



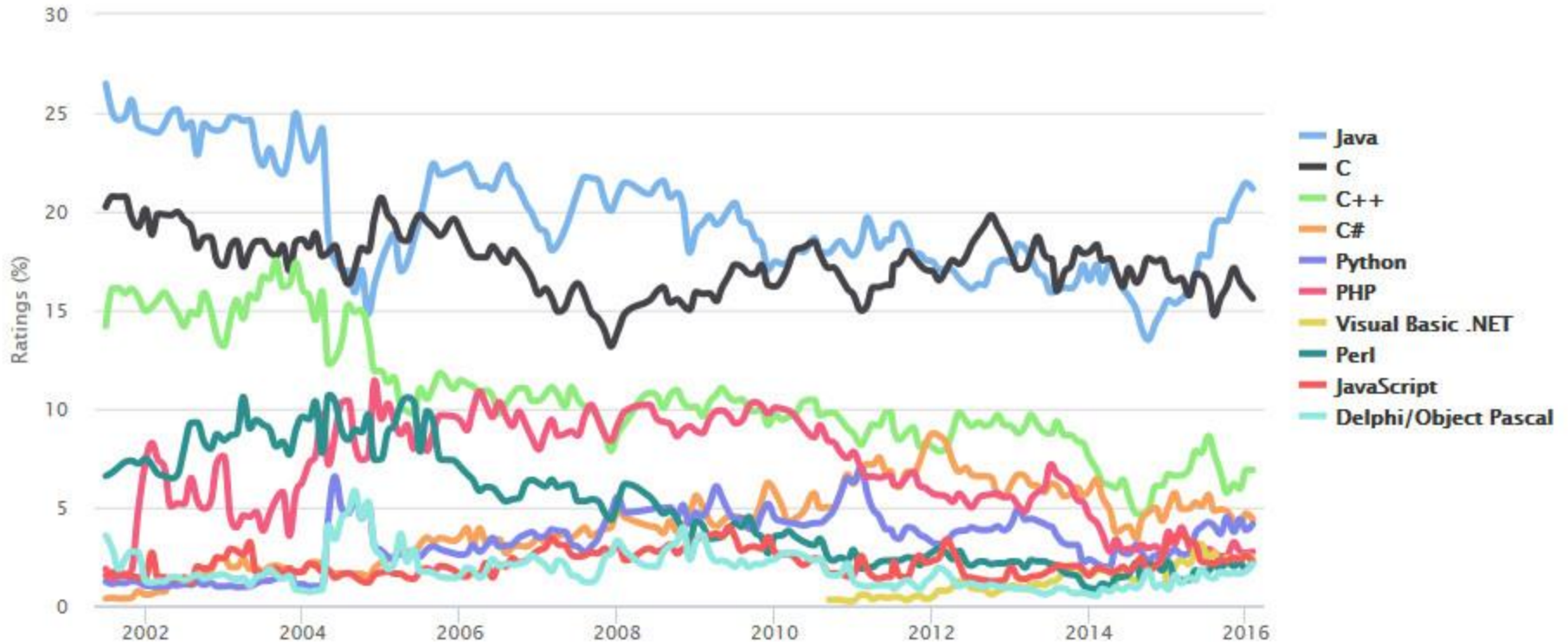
제 1 장

프로그래밍언어와 자바

가장 많이 사용되는 언어는?

