📖 Java Study Skill Up Team Review

자바의 역사

- 1991년 썬의 엔지니어들에 의해서 고안된 오크(Oak)라는 언어에서부터 시작되었다.
    - 오크(Oak)는 처음에 가전제품에서 사용될 목적이었다.
        - 하지만 인터넷의 등장과 함께 인터넷에서 실행되는 프로그래밍 언어로 사용 목적이 바뀌면서 이름도 자바(Java)로 변경되었다.
- 1995년 썬 마이크로시스템즈(Sun Microsystems)에서 처음 자바 언어를 발표했고 지금까지 다양한 분야에서 사용되고 있다.
- 1999년 윈도우(Windows) 프로그램 개발이 주류였으며 C++ 언어에 비해 자바는 열세였다.
    - 하지만 인터넷이 활성화되면서 웹 애플리케이션 구축용 언어로 자바가 급부상하게 되었다.
    - 기업체 및 공공기관의 다양한 서버 운영체제에서 단 한 번의 작성으로 모든 곳에서 실행 가능한 언어는 자바뿐이었기 때문이다.
- 2023년 개인적으로 지금까지 점유율이 가장 높은 건 여전히 자바라고 알고 있다.
    - 치고 올라오는 언어들이 있다고 하더라도 사용도에 대해서는 순위가 변동될 일이 어려울 거라는 생각이 든다.

자바의 특징

1. 이식성이 높은 언어이다.
    - 이식성은 서로 다른 실행 환경을 가진 시스템 간에 프로그램을 옮겨 실행할 수 있는 것이다.
    - 한 마디로 윈도우에서 실행하는 프로그램을 리눅스 또는 유닉스에서 실행할 수 있다면 이식성이 높은 것이다.
    - 이처럼 자바 언어로 개발된 프로그램은 소스 파일을 수정하지 않아도 자바 실행 환경(JRE)이 설치되어 있는 모든 운영체제에서 실행이 가능하다.
2. 객체 지향 언어
    - 자바는 아무리 작은 프로그램이라도 객체를 만들어 사용한다.
    - 이는 처음부터 객체를 고려하여 설계되었기 때문이며 객체 지향 언어가 가져야 할 캡슐화, 상속, 다형성 기능을 완벽하게 지원하고 있다.
3. 함수적 스타일 코딩
    - 대용량 데이터의 병렬처리, 이벤트 지향 프로그래밍을 위해 함수적 프로그래밍의 부각
    - 즉 이를 통해 코드의 가독성과 유지 보수성을 향상시키고, 복잡한 작업을 간결하게 표현할 수 있다.
4. 메모리 자동 관리
    - 객체 생성(자동으로 메모리 영역을 찾아서 할당한다.)

    - 사용 완료(쓰레기 수집기(Garbage Collector))를 실행시켜 자동적으로 사용하지 않는 객체를 제거한다.)

5. 다양한 애플리케이션 개발

    - 윈도우, 리눅스, 유닉스, 맥 등 다양한 OS 에서 실행되는 프로그램을 개발할 수 있다.
    - 클라이언트용 윈도우 애플리케이션, 서버용 웹 앱플리케이션, 모바일용 안드로이드 앱 등 거의 모든 곳에서 실행 가능한 프로그램을 개발할 수 있다.
    - 자바는 다양한 운영체제에서 사용할 수 있는 개발 도구와 API 를 묶어 에디션 형태로 제공하고 있다.

6. 멀티 스레드의 쉬운 구현

    - 하나의 프로그램이 동시에 여러 가지 작업을 처리해야 할 경우와 대용량 작업을 빠르게 처리하기 위해 서브 작업으로 분리해서 병렬 처리를 하려면 멀티 스레드 프로그래밍이 필요하다.
    - 프로그램이 실행되는 운영체제에 따라서 멀티 스레드를 구현하는 방법이 다르지만, 자바는 스레드 생성 및 제어와 관련된 라이브러리 API 를 제공하고 있기 때문에 실행 운영체제에 상관없이 멀티 스레드를 쉽게 구현할 수 있다.

7. 다이나믹 로딩 지원

    - 자바 애플리케이션은 여러 개의 객체가 서로 연결되어 실행되는데 이 객체들은 Class 로부터 생성된다.
    - 동적 로딩하여 객체를 생성하면 애플리케이션이 실행될 때 모든 객체가 생성되지 않고 객체가 필요한 시점에 객체를 생성한다.
    - 또한 개발 완료 후 유지보수(수정)가 발생하더라도 해당 Class 만 수정하면 되므로 유지보수를 쉽고 빠르게 진행할 수 있다.

자바의 특징은 굉장히 많고 사용에 따라 다른 언어와 비교했을 때 장점과 단점도 존재할 것이다.
하지만 역시 자바를 선택하는 이유는 풍부한 자바 오픈소스 라이브러리이지 않을까 싶다.

