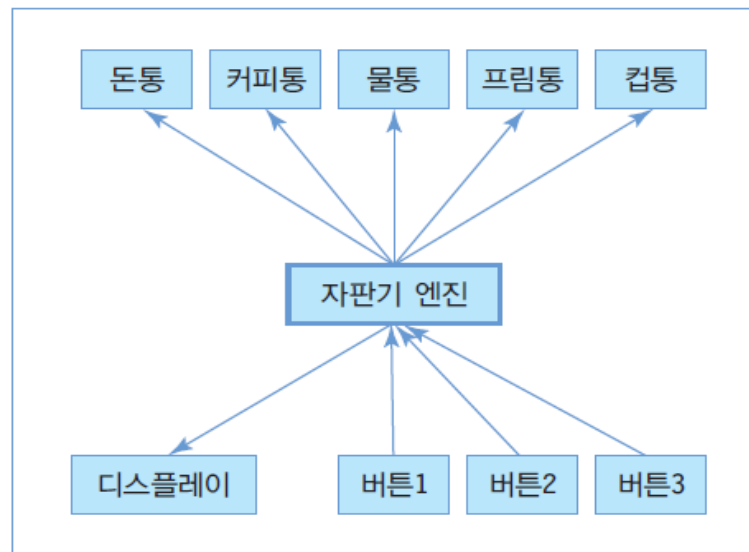
□ 절차 지향 프로그래밍

- 작업 순서를 표현하는 컴퓨터 명령 집합
- 함수들의 집합으로 프로그램 작성

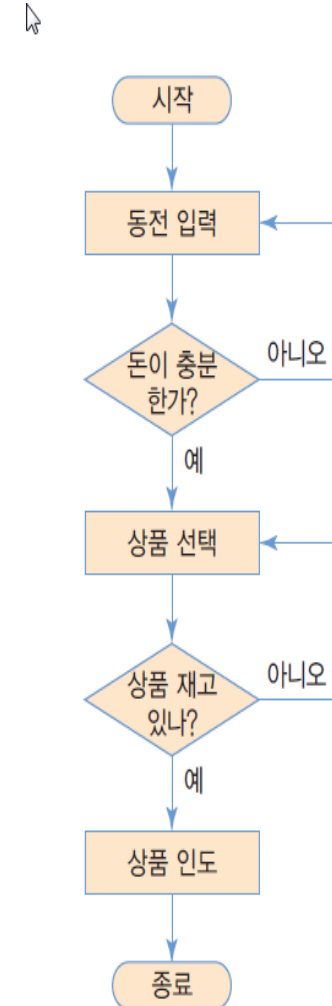
□ 객체 지향 프로그래밍

- 프로그램을 실제 세상에 가깝게 모델링
- 컴퓨터가 수행하는 작업을 객체들간의 상호 작용으로 표현
- 클래스 혹은 객체들의 집합으로 프로그램 작성

커피 자판기



객체지향적 프로그래밍의 객체들의 상호 관련성



절차지향적 프로그래밍의 실행

객체 지향 언어의 특성 : 캡슐화

45

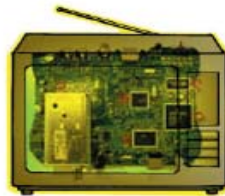
□ 캡슐화

- ▣ 메소드(함수)와 데이터를 클래스 내에 선언하고 구현
- ▣ 외부에서는 공개된 메소드의 인터페이스만 접근 가능
 - 외부에서는 비공개 데이터에 직접 접근하거나 메소드의 구현 세부를 알 수 없음
- ▣ 객체 내 데이터에 대한 보안, 보호, 외부 접근 제한

