JAVA란 무엇일까요!

자바 시작하기

# 0. 학습 목표

1. 자바의 출현 배경과 플랫폼 독립성에 대한 이해
2. 자바 가상 기계와 자바의 실행 환경 이해
3. JDK와 JRE등 자바 개발 환경 이해
4. 자바 응용프로그램의 종류의 특징 이해
5. 자바 언어와 자바 플랫폼의 특징 이해

# 1. 프로그래밍 언어

### 프로그램 작성 언어

* **기계어** : 0, 1의 이진수로 구성된 언어. 컴퓨터의 CPU는 기계어만 처리한다.
* **어셈블리어** : 기계어 명령을 표현하기 쉬운 상징적인 단어인 니모닉 기호로 일대일 대응시킨 언어. (Ex. ADD, SUB, MOVE)
* **고급언어** : 사람이 이해하기 쉽고, 복잡한 작접, 자료 구조, 알고리즘 등을 표현하기 위해 고안된 언어.

> 절차 지향 언어와 객체 지향 언어로 나눌 수 있다.

> C/C++, JAVA, C# 등…

### 프로그래밍과 컴파일

* **소스** : 프로그래밍 언어로 작성된 텍스트 파일
* **컴파일** : 소스 파일을 컴퓨터가 이해할 수 있는 기계어로 만드는 과정!

자바 : .java -> .class

# 2. 자바의 배경과 독립성 이해하기

### 자바의 태동

* 플랫폼 호환성 문제를 해결하고, 플랫폼 독립적은 언어를 개발하기 위해 시작됨.
* 메모리 사용량이 적고 다양한 플랫폼을 가지는 장치에 적용.

### 자바의 플랫폼 독립성

* 기존 언어 : **인텔 CPU, 리눅스 환경에서 개발한 C/C++ 응용 프로그램은 MAC이나 인텔 CPU의 윈도우 노트북에서는 돌아가지 않는다!**

> 기계어가 CPU마다 다름.

> 운영체제마다 API가 다름.

> 운영체제마다 실행파일의 형식이 다름. (우분투에선 exe가 돌아가지 않더라…)

* **WORA** : Write Once Run Anywhere

**한 번 작성된 코드는 모든 플랫폼에서 바로 실행된다!**

C/C++ 등 기존 언어가 가진 플랫폼 종속성을 극복한다!

### 자바 가상 기계와 자바 실행 환경

* **바이트 코드** : 자바 가상 기계에서 실행 가능한 바이너리 코드.

> 바이트 코드는 CPU에 의해 직접 실행되는 게 아니고, JVM이 인터프리터 방식으로

코드를 해석해, 작동 중인 플랫폼에서 실행된다.

* **자바 가상 기계 (JVM : Java Virtual Machine)**

각기 다른 플랫폼에 설치되고, 동일한 자바 실행 환경을 제공한다.

자바 가상 기계 자체는 플랫폼에 종속성이다.

* 자바의 실행 : 자바 가상 기계가 클래스 파일(.class)의 바이트 코드를 실행한다.

# 3. 자바 개발 환경 이해하기

### JDK와 JRE

* **JDK(Java Development Kit)** : 자바 응용 개발 환경. 개발에 필요한 도구를 포함함

Ex) 컴파일러, 클래스 라이브러리, JRE 등

+) bin 디렉터리에 포함된 주요 개발 도구들

* javac : 자바 소스를 바이트 코드로 변환하는 컴파일러
* java : jre의 bin 디렉터리에도 있는 자바 응용프로그램 실행기
* jar : 자바 아카이브 파일을 생성 및 관리하는 유틸리티
* jdb : 자바 디버거
* appieviewer : 웹 브라우저 없이 애플릿을 실행하는 유틸리티
* **JRE(Java Runtime Environment)** : 자바 실행 환경. JVM 포함

### 자바 API

* 자바 패키지 : 서로 관련된 클래스들을 계층 구조로 분류해 묶어 놓은 것
* 자바 API : 개발자가 이용해 쉽고 빠르게 자바 프로그램을 개발할 수 있는 자바 라이브러리

# 4. 자바 응용의 종류

### 데스크톱 응용 프로그램

* 가장 전형적인 자바 응용 프로그램으로, PC에 설치되어 실행.
* JRE가 설치된 어떤 환경에서든 실행될 수 있음.

### 애플릿 응용 프로그램

* 애플릿 : 웹 브라우저에 의해 구동되고 실행이 제어되는 자바 프로그램

애플릿은 사용할 수 있는 자원 접근에 제약이 있음.

### 서블릿 응용 프로그램

* 서블릿 : 애플릿과 반대로 서버에서 실행되는 자바 프로그램

데이터베이스 서버 및 기타 서버와 연동하는 복잡한 기능 구현 시 사용됨

사용자 인터페이스가 필요 없고, 웹 서버에 의해 실행 통제를 받는다.

* 서버 클라이언트 모델에서 서블릿과 애플릿이 각각 통신하면서 실행됨

### 모바일 응용 프로그램 : 안드로이드 앱

* 안드로이드 : 구글의 주도로 모바일 회사가 모여 구성한 무료 모바일 플랫폼.
* 개발 언어로 자바를 사용한다.

5. 자바의 특징

### 플랫폼 독립성

* 하드웨어, 운영체제에 종속되지 않는 바이트 코드로 플랫폼 독립성을 가짐.

### 객체 지향

* 캡슐화, 상속, 다형성 지원

### 캡슐화

* 자바의 모든 변수나 함수는 클래스 내에 선언됨
* 클래스 안에서 클래스(내부 클래스) 작성 가능

### 소스와 클래스 파일

* 하나의 소스 파일에 여러 클래스 작성 가능 **(public 클래스는 하나만 가능!)**
* 소스 파일의 이름과 public으로 선언된 클래스 이름은 같아야 함
* 클래스 파일에는 하나의 클래스만 존재함

### 실행 모듈

* 하나 또는 다수의 클래스 파일로 구성됨.
* 자바 응용프로그램의 실행은 main() 메소드에서 시작됨.

> **하나의 클래스 파일에 두 개 이상의 main() 메소드가 있을 수 없다!**

### 패키지

* 서로 관련 있는 여러 클래스를 패키지로 묶어 관리한다. 폴더 개념으로 보면 됨.

### 멀티스레드

* 여러 스레드의 동시 수행 환경을 지원한다.
* C/C++은 멀티스레드를 위해 운영체제 API를 호출하지만, 자바는 운영체제의 도움 필요X

### 가비지 컬렉션

* 자바 언어는 메모리 할당 기능은 있지만 반환하는 기능은 없다.
* 사용하지 않는 메모리는 JVM에 의해서 자동 반환되는데, 이를 가비지 컬렉션이라고 함.

### 실시간 응용프로그램엔 부적합

* 실행 도중 예측할 수 없는 시점에 가비지 컬렉션이 실행되기 때문에...

### 안정성

* 타입 체크가 엄격하고, 물리적 주소를 사용하는 포인터 개념이 없다.

### 편리함

* 포인터 개념이 없고, 동적 메모리를 반환하지 않으며 다양한 라이브러리를 지원함.

### 실행 속도 개선을 위한 JIT 컴파일러 사용

* 자바는 바이트 코드를 인터프리터 방식으로 실행하기 때문에 기계어 실행보다 느리다.
* JIT 컴파일 : 실행 중 바이트 코드를 기계어 코드로 컴파일해 기계어를 실행하는 기법