[오픈소스 라이브러리]

- 자바는 오픈 소스 언어이다.
    - 그렇기 때문에 자바 프로그램에서 사용하는 라이브러리 또한 오픈소스가 넘쳐난다.
- 고급 기능을 구현하기 위해 코드를 직접 작성할 경우 많은 시간을 투자해야 하며 실행에 대한 안전성 또한 보장되지 않는다.
    - 하지만 검증된 오픈소스 라이브러리를 사용하면 개발 기간을 단축시켜 줌은 물론 높은 안정성의 애플리케이션을 보다 간편하게 개발할 수 있다.

즉 이러한 풍부한 자바 오픈소스 라이브러리로 인해 많은 회사들이 자바를 선택한다.

프로그래밍 언어와 자바

[언어]

- 사람이 사용하는 언어
    - 컴퓨터 입장에서 보면 이해할 수 없는 문자 집합이다.
- 기계어
    - 0과 1로 이루어진 이진 코드이기 때문에 사람이 이해하기는 매우 어렵다.
    - 왜 이진 코드?

        - 내부 동작과 하드웨어 구성과 관련이 있다.
        - 트랜지스터는 전기 신호를 켜고 끄는 역할을 하며, 이 두 가지 상태를 사용하여 정보를 저장하고 처리한다.
        - 즉 컴퓨터는 이러한 트랜지스터의 전기 신호를 ON(1)과 OFF(0) 상태로 사용하여 정보를 저장하고 처리하는데, 이는 컴퓨터에서 가장 정확성이 높고 효율적이라고 할 수 있다.

> 이처럼 사람의 언어와 기계어는 서로 다르기 때문에 다리 역할을 하는 프로그래밍 언어가 필요한 것이다.

[프로그래밍 언어]

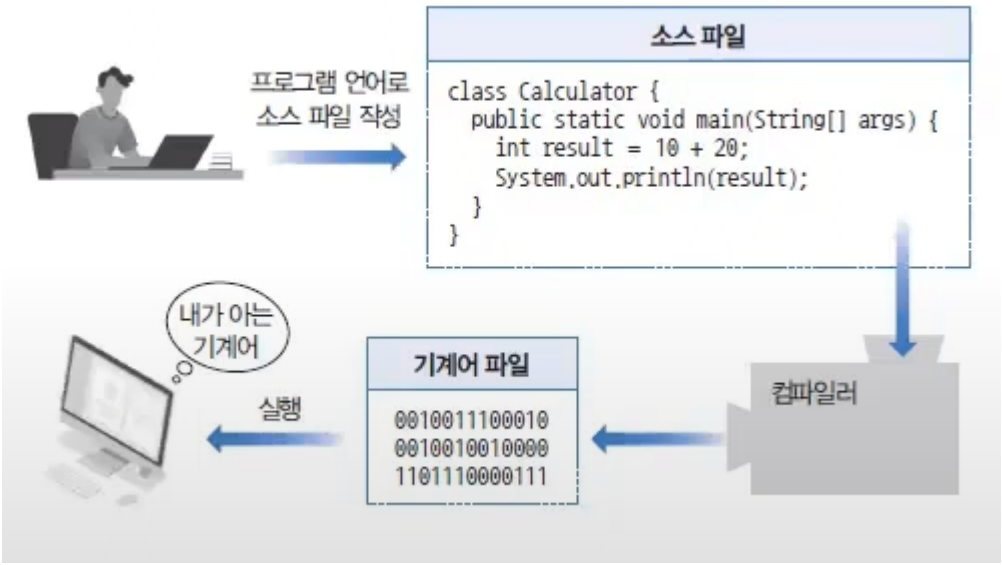
- 프로그래밍(코딩)
- 프로그래밍?
    - 사람이 컴퓨터에게 일을 시키는 것
    - But 컴퓨터는 사람의 말을 알아들을 수 없다.
    - 그렇기에 프로그래밍 언어가 필요하다.
- 언어?
    - 사람이 사용하는 언어
        - 컴퓨터 입장에서 보면 이해할 수 없는 문자 집합이다.
    - 기계어
        - 0과 1로 이루어진 이진 코드이기 때문에 사람이 이해하기는 매우 어렵다.
        - 왜 이진 코드?
            - 내부 동작과 하드웨어 구성과 관련이 있다.
            - 트랜지스터는 전기 신호를 켜고 끄는 역할을 하며, 이 두 가지 상태를 사용하여 정보를 저장하고 처리한다.
            - 즉 컴퓨터는 이러한 트랜지스터의 전기 신호를 ON(1)과 OFF(0) 상태로 사용하여 정보를 저장하고 처리하는데, 이는 컴퓨터에서 가장 정확성이 높고 효율적이라고 할 수 있다.
- 프로그래밍 언어?
    - 컴퓨터가 이해할 수 있는 언어로써 사람과 컴퓨터가 소통하기 위해 필요한 언어이다.

[컴파일러]

- 사람이 작성한 소스를 기계어로 변환시켜 주는 소프트웨어이다.

[고급언어와 저급언어] 고급 언어: 일반적인 프로그래밍 언어 - C / C++ / Java 등 저급 언어: 어셈블리어 - 사람이 쉽게 이해할 수 없어 배우기가 어렵다.

