01-1. 소개

Java Overview

Software Language

언어는 의사소통을 목적으로 합니다.

- 한국어, 영어등은 사람과 사람 간의 의사소통을 목적으로 합니다.
- 소프트웨어 언어도 사람이 사용하는 언어이지만 컴퓨터와 의사소통을 목적으로 사용하는 언어

- 컴퓨터의 OS 혹은 어떤 애플리케이션을 실행시켜 주는 플랫폼에게 일을 시키기 위해서 사용하는 언어가 소프트웨어 언어입니다.

- 소프트웨어 언어를 가지고 어플리케이션이 어떻게 실행되어야 한다는 것을 기록해 놓으면 그 애플리케이션을 실행시키는 플랫폼에서 그 기록을 해석해 원하는 대로 실행시켜 준다는 것입니다.

Software Language

애플리케이션은 플랫폼에서 이해할 수 있는 언어로 개발되어야 한다.

- 사람과 사람 간에 대화를 할 때 그 사람이 이해할 수 있는 언어를 사용해야 하는 것과 동일합니다.
- 애플리케이션이 윈도우 OS 에서 실행되어야 한다면 윈도우 OS 에서 이해할 수 있는 C# 등의 언어를 이용해야 하고, 애플리케이션이 리눅스에서 실행되어야 한다면 리눅스에서 이해할 수 있는 리눅트 C 등의 언어를 이용해야 합니다.
- 우리는 안드로이드 애플리케이션 개발을 목표로 하고 우리의 애플리케이션이 안드로이드 OS에서 실행됨으로 안드로이드 OS에서 이해할 수 있는 언어를 이용해 애플리케이션이 개발되어야 합니다.
- 이 안드로이드에서 실행되는 애플리케이션을 개발하기 위한 언어가 자바 혹은 코틀린입니다.

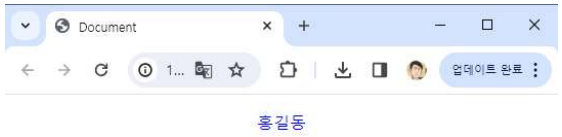
Software Language

동적 컨텐츠를 만들 수 있어야 소프트웨어 언어라고 할 수 있다.

- 변수가 있어야 하며 변수의 값을 변경하면서 동적인 컨텐츠가 나오게 해야 합니다.
- 이런 능력을 가지고 있는 것을 소프트웨어 언어라고 합니다.

- HTML 은 Hyper Text Markup Language 의 약어로 L 이 Language 의 약어입니다.
- HTML 로 화면의 구조를 정의하고 CSS 로 화면 기법을 적용합니다.
- 하지만 HTML CSS 에서 변수 처리가 안되며 HTML CSS 만으로 동적인 애플리케이션을 만들 수 없습니다.

```
<style>
  body { text-align: center; }
  #name { color: blue; }
</style>
<p id="name">홍길동</p>
```



- 언제 누가 실행시키든 화면은 동일하다는 이야기입니다. 이를 정적(static) 컨텐츠라고 합니다.
- 주식사이트에 출력되는 주가가 항상 동일하다면? 뉴스 사이트에 출력되는 뉴스가 항상 동일하다면? 우리는 이런 사이트를 애플리케이션이라고 하지 않습니다.

Software Language

플랫폼이 이해할 수 있는 문법에 맞게 작성되어야 한다.

- 언어를 이용해 애플리케이션을 만든다는 것은 애플리케이션을 실행시켜 주는 플랫폼에 우리의 애플리케이션을 어떻게 실행시켜 주면 된다고 명시하는 일종의 명령서를 만드는 것입니다.
- 애플리케이션을 실행시켜 주는 플랫폼에서 이해할 수 있는 언어로 만들어야 하고 그 플랫폼에서 해석 가능하게 만들어야 합니다.
- 자바라는 언어를 이용한다고 하더라도 안드로이드 플랫폼이 이해할 수 있는 형태로 명령서를 만들어야 함으로 규칙을 준수해서 만들어야 합니다.
- 그런 규칙을 흔히 언어의 문법이라고 하죠.

Software Language

소프트웨어 언어들은 종류가 상당히 많아요.

- 자바(Java)만 있는 것이 아니라 자바스크립트(JavaScript)라고 하는 소프트웨어 언어도 있고요. C라는 소프트웨어 언어도 있고 C++도 있고 C#도 있고 파이썬(Python)도 있고 코틀린(Kotlin)도 있고 스위프트(Swift)도 있고 다트(Dart)도 있고 스칼라(Scala) 라고 하는 소프트웨어 언어도 있습니다.