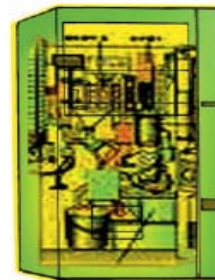
실세계의 캡슐화



캡슐약



TV



자판기



카메라



사람

객체

자바 객체의 캡슐화

```
String name;  
int age;
```

} 필드(field)

```
void speak();  
void eat();  
void study();
```

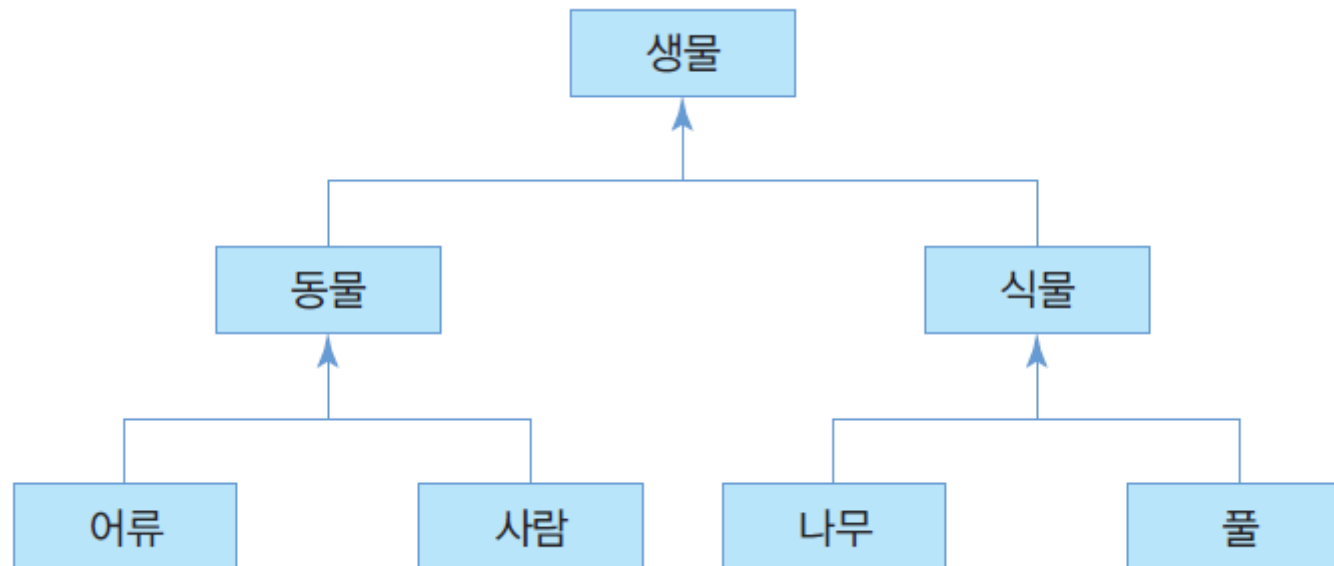
} 메소드(method)

객체 지향의 특성 : 상속

46

□ 유전적 상속 관계 표현

- 나무는 식물의 속성과 생물의 속성을 모두 가짐
- 사람은 생물의 속성은 가지지만 식물의 속성은 가지고 있지 않음



객체 지향의 특성 : 다형성

47

□ 다형성

- 동일한 이름의 기능이 서로 다르게 작동하는 현상
- 자바의 다형성 사례
 - 슈퍼 클래스의 메소드를 서브 클래스마다 다르게 구현하는 메소드 오버라이딩
 - 한 클래스 내에 구현된 동일한 이름이지만 다르게 작동하는 여러 메소드



클래스와 객체

48

□ 사람 세상

자동차 설계도/회로도

- 문 4
- 바퀴 4
- 색상 흰색
- 번호 1234
- 제조사 기아
- 엔진 A
- 연료 수소
- 전진기능 회로도
- 후진기능 회로도
- 좌회전 회로도
- 정지 회로도
- 자율주행 회로도

자동차
1

자동차
2

자동차
3

자동차
N

□ 컴퓨터 세상

➔ 메모리

자동차 클래스

필드 속성

- 문 4
- 바퀴 4
- 색상 흰색
- 번호 1234
- 제조사 기아
- 엔진 A
- 연료 수소

메서드 (METHOD)