[프로그래밍 언어의 실행]



- 프로그래밍 언어로 작성된 내용은 Source(소스)이다.
- 이 소스는 **컴파일러**를 통해 기계어로 변환된 후 실행된다.
- 컴파일러?
  - 사람이 작성한 소스를 기계어로 변환시켜주는 소프트웨어다.
- 즉 순서는 다음과 같다.
  1. 사람이 소스를 작성
  2. 작성한 소스를 컴파일러(컴파일 과정)를 통해 컴퓨터가 이해할 수 있는 0과 1로 이루어진 기계어로 변경
  3. 실행

[Java Virtual Machine]

- 영어권에서는 컴퓨터를 흔히 기계(Machine)라고 부르기 때문에 '자바를 실행시키는 가상의 기계'라고 해서 JVM 이라는 용어가 탄생되었다.
- 자바 프로그램은 완전한 기계어가 아닌 중간 단계의 바이트 코드이다.
  - 이 때문에 이것을 해석하고 실행할 수 있는 가상의 운영체제가 필요한데 이게 '자바 가상 기계'이다.
- Java Virtual Machine은 실 운영체제를 대신해서 자바 프로그램을 실행하는 가상의 운영체제 역할을 한다.
- 운영체제별로 프로그램을 실행하고 관리하는 방법이 다르기 때문에
  - 운영체제별로 자바 프로그램을 별도로 개발하는 것보다 자바 운영체제와 자바 프로그램을 중계하는 JVM을 두어 자바 프로그램이 운영체제와 상관없이 동일하게 실행되도록 설계하였다.
    - 따라서 개발자는 운영체제와 상관없이 자바프로그램을 개발할 수 있다.
- JVM은 운영체제에 맞게 JDK 또는 JRE를 설치하면 자동적으로 설치된다.
  - 즉, Windows 용 Mac 용 등과 같이 운영체제에 따라 설치하면 운영체제에 맞는 JVM이 설치된다.

(실행 단계)

- 자바 프로그램은 확장자가 .java 인 파일을 작성하는 것부터 시작된다.(소스 파일)
- 이 소스 파일을 컴파일러(javac.exe)로 컴파일하면 확장자가 .class인 바이트 코드 파일이 생성
- 바이트 코드 파일은 JVM 구동 명령어(javac.exe)에 의해 JVM 에서 해석되고 해당 운영체제에서 실행할 수 있는 기계어로 변역 후 실행된다.

(장단점)

- 자바의 가장 큰 장점 중 하나 'Write once, run anywhere (한 번 작성하면 어디서든 실행된다.)'는 매우 매력적임에는 틀림 없지만
    - 한 번의 컴파일링으로 실행 가능한 기계어가 만들어지지 않고 JVM에 의해 기계어로 변역되고 실행되기 때문에
        - C/C++의 컴파일 단계에서 만들어지는 완전한 기계어보다는 속도가 느리다는 단점을 가지고 있다.
            - 그러나 기계어로 빠르게 변환해주는 JVM 내부의 최적화된 JIT 컴파일러를 통해서 속도의 격차는 많이 줄어들었다.

[JDK, JRE]

- JDK = 자바 개발 키트 = Java Development Kit
    - 자바 프로그램 개발 도구로써 개발을 위한 Class, 컴파일러, 자바 가상 기계, API, 실행 및 배포 도구를 포함하여 개발을 위한 전반적인 환경을 제공하는 것이다.
        - [유료] Oracle JDK / [무료] Open JDK
            - 무료: 개인이나 소규모 기업에서 사용하기에 충분한 안전성을 가지고 있다.
- JRE = 자바 실행 환경 = Java Runtime Environment
    - JRE는 자바 가상 기계와 라이브러리 API만 포함되어 있는 것으로써, 이미 개발된 프로그램만 실행한다면 JRE만 설치하면 된다.

[외부 라이브러리]

- External Libraries
- System 이라는 것도 자바에서 제공하는 파일이다.
- System.out,.println("안녕");
- 인텔리J 가 선택한 JDK에 포함되어 있다. (Project Settings > Project > SDK)
- JDK 대부분은 거의 공일하게 표준 라이브러리를 제공하고 있다.

[주석]

- 주석은 프로그램 실행과 상관없이 코드에 설명을 붙인 것이다.
- 주석은 컴파일 과정에서 무시된다.
- 실행문만 바이트 코드로 번역된다.
- 그래서 주석을 많이 사용하더라도 프로그램 크기가 커지는 것은 아니다.
- 복잡한 코드일수록 주석을 달면 전체 코드를 이해하기 쉽고, 수정이 용이하다.

```
/*
  주석입니다. 1
  주석입니다. 2
  System.out.println("출력해 주세요.");
  System.out.println("출력해 주세요. 2");
  System.out.pritln("헬로우");
*/

// 주석입니다. 1
// 주석입니다. 2
```

```
// System.out.println("출력해 주세요.");
// System.out.println("출력해 주세요. 2");
// System.out.pritln("헬로우");
```

- 아래 주석을 더 권장: 실제 소스에서 주석 해제가 필요한 경우 해당 라인만 쉽게 해제할 수 있기 때문