- 어떤 애플리케이션을 개발하는 지에 따라 사용하는 소프트웨어 언어라 다르며 현업 개발자들에 따라 주력으로 사용하는 소프트웨어 언어가 다릅니다.

- "나는 자바밖에 안 배웠어. 그러니까 나는 자바만 할 거야!" 가 사실 불가능
- 저희가 목적으로 하는 거는 소프트웨어 언어를 이용을 해서 애플리케이션을 개발하는 것이 목표이지, 소프트웨어 언어 자체가 목표가 아니기 때문

- 소프트웨어 언어라고 하는 것은 도구밖에 안 돼요.
- 소프트웨어 개발자들이 애플리케이션을 만들기 위해서 사용하는 도구밖에 안 된다는 거죠.

Software Language

소프트웨어 언어들은 대등소이 하다.

- 처음 학습한 언어가 뭐가 됐든 그 언어를 열심히 학습을 하면 그 다음 언어를 학습할 때는 언어마다 특징이 있고 문법이 다르니까 학습은 분명히 들어가야 겠지만 처음 언어를 학습했던 것보다는 공이 훨씬 더 작게 될 거다.
- 자바를 학습하면서 변수, 함수, 클래스, 객체, 연산자등의 개념들이 학습될 것인데 이런 것들은 언어별로 약간의 문법적인 차이만 있는 것이지 모든 언어에서 공통으로 나오는 개념이들 입니다.
- 자바를 학습하면서 모든 언어의 공통적인 개념에 대해서 좀 잘 정리해 놓으면 나중에 어떤 이유에 의해 자바스크립트를 하거나 C#을 하거나 이랬을 때 클래스가 뭐지? 변수가 뭐지? 함수가 뭐지? 를 따로 이해할 필요가 없게 됩니다. 단지 이 언어에서는 변수를, 클래스를 이용하는 문법이 어떻게 되는 것이지? 만 정리하면 되겠지요.

Software Language

소프트웨어 언어 학습은 문법 학습이다.

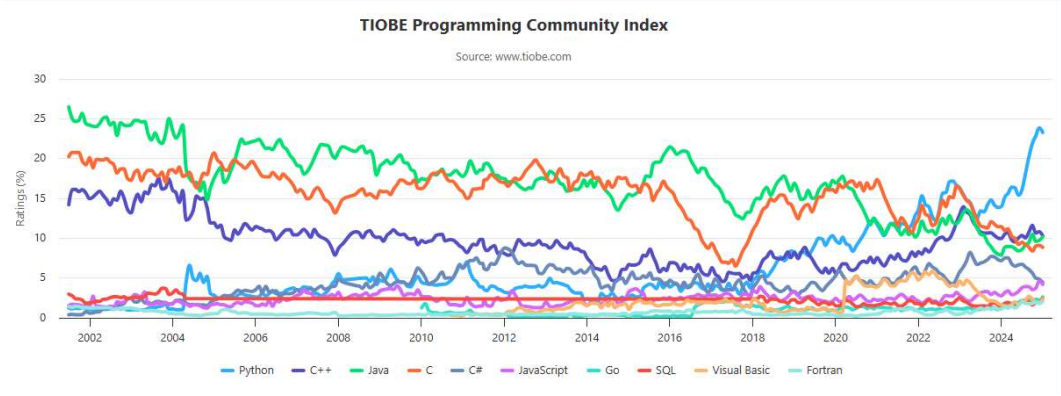
- 모든 소프트웨어 언어는 문법이 있으며 그 문법에 맞게 작성해야 합니다. 결국 어떤 소프트웨어 언어를 학습한다는 것은 그 언어가 가지고 있는 문법을 배우는 것부터 시작합니다.

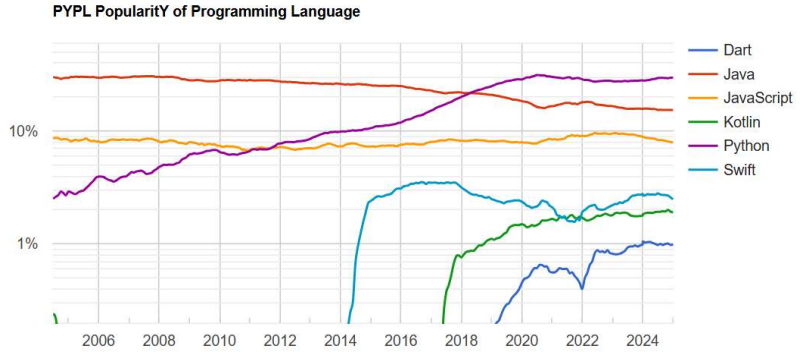
- 우리가 사용하는 한국어와 영어의 문법에 차이가 있듯이 소프트웨어 언어별로 약간의 문법 차이는 있습니다.
- 즉 변수, 함수, 클래스등의 문법은 모두 제공하지만 변수를 선언하는 방법, 함수를 선언하는 방법등의 차이는 있습니다.