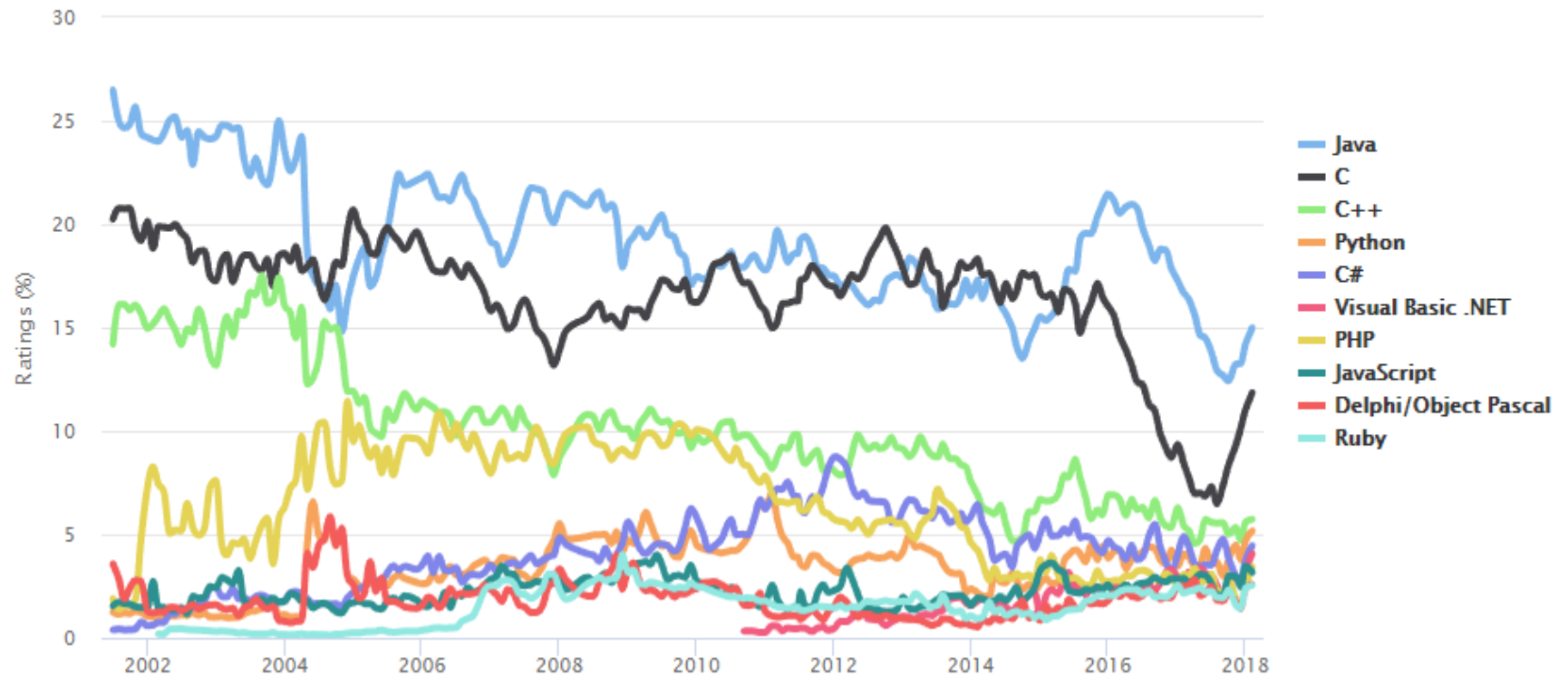
TIOBE Programming Community Index

Source: www.tiobe.com



TIOBE Programming Community Index

Source: www.tiobe.com



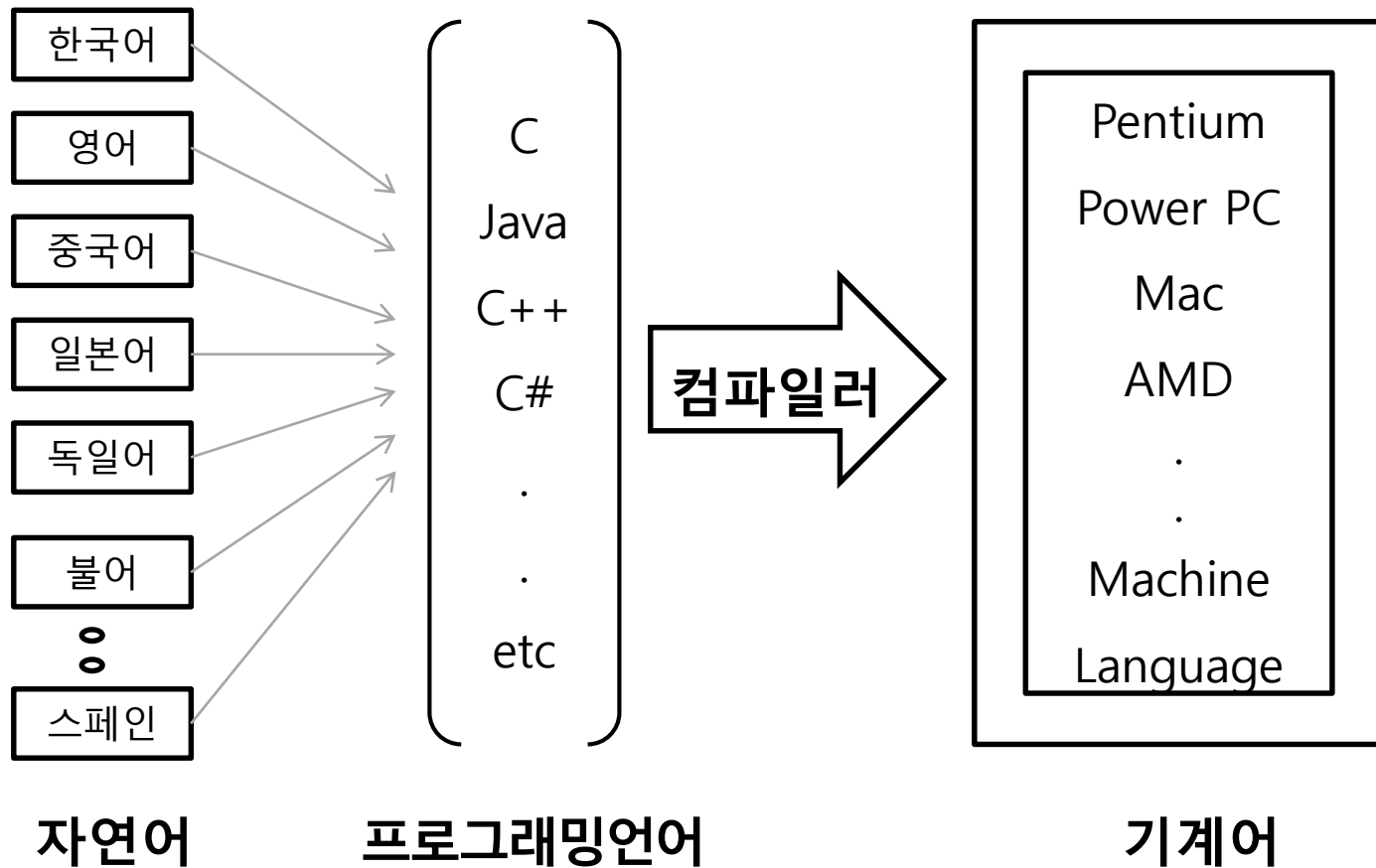
Feb 2016	Feb 2015	Change	Programming Language	Ratings	Change
1	2	⬆	Java	21.145%	+5.80%
2	1	⬇	C	15.594%	-0.89%
3	3		C++	6.907%	+0.29%
4	5	⬆	C#	4.400%	-1.34%
5	8	⬆	Python	4.180%	+1.30%
6	7	⬆	PHP	2.770%	-0.40%
7	9	⬆	Visual Basic .NET	2.454%	+0.43%
8	12	⬆	Perl	2.251%	+0.86%
9	6	⬇	JavaScript	2.201%	-1.31%
10	11	⬆	Delphi/Object Pascal	2.163%	+0.59%
11	20	⬆	Ruby	2.053%	+1.18%
12	10	⬇	Visual Basic	1.855%	+0.14%
13	26	⬆	Assembly language	1.828%	+1.08%
14	4	⬇	Objective-C	1.403%	-4.62%
15	30	⬆	D	1.391%	+0.77%
16	27	⬆	Swift	1.375%	+0.65%
17	18	⬆	R	1.192%	+0.23%
18	17	⬇	MATLAB	1.091%	+0.06%

Feb 2018	Feb 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.988%	-1.69%
2	2		C	11.857%	+3.41%
3	3		C++	5.726%	+0.30%
4	5	⬆	Python	5.168%	+1.12%
5	4	⬇	C#	4.453%	-0.45%
6	8	⬆	Visual Basic .NET	4.072%	+1.25%
7	6	⬇	PHP	3.420%	+0.35%
8	7	⬇	JavaScript	3.165%	+0.29%
9	9		Delphi/Object Pascal	2.589%	+0.11%
10	11	⬆	Ruby	2.534%	+0.38%
11	-	⬆	SQL	2.356%	+2.36%
12	16	⬆	Visual Basic	2.177%	+0.30%
13	15	⬆	R	2.086%	+0.16%
14	18	⬆	PL/SQL	1.877%	+0.33%
15	13	⬇	Assembly language	1.833%	-0.27%
16	12	⬇	Swift	1.794%	-0.33%
17	10	⬇	Perl	1.759%	-0.41%
18	14	⬇	Go	1.417%	-0.69%
19	17	⬇	MATLAB	1.228%	-0.49%
20	19	⬇	Objective-C	1.130%	-0.41%

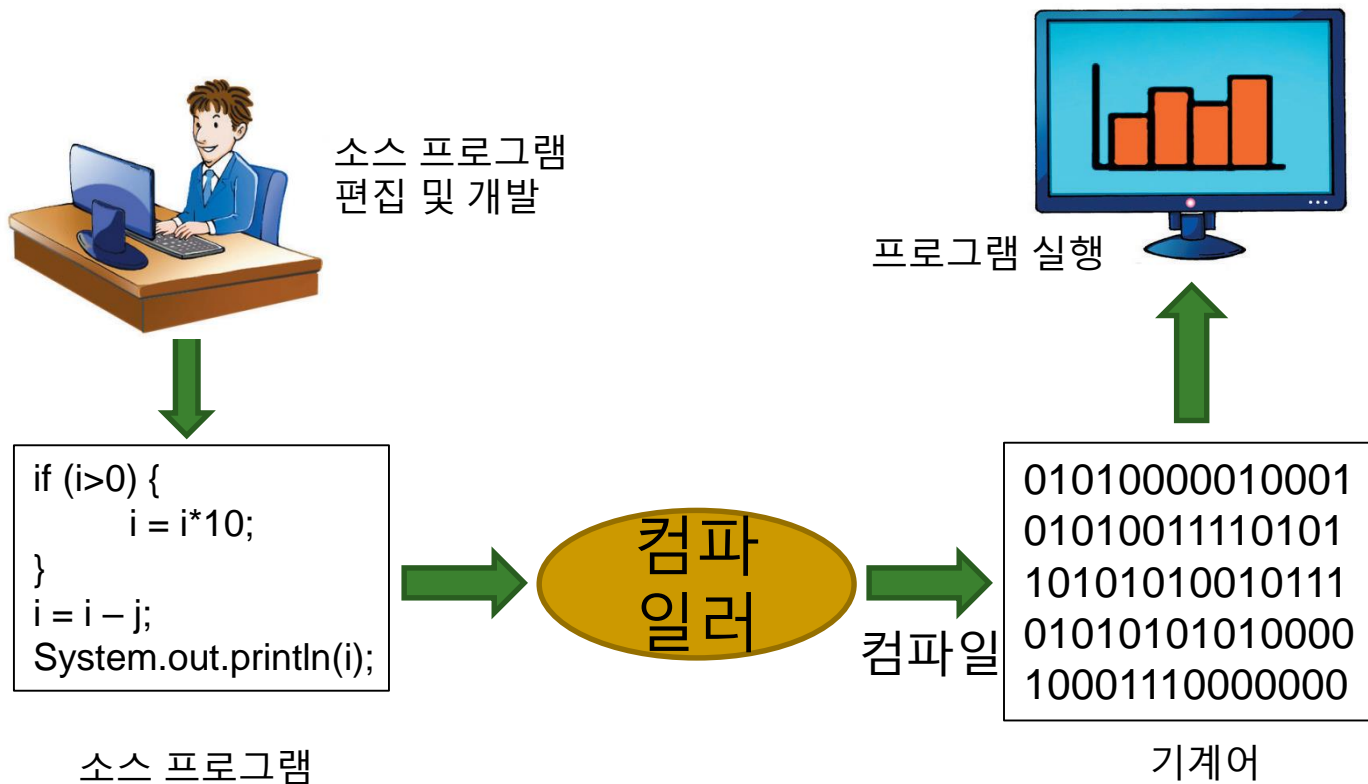
자연어와 프로그래밍언어

- 자연어와 기계어(natural language & machine language)
 - 자연어 : 사람들 간 서로 소통을 위해 사용하는 언어
 - 기계어 : 컴퓨터 간 서로 소통을 위해 사용하는 언어
 - 0, 1의 2진수로 구성 → 사람이 이해하기에는 무척 어려움
 - 컴퓨터는 기계어를 이해하고 처리함

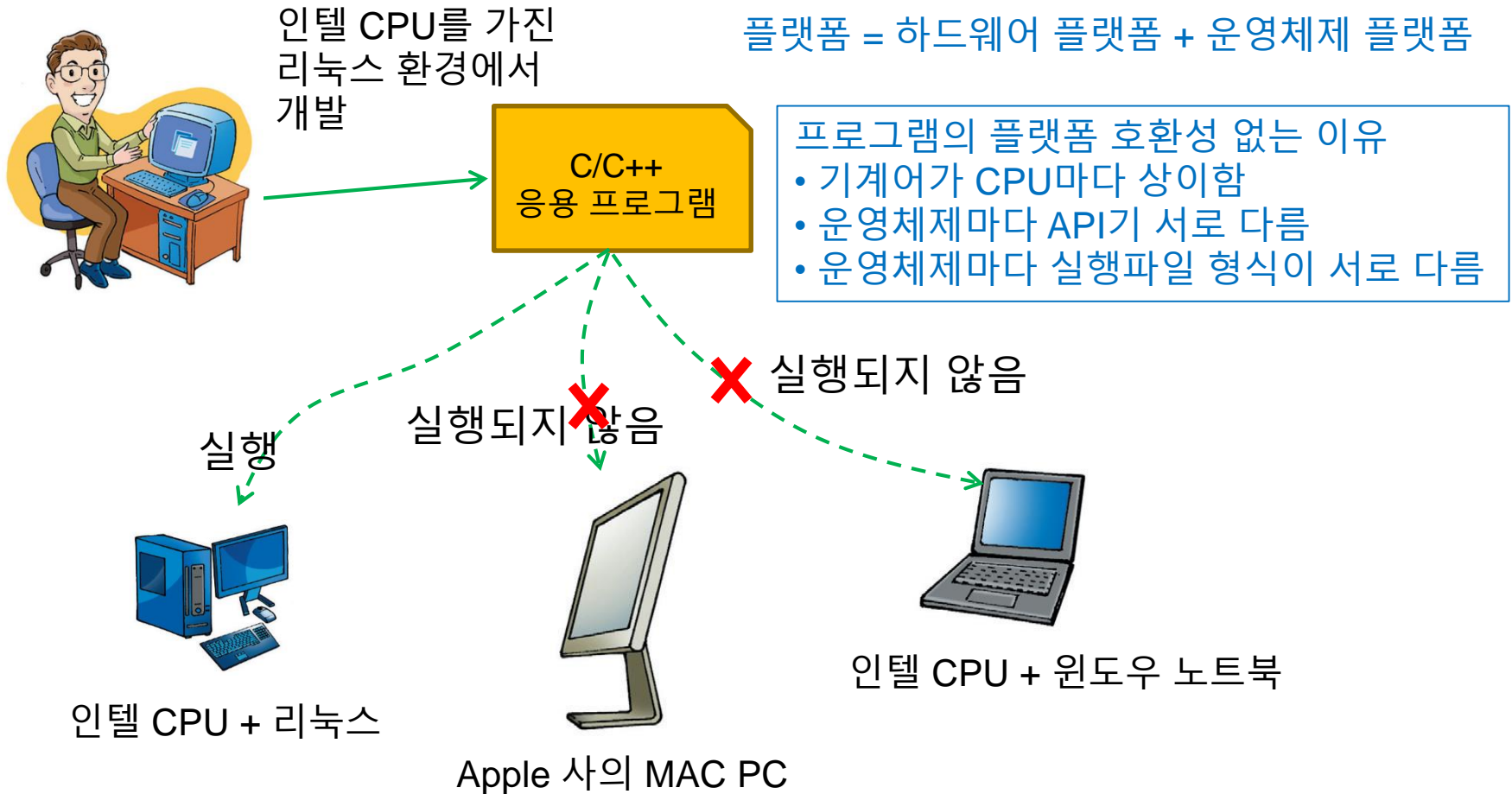
■ 프로그래밍 언어(programming language)



프로그램 편집, 컴파일 및 실행



플랫폼 의존성



- 고급언어와 저급언어(high-level language & low-level language)
 - 고급언어 – 사람이 이해하기 쉬우며 알고리즘 혹은 프로그램 작성을 위해 사용하는 언어
 - 다른 컴퓨터에서의 수행이 용이함 : 소스레벨 이식성이 높음
 - 컴파일 작업은 필요함
 - 어셈블리어
 - 기계어의 명령을 ADD, SUB, MOVE 등과 같은 표현하기 쉬운 상징적인 단어인 니모닉 기호(mnemonic symbol)로 일대일 대응시킨 언어
 - 저급언어
 - 사람이 이해하기 무척 어려우며 컴퓨터가 이해하는 언어
 - 이식성이 없음, 즉 다른 컴퓨터에서는 실행 불가능

자바 소개

- 썬 마이크로시스템즈의 제임스 고슬링(James Gosling)
 - 1991년 그린 프로젝트(Green Project) : 가전 제품에 들어갈 소프트웨어 개발을 목적 → 1995년 자바 발표



- 개발방향
 - 플랫폼 호환성 문제 해결
 - 기존 프로그래밍 언어의 플랫폼 호환성 결여
 - 소스를 재컴파일하거나 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 확산 됨
 - Netscape 웹 브라우저에서 실행
- 2009년에 썬 마이크로시스템스를 오라클에서 인수

■ WORA(Write Once Run Anywhere)

- **기계어 레벨에서의 이식성** 추구 → 한 번 만 작성하면 다른 컴퓨터에서 실행가능
- 네트워크에 연결된 어느 클라이언트에서도 실행 가능
 - 웹 브라우저, 분산 환경에서 활용성이 높아짐

Java 소스 프로그램

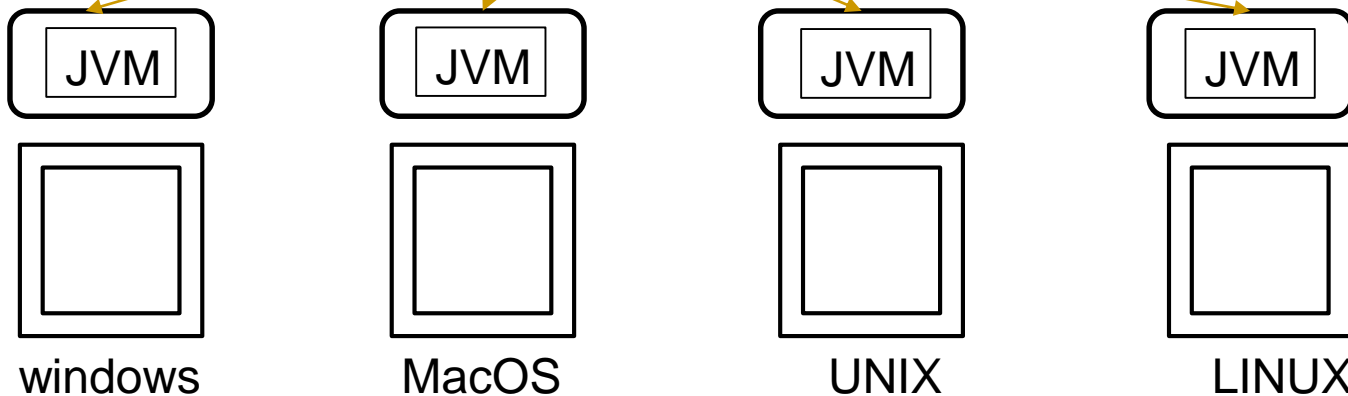
```
public class HelloWorld {  
    private int year = 2020;  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
        System.out.println("This year is " + year);  
    }  
}
```

Write-Once



HelloWorld.class

Run Anywhere

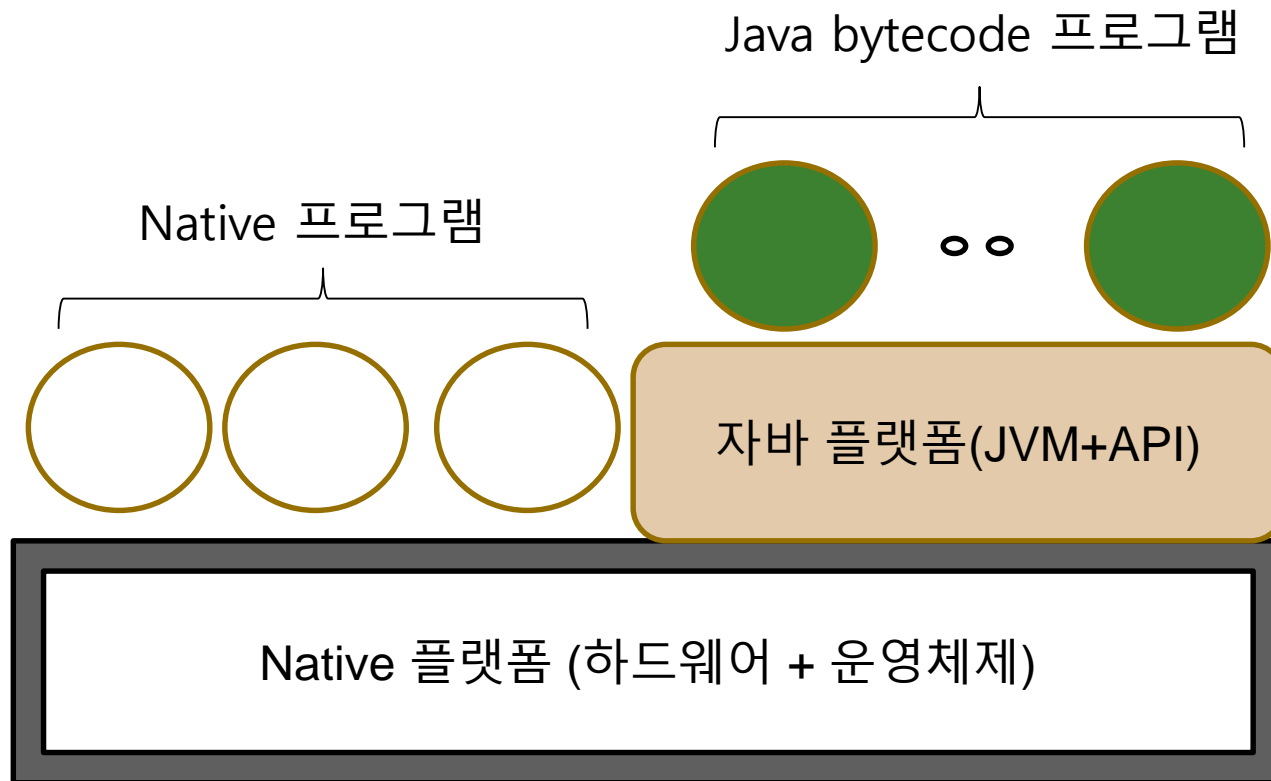


■ 자바 가상기계(Java Virtual Machine)

- 가상 머신으로서 각기 다른 플랫폼에서 동일한 자바실행환경 제공
- 자바 가상 머신 자체는 플랫폼에 종속적
 - 자바 가상 머신은 각 플랫폼에 맞도록 작성된다.
 - 리눅스에서 작동하는 자바 가상머신은 윈도우즈에서 작동하지 않는다.
- 자바 가상머신 개발, 공급
 - 자바 개발사인 오라클 이외 IBM, MS 등 다양한 회사에서 제작 공급

■ 바이트 코드

- 플랫폼 종속성이 없이 자바 가상 기계의 기계어로 JVM 상에서만 동작하는 실행코드
- 클래스 파일(.class)에 저장
- 바이트 코드는 native 컴퓨터가 아닌 JVM에 의해 해석되면서 수행된다



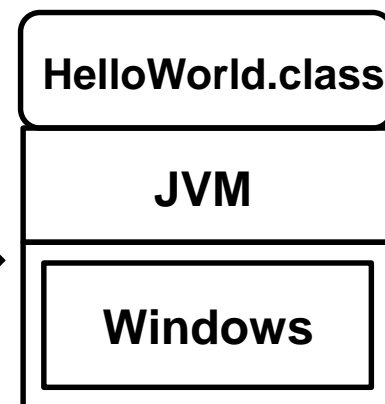
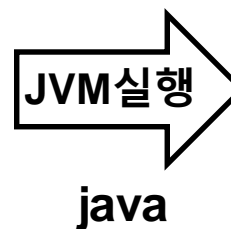
Java 프로그램 실행단계

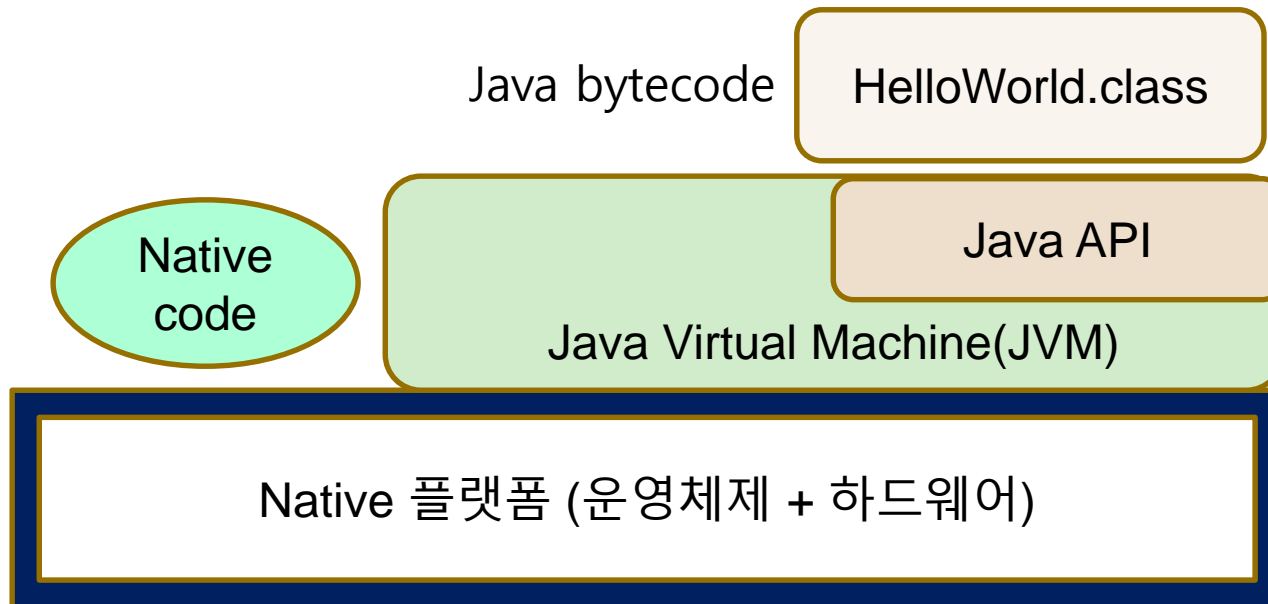
Java 소스 프로그램: HelloWorld.java

```
public class HelloWorld {  
    private int year = 2020;  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
        System.out.println("This year is " + year);  
    }  
}
```



Java 기계어프로그램(Java bytecode)
HelloWorld.class





자바 가상 기계와 프로그램의 실행



자바 프로그래밍

Draw.java
Hello.java
Shape.java

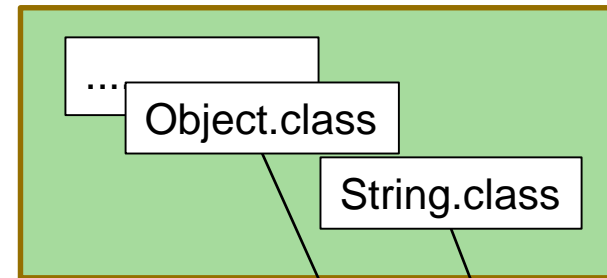
(소스 코드)



Draw.class
Hello.class
Shape.class

(바이트 코드)

실행에 필요한 자바 클래스 라이브러리(JDK APIs)

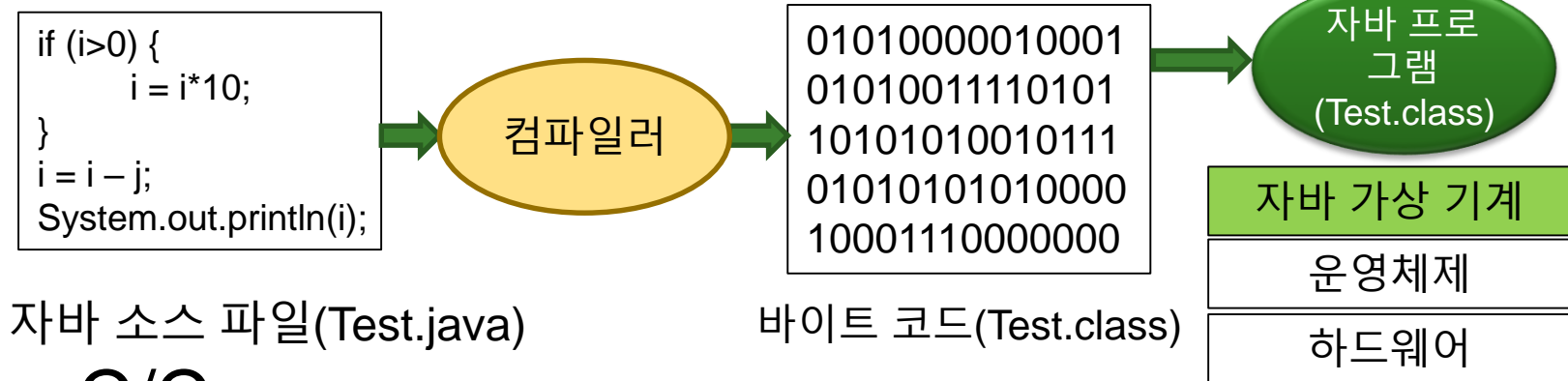


클래스 로딩

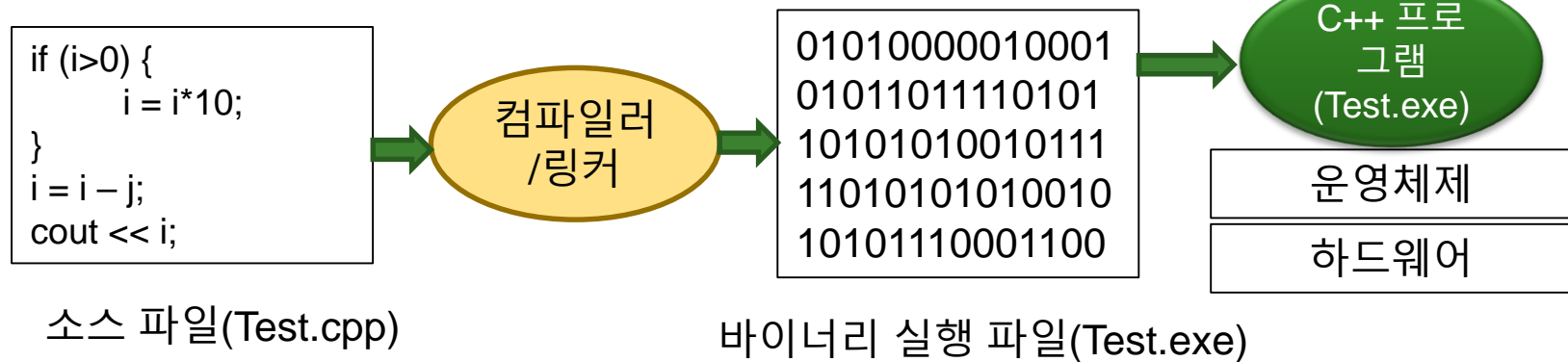


자바와 C/C++의 실행 환경 차이

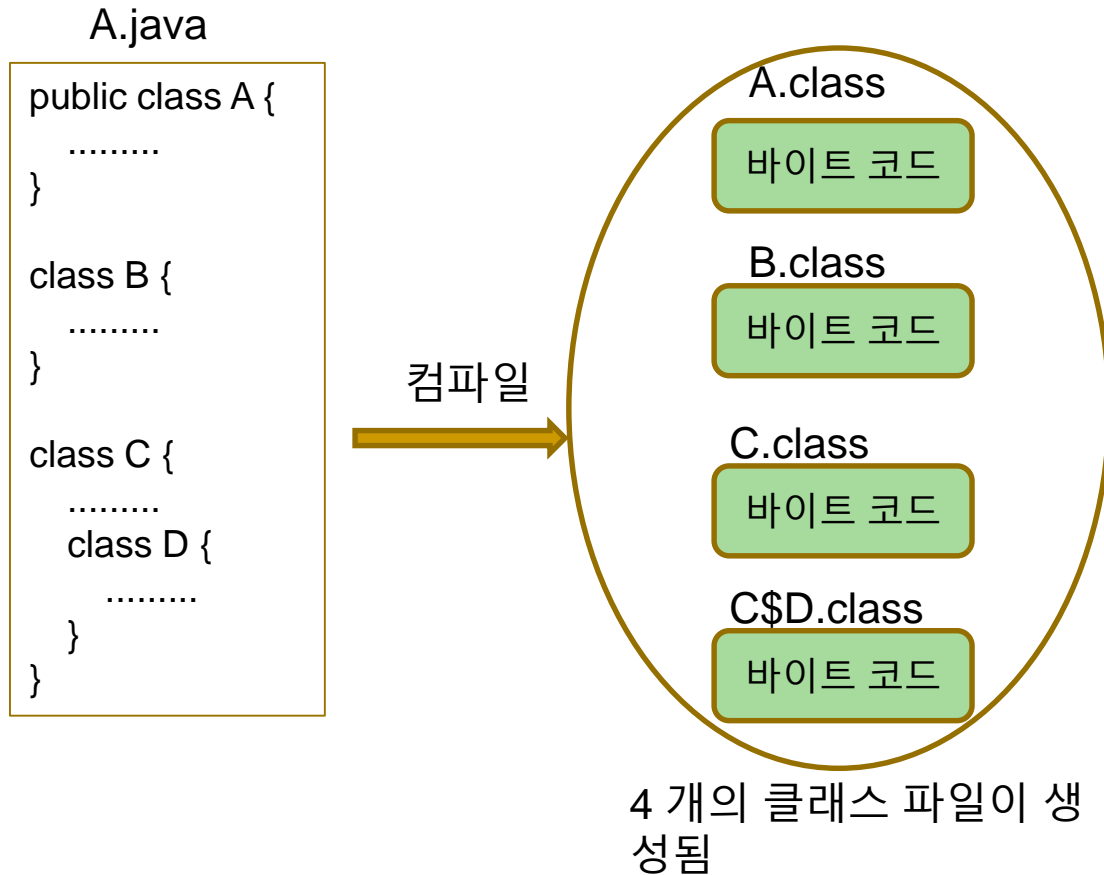
■ 자바



■ C/C++



소스 파일과 클래스, 클래스 파일의 관계



자바와 C/C++ 실행 환경 및 과정

■ 자바

- 자바는 링크 과정이 없이 컴파일러가 바로 바이트 코드 생성
- 바이트 코드는 JVM으로만 실행 가능
- 자바는 런타임에 필요한 클래스들이 JVM에 링크되며 **클래스 로더가 동적으로 필요한 클래스를 로딩**한다.

■ C/C++

- C/C++에서는 컴파일러가 중간 단계인 오브젝트 코드를 생성한 후 링커가 필요한 라이브러리들을 링크하여 최종 실행 가능한 실행 파일을 생성
- 라이브러리의 형태가 정적 라이브러리의 경우는 라이브러리를 실행 파일에 포함시키므로 실행 파일 크기가 커짐.
- 동적 라이브러리의 경우는 런타임에 링크가 일어난다.
- 오브젝트 코드 및 실행 파일은 플랫폼에 따라 다르므로 플랫폼이 바뀌면 컴파일 및 링크를 새로 하여야 한다.

자바의 특성

■ 객체지향

- 객체지향의 특징인 클래스 계층 구조, 상속성, 다형성, 캡슐화 등 지원

■ 멀티스레드

- 하나의 프로그램에서 다수의 스레드의 동시에 수행 환경 지원
- 자바는 운영체제의 도움 없이 자체적으로 멀티스레드 지원
 - C/C++ 등에서는 멀티스레드를 위해 운영체제 API를 호출

■ 플랫폼 독립성

- 하드웨어, 운영체제 등에 독립적인 바이트 코드를 자바 가상 기계로 실행시킴으로써 플랫폼에 종속성을 갖지 않음

■ 소스(.java)와 클래스(.class) 파일

- 클래스 파일에는 단 하나 만의 클래스만 존재.
- 하나의 소스 파일에 여러 클래스를 작성 가능
 - 하나의 public 클래스만 가능
- 다수의 클래스가 작성된 자바 소스를 컴파일하면 각각 별도의 클래스 파일 생성
- 소스 파일의 이름과 public으로 선언된 클래스 이름은 같아야 함

■ 자바 실행 모듈

- 한 개의 class 파일 또는 다수의 class 파일로 구성
- 자바 응용프로그램의 실행은 main() 메소드에서 시작
- 하나의 클래스 파일에 하나 이상의 main() 메소드가 있을 수 없음

■ 클래스 파일의 동적 로딩(dynamic loading)

■ 클래스로 캡슐화

- 자바의 모든 변수나 함수는 클래스 내에서 정의 - 캡슐화
- 클래스 안에서 새로운 클래스 정의 가능 - 내부 클래스

■ 패키지

- 관련된 클래스는 패키지로 묶어 관리
- 패키지는 폴더 개념
 - 예) java..lang.System은 java\lang 디렉터리 밑의 System.class를 나타냄

■ 이식성이 뛰어남 (기계어 레벨에서의 이식성)

- **번역 언어 이다** (Interpreted)
 - 자바는 자바 가상 기계에 의해 수행되는 번역 언어
- **분산처리 언어이다** (Distributed)
 - 자바는 인터넷에 필요한 주요 프로토콜을 지원
 - TCP/IP, HTTP
 - 분산처리를 지원하는 다양한 클래스를 제공
 - RMI(Remote Method Invocation) 기법
- **간편하다.**
 - 배우기 쉽고 간편하다 (기존의 C, C++에 비해)
 - 자바가 제공하지 않는 C++ 언어의 요소
 - 포인터 연산과 구조체 그리고 TypeDef
 - 전처리기(Preprocessor)
 - 메모리 할당 후 회수 불필요(가비지 컬렉션)

자바 버전



Java 1.0

- 1996년
- 211개의 클래스
- 속도는 느림
- 애플릿이 가장 주목받음



Java 1.1-1.4

- 1997-2004년
- 2000여개의 클래스
- 3가지 버전 존재 (ME, SE, EE)
- 웹과 모바일 기반의 엔터프라이즈 프로그래밍 언어로서 부각



Java 1.5-1.6

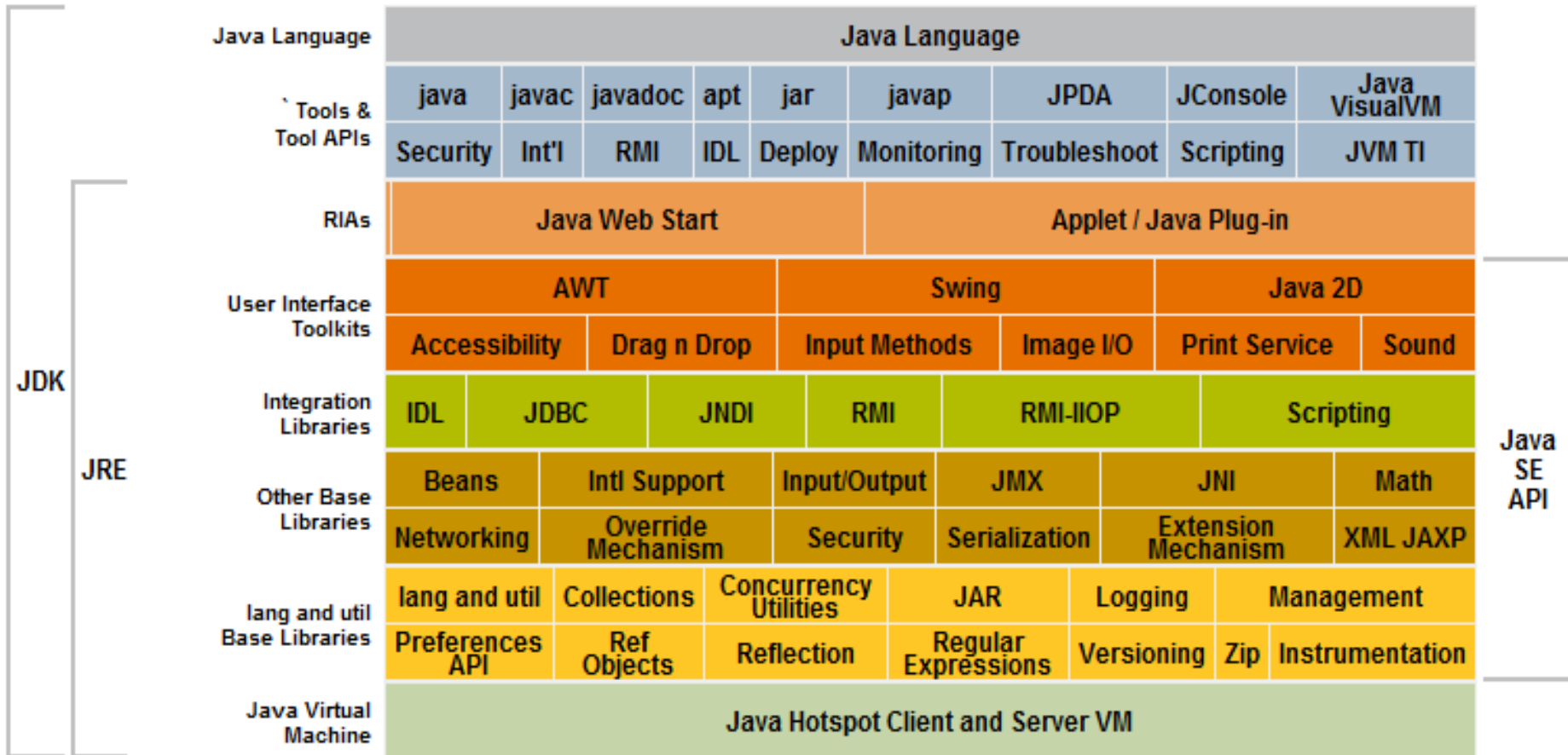
- 2004-2006년
- 3000여개의 클래스
- 제네릭 클래스, “for each” 반복 루프, 가변 인수, 오토 박싱, 메타 데이터, 열거형, 정적 import



Java 1.7-1.8

- 2011-2014년
- 4000개 이상의 클래스
- 람다 표현식 (Lambda expressions), 새로운 날짜, 시간 API (Date & Time API), 강화된 비밀번호 기반 암호화

Java 가상기계(SE) 구성



출처: <http://download.oracle.com/javase/6/docs/>

JDK와 JRE

- JDK란?
 - Java Development Kit의 약자
 - 자바 응용 개발 환경으로, 개발에 필요한 도구 포함
 - 컴파일러, JRE (Java Runtime Environment), 클래스 라이브러리, 샘플 등
- JRE
 - 자바 실행 환경으로 JVM이 포함되어 있음
 - 자바 실행 환경만 필요한 경우 JRE만 따로 다운 가능
- JDK와 JRE의 개발 및 배포
 - 오라클의 Technology Network의 자바 사이트에서 다운로드
- JDK의 bin 디렉토리에 포함된 주요 개발 도구
 - Javac : 자바 소스를 바이트 코드로 변환하는 컴파일러
 - Java : jre의 bin 디렉토리에 있는 자바 응용프로그램 실행기
 - Jar : 자바 아카이브 파일 (JAR)의 생성 및 관리하는 유틸리티
 - Jdb : 자바 디버거
 - Appletviewer : 웹 브라우저 없이 애플릿을 실행 및 디버깅하는 유틸리티

자바 API

- 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 개발자는 클래스 사용
- 자바 패키지(package)
 - 서로 관련된 클래스들을 분류하여 묶어 놓은 것
 - 필요한 클래스가 속한 패키지만 import하여 사용
 - 계층구조로 되어 있음
 - 클래스의 이름에 패키지 이름도 포함
 - 다른 패키지에 동일한 이름의 클래스 존재 가능
 - 자바 API(클래스 라이브러리)는 JDK내에 패키지 형태로 제공
 - 개발자가 자신의 패키지 생성 가능

자바 프로그램 개발

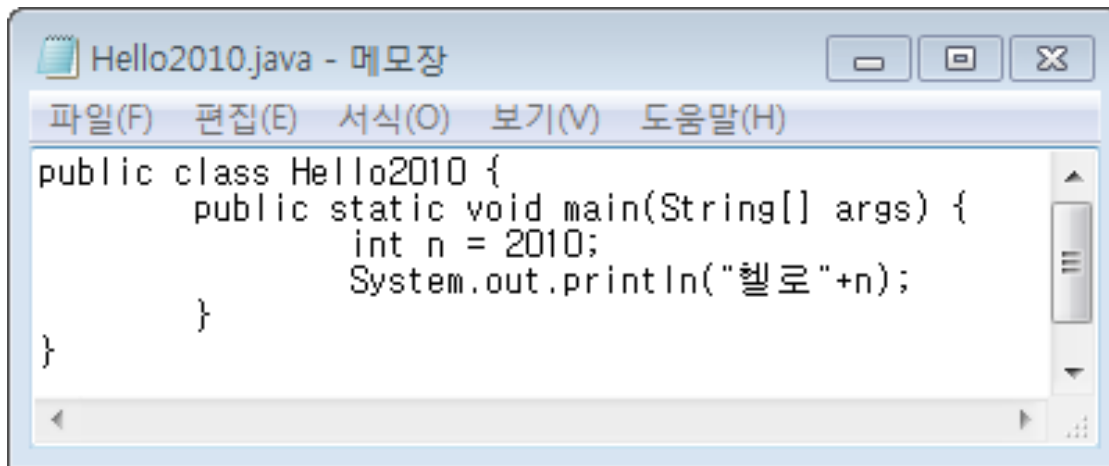
샘플: Hello2020.java

- `public class Hello2020`
 - 클래스의 선언
 - Hello2010 은 클래스 이름
 - 클래스는 {와 } 사이에 정의
 - 자바는 하나 이상의 클래스로 구성
- `public static void main(String[] args)`
 - 자바 프로그램은 main 메소드에서 실행 시작
 - 실행을 시작하는 클래스에 main() 메소드가 반드시 하나만 존재
- `int n = 2020;`
 - 지역 변수 선언
- `System.out.println("헬로"+n);`
 - 화면에 "헬로2010"를 출력하는 실행문
 - JDK에서 제공하는 `java.lang.System.out` 객체 이용

```
public class Hello2010 {  
    public static void main(String[] args) {  
        int n = 2020;  
        System.out.println("헬로" + n);  
    }  
}
```

