

# Java 개요

**강사 : 강병준**

# PROGRAMMING?

- 컴퓨터와 인간간의 커뮤니케이션
- 컴퓨터의 처리를 인간이 원하는 의도대로 하기 위해서 일련의 명령어로 작업을 지시하는 것
- 때문에 컴퓨터가 이해할 수 있는 커뮤니케이션 방식이 필요
- 프로그래밍 언어란 결국 커뮤니케이션용 언어일 뿐
- 실제 세계의 언어마다 특징이 있듯이 프로그래밍 언어들도 그 특징이 있다.
- 사람과 사람 사이에 통역사가 있듯 컴퓨터와 사람 사이에도 통역사가 필요하다, 자바에서는 JDK라는 통역

# 프로그래밍 언어의 발전 방향

## ● Low level 언어

- 기계어 중심의 프로그래밍  
(0과 1로 이루어진 언어)
- CPU 인스트럭션
- 어셈블리어

## ● High level 언어

- 사람이 이해하기 쉬운 언어
- 3세대 : fortran, cobol, c
- 4세대 : Delphi, powerbuilder
- Web : 닷넷/ Java 등 현재 사용되는 언어들

# LOW LEVEL의 언어의 장/단점

## ● 장점

- 세밀한 처리가 가능
- 최적화된 프로그래밍 구현 가능

## ● 단점

- 진입장벽이 높다.
- 프로그램 구현을 위해서 많은 시간

# High level 언어의 장/단점

## ● 장점

- 인간이 이해하는 언어의 형태로 작성
- 빠른 개발
- 쉬운 유지보수

## ● 단점

- 완벽한 성능을 기대하기 어렵다.
- 개발자의 지식 수준이 상대적으로 낮다.

# HIGH LEVEL 언어와 컴파일러의 등장

- High level → CPU 인스트럭션으로 직접 변환은 불가
- 중간에 CPU의 언어로 해석할 수 있는 간접적인 장치가 컴파일러
- 보통의 경우 컴파일러를 통해서 운영체제와 통신하는 코드를 생성

## Interpreter 방식의 언어

- 컴파일을 통해서 운영체제와 통신하는 방식의 대안
- 특정한 프로그램이 실행되면서 프로그램에 필요한 로직이나 데이터를 해석
- 해석기의 존재만 있다면 운영체제에 영향을 덜 받을 수 있다는 장점

# 자바 언어의 역사

- 1991년 선사의 James Gosling에 의해 가전제품에 이용하기 위해 개발이 시작(Green Project-컴퓨터기술을 통합한 통합 리모콘.) 당시 WWW은 별로 알려지지 않았다.
- 초기에 개발된 언어를 Oak라 하였으며 전자기기의 내장된 프로그램을 위해 사용. Oak는 별로 관심을 끌지 못하였다
- 1994년 급격히 성장한 WWW에 자바를 적용 결정. 처음 등장한 것이 웹러너였으며 뜨거운 자바 즉 핫자바로 이름을 바꾸었다.
- 처음으로 공식 회의에서 웹 문서 안에서 실행되는 3D 분자 구조 모형을 선보였다. 운영체제에 구해 받는 애플릿 프로그램을 작성할 수 있었다.Gosling은 웹브라우저 안에서 실행되는 자바 프로그램을 작은 프로그램이라는 뜻에서 Applet이라 하였다.
- 1995년 5월 23일 넷스케이프 사의 브라우저가 자바 기술을 채택하게 되었다.
- 자바라는 이름은 인도네시아 산 커피 원료 이름 "자바"에서 유래 (잠들지 않는 인터넷과 의미가 상통)

# 자바 언어의 역사

- 플랫폼 : Sun SPARC Solaris, Windows NT, Windows95, Linux
- Java beta1 발표(Sun Microsystems)
- Netscape 지원결정(1995)
- Java beta2 발표
- JavaScript 발표(Sun & Netscape)
- 1996년 자바1.0 발표 Netscape2.0 자바 지원
- 2005년 자바 5.0 사용
- 2009년 자바6.0 사용
- 2010년 Oracle인수 자바 7.0사용
- 2014년 자바 8.0

# 자바의 특징

## 1-1. JAVA의 장점

### 1. 확장성(Scalability)

#### ▶ 적용분야 광범위

- 기업용(Enterprise급) ~ 핸드폰(Mobile)
- 다양한 분야에 적용 가능한 프로그램 개발

#### ▶ 기존 시스템 확장 용이

- 인사 시스템 → 급여 시스템



# 자바의 특징

## 2. 보안성(Security)

### ▶ 자체 강력한 보안 능력 보유

- 클래스의 캡슐화
- 외부접근자 : public vs private
- 자바가상머신 보안 제공

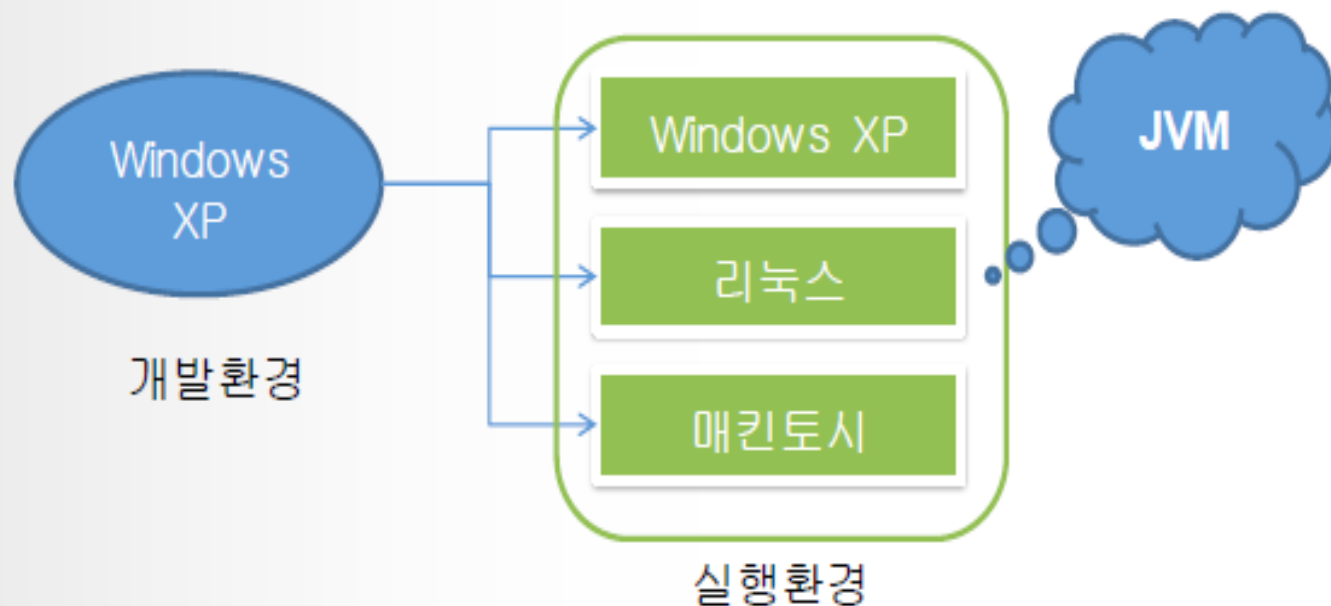


# 자바의 특징

## 3. 보편성(Universality)=이식성, 호환성(Compatible)

### ▶ 플랫폼(운영체제:OS) 독립성

- JVM(Java Virtual Machine) : java 프로그램 해석&실행 기능



# 자바의 특징

## 4. 재사용성(Reuse)=Modularity

- Modularity : 생산에 규격화된 부품을 사용함
- 기 개발된 java 프로그램 재사용
- 객체지향프로그램의 가장 큰 이점



# 자바의 특징

## 1-2. JAVA의 특징

### 1. 쉽고, 편한 프로그램 개발 환경 제공

- 복잡한 부분 제외
  - C언어의 포인터 기능 없음
- 객체지향 프로그램(Object Oriented Program)
  - 절차지향언어에 비해 컴포넌트(Component)화 용이 = Modul화
  - CBD(Component Based Development) 개발방법론 적용
  - 객체지향 개념 이해 선행
- 풍부한 API(Application Programming Interface) 제공
  - API : JAVA에 포함된 기 개발 프로그램 → 프로그램 개발 시 사용

# 자바의 특징

## 2. 플랫폼에 독립적인 환경 제공

### ➤ 프로그램 언어(해석기로 구분)

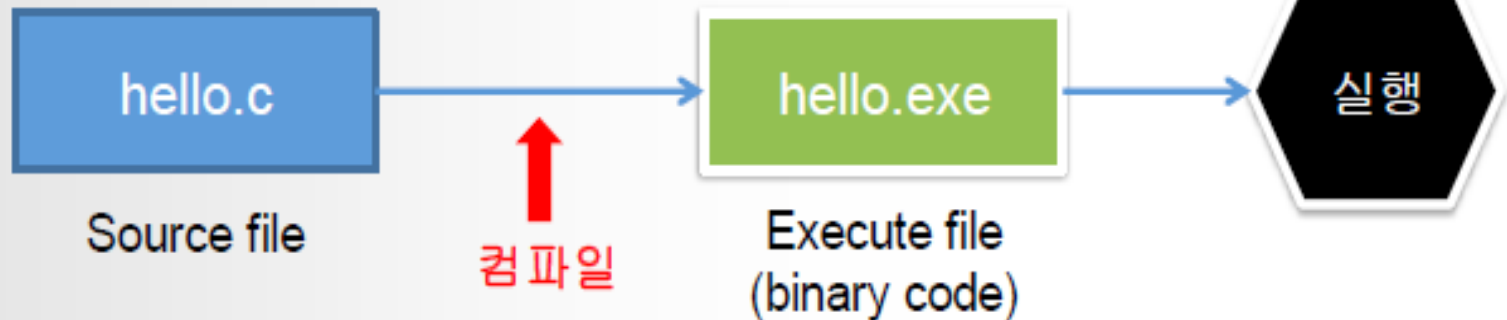
- 컴파일(Compile)되는 언어 → 베이직, 포트란, C언어
- 인터프리터(Interpreter)되는 언어 : HTML, Script언어
- 컴파일 + 인터프리터 되는 언어 : JAVA