자바 프로그래밍 – 자바 언어

- 1990년말 가전제품에서 동작하는 언어 개발-오크(Oak)
- 1995년 자바(Java) 1.0 버전 발표
- 2000년대 중반이후 가장 많이 사용하는 프로그래밍 언어
- GPL(General Public License) 라이선스
- 2009년 오라클(Oracle)로 인수합병
- 2019년부터 유료화 정책

자바 프로그래밍 – 자바 언어





자바 프로그래밍 – 컴파일러 vs. 인터프리터

- 소프트웨어 언어는 크게 또 두 가지 타입으로 구분
- 소스코드는 사람이 이해할 수 있는 문서이기 때문에 엄밀히 말하면 텍스트 파일

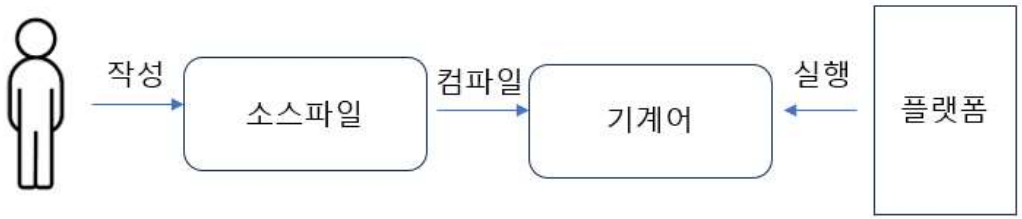
스크립트 언어

- 사람이 해석할 수 있는 소스 파일을 특정 플랫폼에서 직접 해석해 실행시켜 준다면 그 소스 파일을 개발한 소프트웨어 언어를 스크립트 언어라고 분류합니다.
- 대표적인 스크립트 언어가 자바스크립트.
- 자바스크립트로 개발된 소스파일이 브라우저에 의해 해석되어 애플리케이션이 실행되게 됩니다.
- 자바는 스크립트 언어가 아니다.

자바 프로그래밍 – 컴파일러 vs. 인터프리터

컴파일 언어

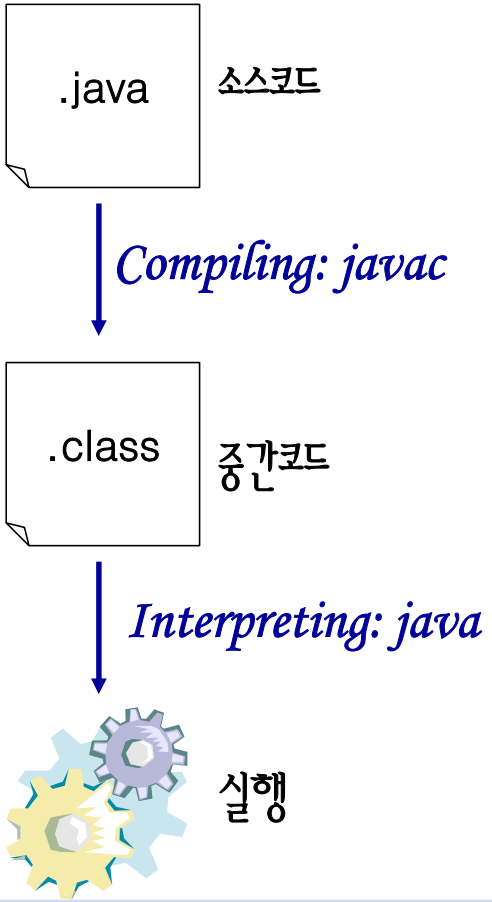
- 소스 파일을 다시 플랫폼에서 해석 가능한 형태로 변형시켜 실행
- 소스파일을 플랫폼이 이해할 수 있는 형태로 변형하는 작업을 컴파일이라고 부르며,컴파일에 의해 사람은 해석 불가능한 특정 플랫폼만 이해할 수 있는 기계어로 바뀌었다고 표현합니다.
- 이 컴파일 과정이 필요한 소프트웨어 언어를 컴파일 언어라고 부릅니다.



