

경기과학기술대학교 인공지능학과 · 전공기초

자바 프로그래밍

1주차

객체지향의 문을 열고, 내 손으로 개발 환경을 세운다

자바를 배우기 전에,
자바가 왜 태어났는지부터 본다.

오늘의 강의

1~2 교시에 기초를 다지고, 3 교시에 실습 진행

1교시



왜 자바인가

수업 운영 안내

2교시



무엇으로 만드나

개발 도구와 절차
자바 활용과 특징

3교시



직접 만든다

이클립스 설치
첫 프로그램 실행

PART 01 · 수업 운영 안내

1교시 · 오리엔테이션

수업 방법

이론과 실습을 병행

- 매주 3시간, 개념 2교시 + 실습 1교시
- 목표는 객체지향 사고방식 체득
- JAVA 프로그래밍 지식 습득





교수 소개

Me.monday9.com

15주 로드맵



1~7주

문법 · 반복문
배열 · 클래스



8주

중간고사로
기초 점검



9~14주

상속 · 인터페이스
파일 입출력



15주

기말고사로
마무리

주교재와 참고서



강의 자료



주차별
강의 내용
PDF 다운로드
홈페이지



주교재

명품 자바 에센셜

황기태 지음 · 생능출판사

강의 순서와 예제는 이 주교재를
그대로 따라갑니다.

평가는 이렇게 100점

30%

중간고사

1~7주

30%

기말고사

9~14주

20%

출석

u-check 전자출결

20%

과제

4·7·12·14주차



출석 기준 · 예외 없음

결석 4회면

무조건 F입니다

출결 방법



헤이영

헤이영 스마트캠퍼스
앱을 스토어에서 받는다

출석 체크

수업 시작 시점에
직접 체크한다

출석

3교시 종료 후
퇴실 시 수기로 서명

오류 신고

오류가 나면 즉시
교수에게 말한다

앱과 수기 서명한 자료가 남지 않은 출석은(둘 중 하나만 있거나 둘다 없으면) 불출석입니다.
정정은 해당 주 안에만 받습니다.

매주 이렇게 굴러갑니다

1

개념 강의

1~2교시에
이론 지식 습득

50분 강의 10분 휴식

2

코드로 구현

3교시에
직접 코딩

오늘 준비할 것



실습실 PC

설치 여부만 확인하면 됩니다



개인 노트북

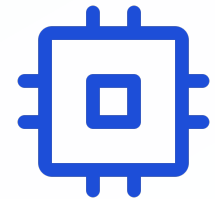
3교시에 JDK와 이클립스를 설치합니다



윈도우든 맥이든 상관없습니다

컴퓨터는 어떻게 일하나

프로그래밍은 기계에게
순서를 알려주는 일이다.



만능 요리 기계

재료와 순서를 넣으면 요리가 나온다

요리사 = 프로그래머

재료와 순서를 정해 레시피로 적는다

요리 기계 = 컴퓨터

레시피대로만 움직인다. 판단하지 않는다

기계가 똑똑한 게 아니라, 레시피가 똑똑한 것이다 · 레시피 = 소프트웨어



컴퓨터를 이루는 것



CPU

Central Processing Unit

명령을 실제로
처리하는 두뇌



메모리

프로그램과
데이터를 담아 둔다



입출력 장치

키보드로 넣고
화면으로 내보낸다



GPU

Graphics Processing Unit)