- 전진기능 회로도
- 후진기능 회로도
- 좌회전 회로도
- 정지 회로도
- 자율주행 회로도

자동차
객체

자동차
객체

자동차
객체

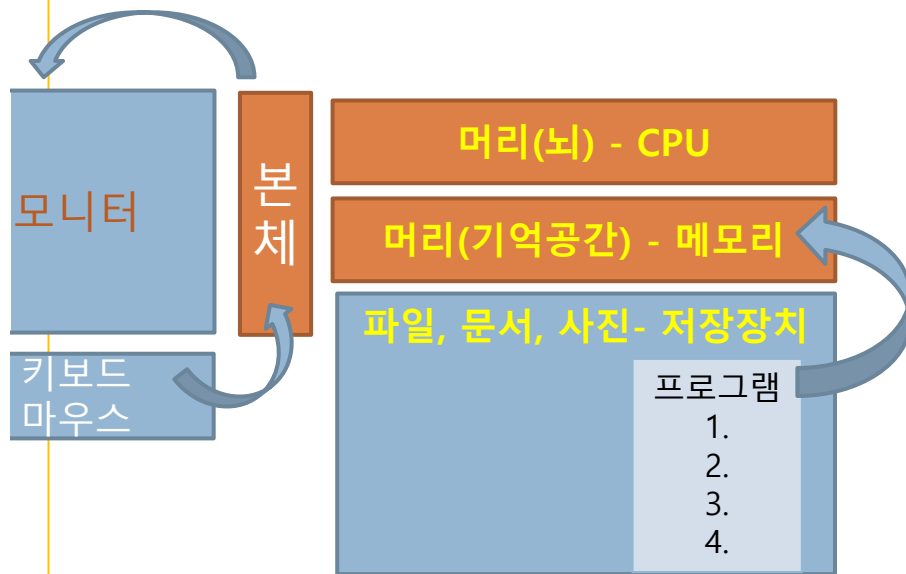
자동차
객체

자동차
객체

클래스와 객체

49

□ 컴퓨터 세상 → 메모리



자동차 클래스

필드
속성

- 문 4
- 바퀴 4
- 색상 흰색
- 번호 1234
- 제조사 기아
- 엔진 A
- 연료 수소

메서드
(METHOD)

- 전진기능 회로도
- 후진기능 회로도
- 자회전 회로도
- 정지 회로도
- 자율주행 회로도

자동차
객체

자동차
객체

자동차
객체

자동차
객체

자동차
객체

클래스와 객체

50

□ 클래스

- ▣ 객체의 속성/외관/특징 & 행위/행동 선언
- ▣ 객체의 설계도 혹은 틀

□ 객체 => 메모리에 생성(존재)

- ▣ 클래스의 틀로 찍어낸 실체
 - 메모리 공간을 갖는 구체적인 실체
 - 클래스를 구체화한 객체를 인스턴스(instance)라고 부름
 - 객체와 인스턴스는 같은 뜻으로 사용

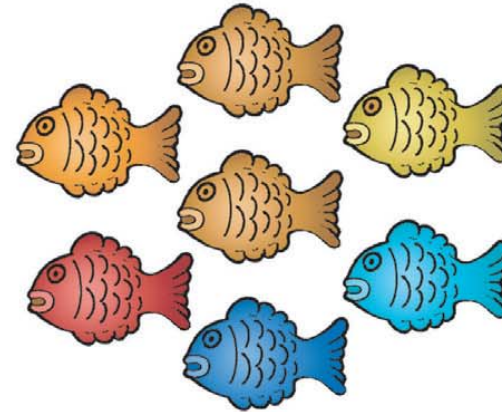
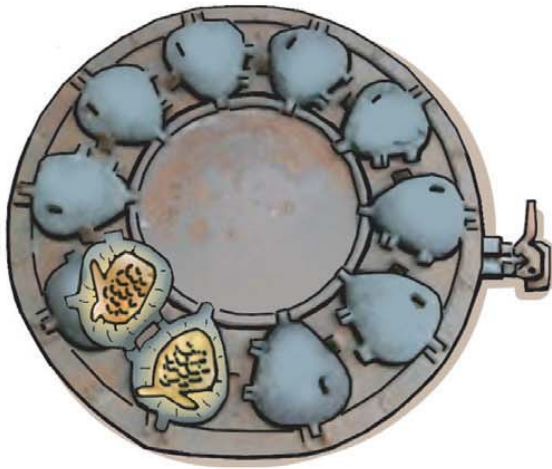
□ 사례