- 대표적으로 자바, C 등의 소프트웨어 언어들이 컴파일 언어

자바 프로그래밍 – 컴파일러 vs. 인터프리터

- 자바 프로그램의 동작 방식
  - 소스 코드 작성
  - 중간 코드(Bytecode)로 컴파일 (Compile)
  - 기계어로 인터프리팅 (Interpreting)
  - 자바 가상 머신 (JVM)에서 실행



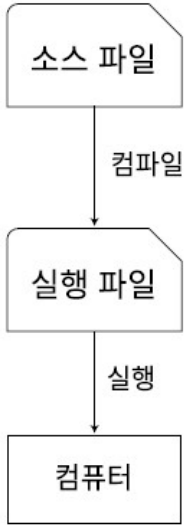
자바 프로그래밍 – 컴파일러 vs. 인터프리터

- 컴파일 (Compile)
  - 컴파일러를 이용하여 고급언어를 기계어로 번역하는 과정
  - 프로그램 실행 전에 미리 수행되어야 함

- 인터프리팅 (Interpreting)
  - 중간코드를 각 하드웨어에 따른 기계어로 번역
  - 실행시간(Run-time)에 번역됨
  - 자바 가상 머신(Java Virtual Machine)에 의하여 수행됨

- 자바 언어: 컴파일러와 인터프리터를 모두 사용

자바 프로그래밍 – 컴파일러 vs. 인터프리터

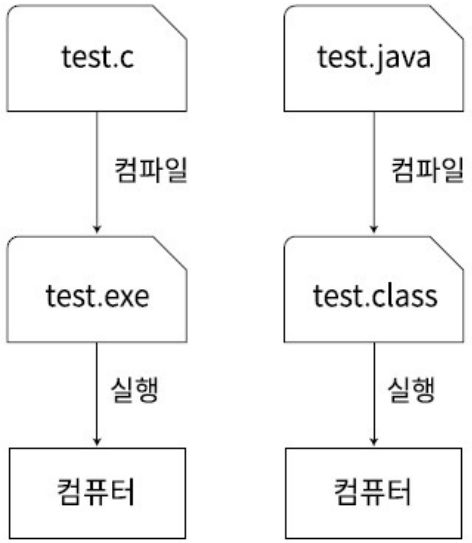


test.c (C 언어 소스 파일)

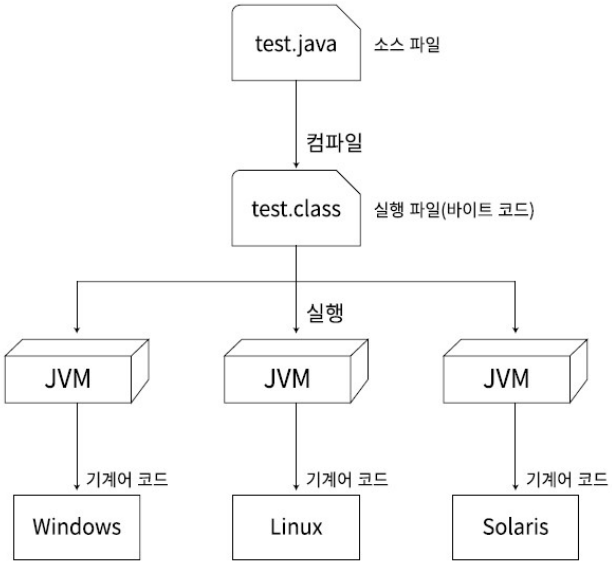
```
printf("%s", "직진하시오");
printf("%s", "50M 앞에서 좌회전하시오");
```

test.java (자바 언어 소스 파일)

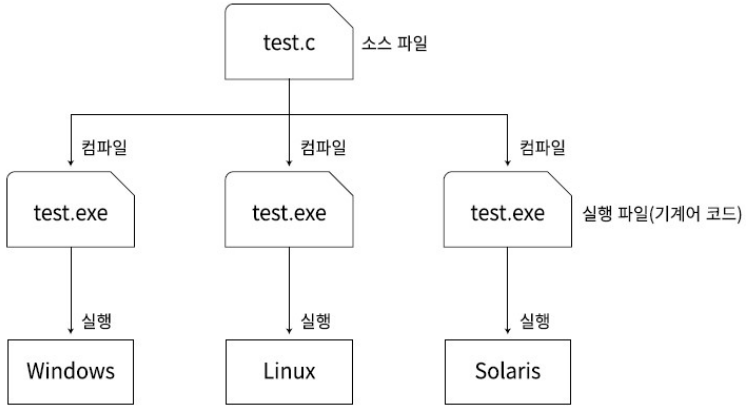
```
System.out.print("직진하시오");
System.out.print("50M 앞에서 좌회전하시오");
```