컴퓨터에서 이미지와 그래픽을
빠르게 그려내기 위해 만들어진 핵심 연산 장치입니다

같은 기계, 다른 변신



컴퓨터

책상 위 하드웨어는 하나



게임기



문서편집기



통계처리기



음악재생기

여기서 잠깐

<https://www.monday9.com/r/NCZA>

언어가 올라온 계단

사람 쪽



고급 언어

사람의 표현으로 올라온다

Pascal · C/C++ · C# · 자바



어셈블리어

명령을 기호로 바꾼다



기계어

0과 1로 CPU에 직접 명령

기계 쪽

어셈블리어의 한계

니모닉 기호

ADD SUB MOVE



기계어 명령

1011 0000

일대일 대응

명령 하나에 기계어 하나

CPU 종속

CPU가 바뀌면 다시 쓴다

저급 언어

기계에 가까운 언어다

절차 지향 vs 객체 지향

1≡
2≡

절차 지향

일의 순서를 중심으로 코드를 쓴다

대표 언어 · C



객체 지향

객체와 관계로 문제를 모델링한다

대표 언어 · 자바 · C++ · C#

자바는 객체 지향 쪽에 서 있는 언어다

객체지향 프로그래밍 vs 절차지향 프로그래밍

자바 프로그래밍 수업용 핵심 비교

절차지향 프로그래밍

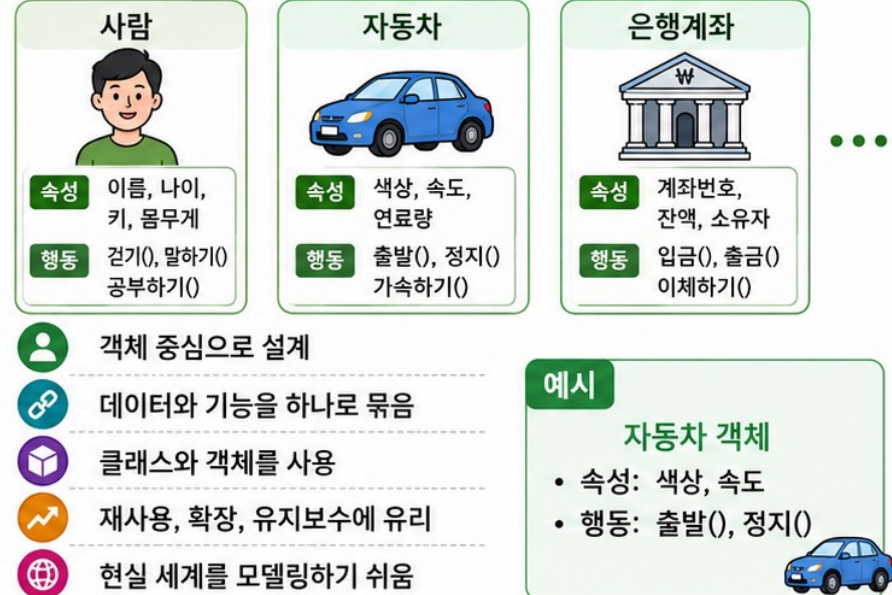


- 순서와 절차 중심
- 함수 단위로 작업 수행
- 데이터와 기능이 분리됨
- 규모가 커지면 유지보수가 어려울 수 있음

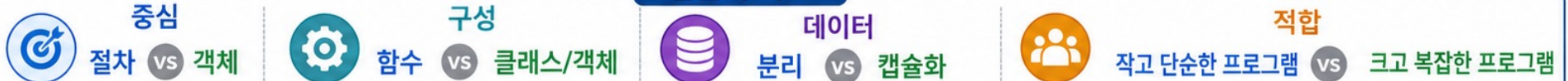
예시

- 입력 → 계산 → 출력
- 처리 흐름을 따라 실행

객체지향 프로그래밍



한눈에 비교



★ 자바는 대표적인 객체지향 언어입니다.

면접 질문: 객체지향형 언어란?

아주 단순하고 깔끔한 답변 예시

“ 객체지향형 언어는 데이터와 기능을 하나의 객체로 묶어 프로그램을 구성하는 언어입니다. ”

핵심 포인트



객체 중심 설계



속성과 기능을
함께 표현



재사용과
유지보수에 유리



현실 세계를
모델링하기 쉬움

쉬운 예시



자동차 객체

속성 : 색상, 속도

기능 : 출발(), 정지()