- | | |
|----------------|---------------------|
| ▣ 클래스: 소나타자동차, | 객체: 출고된 실제 소나타 100대 |
| ▣ 클래스: 벽시계, | 객체: 우리집 벽에 걸린 벽시계들 |
| ▣ 클래스: 책상, | 객체: 우리가 사용중인 실제 책상들 |

클래스와 객체와의 관계

51

붕어빵 틀은 클래스이며, 이 틀의 형태로 구워진 붕어빵은 바로 객체입니다. 붕어빵은 틀의 모양대로 만들어지지만 서로 조금씩 다릅니다. 치즈붕어빵, 크림붕어빵, 앙코붕어빵 등이 있습니다. 그래도 이들은 모두 붕어빵입니다.

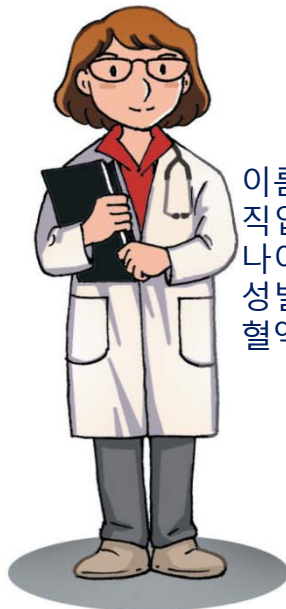


사람을 사례로 든 클래스와 객체 사례

52

클래스: 사람