자바 프로그래밍 – 컴파일러 vs. 인터프리터



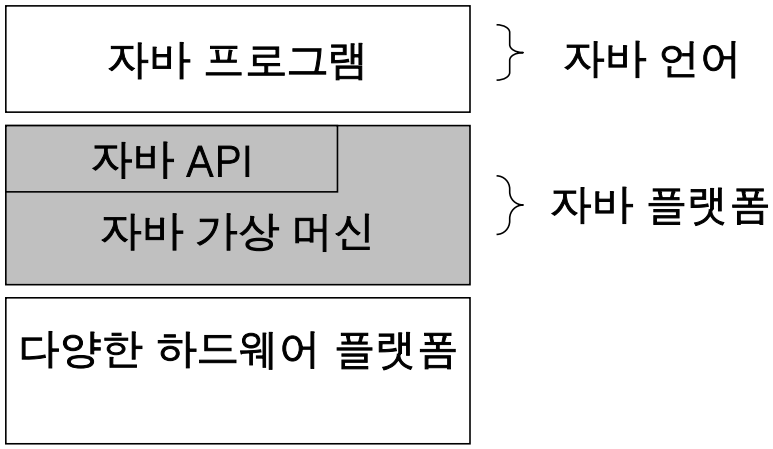
자바 실행 구조

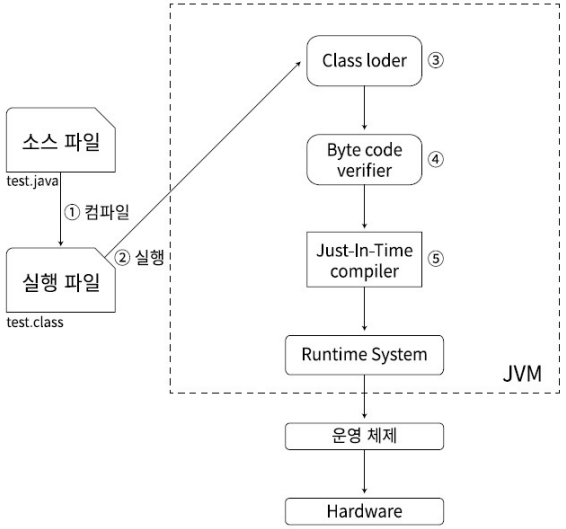


C 실행 구조

자바 프로그래밍 – 컴파일러 vs. 인터프리터

- 자바 가상 머신 (JVM)
  - 자바의 중간코드(Bytecode)를 실행시키는 플랫폼 역할
  - 중간코드(Bytecode)는 컴퓨터 하드웨어에 무관하게 공통으로 사용할 수 있는 자바 전용 어셈블리 언어





자바 프로그래밍 언어의 특징

- 자바 언어는 단순하며 상대적으로 배우기 쉽다.
  - 자바 언어는 C/C++ 언어의 구문과 C++ 언어의 객체지향성을 채택하면서 포인터, 연산자 중복 등의 복잡한 기능은 제외시킴
  - 동적으로 할당되는 메모리를 프로그래머가 신경 쓸 필요 없이 Garbage Collector를 통해 자동으로 메모리를 관리해줌
  - 예외처리(Exception Handling) 개념 도입

❑ 자바 언어는 플랫폼 독립적이다.

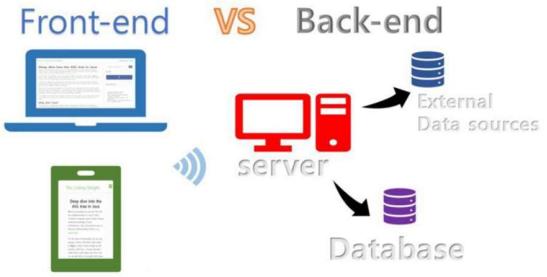
❑ 객체 지향 프로그래밍(Object-Oriented Programming: OOP)

자바 Platform

- Java SE(Java Platform, Standard Edition)
  데스크톱, 서버, 임베디드 시스템 개발을 위한 표준 자바 플랫폼으로 자바의 기본 개발환경을 제공
- Java EE(Java Enterprise Edition)
  Java SE에 웹 서버 역할을 추가한 것으로 자바 애플리케이션을 동작시킬 수 있는 컨테이너 등을 표준화한 플랫폼
- Java ME(Java Micro Edition)
  모바일이나 내장형 장치처럼 리소스가 제한된 소형 장치에서 실행되는 자바 애플리케이션을 위해 경량화된 기술들을 지원하는 플랫

자바로 개발되는 애플리케이션

- Back-End Web Applicaiton
  Java, Servlet & JSP, Spring Framework



- Android Application

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare