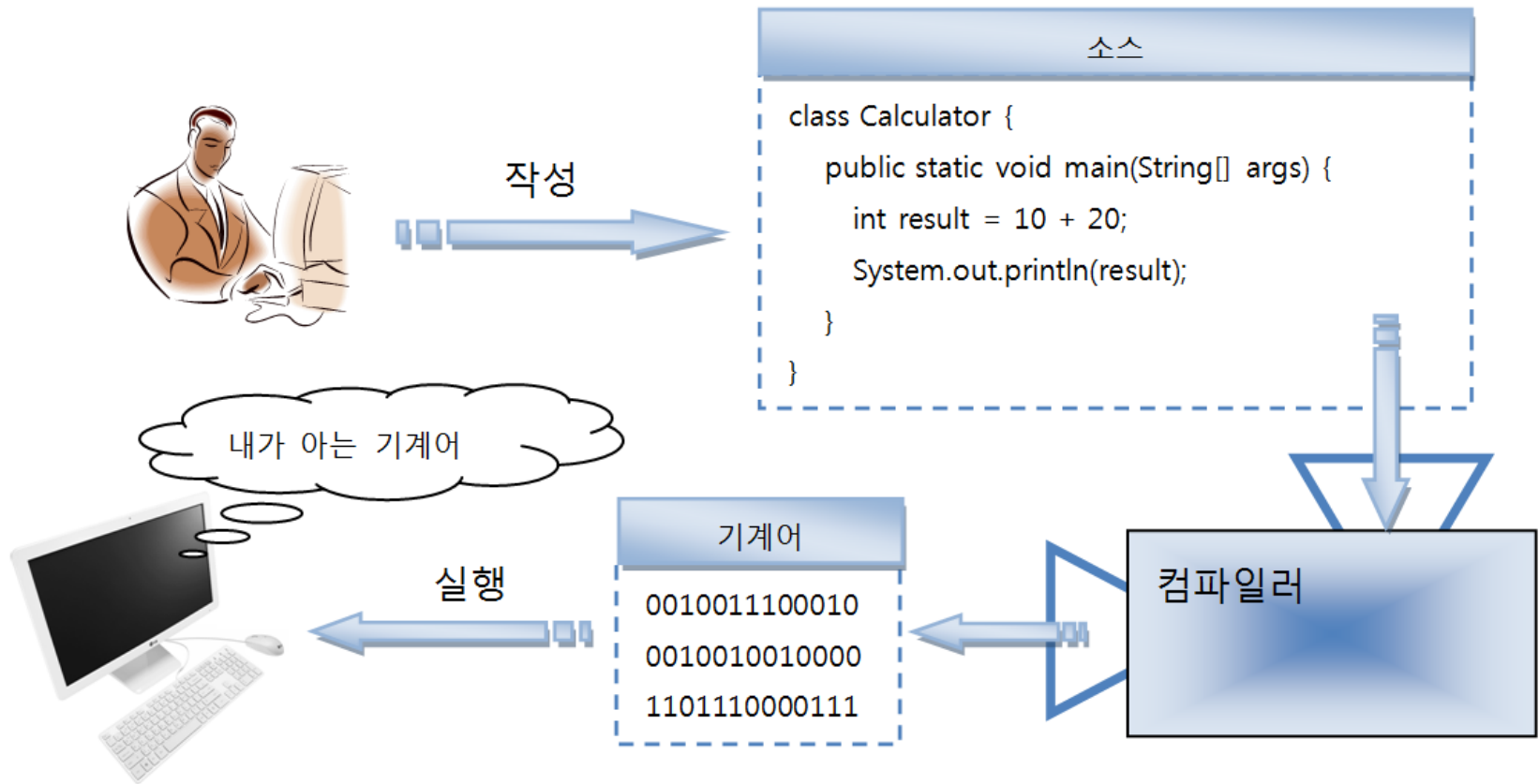


■ 프로그래밍 언어의 역할은?

- 사람과 컴퓨터의 대화
- 사람의 언어와 기계어 사이에서 다리와 같은 역할
- 저급 언어와 고급 언어로 구분

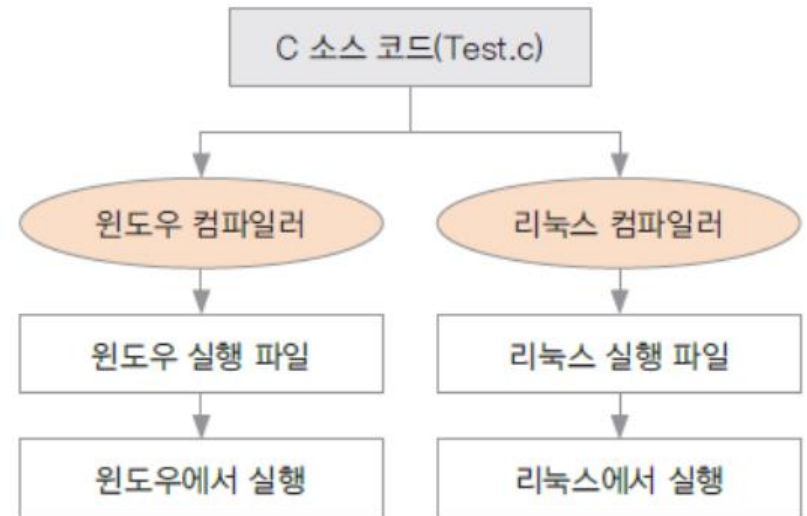
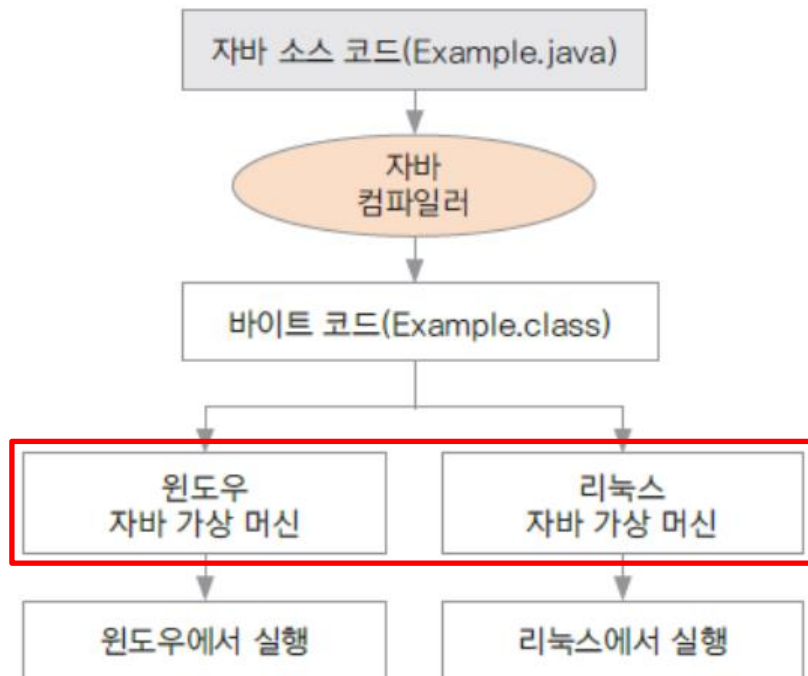


■ 자바?