면접 답변 예시

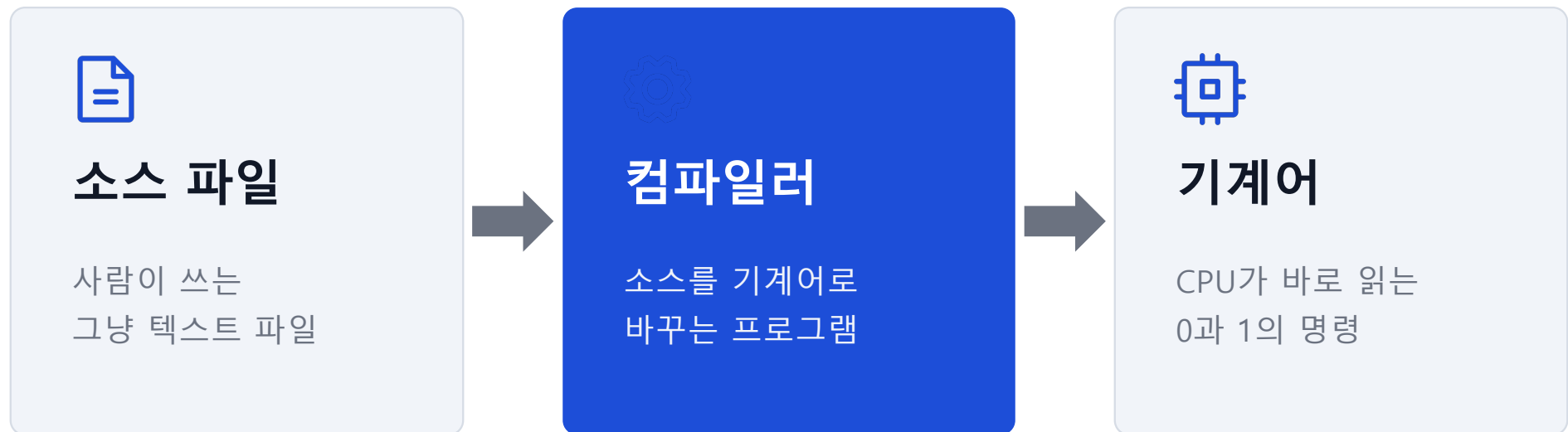


“ 객체지향형 언어는 데이터와 기능을 객체로 묶어 프로그램을 구성하는 언어입니다. 현실 세계의 사물을 객체처럼 표현할 수 있어서 구조화에 유리하고, 재사용성과 유지보수성이 좋습니다. 대표적으로 자바가 있습니다. ”



면접에서는 ‘객체’, ‘데이터와 기능의 묶음’, ‘재사용성’ 이 3가지를 말하면 충분합니다.

컴파일과 컴파일러



언어마다 전용 컴파일러가 따로 있다

확장자로 알아보는 언어

.asm

어셈블리어

.c

C 언어

.cpp

C++

.vb

비주얼 베이직



.java

사람이 쓴 자바 소스

컴파일



.class

컴파일하면 생기는 파일

자바 ≠ 자바스크립트



자바

- 어디에나 쓰는 범용 언어
- 컴파일 후 JVM이 실행



자바스크립트

- HTML 문서에 내장되는 코드
- 브라우저가 번역·실행

≠

이름만 닮았을 뿐, 태생도 실행 방식도 다릅니다



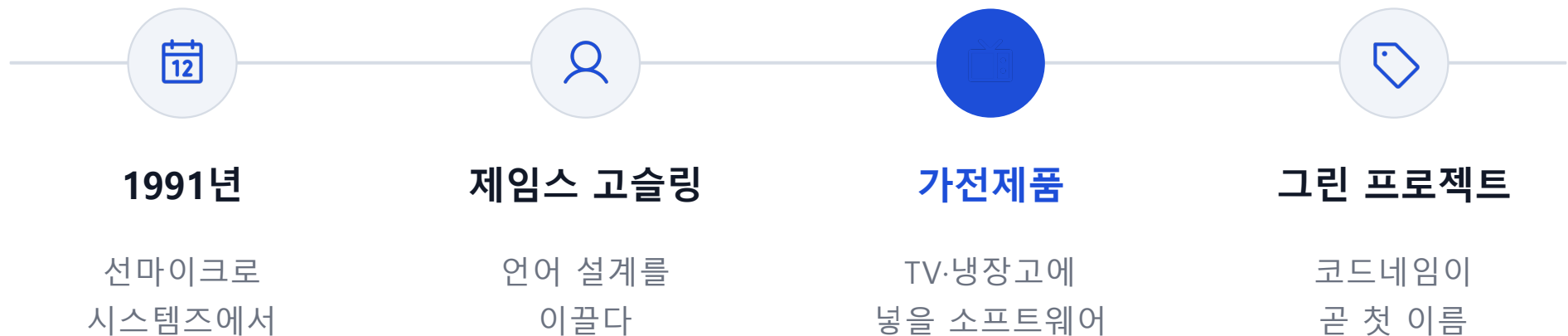
CHAPTER 3 · 자바와 WORA

자바는 왜 태어났나

자바의 최대 강점은
PC가 아니라 가전제품에서 나왔다.