이름, 직업, 나이, 성별, 혈액형
밥 먹기, 잠자기, 말하기, 걷기



이름 최승희
직업 의사
나이 45
성별 여
혈액형 A

객체 : 최승희



이름 이미녀
직업 골프선수
나이 28
성별 여
혈액형 O

객체 : 이미녀

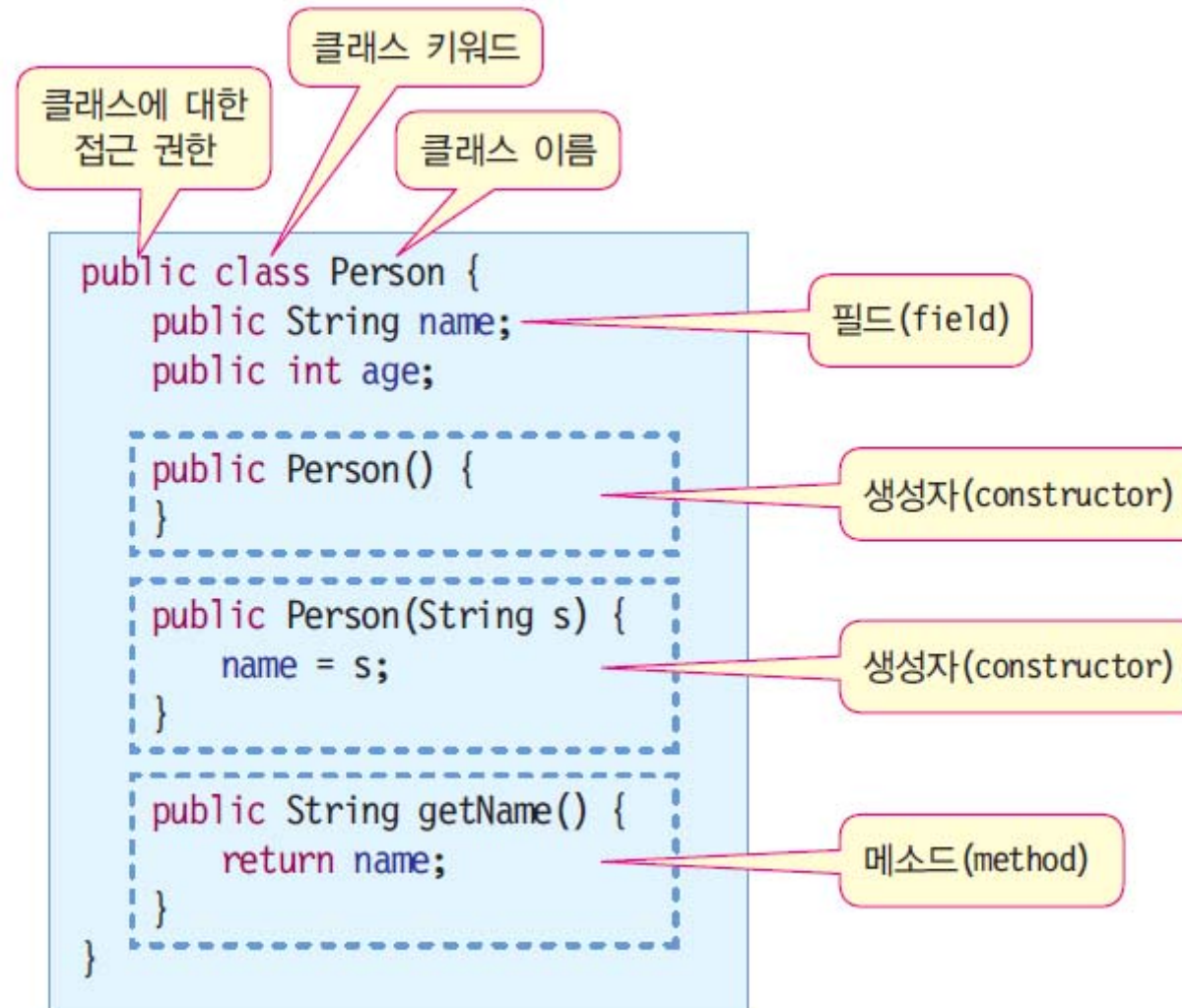


이름 김미남
직업 교수
나이 47
성별 남
혈액형 AB

객체:김미남

클래스 구성

53



- 특징 : 외형, 성격
 - 꼬리 있음
 - 다리 4개
 - 털 색상
 - 주둥이 짧음
 - 성격 온순
- 행동 : 동작
 - 짖는다.
 - 꼬리를 흔들
- 특징 : 외형, 성격
 - 꼬리 있음
 - 다리 4개
 - 털 색상
 - 주둥이 짧음
 - 성격 온순
- 행동 : 동작
 - 짖는다.
 - 꼬리를 흔들