구분	컴파일 언어	인터프리터 언어
장점	실행 속도 빠름	OS 독립적
단점	OS 종속적	실행 속도 느림

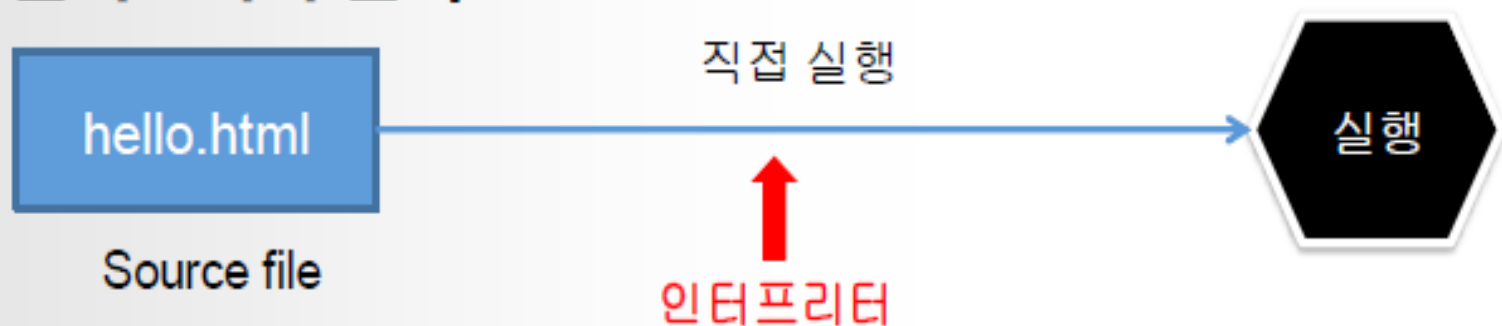
# 자바의 특징

## 2. 플랫폼에 독립적인 환경 제공

### ▶ 컴파일 언어



### ▶ 인터프리터 언어

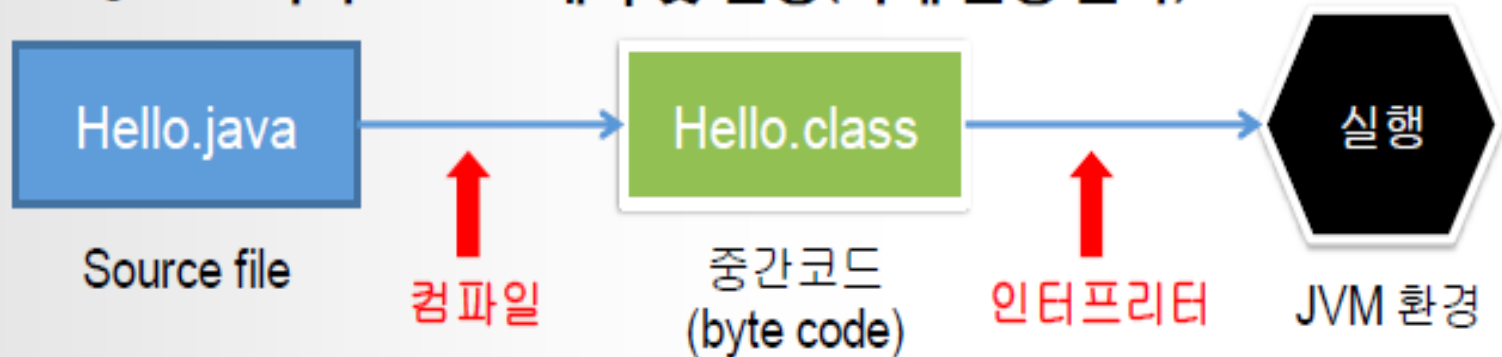


# 자바의 특징

## 2. 플랫폼에 독립적인 환경 제공

### ▶ JAVA 언어

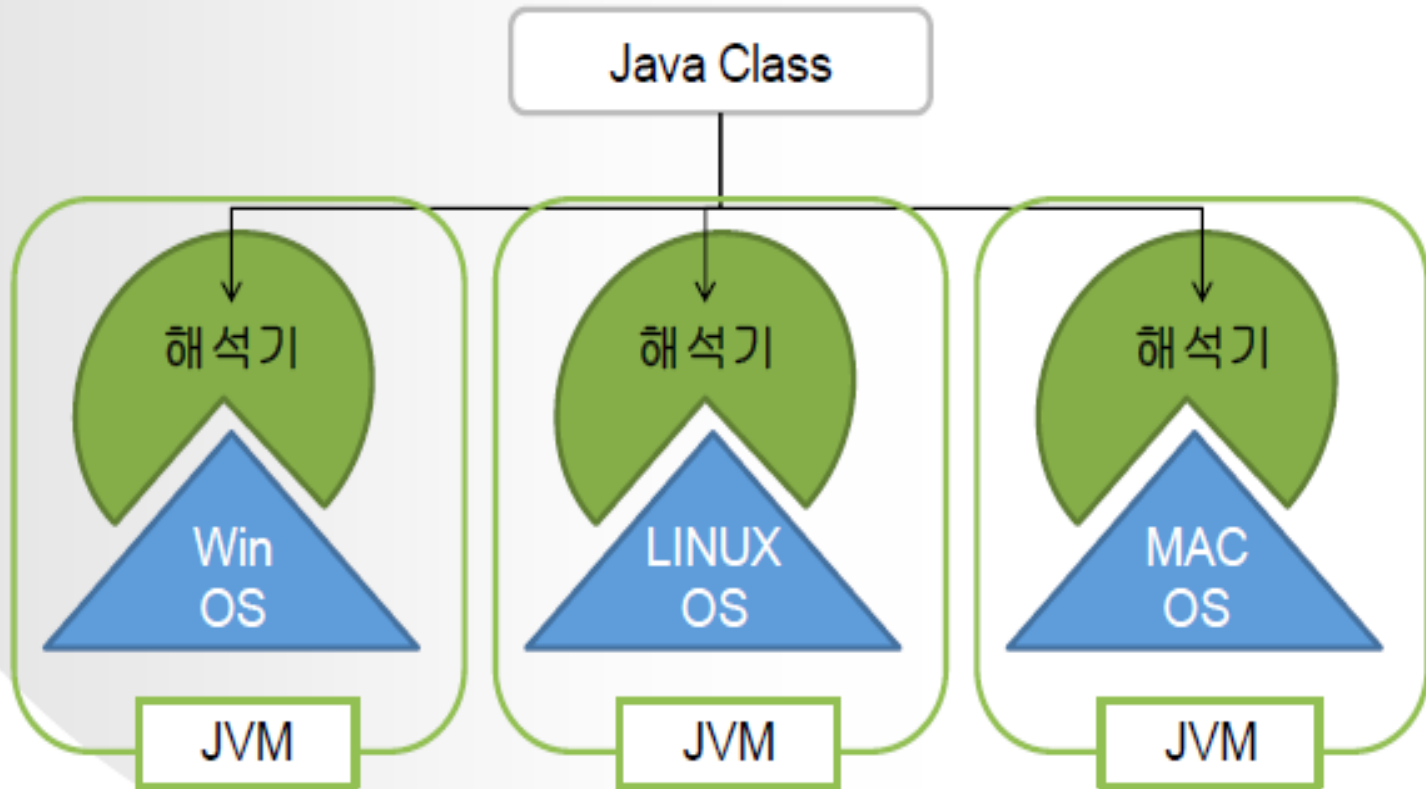
- 컴파일 언어 장점 : 빠른 기계어 생성
- 인터프리터 언어 장점 : 플랫폼 독립성
- JVM : 바이트 코드 해석 및 실행(자체 실행 불가)



# 자바의 특징

## 3. JVM(Java Virtual Machine)

- Byte code(class) 파일 자체 실행 못함
- JVM : Byte code 해석기를 포함한 시스템





# 자바의 특징

## 플랫폼 종속성(PLATFORM DEPENDENCY)



인텔 CPU를 가진  
리눅스 환경에서  
개발

C/C++  
응용 프로그램

플랫폼 = 하드웨어 플랫폼 + 운영체제 플랫폼

프로그램의 플랫폼 호환성 없는 이유

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

실행



인텔 CPU + 리눅스

실행되지  
않음



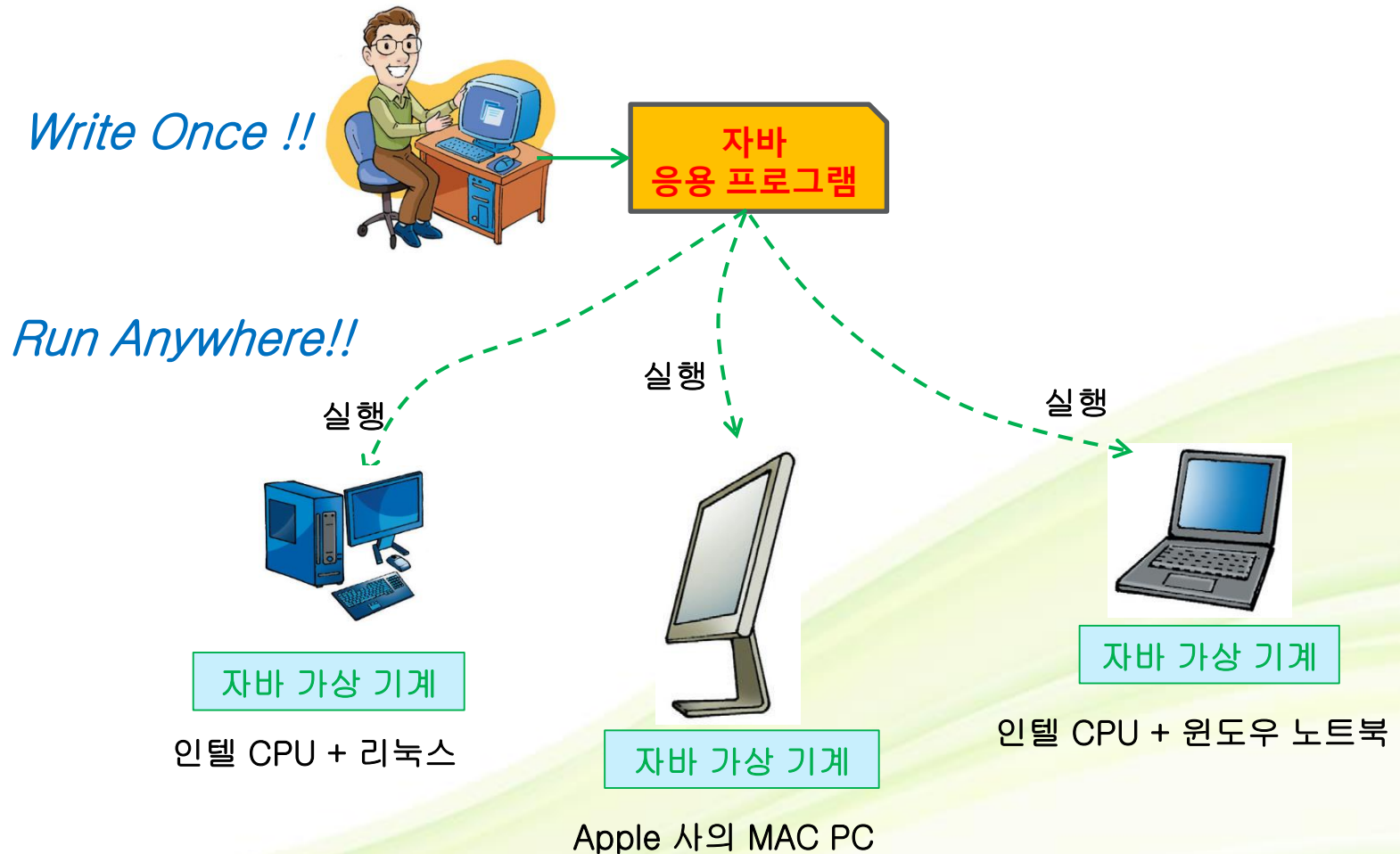
Apple사의 MAC PC

실행되지  
않음



인텔 CPU + 윈도우 노트북

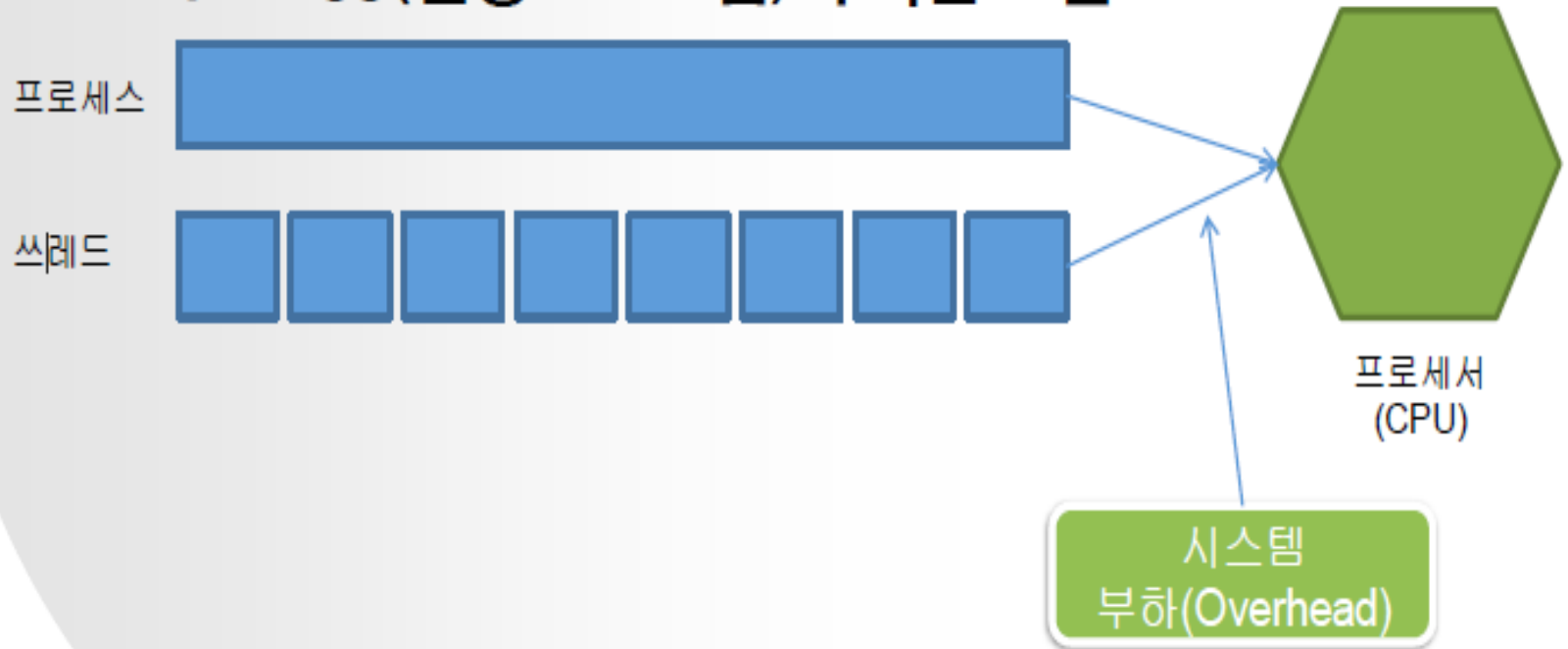
# 자바의 특징



# 자바의 특징

## 4. Thred기반 멀티태스킹

### ▶ Process(실행 프로그램)의 작은 모듈



# 자바의 특징

## 5. Garbage Collection

- 프로그램 종료 후 사용한 메모리 시스템에 반환
- 효율적인 메모리 사용
  - “메모리가 부족하다.” 메시지 해결
- 시기 : 유휴 시간(idle time)



# JDK(JAVA DEVELOPER KIT)

## ❖ 구성

- ❖ 자바 API(Application Programming Interface): 자바에서 사용하는 클래스와 인터페이스들의 모임이며 패키지의 형태로 제공
- ❖ JVM(Java Virtual Machine): 자바 프로그램을 실행할 수 있도록 해주는 물리적인 하드웨어들을 추상화 한 것으로 명령어, 레지스터, Stack, Heap, Method 영역 등으로 구성 - JRE(Java Runtime Environment)라고도 합니다.