- Sun Microsystems 에서 개발 (현 Oracle)
- 가전기기에서 사용할 목적으로 제임스 고슬링(James Gosling)과 그 외 기술자들이 오크(Oak)라는 언어를 개발
- 이후 1994년 개발방향이 바뀌면서 자바(JAVA)로 명명
(James Gosling, **A**rthur **V**an Hoff, **A**ndy Bechtolsheim)
- 2010년 오라클에서 썬을 인수하여 Java 개발, 관리, 배포 주관

■ 자바의 특징

- 이식성이 높은 언어
 - 여러 환경에서 사용 가능



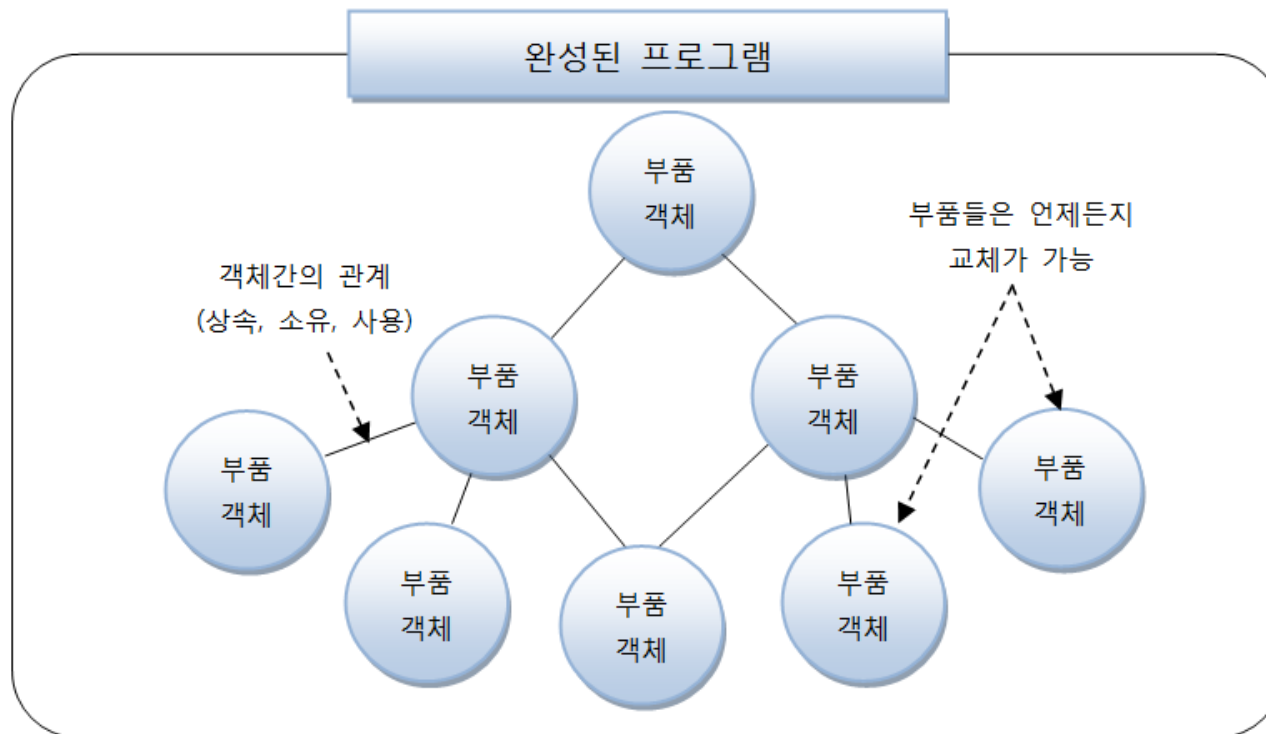
■ 자바의 특징

- 객체 지향 언어 -> OOP(Object Oriented Programming)

부품 객체를 먼저 만들고, 이것들을 조합해 전체 프로그램을 완성하는 기법

- 자바는 처음부터 OOP 개발용 언어로 설계

- 캡슐화, 상속, 다형성



■ 자바의 특징

- 메모리를 자동으로 관리 (Garbage Collection)
- 다양한 애플리케이션 개발 가능
 - 웹서버 / 안드로이드 / Stand Alone(유틸, 게임, ERP, ...) 등
- 풍부한 오픈 소스 라이브러리

■ 자바의 종류

- JAVA SE (Standard Edition) – 표준 개발
- JAVA EE (Enterprise Edition) – 웹애플리케이션 개발
- JAVA ME (Micro Edition) – 소형기기 (가전제품)

■ JDK(Java Development Kit) = JRE + 개발 도구

자바 프로그램 개발하고 실행하기 위해 반드시 설치

■ JRE(Java Runtime Environment) = JVM + 표준 클래스 라이브러리

자바 프로그램을 실행만 할 경우 설치

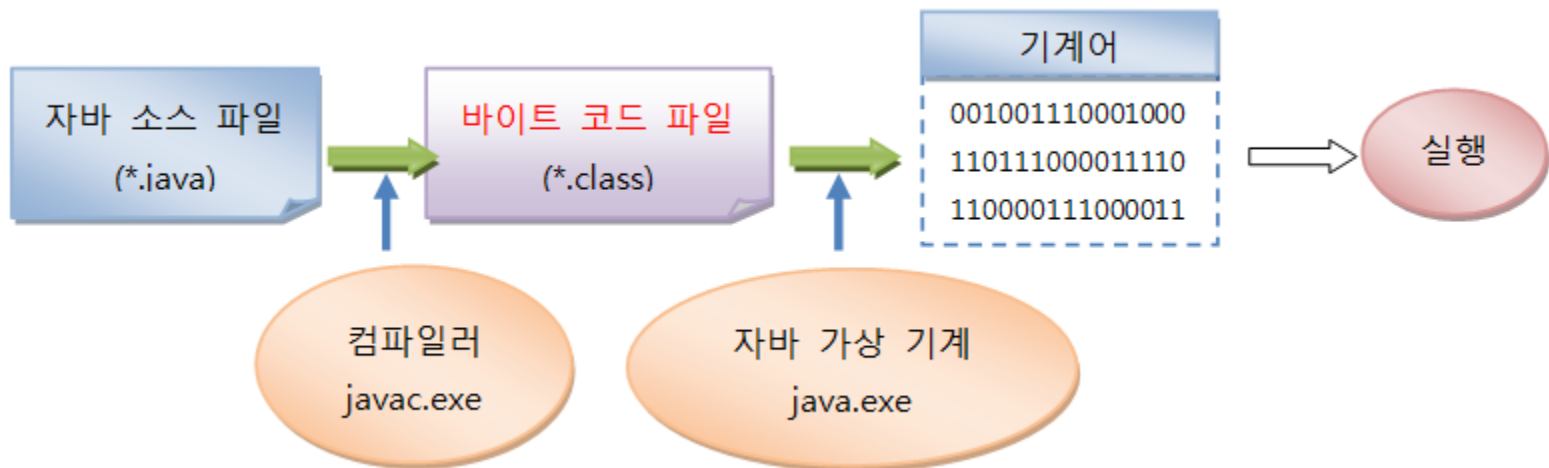
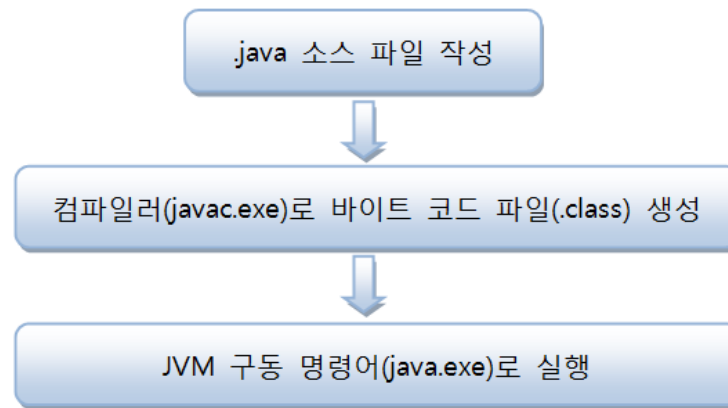
■ JDK(Java Development Kit) 설치

설치 파일 다운로드 사이트: <http://www.oracle.com>

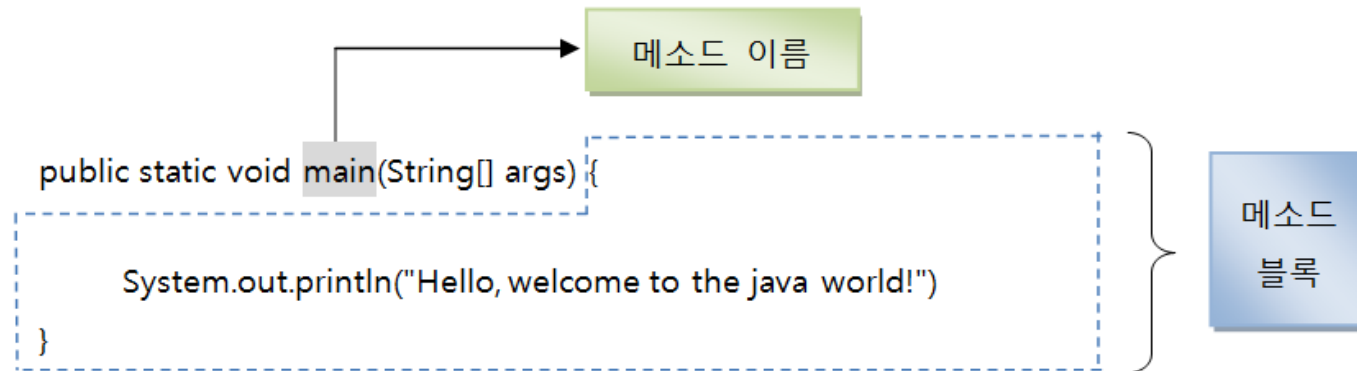
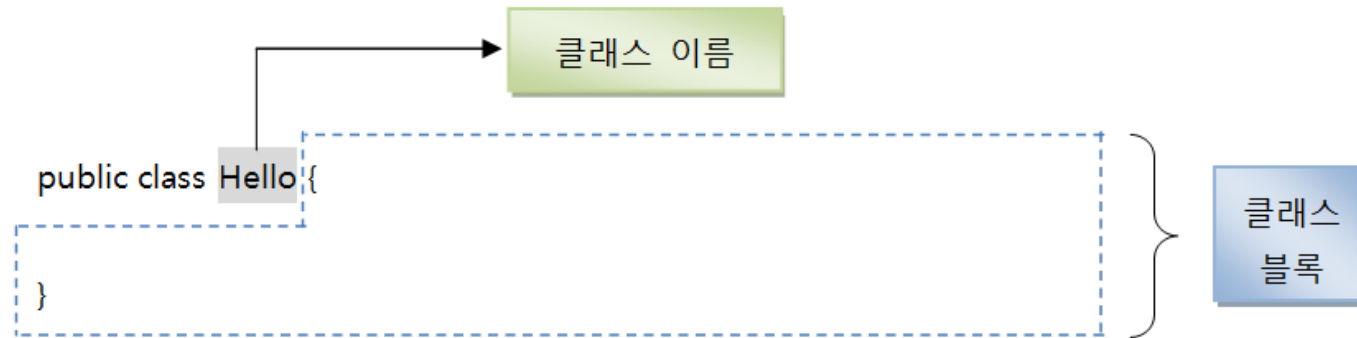
■ API 도큐먼트

- JDK에서 제공하는 표준 클래스 라이브러리 설명해 놓은 HTML 페이지들
- <http://docs.oracle.com/javase/버전/docs/api/>

■ 소스 작성에서부터 실행까지

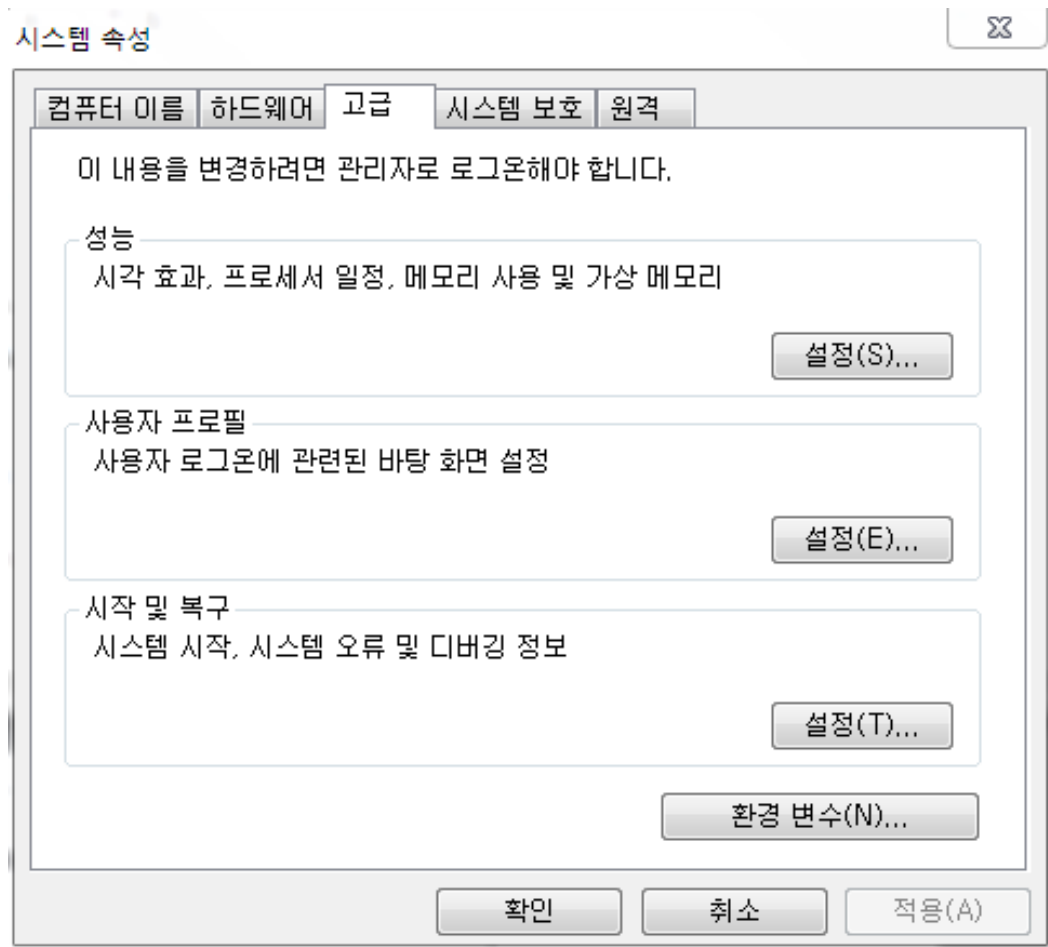


■ 자바 프로그램(클래스) 구성



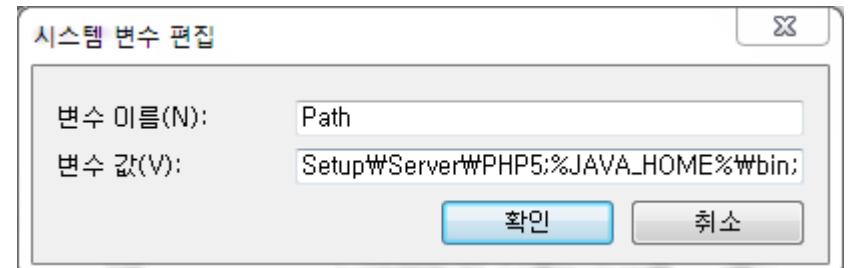
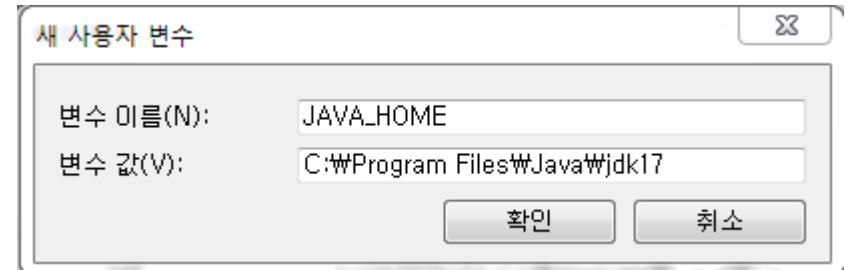
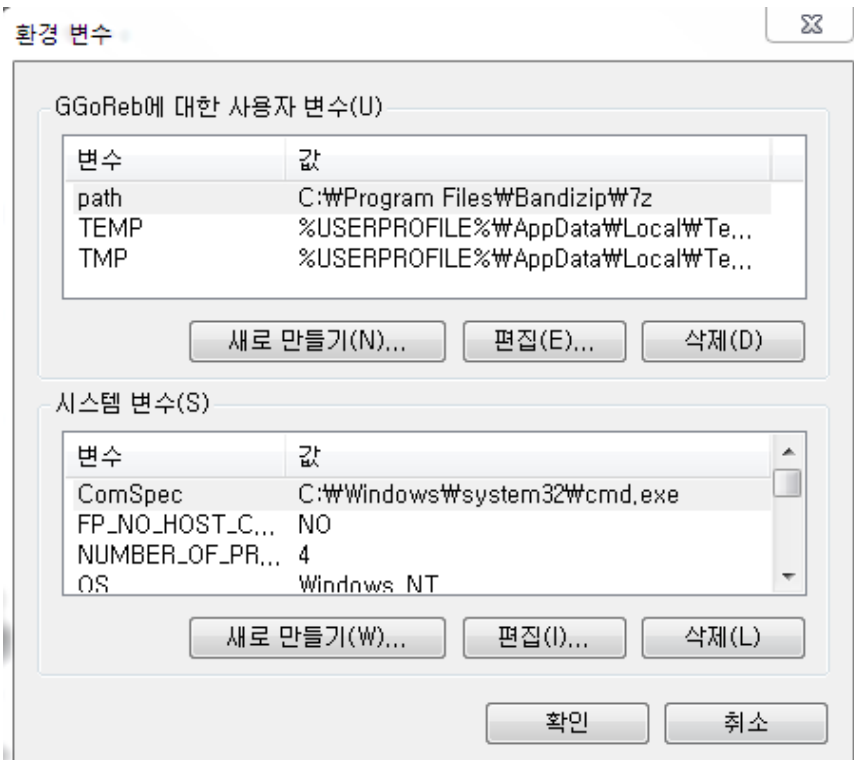
■ 환경변수

● 제어판 – 시스템 – 고급 시스템 설정

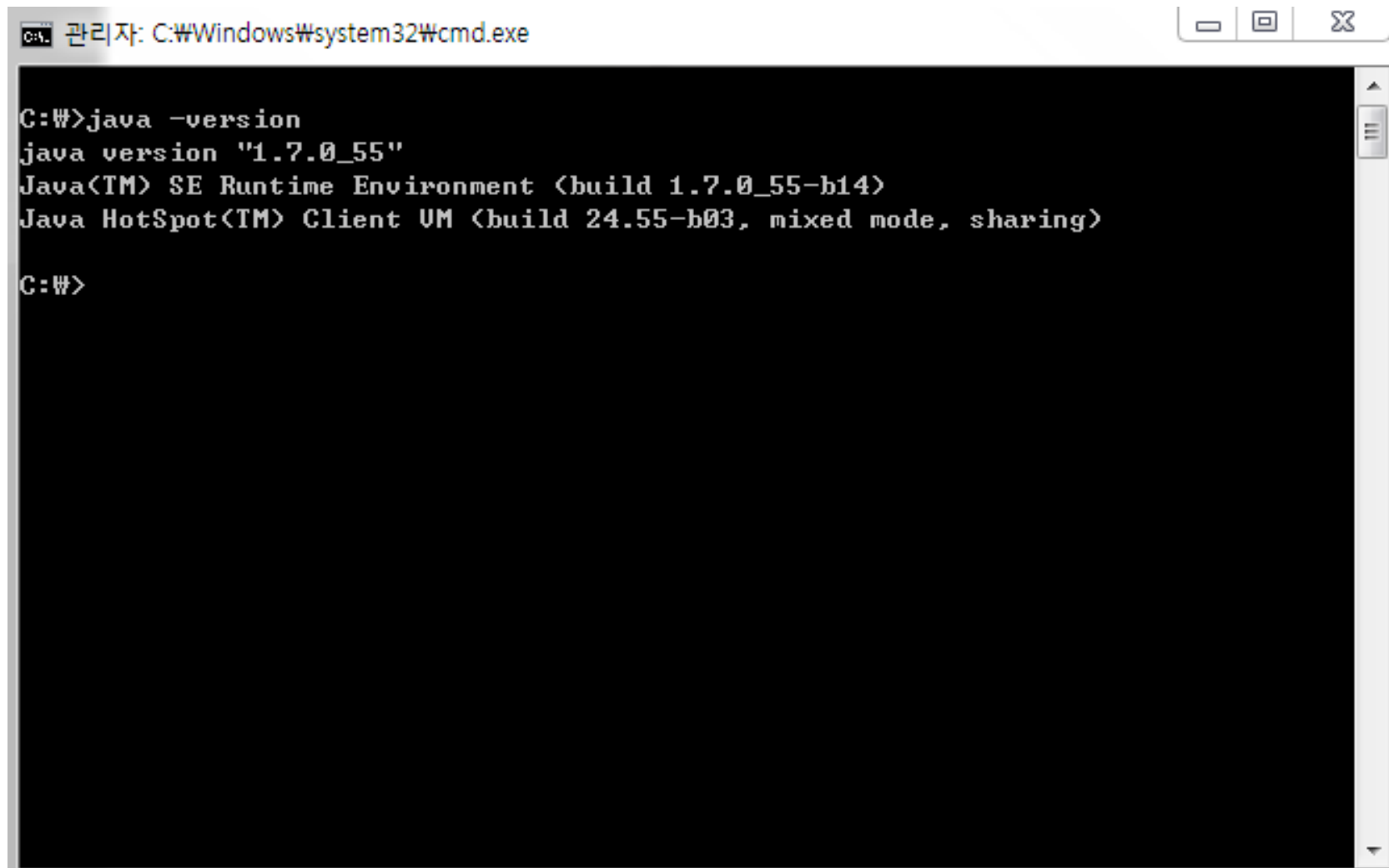


■ 환경변수

● 환경변수



■ 설치 확인



A screenshot of a Windows command prompt window. The title bar at the top reads "관리자: C:\Windows\system32\cmd.exe". The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt shows the following text:

```
C:\>java -version
java version "1.7.0_55"
Java(TM) SE Runtime Environment (build 1.7.0_55-b14)
Java HotSpot(TM) Client VM (build 24.55-b03, mixed mode, sharing)

C:\>
```

■ 자바 컴파일 및 실행

● 텍스트 편집기 (메모장 등) 이용 작성

```
class Launcher {  
    public static void main(String[] args) {  
        System.out.println("First Java Program");  
    }  
}
```

● 컴파일

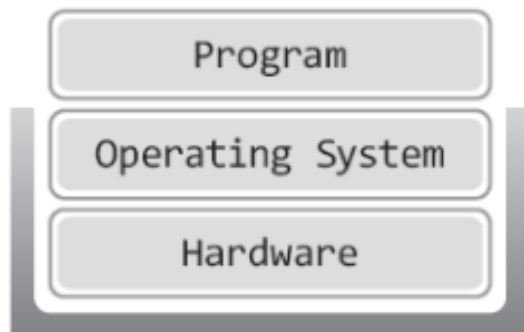
```
D:\#Dev>javac Launcher.java  
  
D:\#Dev>
```

● 실행

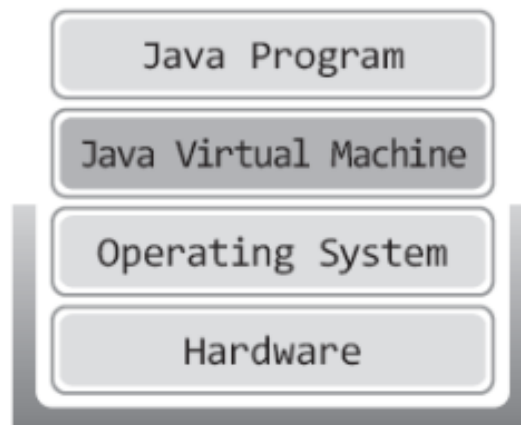
```
D:\#Dev>java Launcher  
First Java Program  
  
D:\#Dev>
```

■ 자바 프로그램의 실행 구조와 가상머신

45/11



일반적인 프로그램의 실행구조



자바 프로그램의 실행구조



운영체제에 독립적인 자바 프로그램

자바 프로그램은 운영체제에 독립적!

자바 가상머신은 운영체제에 의존적!

■ 이클립스(Eclipse)

- 2003년 IBM에서 개발
- 자바 통합 개발 환경(IDE: Integrated Development Environments) 제공
 - 프로젝트 생성 기능 제공
 - 자동 코드 완성 기능 제공
 - 디버깅 기능 제공
- 이클립스 연합(Eclipse Foundation) 설립 - 지속적 버전업과 배포
- 다양한 개발 환경을 구축할 수 있도록 플러그인(Plug-In) 설치 가능
 - 안드로이드 개발 환경
 - 스프링(Spring) 개발 환경
 - C, C++, PHP 등 개발 환경

■ 이클립스(Eclipse)

● 다운로드 사이트 : <http://www.eclipse.org>

- Eclipse IDE for Java Developers 버전
 - 순수 자바 학습용
- Eclipse IDE for Java EE Developers 버전 (책에서 사용하는 버전)
 - 웹 애플리케이션 등의 Enterprise (Network) 환경에서 실행
- CPU 사양에 맞게 다운로드

■ 이클립스(Eclipse)

● 워크스페이스 (Workspace)

- 이클립스에서 생성한 프로젝트가 기본적으로 저장되는 디렉토리
- 최초 실행 시 워크스페이스 런처(Workspace Launcher)에서 설정
- .metadata 디렉토리
 - 자동 생성되며 이클립스 실행 시 필요한 메타데이터 저장
 - 이 디렉토리를 삭제하면 이클립스의 초기 설치 상태로 돌아감

■ 이클립스(Eclipse)

● 퍼스펙티브 (Perspective)

- 개발 프로젝트 종류별로 유용한 View 들을 묶어놓은 것



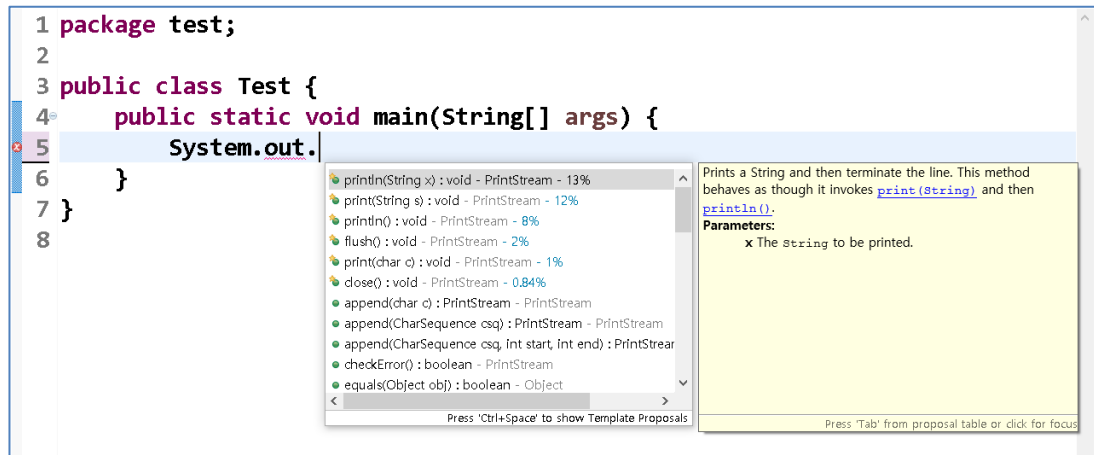
● 뷰 (View)

- 퍼스펙티브를 구성하는 작은 창으로 여러가지 목적에 맞는 내용 보여줌
- 자유롭게 제거하거나 추가 가능
- 학습 중 주로 사용되는 View
 - Package Explorer
 - Console

■ 이클립스(Eclipse)

● 자동 완성 기능 (Code Assistant)