이번 챕터가 오늘의 최중요 구간입니다.

1991년, 그린 프로젝트



PC가 아니라 가전제품에서 출발했다

가전제품이 낸 두 숙제

숙제 ① CPU가 제각각

제품마다 CPU 등 하드웨어가 다르다

→ 플랫폼 독립적 언어

숙제 ② 메모리가 부족

제어 장치에 실을 메모리가 아주 적었다

→ 작게 도는 가벼운 언어

이 두 숙제가 자바의 설계를 그대로 결정했다



참나무가 커피가 된 사연

Oak



Java

1995년, 출시 직전의 개명

- 첫 이름은 사무실 앞 참나무, Oak
- 웹이 발전하면서 oak와 웹을 융합하는 언어 탄생
- 커피 산지 자바 섬에서 따왔다는 설
- 제임스 고슬링은 하루 10여 잔의 커피광



자바 로고는 김이 나는 커피잔

1995년에서 오라클까지

1995

자바 공식 발표

선월드 콘퍼런스에서
세상에 공개됐다

1996

넷스케이프 탑재

브라우저에서 바로
실행되기 시작했다

2009

오라클 인수

선마이크로시스템즈가
오라클 품으로 갔다



웹이 원하는 '한 번 만들어 어디서나 실행'을 자바는 이미 갖고 있었다

C/C++의 한계



기계어

인텔과 ARM은 명령어
부터 다르다.



OS API

화면 출력 함수부터
OS마다 다르다.



파일 형식

윈도우는 exe, 리눅스
는 ELF.

결국 플랫폼마다 고쳐서 다시 컴파일해야 한다

W-O-R-A, 이번 주 중요 키워드

Write Once, Run Anywhere



C/C++

- 플랫폼마다 다시 컴파일한다
- 소스를 플랫폼별로 관리한다



자바

- ✓ 한 번 컴파일하면 그대로 쓴다
- ✓ 맥·윈도우·리눅스 어디서나

한 번 작성하면 JVM 있는 곳 어디서나 그대로 실행된다

A dense, dark background image filled with a variety of mobile devices, including smartphones, feature phones, and PDAs, illustrating the ubiquity of Java-powered devices.

30억 대의 기기

3 Billion Devices Run Java

— 자바 설치 화면에 오래 떠 있던 문구



셋톱박스, 블루레이, 카드 단말기까지 실제 수십억대의 디바이스에서 실행되고 있다

당신이 사장이라면

당신은 소프트웨어 회사의 사장입니다. 어느 쪽을 택하시겠습니까?

선택 A



OS마다 따로 개발

- 윈도우 팀, 맥 팀, 리눅스 팀
- 한 줄 고치면 세 번 고친다

선택 B



자바로 한 번만 개발

- 코드 하나, 모든 OS에서 실행
- 팀도 하나, 유지보수도 하나

인력 3배, 기간 3배, 유지보수 3배 — 플랫폼 독립성은 결국 돈 문제입니다.

바이트 코드의 정체



컴파일 결과

자바 컴파일러가 만들어
내는 바이너리 코드다.



CPU와 무관

특정 CPU의 명령이 아니
라서 직접 실행되지 않는
다.



JVM 기계어

JVM이 한 명령씩 읽어 해
석하고 실행한다.

바이트 코드는 .class 파일에 담긴다. 어떤 CPU에도 묶이지 않은 중간 언어다.

JVM은 종속, 코드는 독립



JVM은 종속

- 플랫폼마다 다르게 구현
- 오라클·IBM 등이 공급



코드는 독립

- 명령어 체계가 동일하다
- 어떤 JVM에서도 그대로

자바 프로그램에게 JVM은 늘 같은 컴퓨터다

실행까지 가는 길



실행 파일은 만들어지지 않는다. JVM이 곧 실행기다.

링크가 없다는 차이



C / C++

- 컴파일 후 링크를 거친다
- 결과물은 .exe 실행 파일
- CPU가 직접 실행한다



자바

- 링크 단계가 아예 없다
- 결과물은 .class 바이트 코드
- JVM이 대신 실행한다

링크도, 실행 파일도 없다. 자바는 JVM 하나로 실행한다.

1교시 확인 퀴즈

1

WORA는 무슨 뜻인가?

2

바이트 코드를 실행하는 건?

2교시

개발 도구와 절차

설치는 3교시. 지금은 이론부터.

JDK 구성

API와 패키지

편집·컴파일·실행

JDK가 곧 자바 배포판

JDK

Java Development Kit · 자바 개발 키트



컴파일러

소스를 번역하는 javac와 개발 도구 모음



실행 환경

바이트 코드를 실제로 돌리는 JVM



표준 API

미리 만들어진 클래스 라이브러리 묶음

JDK와 JRE의 차이



JDK · 개발자용

- 컴파일러 javac 포함
- JRE 전체를 그대로 포함
- 만들 사람에게 필요



JRE · 사용자용

- JVM + 표준 클래스 파일
- 컴파일러는 없다
- 실행만 할 사람에게

JDK는 JRE를 통째로 품고 있다

SE · ME · EE



Java SE

데스크톱과 서버용
응용 프로그램



Java ME

휴대전화와
셋톱박스 등 모바일



Java EE

다중 사용자 기업용
대규모 시스템

이 수업과 교재는 처음부터 끝까지 Java SE 기준

자바와 오픈 소스

2006년, 자바 소스 코드가 GPL로 공개됐다



얻은 것

- 개발 시간과 비용 단축
- 많은 참여자가 품질을 올린다



치른 대가

- 개량 버전이 난립한다
- 버전 간 호환성이 흔들린다

완전히 새로 시작하지 않는다



패키지

관련 클래스를 묶어 둔
단위



자바 API

미리 만들어 둔 클래스
라이브러리



JDK 포함

JDK에 처음부터 들어
있다

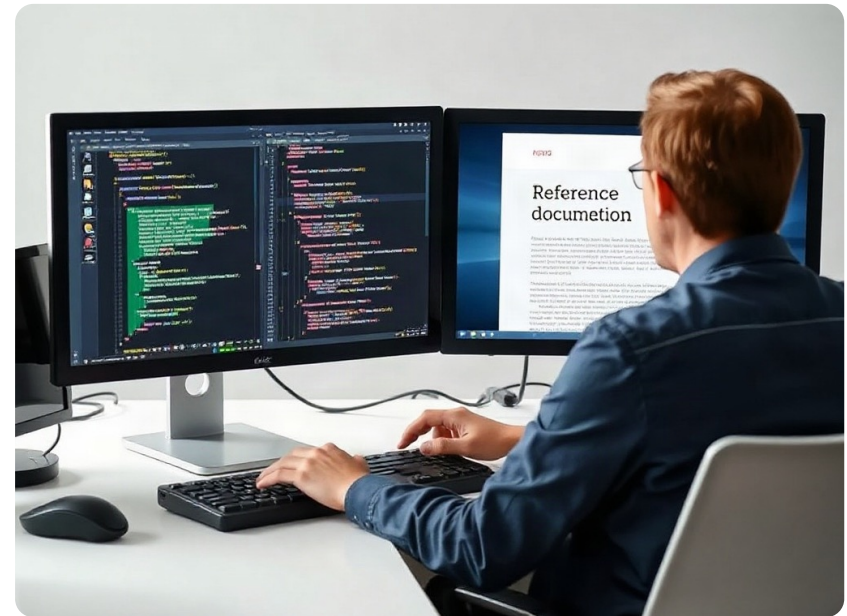
`System.out.println`도 남이 만들어 둔 클래스다

API 문서를 열어두자

상황 어떤 기능이 있는지 모른다

적용 온라인 API 문서에서 클래스 검색

결과 메소드 이름과 사용법 확인



편집 · 컴파일 · 실행

1

편집

Hello2030.java

메모장으로 소스를 쓴다
.java로 저장한다

2

컴파일

javac Hello2030.java

javac가 바이트 코드
.class를 만든다

3

실행

java Hello2030

확장자 없이
클래스 이름만 쓴다

클래스 이름과 파일 이름은 대소문자까지 똑같아야 한다

Hello2030 소스 읽기

```
public class Hello2030 {  
    public static void main(String[] args) {  
        int n = 2030;  