## ❖ 플랫폼의 종류

- ❖ J2EE(Java2 Enterprise Edition)
  - ❖ 기업환경에서 서버 측 프로그램을 개발할 때 사용하는 개발 도구
  - ❖ DB연동, JSP(서버 스크립트 언어), EJB(Enterprise Java Beans)를 쉽게 사용 가능
  - ❖ JDK와 JRE를 별도로 설치
- ❖ J2SE(Java2 Standard Edition)
  - ❖ 표준 판 자바라고 일컬어지는 자바의 기본 개발 환경
  - ❖ 모든 운영체제에서 사용할 수 있고 그래픽, 네트워크 등의 기능이 있음
  - ❖ android는 J2SE 플랫폼을 이용
  - ❖ JDK와 JRE가 같이 설치
  - ❖ 외부 프레임워크를 이용하면 웹 프로그래밍을 할 수 있음(토크, 스프링, 스트럿츠)
- ❖ J2ME(Java2 Micro Edition)
  - ❖ 휴대폰이나 PDA와 같은 embedded 관련 프로그램을 개발할 때 사용
  - ❖ 메모리 관리에는 최적화 되어 있지만 전원 처리, 장치간의 입출력 기능 등은 제한적으로 제공

# JDK 개발 플랫폼

## ❖ 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
- ❖ JRE(Java Runtime Environment): 실행 환경은 JVM(자바 가상 머신)과 하위 운영체제를 연결해주는 글루(glue)로 구성- 플랫폼 고유의 라이브러리와 JNI 코드가 포함되어 있음
- ❖ 자바 가상 머신: JVM(Java Virtual Machine) - 자바 언어로 작성된 코드는 운영체제에서 직접 인식할 수 없으므로 JVM이 이해할 수 있는 중간 코드 형태인 바이트 코드를 생성(클래스 파일)
- ❖ 컴파일 명령으로 생성한 자바 바이트코드를 실행할 수 있는 가상의 운영체제 및 CPU가 JVM
  - ❖ 자바 바이트코드는 주로 자바 소스코드를 compile해서 생성하지만 다른 언어의 컴파일러에서도 생성할 수 있습니다.
  - ❖ JVM은 MS Windows, Linux, Unix, Mac OS X 등 대부분의 운영체제는 물론 핸드폰이나 가전기기와 같은 대부분의 하드웨어에도 설치 가능
  - ❖ 자바 플랫폼은 여러 플랫폼을 지원하여 MiddleWare로서의 역할과 플랫폼 스스로의 역할을 동시에 수행할 수 있습니다.

# Java 개발환경 구축

The background features a series of flowing, wavy lines in various shades of green and yellow, creating a sense of movement. Scattered throughout are several circles of different sizes and colors, including green, yellow, and light green, adding a playful and modern aesthetic to the slide.



# JAVA 프로그래밍 준비물

- JRE - 사용자를 위한 실행환경만 제공
- JDK - 실행환경과 반기계어로 변환하는 컴파일러까지 포함
- Java 사이트

과거 (<http://java.sun.com>)

현재 : 오라클

<http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html>

- 저장위치 : 임의 가능
  - JDK의 설치와 환경 설정
    - JAVA\_HOME : 자바의 홈 Directory
    - PATH : 실행 명령어 위치 지정 (bin)
- ※. 위와 동일한 방법으로 lib 폴더를 CLASSPATH라는 환경변수로 추가하기도 하는데 CLASSPATH는 이 경로에 있는 클래스들을 JVM에 먼저 로드하겠다는 의미입니다.



# JDK 다운 받기

The screenshot shows the Oracle Java SE Downloads page in a Windows Internet Explorer browser. The address bar displays the URL: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. The page features the Oracle logo at the top, followed by navigation links for Products and Services, Downloads, Store, Support, Education, Partners, and About. A search bar is also present.

The main content area is titled "Java SE Downloads" and includes tabs for Overview, Downloads, Documentation, Community, Technologies, and Training. Under the Downloads tab, there are links for Latest Release, Next Release (Early Access), Embedded Use, Real-Time, and Previous Releases. Below these links are four download buttons: Java Platform (JDK), JDK + JavaFX Bundle, JDK + NetBeans Bundle, and JDK + Java EE Bundle. Each button has a "Download" link.

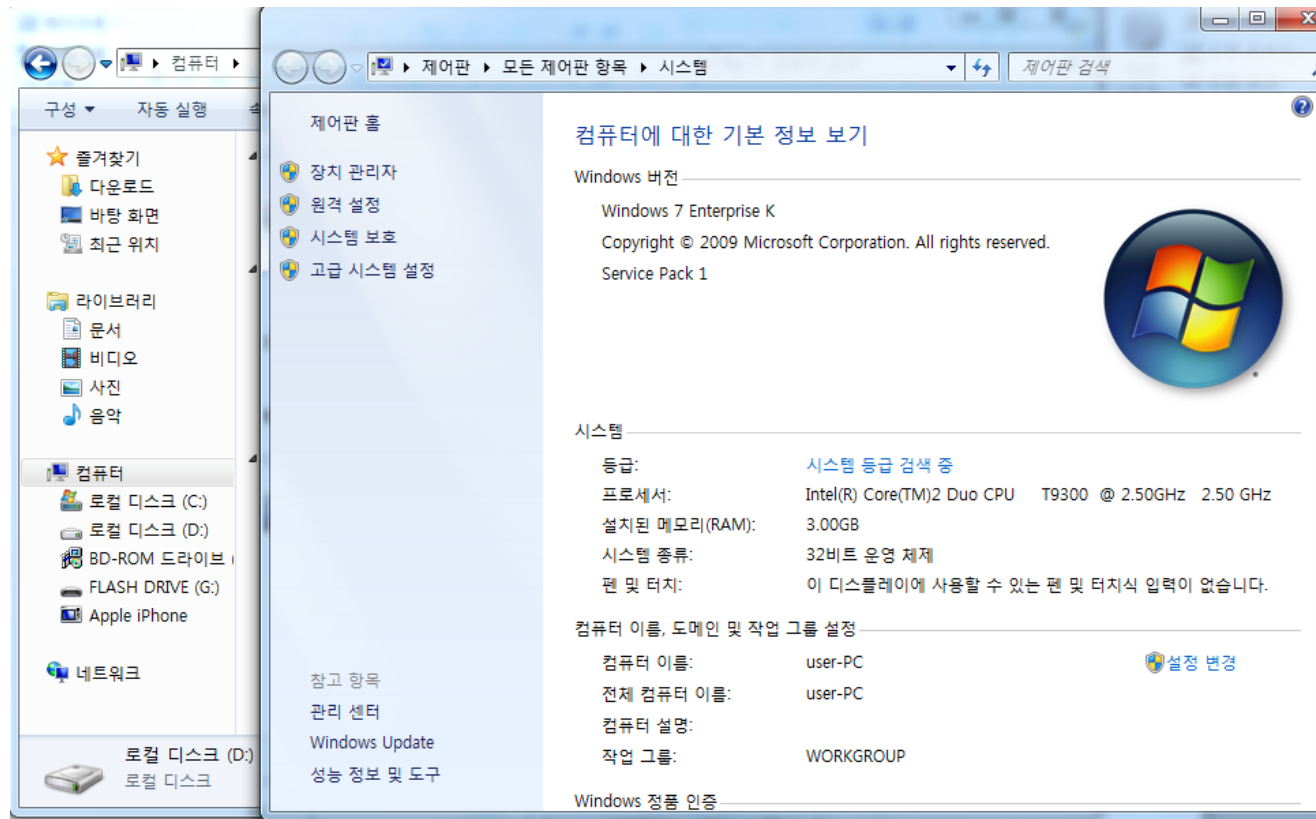
On the right side, there are sections for "Java SDKs and Tools" (listing Java SE, Java EE and Glassfish, Java ME, JavaFX, Java Card, and NetBeans IDE) and "Java Resources" (listing New to Java?, APIs, Code Samples & Apps, Developer Training, Documentation, Java BluePrints, Java.com, and Java.net).

At the bottom, there is a section for "Java Platform, Standard Edition" with the title "JDK 6 Update 21 (JDK or JRE)". It states: "This release includes performance improvements, support for Oracle Enterprise Linux, Oracle VM, and Google Chrome. [Learn more](#)". Below this, there are links for "Download JDK" and "Download JRE".

The browser's status bar at the bottom shows the text: "작업을 마쳤으나 페이지에 오류가 있습니다." (Job completed, but there is an error on the page).