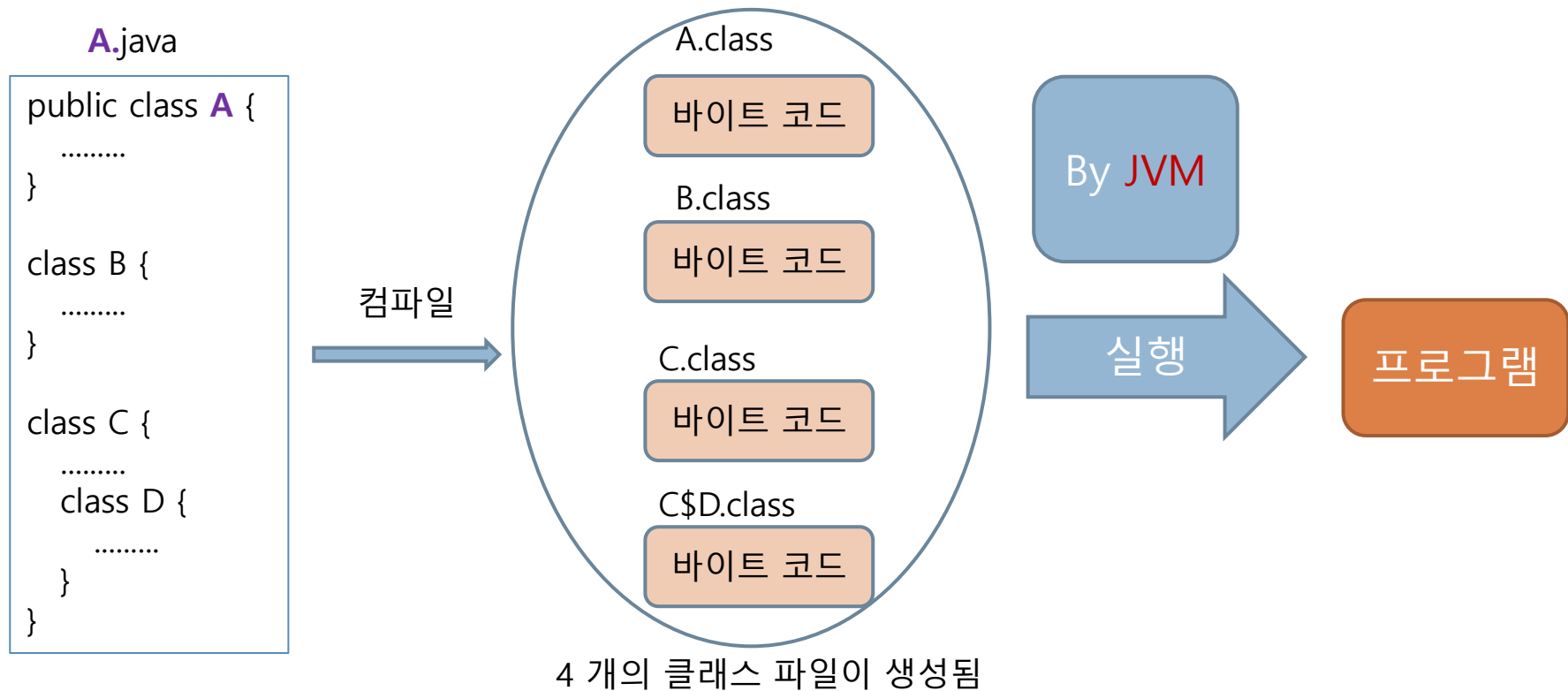
자바의 특성

54

- 객체지향
 - ▣ 객체지향의 특징인 클래스 계층 구조, 상속성, 다형성, 캡슐화 등 지원
- 멀티스레드
 - ▣ 다수 스레드의 동시 수행 환경 지원
 - 자바는 운영체제의 도움 없이 자체적으로 멀티스레드 지원
 - C/C++ 등에서는 멀티스레드를 위해 운영체제 API를 호출
- 플랫폼 독립성
 - ▣ 자바 가상 기계가 바이트 코드 실행
 - 플랫폼에 종속성을 갖지 않음
- 소스(.java)와 클래스(.class) 파일
 - ▣ 하나의 소스 파일에 여러 클래스를 작성 가능
 - 하나의 public 클래스만 가능
 - ▣ 소스 파일의 이름과 public으로 선언된 클래스 이름은 같아야 함
 - ▣ 클래스 파일에는 단 하나 만의 클래스만 존재
 - 다수의 클래스를 가진 자바 소스를 컴파일하면 클래스마다 별도 클래스 파일 생성

소스 파일과 클래스, 클래스 파일의 관계

55



자바의 특징

56

- 실행 모듈
 - ▣ 한 개의 class 파일 또는 다수의 class 파일로 구성
 - ▣ 여러 폴더에 걸쳐 다수의 클래스 파일로 구성된 경우
 - jar 파일 형태로 배포 가능
- **main() 메소드**
 - ▣ 자바 응용프로그램의 실행은 main() 메소드에서 시작
 - ▣ **하나의 클래스 파일에 하나 이상의 main() 메소드가 있을 수 없음**
 - 각 클래스 파일이 main() 메소드를 포함하는 것은 상관없음
- 클래스로 캡슐화
 - ▣ 자바의 모든 변수나 함수는 클래스 내에 선언
 - ▣ 클래스 안에서 새로운 클래스(내부 클래스) 작성 가능
- 패키지
 - ▣ 관련된 여러 클래스를 패키지로 묶어 관리
 - ▣ 패키지는 폴더 개념
 - 예) java.lang.System은 java\lang 디렉터리의 System.class 파일