        System.out.println("Hello " + n);  
    }  
}
```

파일 이름은 **Hello2030.java**여야 한다

public class Hello2030

코드를 담는 껍데기

public static void main

여기서 실행이 시작된다

int n = 2030;

변수 선언과 초기화

System.out.println

화면에 한 줄 출력

명령창에서 돌려보기

```
C:\WTemp>javac Hello2030.java
```

Hello2030.class 생성

```
C:\WTemp>java Hello2030
```

```
C:\WTemp>java Hello2030.class
```

오류: 클래스를 찾을 수 없음

확장자 규칙

```
javac Hello2030.java
```

파일 이름 그대로

```
java Hello2030
```

확장자 떼고 이름만

시작은 언제나 main

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

public

누구나 호출할 수 있게

static

객체 없이 바로 실행

void

돌려줄 값이 없다

main

JVM이 찾는 이름



이름이나 형식이 하나라도 틀리면 실행되지 않는다



자바 활용과 특징

05

자바는 어디에 쓰이고 있나

자바가 지금도 살아 있는 이유는 쓰이는 자리가 넓기 때문이다.

데스크톱 응용

웹 · 서블릿

안드로이드

임베디드 · IoT

20년째 최상위 언어

프로그래밍 언어 순위

20년

**TIOBE 인덱스
최상위 유지**



데스크톱부터 모바일까지

웹, 모바일, 서버까지 한 언어로



기업 시스템의 뼈대

한 번 깔린 시스템은 쉽게 못 바꾼다



신입 채용 요구 기술

그래서 채용 공고에 계속 남는다

데스크톱 응용프로그램

설치해서 혼자 돌아간다

- PC에 설치해 단독 실행
- JRE만 있으면 어디서든 동작
- 교재가 다루는 대상도 이 분야

오늘 설치할 이클립스가 바로 그 예다



애플릿, 웹 속의 자바

애플릿(Applet)은 웹 브라우저 안에서 특정한 작은 기능을 수행하는 응용 프로그램을 뜻합니다.
단독으로 실행되지 않고 다른 소프트웨어의 구성 요소로 작동합니다.(웹 브라우저용 계산기, 그림판 등)



웹 페이지에 내장

HTML 문서에 심어 배포한다



단독 실행 불가

브라우저가 있어야 돌아간다



보안 제한

사용자 파일에 손대지 못한다



지금은 지원 종료

이름과 성격만 기억하면 된다

애플릿 vs 서블릿

서블릿(Servlet)은 자바를 사용해 웹 클라이언트의 요청을 처리하고 동적 응답을 생성하는 서버 사이드 자바 프로그램(API/규격)입니다



클라이언트 측

애플릿

- 사용자 브라우저에서 실행
- 화면을 보여주는 역할



서버 측

서블릿

- 웹 서버에서 실행
- 사용자 인터페이스가 없다

같은 자바라도 실행되는 자리가 다르면 역할이 완전히 갈린다

안드로이드의 뿌리



Java ME

단말기 제조사가
채택한 모바일 판



안드로이드

구글 주도 OHA가
만든 모바일 OS



Dalvik

언어는 자바지만
가상 기계는 별개

언어는 그대로, 가상 기계만 갈아 끼운 것이 안드로이드다.