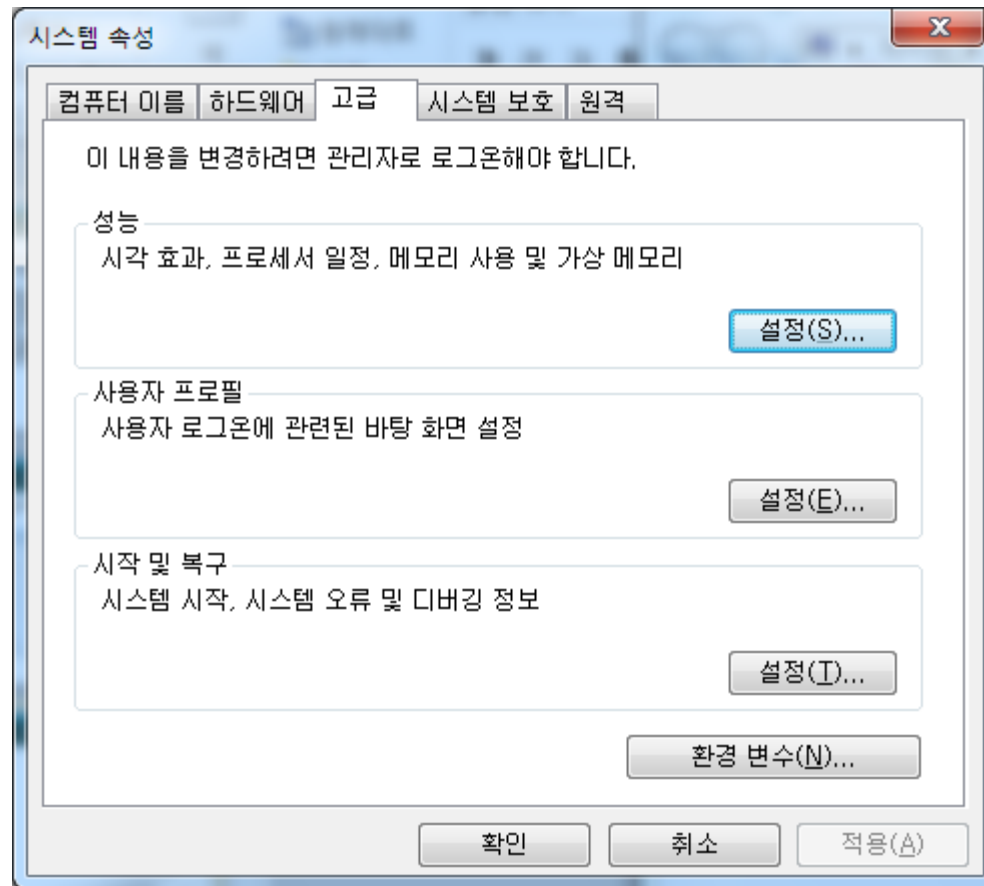
# 자바 환경변수 설정

- ❖ 탐색기에서 컴퓨터를 선택하고 마우스 오른쪽쪽을 눌러서 속성을 선택
- ❖ 제어판에서 시스템을 선택



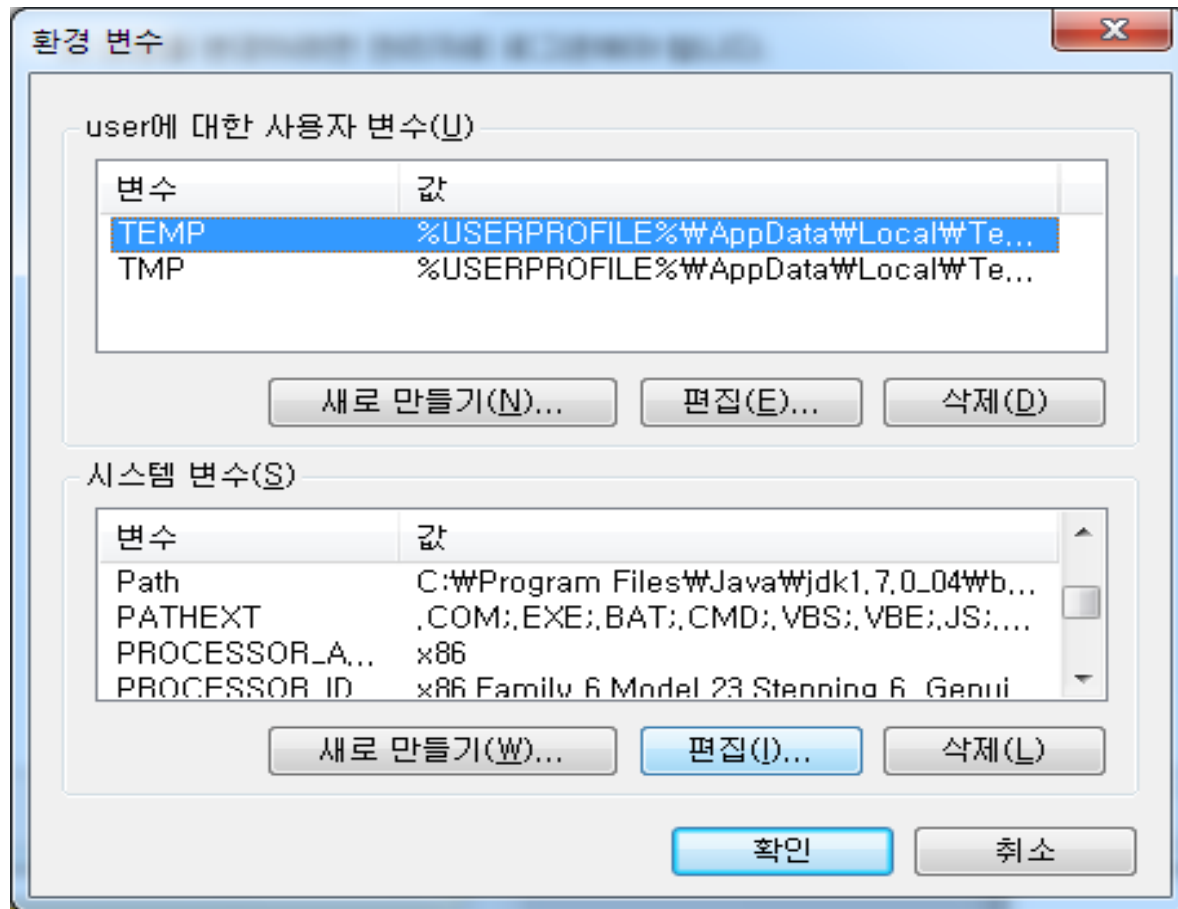
# 자바 환경변수 설정

## ❖ 고급 시스템 설정을 클릭



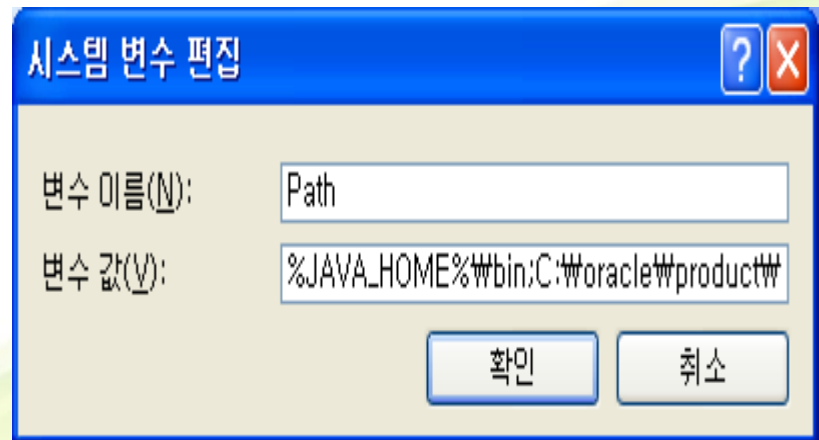
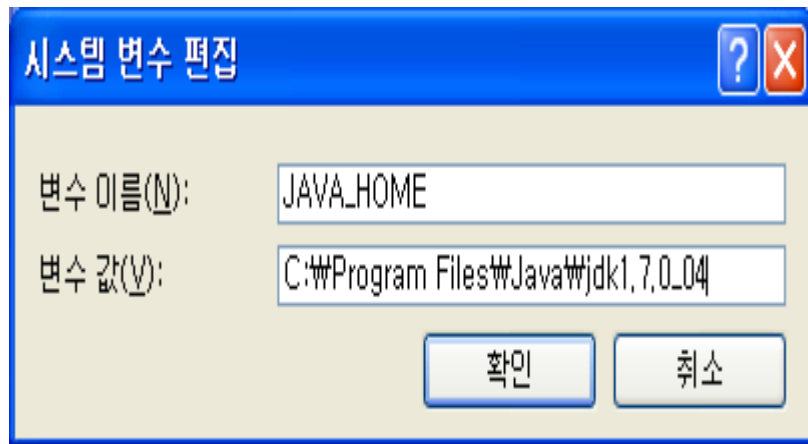
# 자바 환경변수 설정

## ❖ 환경 변수를 클릭



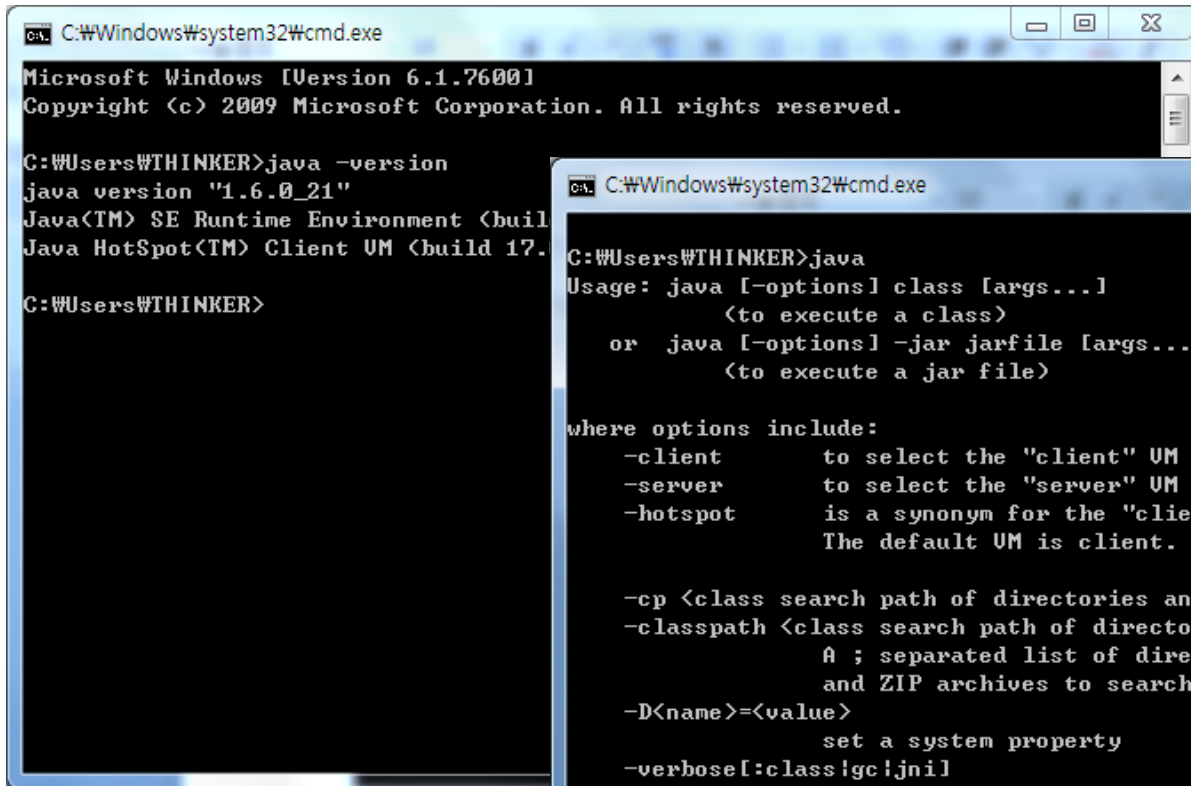
# 자바 환경변수 설정

- ❖ JAVA\_HOME 에 JDK가 설치된 디렉토리를 설정
- ❖ Path 항목에는 자바가 설치된 디렉토리의 bin 디렉토리를 추가



# JDK 설치 후 확인

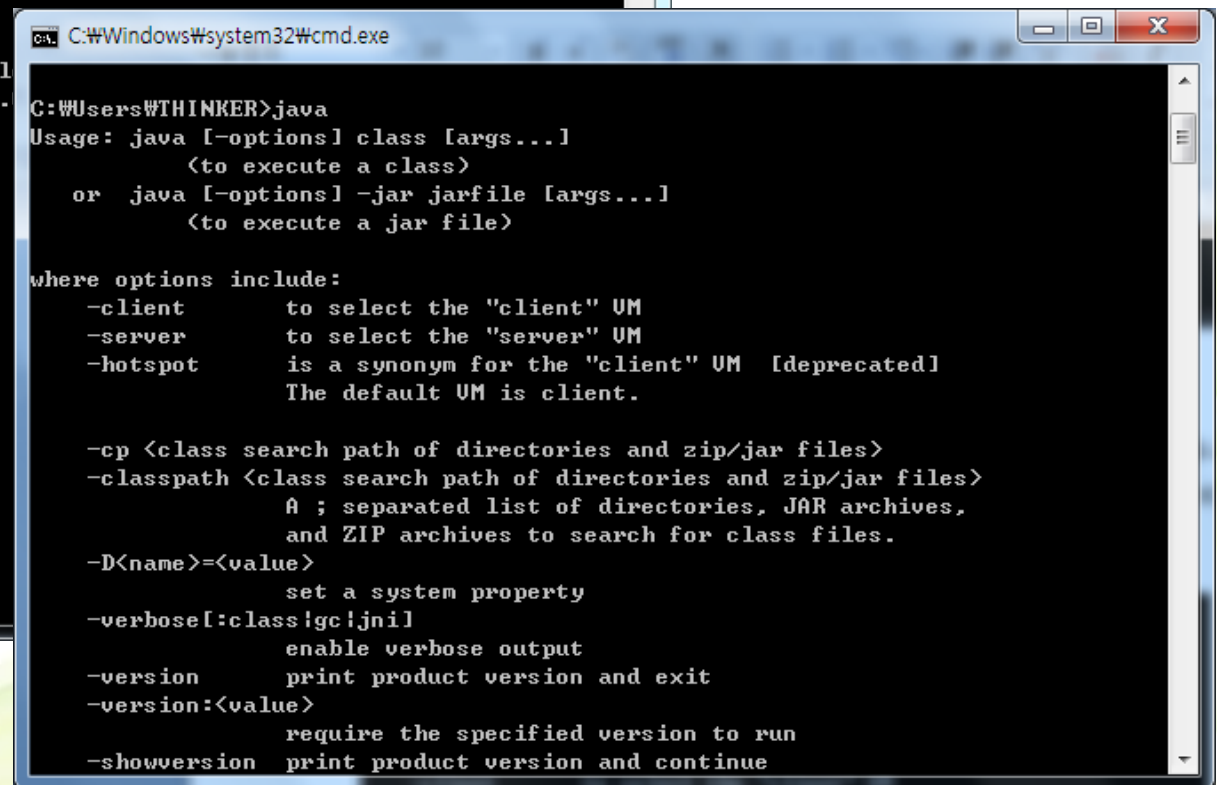
- java 명령어 실행
- java -version 명령어 실행



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\THINKER>java -version
java version "1.6.0_21"
Java(TM) SE Runtime Environment (build 1.6.0_21-b05)
Java HotSpot(TM) Client VM (build 17.0-b, mixed mode)

C:\Users\THINKER>
```



```
C:\Windows\system32\cmd.exe
C:\Users\THINKER>java
Usage: java [-options] class [args...]
           (to execute a class)
or  java [-options] -jar jarfile [args...]
       (to execute a jar file)

where options include:
    -client           to select the "client" VM
    -server           to select the "server" VM
    -hotspot          is a synonym for the "client" VM [deprecated]
                    The default VM is client.