- Ctrl + space : 현재 상황에서 사용할 수 있는 속성 및 기능 목록 표시

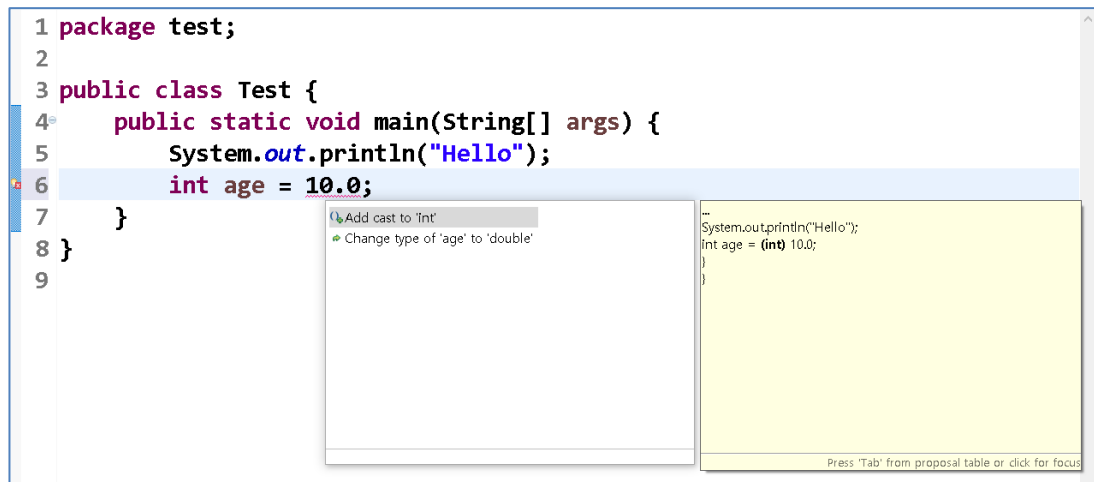


The screenshot shows the Eclipse IDE with a Java file. The code is as follows:

```
1 package test;
2
3 public class Test {
4     public static void main(String[] args) {
5         System.out.
6     }
7 }
8
```

At line 5, after `System.out.`, the Code Assistant is open, displaying a list of methods from the `PrintStream` class. The methods listed include `println(String x)`, `print(String s)`, `println()`, `flush()`, `print(char c)`, `close()`, `append(char c)`, `append(CharSequence csq)`, `append(CharSequence csq, int start, int end)`, `checkError()`, and `equals(Object obj)`. The `println(String x)` method is highlighted. To the right of the list, a tooltip for the `println()` method is visible, stating: "Prints a String and then terminate the line. This method behaves as though it invokes `print(String)` and then `println()`." The parameters section indicates: "Parameters: x The string to be printed."

- Ctrl + 1 : 개발자가 실수하고 있는 상황을 해결할 수 있는 기능 제공



The screenshot shows the Eclipse IDE with the same Java file as before, but now with the following code:

```
1 package test;
2
3 public class Test {
4     public static void main(String[] args) {
5         System.out.println("Hello");
6         int age = 10.0;
7     }
8 }
9
```

At line 6, after `int age = 10.0;`, the Code Assistant is open, displaying two quick fix proposals:

- 1. Add cast to 'int' (indicated by a lightning bolt icon)
- 2. Change type of 'age' to 'double' (indicated by a green arrow icon)

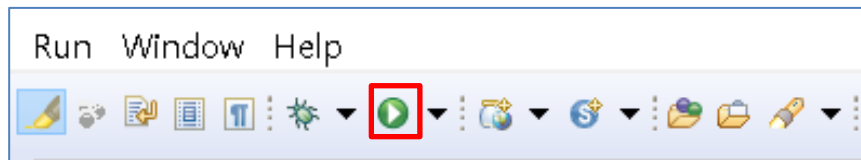
To the right of the list, a tooltip for the "Add cast to 'int'" proposal is visible, showing the corrected code:

```
...
System.out.println("Hello");
int age = (int) 10.0;
}
```

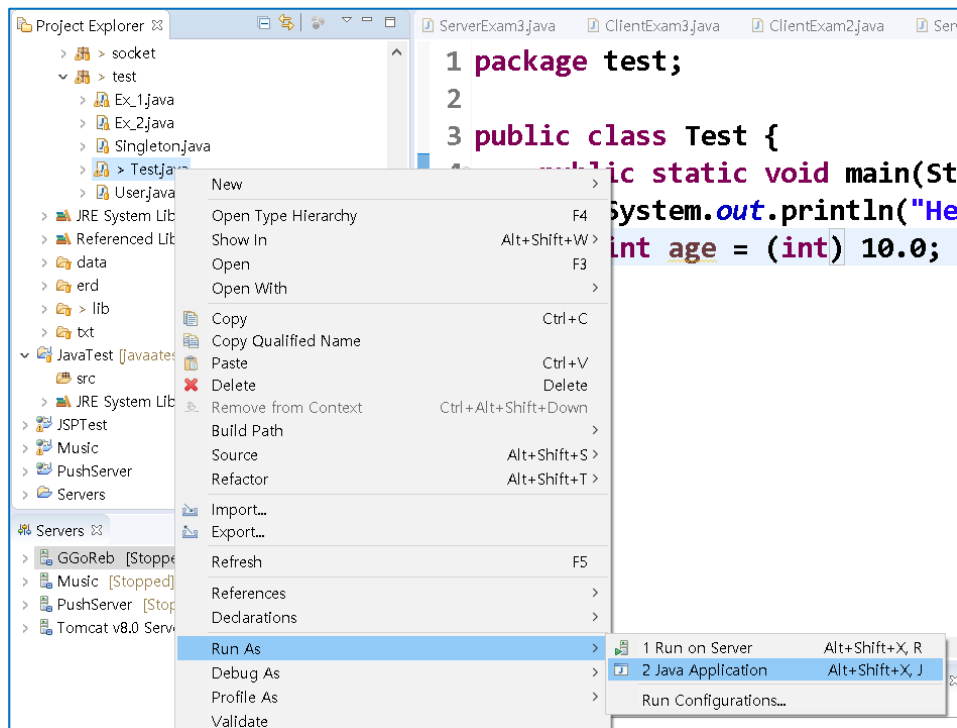
■ 이클립스(Eclipse)

● 자바 프로그램(바이트 코드) 실행

- 툴바



- Package Explorer – 우클릭 – Run As – Java Application



■ 실행문과 세미콜론 (;)