30년 전에 본 IoT

1991년

제임스 고슬링의 메모에 있던 것

- 가전이 서로 연결될 것이다
- 공장이 아니라 네트워크로 교체
- 10년 만에 휴대폰·셋톱박스로
- 지금의 IoT와 OTA 업데이트



밤사이 스스로 업데이트되는 전기차

JAVA 특징 ① 독립성과 객체



플랫폼 독립성

- 바이트 코드와 JVM이 만든다
- 한 번 작성해 어디서나 실행



객체 지향

- 캡슐화 · 상속 · 다형성
- 현실의 객체로 문제를 모델링

두 기둥 모두 사람의 사고에 가깝게 만드는 장치다

JAVA 특징 ② 클래스가 전부

클래스 밖에는 변수도 메소드도 없다



모두 클래스 안

- 변수는 클래스 안에
- 메소드도 클래스 안에
- 내부 클래스도 가능



C언어는 전역 가능

- 전역 함수 존재 가능
- 클래스 자체가 없다

클래스 파일은 하나씩

소스 파일 W.java

```
public class W { }  
class X { }  
class Y { class Z { } }
```

클래스 4개를 한 파일에 담았다



javac 컴파일 결과

- W.class
- X.class
- Y.class
- **Y\$Z.class — 내부 클래스**

public 클래스는 파일당 하나 — 이름은 파일 이름과 같아야 한다

면접 질문: 자바의 클래스란?

아주 단순하고 깔끔한 답변 예시

한 줄 정의

“ 클래스는 객체를 만들기 위한 설계도이며, 속성(필드)과 기능(메서드)을 정의하는 틀입니다. ”

핵심 포인트



객체 생성의
기준



필드와
메서드를
함께 정의



같은 구조의
객체를 여러 개
생성 가능



코드 재사용과
구조화에 유리

쉬운 예시



Car 클래스

필드 : color, speed

메서드 : start(), stop()

객체 예: myCar, yourCar

면접 답변 예시



자바의 클래스는 객체를 만들기 위한 설계도입니다.
클래스 안에는 객체의 상태를 나타내는 필드와,
동작을 나타내는 메서드가 정의됩니다.
이 클래스를 바탕으로 여러 객체를 만들 수 있습니다.



면접에서는 ‘설계도’, ‘필드와 메서드’, ‘객체 생성’ 이 3가지를 말하면 충분합니다.

JAVA 특징 ③ jar로 배포



여러 클래스를 하나로

- 클래스 파일 여러 개를 압축
- jar 하나로 배포한다



실행은 main에서

- 프로그램 시작점은 main
- 한 클래스에 main은 하나

한 클래스에 main은 하나, 하지만 클래스마다 각자 가질 수 있다

JAVA 특징 ④ 패키지는 폴더



- 관련 클래스를 묶어 두는 분류 단위
- 패키지가 다르면 같은 이름의 클래스도 충돌하지 않는다

JAVA 특징 ⑤ 스레드와 GC



C / C++

스레드

운영체제에 의존한다

메모리

직접 반환해야 한다



자바

스레드

언어가 직접 지원한다

메모리

GC가 자동 회수한다

운영체제에 기대지 않고, 언어가 직접 챙긴다

JAVA 느리다는 오해와 JIT

오해

해석 실행이라
C/C++보다 느리다

성능이 중요한 곳엔
못 쓴다

사실

JIT가 실행 중
기계어로 컴파일한다

C/C++에 근접한
속도를 낸다

JIT = Just In Time · 필요한 순간에 기계어로 번역한다

실시간에는 부적합



예측 불가

GC 실행 시점을
아무도 모른다



순간 정지

그 순간 프로그램이
잠시 멈춘다



마감이 있다

정해진 시간 안에
답해야 하는 일

그래서 자바는 실시간 제어 시스템에는 부적합하다

포인터가 없다는 축복

// 포인터를 없앤 것은 기능을 뺀 게 아니라
사고를 원천 봉쇄한 설계다

안전하다 · 쉽다 · GC는 한 뿌리다



잘못된 주소를
건드릴 수 없다



메모리 반환은
가비지 컬렉터 몫

2교시 확인 퀴즈

1

JDK와 JRE는 무엇이 다르고, 어느 쪽이 어느 쪽을 포함하는가?

2

하나의 .class 파일에 여러 개의 클래스가 들어갈 수 있는가?