    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A ; separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D<name>=<value>  set a system property
    -verbose[:class!gc!jni]
                    enable verbose output
    -version          print product version and exit
    -version:<value>  require the specified version to run
    -showversion      print product version and continue
```

# JDK가 설치후 부수적으로 갖춰지는 요소들

- JRE - 실행 환경
- API
- JVM

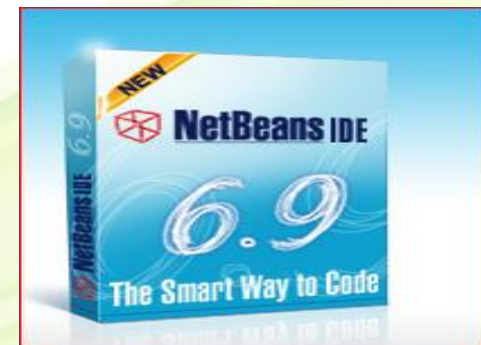
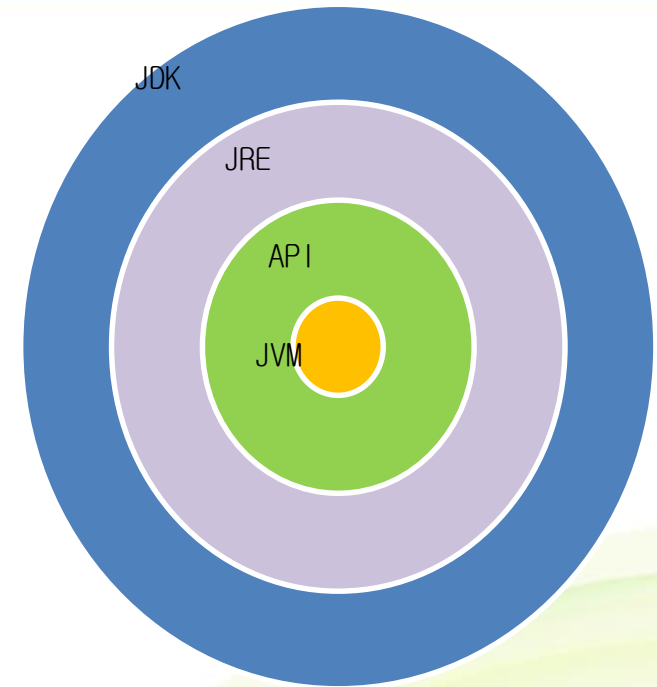
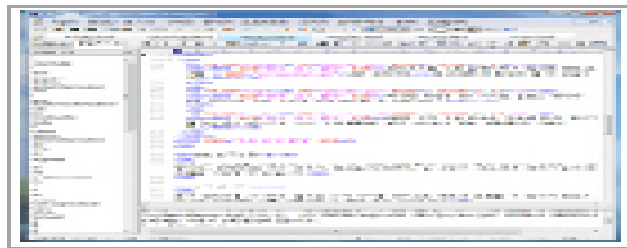
개발 환경

- Eclipse



- NetBeans

- EditPlus





## 컴파일(compile)이란?

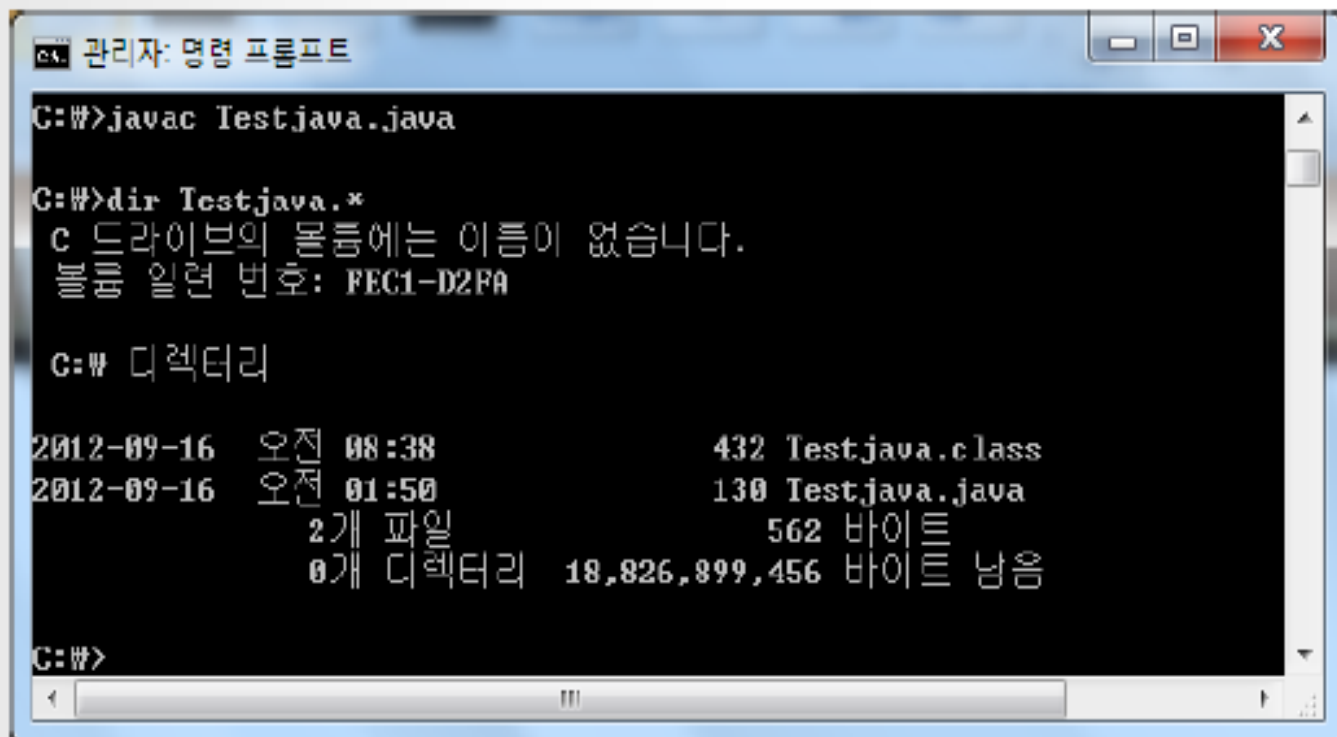
- 컴파일러에 의해서 소프 프로그램을 기계어(0과 1로 구성된 이진 파일)로 번역한 후, 컴퓨터가 기계어 파일을 실행하는 방식





## ● 바이트코드(class 파일) 확인

➤ 명령 수행 : `dir TestJava.*`



```
관리자: 명령 프롬프트
C:\>javac Testjava.java

C:\>dir Testjava.*
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: FEC1-D2FA

C:\> 디렉터리

2012-09-16 오전 08:38                432 Testjava.class
2012-09-16 오전 01:50                130 Testjava.java
                2개 파일                562 바이트
                0개 디렉터리 18,826,899,456 바이트 남음

C:\>
```

## 바이트코드란?

- 플랫폼에 독립적인 자바 코드



- 명령 프롬프트에서 실행

- ▶ 명령 수행 : java TestJava



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

C:\Users\user>cd W

C:\W>javac JavaTest.java

C:\W>java JavaTest
Java Test!!!

C:\W>_
```

# HELLO JAVA

- 메모장을 이용하여 작성 및 실행
- C:/java/src에 Hello.java란 파일이름으로 저장한다.

// 프로그램 : HelloJava.java

```
public class HelloJava{  
    public static void main (String args[]){  
        System.out.println( "Hello, Java!!" );  
    }  
}
```

## (1) 클래스 정의

- 자바 프로그램은 확장자가 “java”인 소스 파일을 하나 만들어서 자바 문법에 맞는 내용을 기술해야 한다.
- 자바는 클래스를 하나의 단위로 프로그램을 작성하기에 자바 소스 파일 안에 클래스를 정의해야 한다.
- 제대로 동작하려면 자바 소스 파일명이 클래스명과 동일해야 한다.

## (2) main 메소드 정의

- 프로그램의 진입점이 다.
- 프로그램을 실행시키면 main 함수 내부에 기술된 내용들을 순차적으로 수행한다.

## (2) MAIN 메소드 정의

○ `public static void main(String[] args)`

①      ②      ③      ④              ⑤

- ① `public` : 누구나 접근 가능하도록 하기 위한 예약어로 접근 지정자의 일종이다.
- ② `static` : `static`으로 선언된 메소드는 클래스만 존재하면 수행할 수 있도록 한다.
- ③ `void` : 값을 갖지 않는다는 의미를 갖는 자료형태이다
- ④ `main`
  - 자바는 JVM에 의해서 실행되는데 자바 애플리케이션을 실행시키면 JVM은 이름이 `main` 메소드를 찾아 이 내부에 기술된 내용들을 순차적으로 실행한다.
  - 그래서 `main` 메소드를 프로그램의 진입점이라고 한다.
- ⑤ `String[] args`
  - 메소드를 실행시키기 위한 재료가 될 만한 데이터를 전달받아야 할 경우 메소드\_이름 다음에 기술하는 `()`을 사용한다.
  - `()` 안에 기술한 `args`가 메소드에 값을 전달했을 경우 이를 받아 올 수 있는 전달인자가 된다.

### (3) 문장

- main 메소드까지 정의했다면 이 메소드 안에 수행할 내용을 기술할 차례이다.
- 자바는 문장 단위로 프로그램을 작성해야 한다.
- 메소드 내부에 기술할 문장으로는 변수의 선언문이나 다른 메소드를 호출하는 문장들이 있다.
- JAM은 세미콜론으로 끝나면 이를 하나의 문장으로 인식한다.
- 반드시 문장의 끝을 세미콜론으로 마감해야 한다.

`System.out.println( "Hello Java" );`

문장의 끝을 세미콜론으로 마감

### (4) 출력을 위한 문장

`System.out.println( "Hello Java" );` ← 출력할 내용

- 위 문장은 화면에 "Hello Java"를 출력하라는 내용이다.
- `System.out.println( )` 메소드는 "(큰 따옴표)로 둘러싸인 문자열을 화면에 출력하는 역할을 한다

# 자바 프로그래밍하기

## (1) 편집기로 소스 작성하기

### ● 자바 프로그램을 작성할 때 유의할 점

1. 자바 소스 파일명이 클래스명과 동일한 "Hello"로 확장자는 "java"로 지정해야 한다.
2. 클래스 내부에 메소드가 포함되어 있다는 것을 한 눈에 볼 수 있도록 들여쓰기를 한다. 이렇게 작성해 두어야만 프로그램을 분석하기가 쉽다.
3. 반드시 대소문자를 구분해서 기술해야 한다.

## (2) 컴파일

● 컴파일은 자바 컴파일러인 "javac.exe" 명령어로 수행해야 한다.

● javac 다음에 자바소스파일명과 함께 확장자 java를 반드시 입력해야 한다.

```
c:\W>javac Hello.java
```

● 자바로 만들어진 소스(파일명.java)가 에러 없이 성공적으로 컴파일하면 결과물로 파일명.class 형태의 클래스 파일(바이트 코드)을 얻을 수 있다.



### (3) 실행

- 컴파일의 결과로 생성된 클래스 파일을 실행하기 위한 도구가 바로 자바의 인터프리터(Interpreter)인 "java.exe" 명령어이다.
- 이 명령어는 컴파일러에 의해서 생성된 바이트 코드(파일명.class)를 자바 가상머신(JRE)에서 실행하도록 해주는 개발도구이다.
- 이 명령어로 실행하려면 java 다음에 확장자는 생략하고 자바클래스파일명만 기술해야 한다.
- c:\>java Hello

# 자주 발생하는 에러

- ❖ **cannot find symbol, cannot resolve symbol**
  - ❖ 변수나 Method를 찾을 수 없는 경우
- ❖ **‘;’ expected**
  - ❖ ;이 빠진 경우
- ❖ **exception in thread main java.lang.NoSuchMethodError: main**
  - ❖ main Method를 찾지 못하는 경우
- ❖ **exception in thread main java.lang.NoClassDefFoundError : 클래스명**
  - ❖ 클래스를 찾을 수 없는 경우
- ❖ **illegal start of expression**
  - ❖ 문법적 오류(여는 괄호와 닫는 괄호의 불일치 등)
- ❖ **class, interface, or enum expected**
  - ❖ 키워드가 생략된 경우

## ❖ 이클립스(Eclipse) 소개

- 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++ 개발 환경

# 이클립스 설치

## ❖ 이클립스 다운로드

- 이클립스는 자바 언어로 개발된 툴 - JDK 필요
- 다운로드 사이트: <http://www.eclipse.org>
  - Eclipse IDE for Java Developers 버전
- 순수 자바 학습용
  - Eclipse IDE for Java EE Developers 버전
- 웹 애플리케이션 등의 Enterprise (Network) 환경에서 실행
  - CPU 사양에 맞게 다운로드

# 이클립스 설치

## ❖ 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 디렉토리 삭제하고 이클립스 실행 - 초기 상태로 다시 실행

# 이클립스 설치

## ❖ 퍼스펙티브(Perspective)

- 개발 프로젝트 종류별로 유용한 View들을 묶어놓은 것

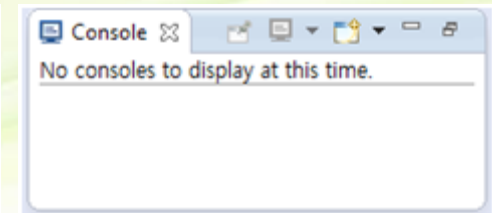
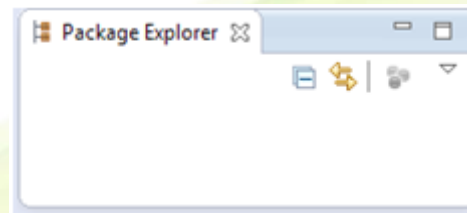


- Eclipse IDE for Java EE Developers
  - 기본적으로 Java EE 퍼스펙티브
  - 책에서는 Java 퍼스펙티브로 변경해 사용



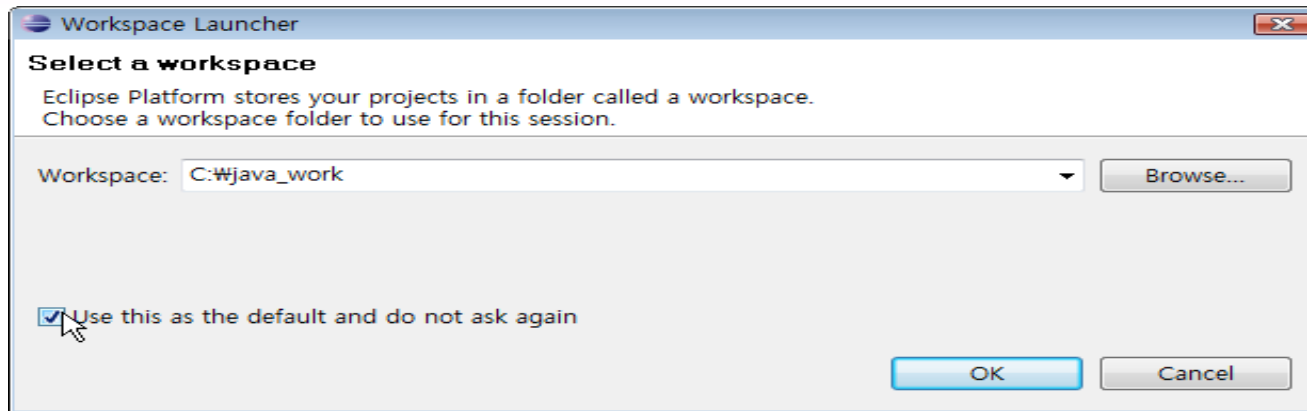
## ❖ 뷰(View)

- 퍼스펙티브를 구성하는 작은 창으로 여러가지 목적에 맞게 내용 보여줌
- 자유롭게 제거 하거나 추가 가능
- 우리 책에서 유용한 뷰들
  - Package Explorer
  - Console

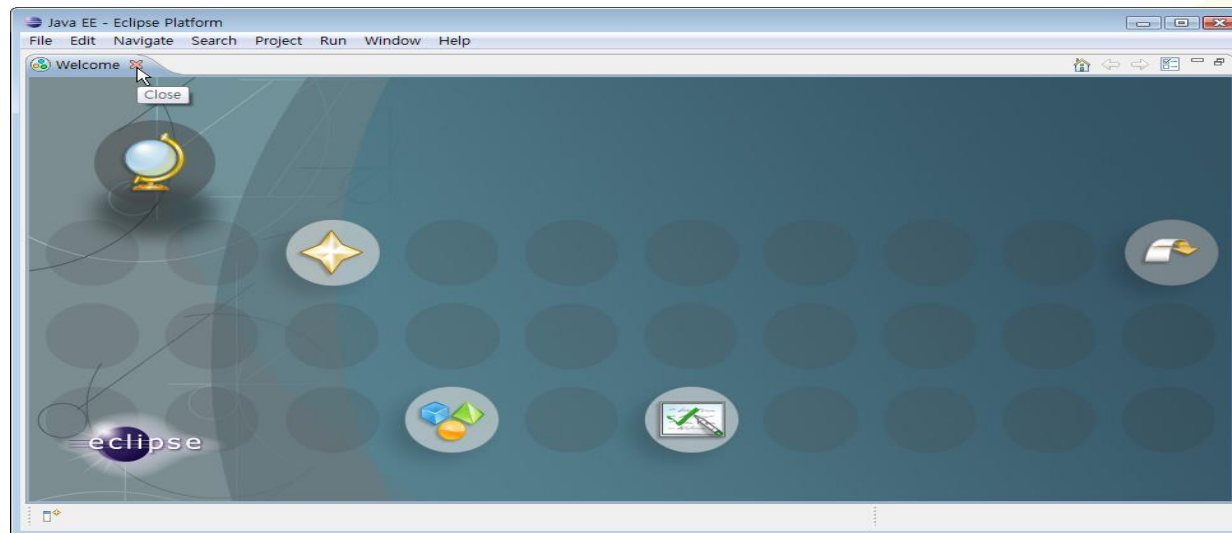


# 개발 툴인 이클립스를 설치하여 실행하기

- 자바 기반의 프로그램을 작성하려면 편집 프로그램을 설치해야 한다. 사용자가 편리한 것으로 JBuilder, KAWA, Visual Age For Java, Forte 등의 자바용 편집기나, EditPlus, 메모장 같은 범용 편집기를 사용해도 된다. 그러나 요즘은 자바 기반 오픈소스 플랫폼인 이클립스를 사용한다. 이클립스는 무료로 사용할 수 있는 개발 툴로, 편리하게 어플리케이션을 개발할 수 있어서 각광을 받고 있다.
- 이클립스 프로젝트가 지향하는 바를 요약하면 다음과 같다.
  - 애플리케이션 개발 툴을 위한 개방형 플랫폼  
: 다양한 OS 지원, GUI와 비GUI를 모두 지원한다.
  - 언어 중립성 : 콘텐츠의 형식(HTML, JAVA, C, JSP, EJB, XML,...)을 제한하지 않는다.
  - 여러 다양한 툴을 조화롭게 통합 : UI(User Interface)는 물론, 더 깊은 수준에서 통합이 가능하다. 또한 이미 설치된 제품에 새로운 도구를 자유롭게 추가할 수 있게 한다.



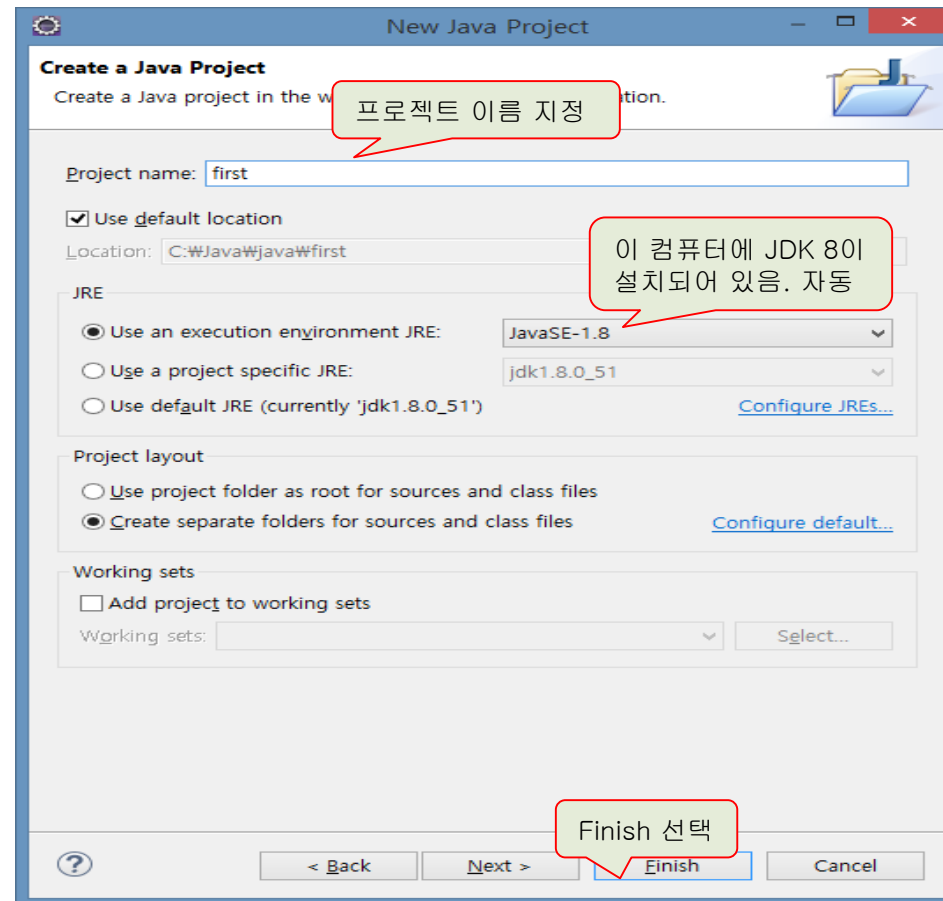
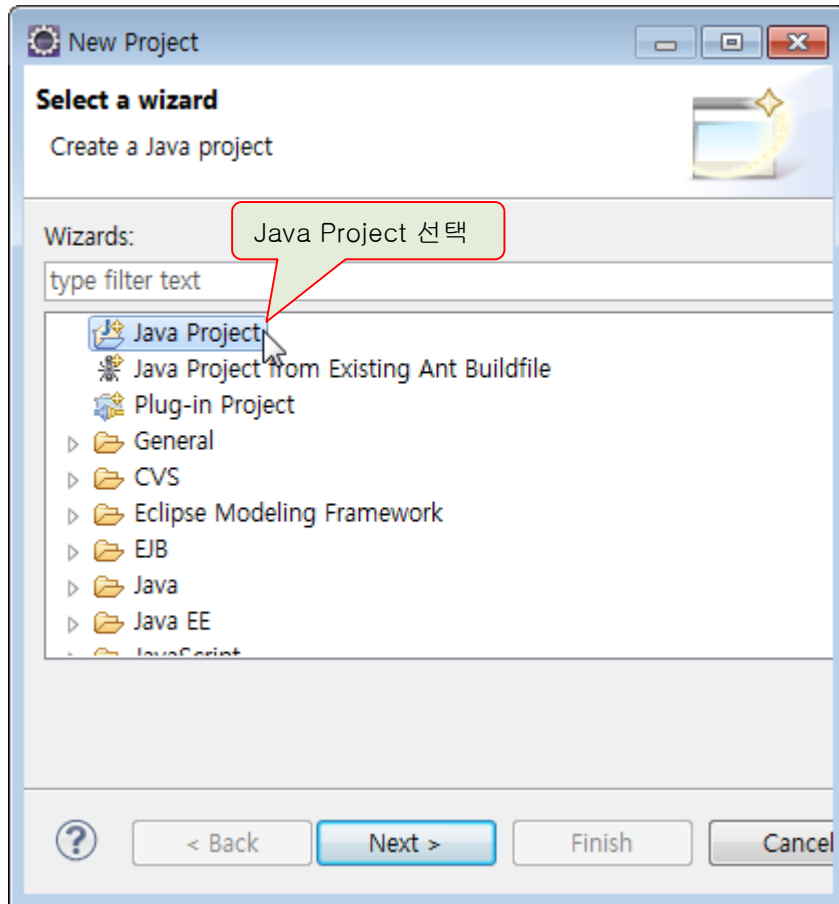
- [Welcome] 창이 나타난다. [Welcome] 창의 닫기[x] 버튼을 클릭한다.





# Eclipse

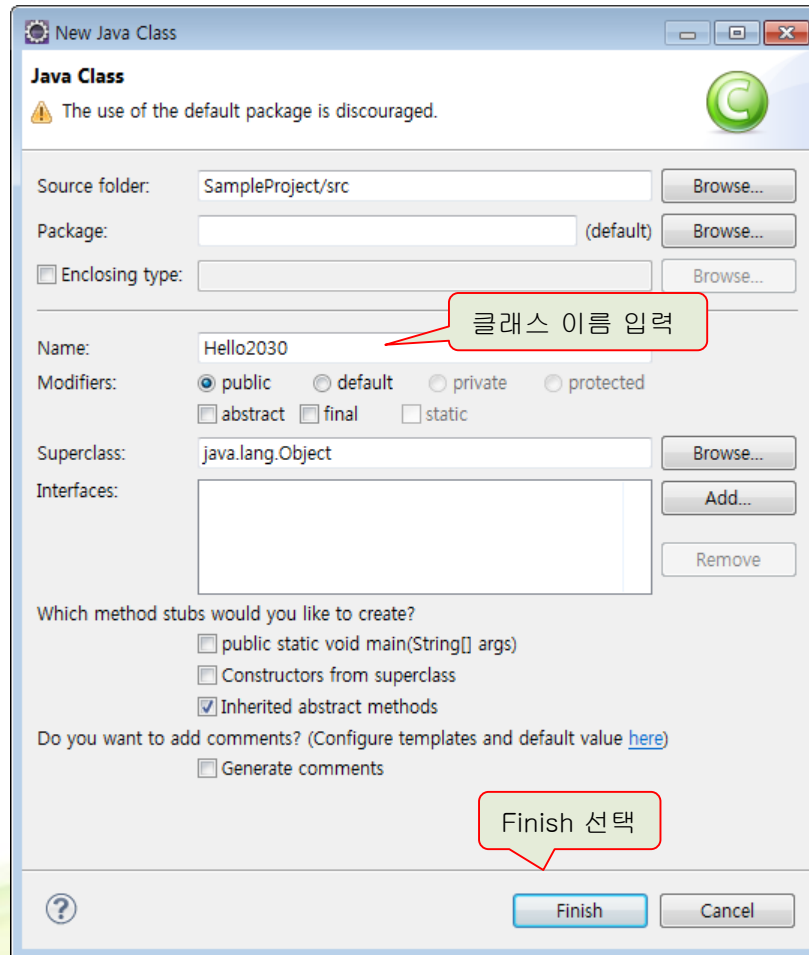
## 프로젝트 생성




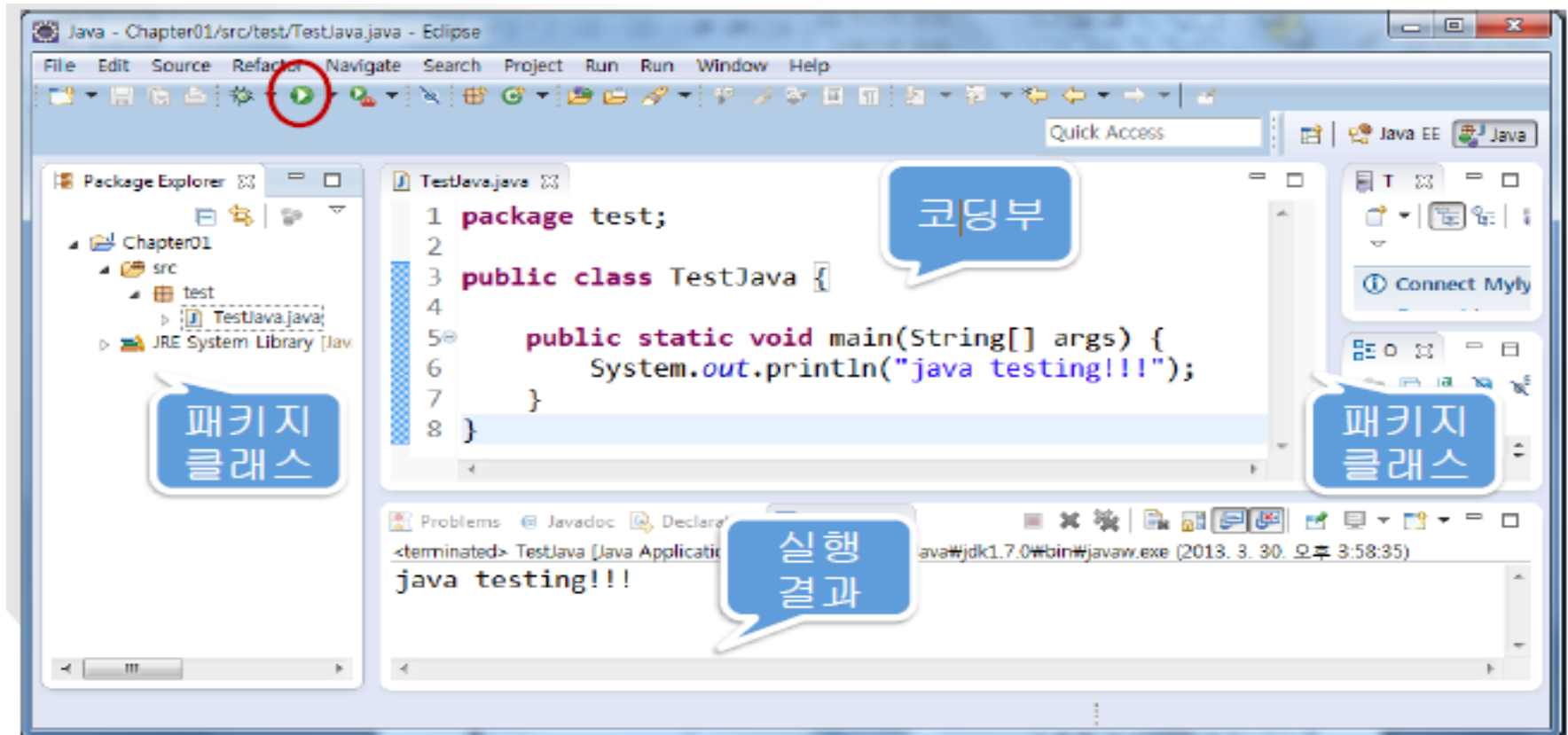
# Eclipse

## 클래스 생성

File->New->Class 메뉴 선택



- 이렇게 생성된 클래스 파일을 실행해보자.  아이콘에서 마크를 클릭해서 표시되는 메뉴에서 [Run As]-[Java Application] 메뉴를 선택한다.
- 화면 아래쪽 [Console] 창에 "Hello World !"가 출력되었음을 확인할 수 있다.



# 디버깅하기

- 자바 컴파일러는 문법에 맞게 자바 소스 파일이 작성 되어야만 클래스 파일을 생성한다. 아래는 오류가 발생하는 문장이다.

The screenshot shows the Eclipse IDE interface. The main editor displays the following Java code:

```
/* Hello World !를 콘솔창에 출력하는 자바 애플리케이션 */
public class HelloWorld {

    /**
     * @작성일 : 2009년 4월
     * @작성자 : 김도형
     */

    public static void main(String[] args) {
        // 메시지를 출력하는 메소드를 호출하는 문장
        System.out.println("Hello World !")
    }
}
```

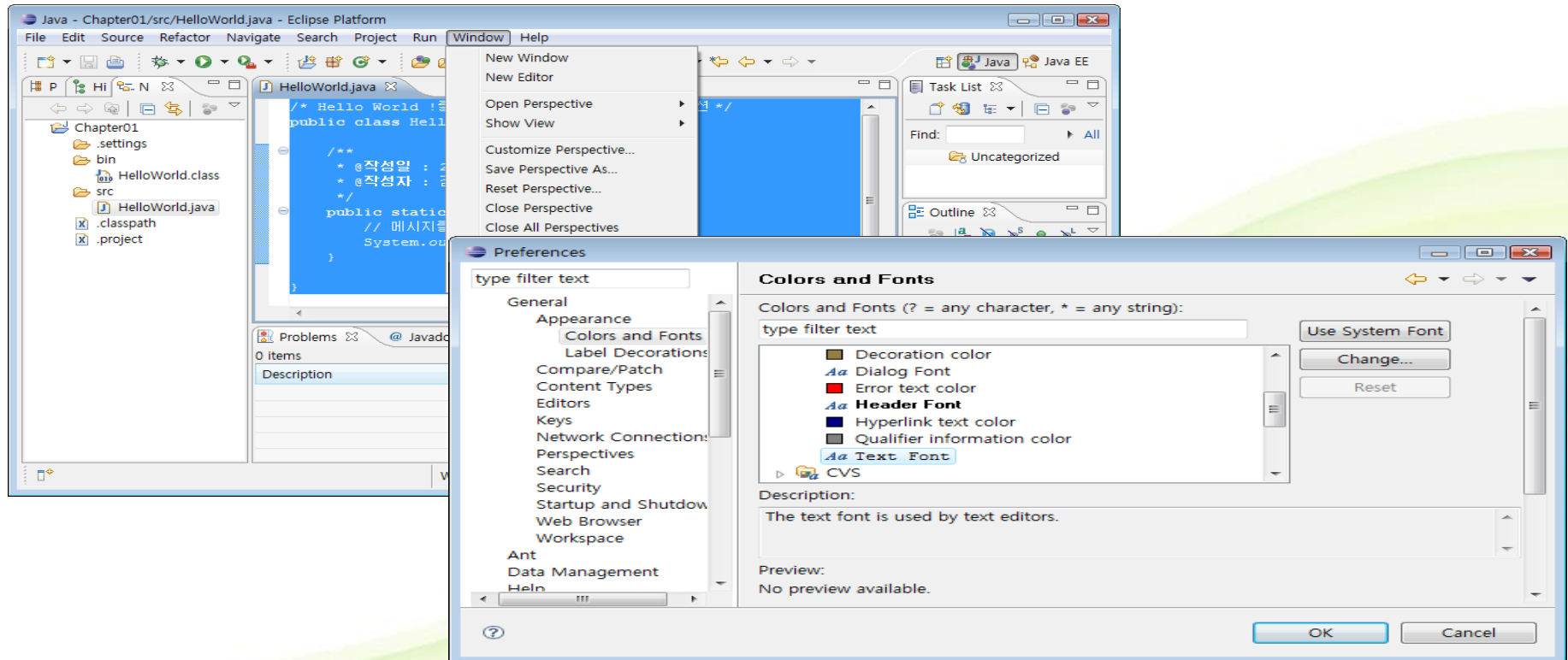
The IDE shows two syntax errors in the Problems view:

Description	Resource	Path	Location	Type
2 errors, 0 warnings, 0 others				
Errors (2 items)				
Syntax error, insert "}" to complete ClassBody	HelloWorld.java	Chapter01/src	line 13	Java Problem
Syntax error, insert ";" to complete BlockStatements	HelloWorld.java	Chapter01/src	line 10	Java Problem

The status bar at the bottom indicates: "Syntax error, insert ";"...complete BlockStatements | Writable | Smart Insert | 10 : 44".

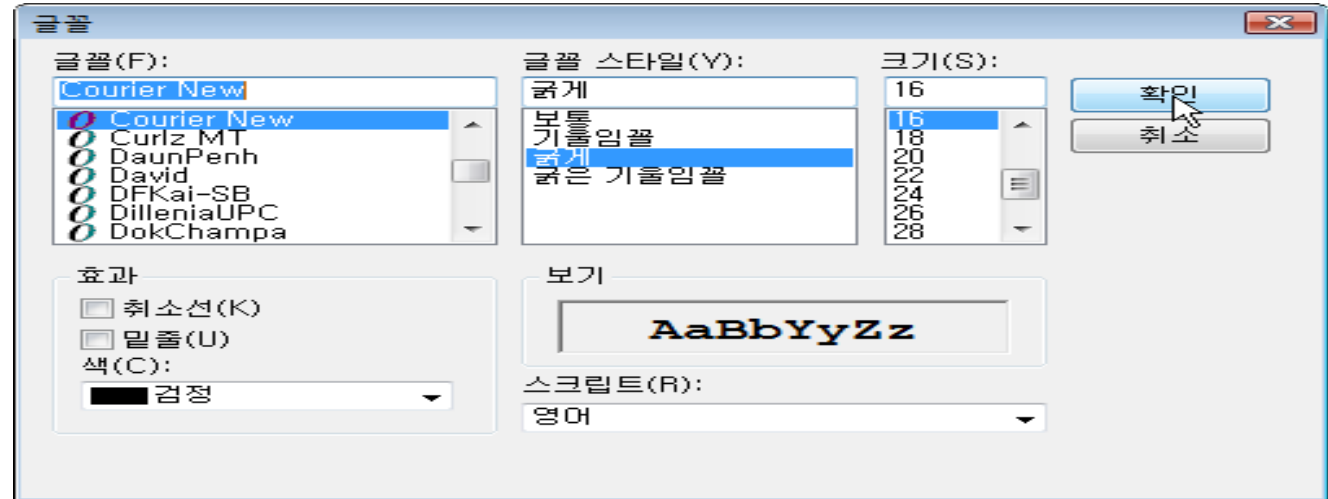
# 글꼴 변경과 라인 번호 표시하기

- 메뉴바에서 [Window]-[Preferences]를 선택하면 [Preferences] 창이 나타나는데 이곳에서 다양한 환경설정을 할 수 있다.
- [Preferences] 창이 나타나면 [General]-[Appearance]-[Colors and Fonts] 항목을 선택한다. 화면 가운데 [Basic]-[Text Font] 항목을 선택한 후 [Change...] 버튼을 클릭한다.

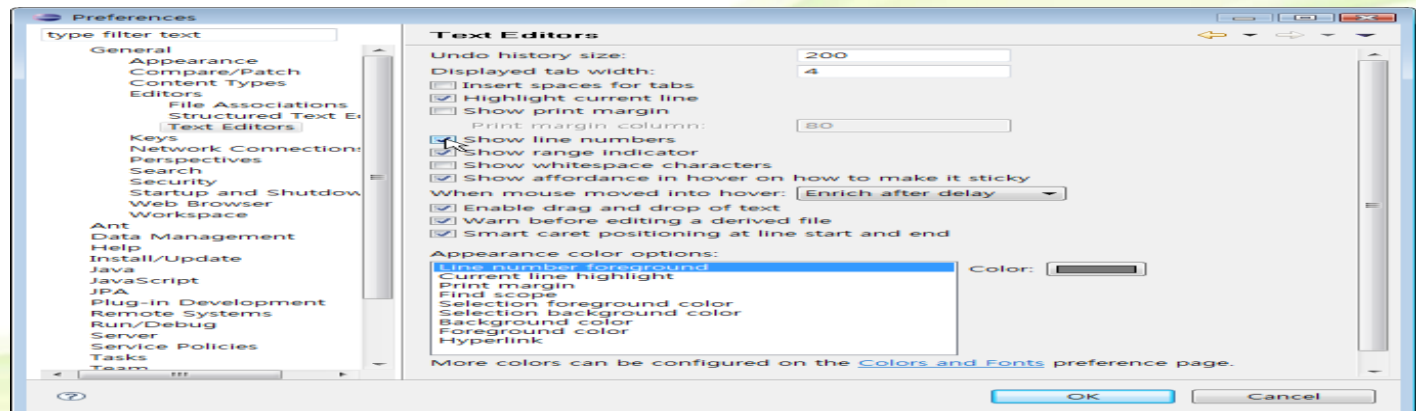


# 글꼴 변경과 라인 번호 표시하기

- [글꼴] 대화상자가 나타나면 원하는 글꼴을 선택하고 [확인] 버튼을 클릭한다.



[에디터] 뷰에 라인번호가 표시되지 않아서 불편하므로, 라인번호가 표시되도록 이번에는 [General]-[Text Editors] 항목을 선택한 후 [Show line numbers] 항목을 체크한 후에 [Ok] 버튼을 클릭한다.



# 자바 프로그래밍하기

- 우리가 보통 "자바를 한다!"하고 이야기할 때는 자바 프로그램을 입력하여 컴파일하고 디버깅하고, 필요에 따라서는 문서화도 시키고, 결과로 나온 실행파일(클래스)을 실행하고 관리하는 작업을 전부 말한다. 여러분이 본서를 통해서 학습할 내용은 프로그램을 개발과 자바를 이해하는 부분으로 나뉜다.



# 자바 프로그램의 기본 구조

## ● 클래스와 메소드 정의

클래스를 정의하는 문장이다. 자바로 프로그램을 작성한다는 것은 확장자가 “java” 인 소스 파일을 하나 만들어서 자바 문법에 맞는 내용을 기술하는 것을 의미한다.

## ● System.out.println( “Hello World !” );

“Hello World !”를 출력하기 위한 메소드를 호출하는 문장이다.

① public class HelloWorld {

클래스 정의

② public static void main(String[] args) {

메소드 정의

③ System.out.println(“Hello World !”);

}

}



# 주석문

- 주석문은 프로그램에 대한 설명을 덧붙이므로 코드를 이해하기 쉽게 하기 위해서 사용한다.
- 주석문은 실행 중에 영향을 미치지 않는다

## (1) /\* ~ \*/ 주석문

/\* 로 시작해 \*/ 이 나올 때까지 모든 내용이 주석 처리가 된다. 여러 줄에 걸쳐 블록 단위로 주석 처리할 경우 사용한다.

## (2) // 주석문

// 뒤에 있는 한 줄만 주석처리 된다.

## (3) /\*\* ~ \*/ 주석문

여러 문장을 주석 처리할 수 있다는 면에서 /\* ~ \*/ 와 유사한 기능을 갖는다.

## 가독성을 높이기 위한 들여쓰기

- 클래스나 메소드의 시작과 종료를 나타내는 { } 내에 내용을 기술할 때는 가독성을 높이기 위해 들여쓰기를 한다.

## ● 주석 예제

```
class Comment {    // 프로그램 시작 부분
    public static void main(String[] args) {
        boolean a = (3 > 10);
        boolean b = (10 > 3);
        //System.out.println( "a = " + a + " b = " + b );
    }
}
```

```
class Comment {    // 프로그램 시작 부분
    public static void main(String[] args) {
        //    boolean a = (3 > 10);
        boolean b = (10 > 3);
        System.out.println( "a = " + a + " b = " + b );
    }
}
```

## 주석문

```
public class Hello {  
    /**  
     * @작성자 : 홍길동  
     */  
    public static void main(String[] args) {  
        /* 모니터에 Hello Java를 출력하는  
         간단한 예제이다. */  
        System.out.println("Hello Java"); //출력을 위한 문장  
    }  
}
```

```
public class Comment2 {
```

```
/**
```

```
    boolean 값 변수 a, b 처리 확인
```

```
    프로그램 최종 수정일 : 2032/12/25
```

```
*/
```

```
    public static void main(String[] args) {
```

```
        boolean a = /* a정의 */ ( 3 > 10 );
```

```
        /* boolean b = ( 10 > 3 );
```

```
        System.out.println( "a = " + a + " b = " + b);
```

```
    */
```

```
    }
```

```
}
```

# 자바의 기본 출력

● `System.out.println("Hello Java");`

필드	설명
in	키보드로부터 입력받기 위한 필드
out	콘솔로 출력하기 위한 필드
err	에러 메시지를 콘솔로 출력하기 위한 필드

# 자바의 기본 출력

메소드	설명
<code>println(내용)</code>	<code>ln</code> 이 <code>LiNe</code> 의 약자인 것을 보면 알 수 있듯이 이 메소드는 자동 개행하기에 내용을 출력하고 줄이 바뀐다.
<code>print(내용)</code>	내용을 출력한 후에 마지막 출력한 문자 뒤에 다음 내용을 출력하므로 줄이 바뀌지 않는다.
<code>printf("형식지정자", 내용)</code>	형식지정자에 맞게 내용을 출력한다.

# 출력을 위한 메서드 살피기

```
public class Ex02 {  
    public static void main(String[] args) {  
        //이름 : 을 출력한 후 줄이 바뀌지 않는다.  
        System.out.print("이름 : ");  
        //이름 : 뒤에 연속해서 홍길동을 출력한 후 줄이 바뀐다.  
        System.out.println("홍길동");  
        //직업 : 을 출력한 %s 위치에 교수를 출력한 후 줄이 바뀌지 않는다.  
        System.out.printf("직업 : %s ", "교수");  
    }  
}
```

```
/*
파일 이름: BlockComment.java
작성자: 홍길동
작성일: 2012년 9월 25일
작성이유: System.out.println 메소드 기능 테스트
*/
```

```
class SystemOutPrintln{
    public static void main(String[] args)    {
        System.out.println(7);    /* 정수의 출력 */
        System.out.println(3.15);
        System.out.println("3+5=" + 8);
        System.out.println(3.15 + "는 실수입니다.");
        System.out.println("3+5" + "의 연산결과는 8입니다.");
        System.out.println(3+5);    /* 덧셈 결과 출력 */
    }
}
```



# ECLIPSE 단축키

## ❖ 자주 사용하는 이클립스 단축키

1. `/* */` : 주석처리 -> 블록 지정한후에 `ctrl+alt+/*` 주석 해제는 `ctrl+alt+\\`  
`//` : 한줄 주석처리 -> `ctrl+/*`
  2. 자동 완성 기능 : `ctrl+spacebar`
  3. 줄이동 : `atl+방향키(위,아래)` -> 커서가 있는 줄의 모든 글자가 방향키에 따라 이동  
줄삭제 : `ctrl+D` -> 커서가 있는 줄을 삭제
  4. 에러 픽스 : 에러난 부분(벌건줄)에서 `ctrl+1` ->에러난 곳에대해 해결방법을 제시함
  5. Undo/Redo : `ctrl+Z/ctrl+Y`
  6. `System.out.println();` 생성 : `sysout` 입력하고 `ctrl+spacebar`
  7. 들여쓰기 자동 수정 : `ctrl+i` -> 커서가 있는 줄의 들여쓰기를 자동으로 맞춰준다. 블록을 지정하고 실행시 블록내에서 자동 들여쓰기
  8. `shift+alt+s r` : getter/setter 자동 생성
- F11 : 디버깅 시작
- F4 : 상속 구조 클래스 보기(Method 등)
- `alt+shift+r` : 변수 및 Method 변경(변경할 변수 에서 단축키를 누르고 변경 후에 엔터를 누르면 변수명이 모두 변경)
- `ctrl+m` : 에디터 화면 넓게/좁게

# ECLIPSE 단축키

## \* Alt

- > Alt + Shift + J : JavaDoc 주석
- > Alt + ←→(좌/우) : 뷰 화면의 탭에 열린 페이지 이동
- > Alt + ↑↓(상/하) : 커서가 있는 줄을 위 아래로 이동

## \* Ctrl

- > Ctrl + 1 : Quick Fix
  - >> 구현하지 않은 메소드 추가
  - >> 로컬 변수 이름 바꾸기
  - >> Assignment 입력
  - >> 행 둘러싸기 (if/where/for이나 블록으로 둘러 싸려면 해당영역을 선택하고 Ctrl + 1)
- > Ctrl + D : 한줄 삭제
- > Ctrl + E : 뷰 화면의 탭에 열린 페이지 이동
- > Ctrl + L : 라인 이동
- > Ctrl + I : 자동 들여쓰기 수정
- > Ctrl + K : 문자열 찾기 (찾고자 하는 문자열을 블록으로 설정한 후...)
  - >> Ctrl + Shift + K : 역순으로 찾기
- > Ctrl + M : 전체화면
- > Ctrl + O : 현재 보고있는 파일의 아웃라인 (메소드 리스트 확인, 메소드나 필드 이동가능)
- > Ctrl + W : 창 닫기

# ECLIPSE 단축키

- > Ctrl + , or. : 다음 Annotation(Error, Warning, Bookmark)으로 이동
- > Ctrl + / : 주석 처리 (여러줄 블록 처리 가능)
- > Ctrl + Alt + ↑↓(상/하) : 한줄(블록) 복사
- > Ctrl + Shift + E : Switch to Editor (탭에 열려있는 Editor 이동)
- > Ctrl + Shift + G : 클래스의 메소드나 필드를 Reference하고 있는 곳으로 이동
  - >> 반대 : F3 (Reference하는 클래스로 이동)
- > Ctrl + Shift + L : 단축키 보기
  - >> Ctrl + Shift + L + L : 단축키 지정
- > Ctrl + Shift + W : 열린 파일 모두 닫기
- > Ctrl + Shift + O : 자동 import 처리 (사용하지 않는 Class는 삭제)
- > Ctrl + Shift + R : Open Resource
- > Ctrl + Shift + ↑↓(상/하) : 다음/이전 메소드로 이동
- > Ctrl + Shift + / : JavaDoc주석 추가
- > Ctrl + Shift + Space : 메소드 파라미터 힌트 (메소드에 입력해야 하는 파라미터 정보가 표시)
- > Ctrl + F3 : 클래스 아웃라인
- > Ctrl + F6 : View 화면의 탭에 열린 페이지 이동
- > Ctrl + F7 : View간 화면 전환
- > Ctrl + F8 : Perspective간 화면 전환
- > Ctrl + F11 : 바로 전에 실행했던 클래스 실행
- > Ctrl + PageDown : 뷰 화면의 탭에 열린 페이지 이동

# ECLIPSE 단축키

## \* Function Key

- > F2 : 에러의 원인에 대한 힌트 (에러 라인에 커서를 위치시키고...)
- > F3 : Java 편집기에서 Reference하는 클래스의 자바파일로 이동
  - >> Ctrl + 클릭
  - >> 반대 : Ctrl + Shift + G (클래스의 메소드나 필드를 Reference하고 있는 곳으로 이동)
- > F4 : 해당 클래스의 Hierarchy
- > F12 : Editor로 포커스

## \* E.T.C.

- > CTRL + 휠 : 페이지 단위 이동