자바 소스 편집

- 어떤 편집기를 사용해도 무관
 - 보조 프로그램의 메모장 이용한 샘플



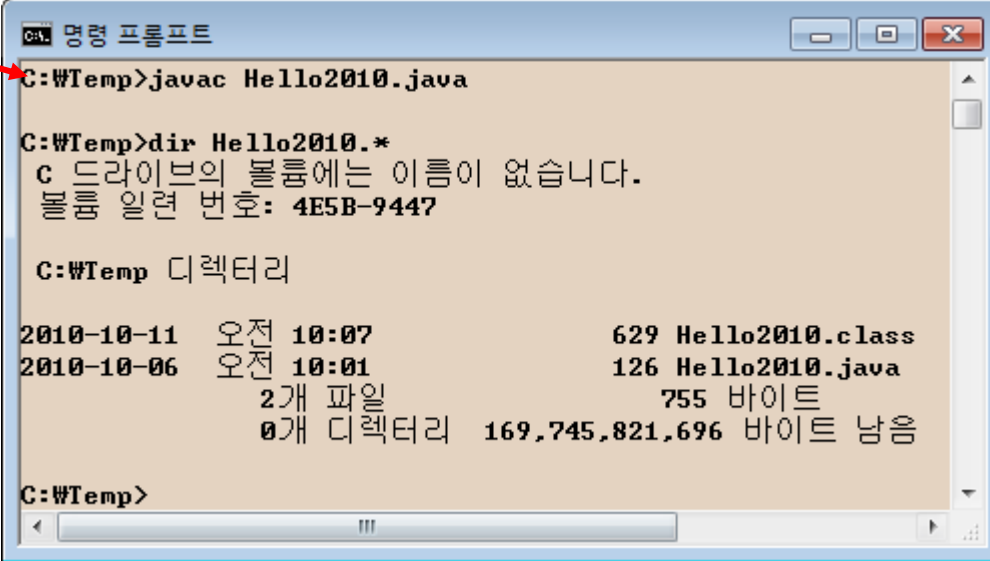
A screenshot of a Notepad window titled "Hello2010.java - 메모장". The window has a menu bar with "파일(F)", "편집(E)", "서식(O)", "보기(V)", and "도움말(H)". The text area contains the following Java code:

```
public class Hello2010 {  
    public static void main(String[] args) {  
        int n = 2010;  
        System.out.println("헬로"+n);  
    }  
}
```

- 작성 후 임의의 디렉토리에 Hello2010.java로 저장
 - 클래스의 이름과 동일한 파일 이름으로 저장
 - 확장자는 .java

소스 컴파일 및 실행

컴파일



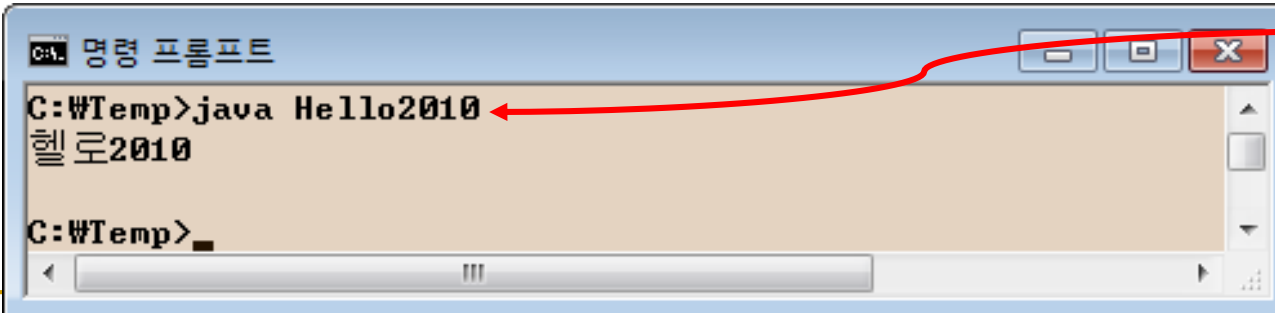
```
C:\WTemp>javac Hello2010.java

C:\WTemp>dir Hello2010.*
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 4E5B-9447

C:\WTemp 디렉터리

2010-10-11 오전 10:07                629 Hello2010.class
2010-10-06 오전 10:01                126 Hello2010.java
                2개 파일                755 바이트
                0개 디렉터리 169,745,821,696 바이트 남음

C:\WTemp>
```



```
C:\WTemp>java Hello2010
헬로2010

C:\WTemp>
```

실행 시 class 확장자(.class)를 붙이지 않는다.

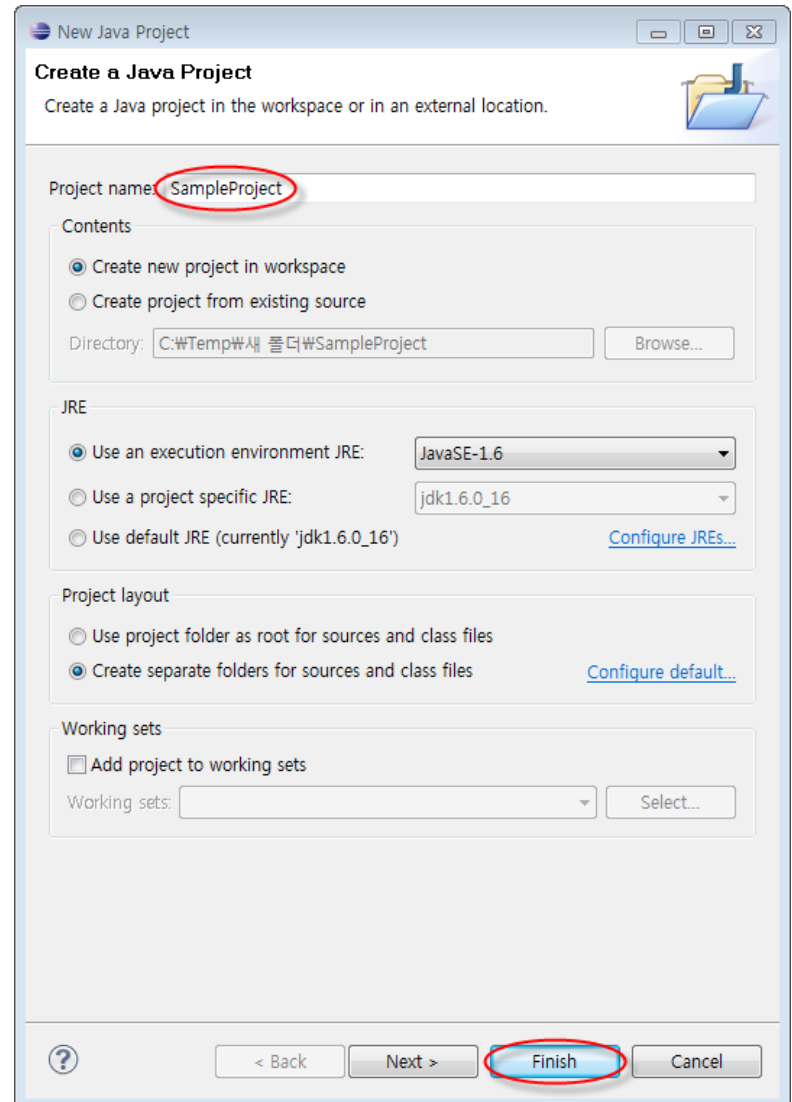
자바 통합 개발 환경(자바 IDE) - Eclipse

- IDE란?
 - Integrated Development Environment의 약자
 - 통합 개발 환경
 - 편집, 컴파일, 디버깅을 한번에 할 수 있는 통합된 개발 환경
- Eclipse
 - 자바 응용 프로그램 개발을 위한 통합 개발 환경
 - IBM에 의해 개발된 오픈 소스 프로젝트
 - <http://www.eclipse.org/downloads/> 에서 다운로드

이클립스를 이용한 자바 프로그램 개발

■ 프로젝트 생성

- File->New->Java Project를 선택
- Project name란에 프로젝트 이름을 입력
- Finish 버튼을 눌러 프로젝트 생성



■ 클래스 생성

- File->New->Class를 선택
- 클래스 이름은 Hello2010을 입력
- Finish 버튼을 눌러 클래스 생성

New Java Class

Java Class

⚠ The use of the default package is discouraged.

Source folder: SampleProject/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: **Hello2010**

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

Superclass: java.lang.Object Browse...