3교시

이제부터는 직접 실습



컴퓨터 전원 ON

JDK 21 설치하기

1

다운로드

오라클 사이트에서
JDK 내려받기

2

설치 실행

설치 파일 실행,
경로는 기본값 그대로

3

설치 확인

명령창에서
java -version 확인



실습실 PC는 이미 설치돼 있을 수 있으니 먼저 확인.

터미널 명령어 : java -version

<https://www.oracle.com/kr/java/technologies/downloads/#java21>

IDE와 이클립스

Integrated Development Environment · 통합 개발 환경

Eclipse IDE for Java Developers



편집

오타를 즉시 표시



컴파일

저장하면 자동으로



디버깅

한 줄씩 멈춰본다

이클립스 · IBM이 자바로 만든 오픈 소스 · [eclipse.org](https://www.eclipse.org)에서 무료

<https://www.eclipse.org/downloads/packages/>

일식이 태양을 가린다

- 이클립스는 일식이라는 뜻
- 당시 자바의 주인은 Sun, 곧 태양
- Sun은 맞불로 NetBeans를 밀었다

그 이클립스도
자바로 만든 응용이다



이름에 담긴 경쟁의 흔적

workspace 지정하기



workspace

C:\javaworks

HelloProject

Week02

Homework

- 작업 폴더를 미리 만들어 둔다
- 실행할 때 Browse로 지정한다
- 프로젝트 여러 개를 담는 상자
- 학기 내내 같은 폴더를 쓴다



경로는 영문으로 잡는다

프로젝트 만들기

1

메뉴 열기

File → New →
Project 클릭

2

종류 선택

목록에서 Java
Project 선택

3

이름 입력

이름 칸에
SampleProject

4

생성 완료

Finish 누르면
폴더가 생긴다

프로젝트 하나 = 자바 응용 하나를 통째로 담는 폴더

클래스 만들고 코드 치기

- 1 File → New → Class 열기
- 2 Package 칸에 OpenChallenge1 입력
Name 칸에 Hello2030 입력
- 3 **Finish**로 클래스 선언 자동 생성
- 4 본문 코드는 손으로 직접 입력

Finish 누르면 여기까지 자동

```
public class Hello2030 {  
  
}
```

중괄호 안에 2교시 코드를 입력

실행하고 콘솔 확인



실행은 클릭 한 번

- 툴바의 실행 아이콘 클릭
- 메뉴 Run 또는 Ctrl+F11

디버깅을 위한 실행 방법 별도 존재.

자기소개 프로그램



미션 조건

- 프로젝트: OpenChallenge1
- 클래스: MyInfoApp
- 출력: 이름·나이·학과 3줄

Hello2023.java

```
public class Hello2023 {  
    public static void main(String[] args) {  
        System.out.println("Jisoo Kim");  
        System.out.println("20");  
        System.out.println("Dept. of AI");  
    }  
}
```

안 될 때 여기부터

틀림



- 클래스 이름 ≠ 파일 이름
- main 시그니처 오타
- 세미콜론.괄호 짝 누락

맞음



- 클래스 이름 = 파일 이름
- main 시그니처 그대로 복사
- 빨간 밑줄에 마우스 올리기

오류 메시지를 읽는 습관이 1년 차이를 만든다

먼저 끝냈다면

교재 실습 문제 두 개를 더 풀어 본다



실습 1-1 · Welcome

- Welcome 클래스를 만든다
- Welcome!! 을 출력한다



실습 1-3 · One2Nine

- One2Nine 클래스를 만든다
- 1부터 9까지 한 줄로

오늘 공부한 내용

1. 소스 .java → 컴파일 → .class → JVM 실행
2. **WORA: 프로그램은 독립, JVM은 플랫폼 종속**
3. JDK는 개발용, JRE는 실행용
4. 특징: 객체 지향 · 캡슐화 · GC · JIT
5. workspace ⊃ project ⊃ class

다음 주엔 변수부터

오늘 만든 workspace를 그대로 들고 2장으로 넘어갑니다



다음 주 예고

- 변수와 자료형
- 연산자와 조건문



반드시 지참

- 노트북과 충전기
- 교재 명품 자바 에센셜