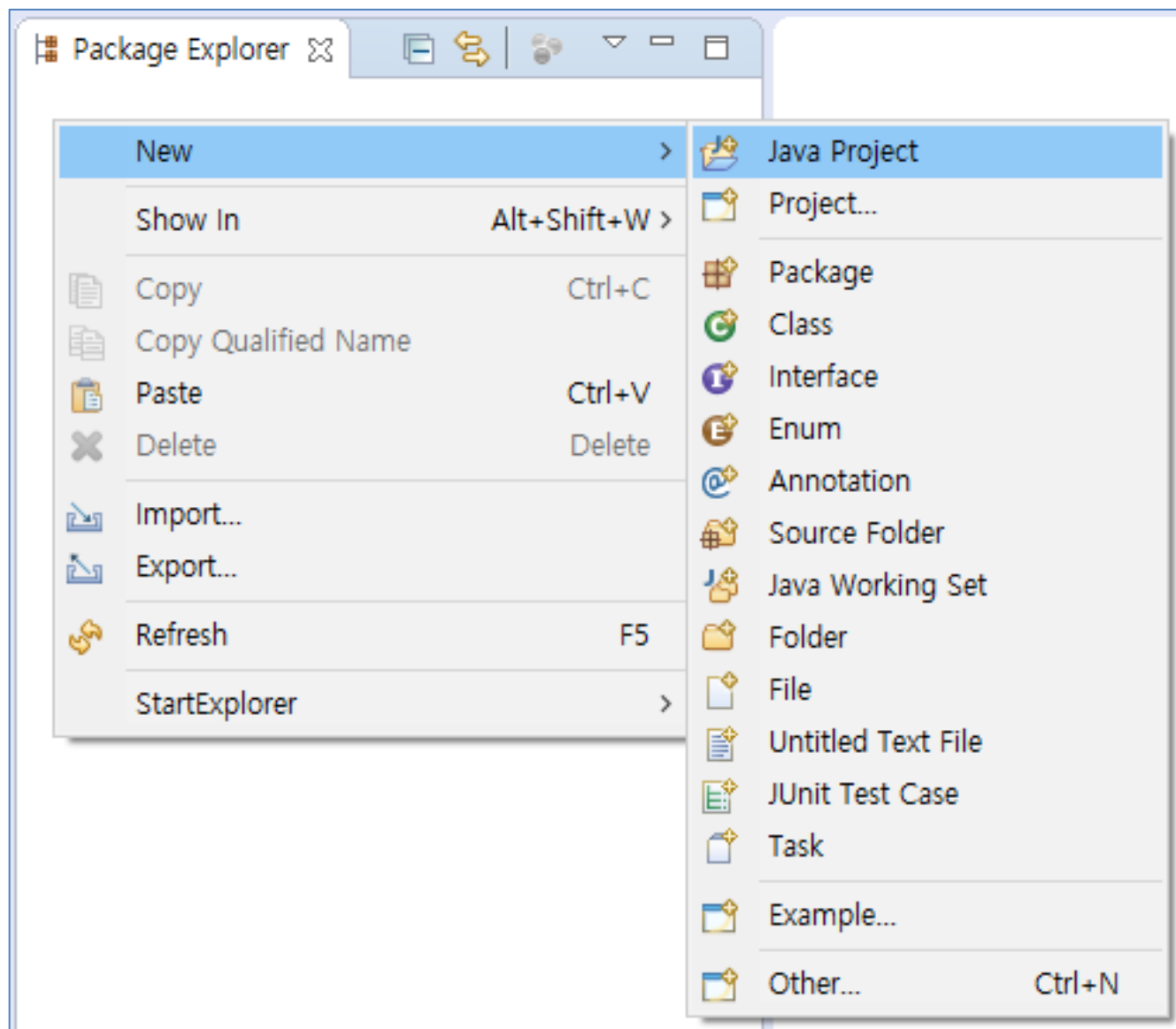
● 실행문

- 변수 선언, 값 저장, 메소드 호출 등의 코드
- 실행문 끝에는 반드시 세미콜론(;)을 붙여 실행문의 끝 표시

```
int x = 1;           //변수 x를 선언하고 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); //콘솔에 출력하는 메소드 호출
```

```
int x = 1; int y = 2;
int result =
    x + y;
```

■ 자바 프로젝트 생성 (1 / 2)



■ 자바 프로젝트 생성 (2 / 2)

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'jre1.8.0_101') [Configure JREs...](#)

Project layout

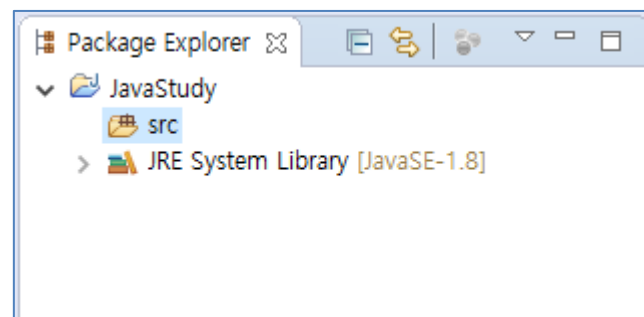
☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

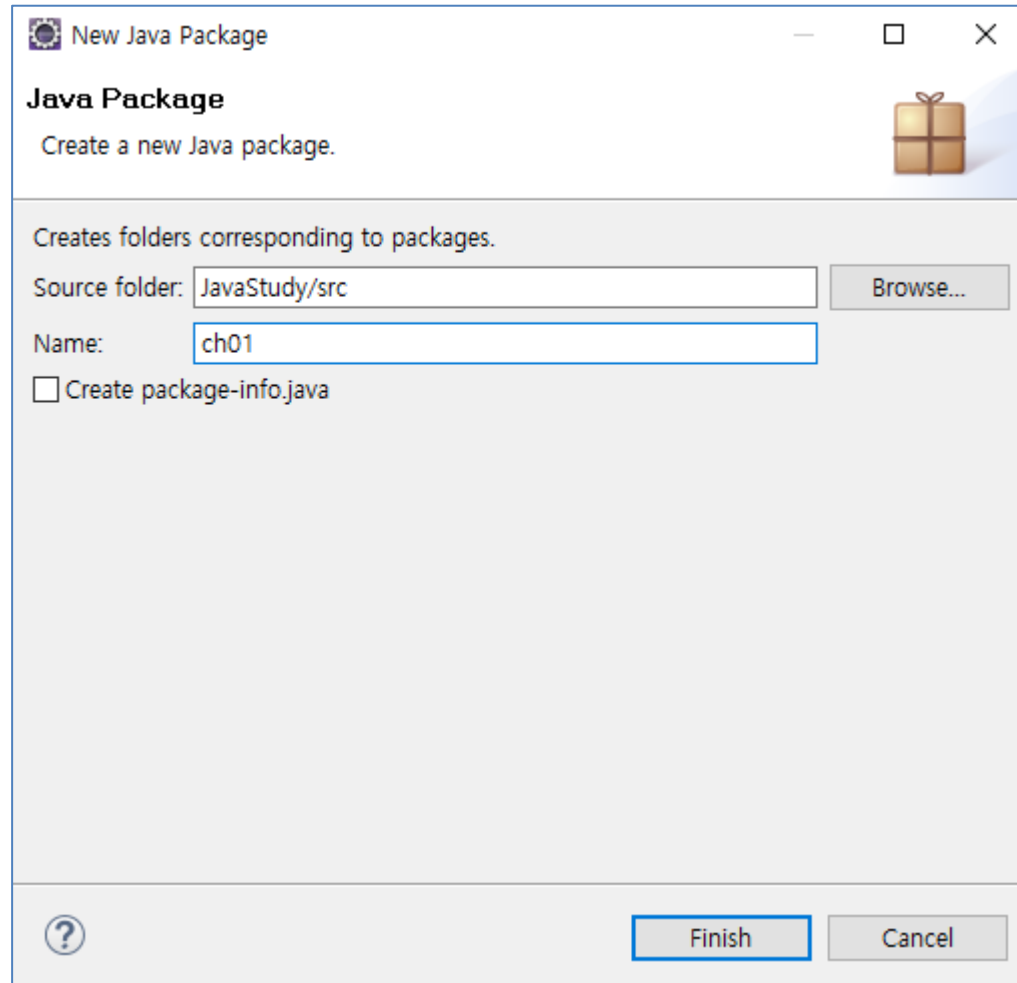
☐ Add project to working sets

Working sets:



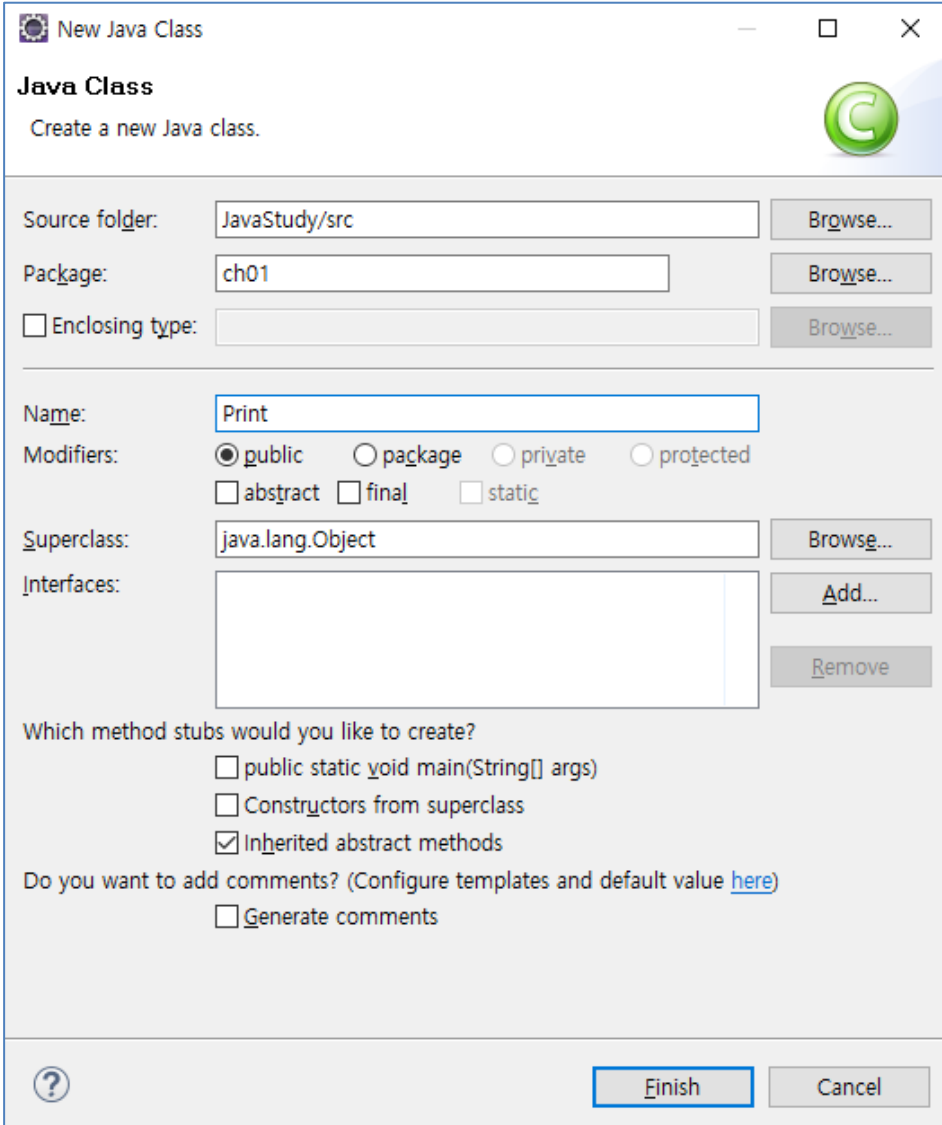
■ 콘솔 출력문 (1 / 4)

● 패키지 생성



■ 콘솔 출력문 (2 / 4)

● 클래스 생성



The image shows the 'New Java Class' dialog box in an IDE. It has a title bar with a question mark icon, a maximize button, and a close button. The main area is titled 'Java Class' with a subtitle 'Create a new Java class.' and a green circular icon with a 'C' on the right. The dialog is divided into several sections: 'Source folder:' with a text field containing 'JavaStudy/src' and a 'Browse...' button; 'Package:' with a text field containing 'ch01' and a 'Browse...' button; 'Enclosing type:' with an empty text field and a 'Browse...' button. Below these is the 'Name:' field containing 'Print'. The 'Modifiers:' section has radio buttons for 'public' (selected), 'package', 'private', and 'protected', and checkboxes for 'abstract', 'final', and 'static'. The 'Superclass:' field contains 'java.lang.Object' with a 'Browse...' button. The 'Interfaces:' section has an empty list box with 'Add...' and 'Remove' buttons. The 'Which method stubs would you like to create?' section has checkboxes for 'public static void main(String[] args)', 'Constructors from superclass', and 'Inherited abstract methods' (checked). The 'Do you want to add comments? (Configure templates and default value [here](#))' section has a checkbox for 'Generate comments'. At the bottom are buttons for '?', 'Finish', and 'Cancel'.

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

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

Superclass:

Interfaces:

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

■ 콘솔 출력문 (3 / 4)

// 문자열(문장)은 항상 큰따옴표를 사용하여 출력

```
System.out.println("First Program");
```

// 숫자 더하기 숫자는 더하기 연산 결과로 출력

```
System.out.println(1 + 2);
```

// 숫자 곱하기 숫자는 곱하기 연산 결과로 출력

```
System.out.println(3 * 4);
```

// 문자열 더하기 문자열은 문자열이 합쳐진 형태로 출력

```
System.out.println("1" + "2");
```

// 출력결과는 숫자이지만 실제 데이터는 문자열

```
System.out.println("12");
```

// 숫자 더하기 문자열, 또는 문자열 더하기 숫자는 문자열이 합쳐진 형태로 출력

```
System.out.println(1 + "2");
```

■ 콘솔 출력문 (4 / 4)

// 더하기와 곱하기 연산의 경우 곱하기가 더 우선 순위로 연산

```
System.out.println(1 + 3 * 4);
```

// 괄호를 사용하면 최우선 순위로 연산

```
System.out.println((1 + 3) * 4);
```

```
System.out.println("큰따옴표 안에서는 '작은따옴표' 사용 가능");
```

```
System.out.println("큰따옴표 안에서 ₩큰따옴표₩를 사용하려면 예외문자(₩₩)를 사용");
```

// 작은따옴표는 문자 한개를 표현할때 사용

```
System.out.println('문');
```

// 여러 기호의 조합

```
System.out.println("A의 점수 합은 : " + (1 + 2) + ", B의 점수 합은 : " + (3 + 4));
```

■ 주석 (Comment)

- 컴파일 대상에서 제외되는 문장
- 프로그램 실행과는 상관없이 코드에 설명을 붙인 것
 - 본인이나 타인이 소스코드를 쉽게 분석할 수 있도록 작성
- 소스코드 활용 주의사항 작성
- 주석의 종류
 - 행 단위 : //
 - 블록 단위 : /* */
 - 문서화 : /** */

■ 주석 (Comment)

```
package ch01;

//파일이름 : Comment.java
//작성자 : 김남현
//작성일 : 2015년 3월
//작성이유 : System.out.println 메소드 기능 테스트
public class Comment {
    /**
     * 주석 사용법
     */
    public static void main(String[] args) {
        /* 이렇게도 사용 가능 */
    }
}
```