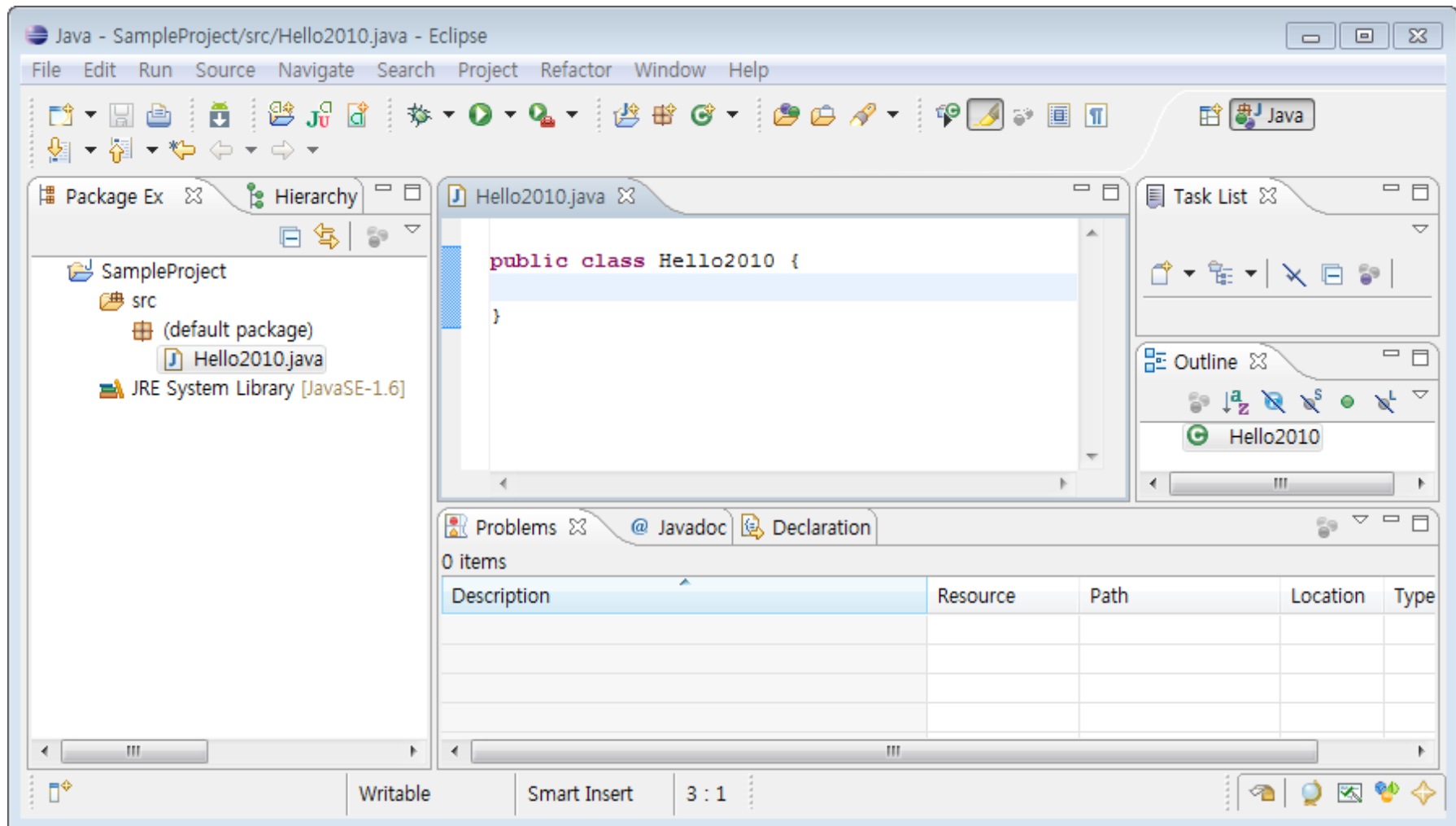
Interfaces: Add...
Remove

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** Cancel

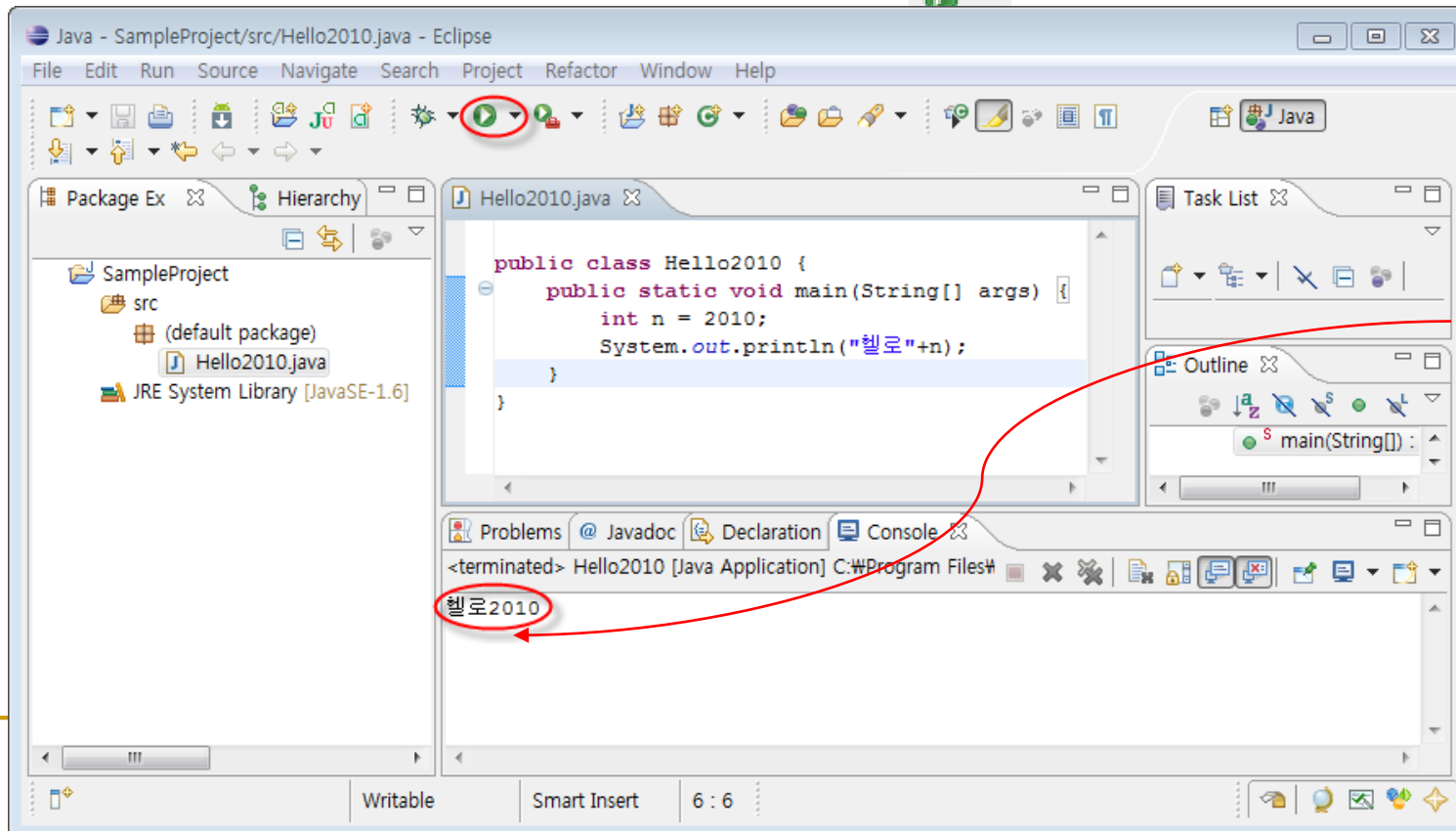
생성된 클래스와 소스



소스 편집과 컴파일 및 실행

■ 자바 응용프로그램의 실행

- 이클립스의 소스 편집 윈도우에 앞서 메모장을 이용하여 소스를 작성한 것과 동일하게 소스 작성
- Run->Run 메뉴를 선택 또는



콘솔 출력 메시지

자바와 오픈 소스

- 오픈 소스란?
 - 소프트웨어 제작자의 권리를 보존하며 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 소프트웨어**에 사용할 경우 저작권 침해가 발생
 - 다양한 개량 버전의 소프트웨어로 인한 호환성 문제
- 오픈 소스 소프트웨어 사례
 - Linux, OpenOffice, Open Solaris, Mozilla, Apache, GNU, WebKit 등
 - 2006년 11월, 썬 마이크로시스템스는 대부분의 자바를 GPL 라이선스로 소스 오픈
 - <http://sourceforge.net> : 오픈 소스를 위한 사이트

자바의 배포판 종류

- 오라클은 개발 환경에 따라 3 개의 배포판 제공
- **Java SE**
 - 자바 표준 배포판
 - 데스크탑과 서버 응용 개발 플랫폼
 - 임베디드 및 실시간 환경 지원
- **Java ME**
 - 자바 마이크로 배포판
 - 휴대 전화나 PDA, 셋톱박스 등제한된 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 닷넷(.Net)

■ 자바의 경쟁자 닷넷

- “Write Once, Run Anywhere”

■ CIL과 Bytecode

- 닷넷이 작동되기 위한 환경인 닷넷 프레임워크(.Net Framework)는 자바 플랫폼의 자바 운영 환경(JRE)이다.
- 닷넷의 공통 중개 언어(Common Intermediate Language)는 자바의 바이트코드와 대응

■ CLR과 JVM

- CLR(Common Language Runtime)과 JVM(Java Virtual Machine)
- 각각 하나의 플랫폼에서 개발된 프로그램을 모든 플랫폼에서 실행할 수 있게 만드는 새로운 프로그래밍 패러다임을 가능하게 하는 공통된 새로운 기술

■ 자바 언어와 C# 언어

- 2000년에 소개된 C#은 자바와 C++를 결합한 닷넷의 주력 언어

■ 전망

- Windows 플랫폼과 Unix 플랫폼이 경쟁하면서 공존하듯이 자바와 닷넷도 함께 발전

자바 버전 JKD8

- 람다식(Lambda expressions)
- 작은 가상기계(VM)
- 병렬 배열 정렬(Parallel Array Sorting)
- 컬렉션을 위한 대용량 데이터 처리
- Base64 인코딩과 디코딩을 위한 표준 API
- 새로운 날짜, 시간 API(Date & Time API)
- 강화된 비밀번호기반 암호화>Password-Based-Encryption (PBE))