한 시대를 풍미한 언어
자바에 대한 이야기들

이윤성

JRAS
Java Reminder And Starter

"일반적인 가전제품 또는 휴대용 기기 간의
커뮤니케이션을 가능케 해주는 기술이 없을까?"

이미지 출처_투인 http://www.2uin.com/wp-content/uploads/2015/08/news2_0.png

"일반적인 가전제품 또는 휴대용 기기 간의
커뮤니케이션을 가능케 해주는 기술이 없을까?"

1991

Sun Microsystems

"일반적인 가전제품 또는 휴대용 기기 간의
커뮤니케이션을 가능케 해주는 기술이 없을까?"

1991

Sun Microsystems

Green Team

이미지 출처_Tech-Insider http://tech-insider.org/java/research/1998/images/GreenBBQ.jpg

"일반적인 가전제품 또는 휴대용 기기 간의
커뮤니케이션을 가능케 해주는 기술이 없을까?"

1991

Sun Microsystems

Green Team

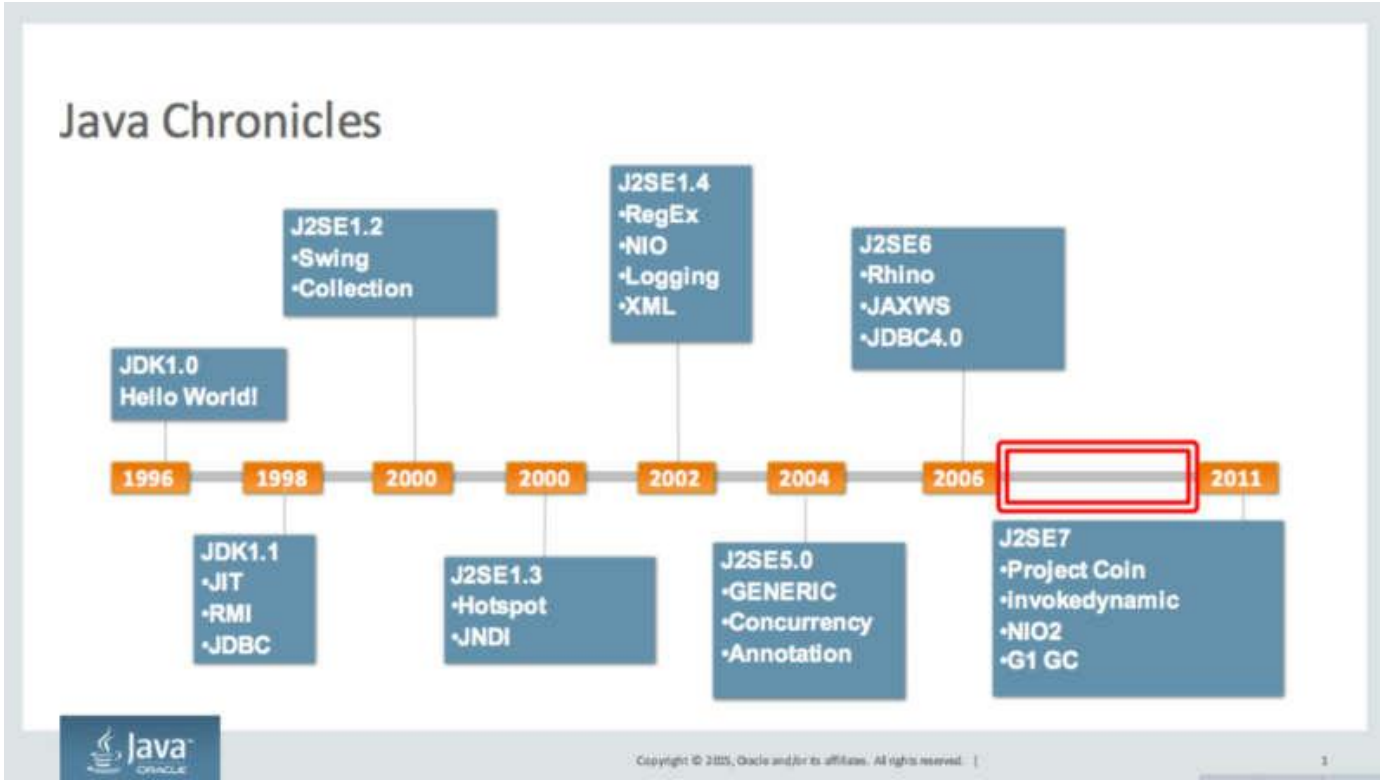
Code Name : Oak

휴대용 장치 + 운영체계 + Oak로 작성된 애플리케이션을 만들 수 있도록 하자

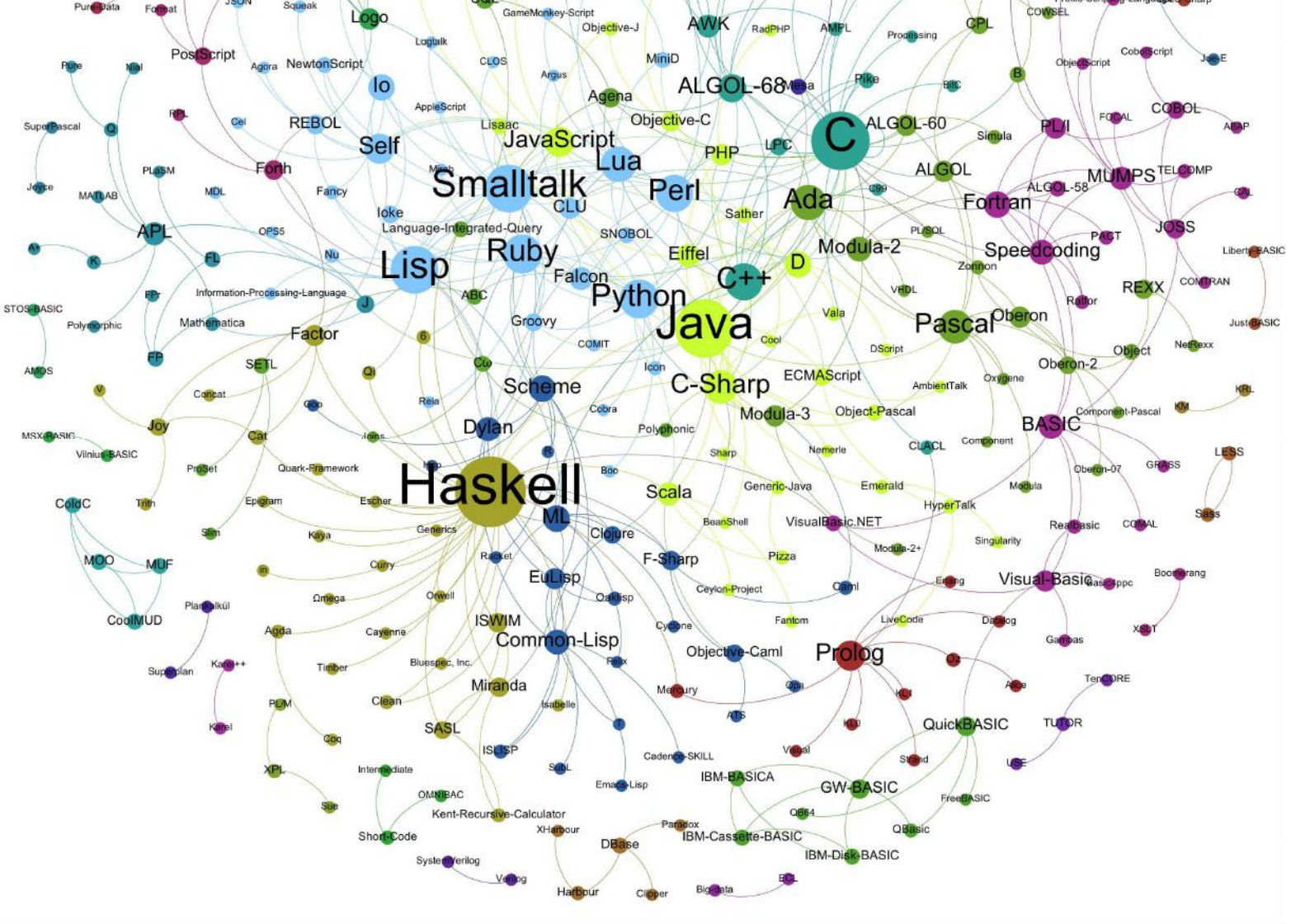
다음의 요구사항을 해결할 것

첫째, 절대 충돌이나 다운되지 않도록 할 것
둘째, CPU 및 운영체제에 독립적일 것
셋째, 네트워크 통신에 적합해야 한다


TV 셋톱박스 시장 진출을 목표로 자회사 FirstPerson 설립
언어 이름을 Oak에서 Java로 변경
셋톱박스 시장 진출 실패
당시 인지도가 낮았던 Sun은 그래픽 워크스테이션의 강자
실리콘 그래픽스(SGI)에 밀려 타임 워너 사와의 계약을 실패함
자바 1.0 발표, 웹 시장 진출
웹의 가능성을 확인하고 1994년부터 '웹 러너'라는 웹브라우저 개발
1995년 핫자바(HotJava)라는 이름으로 TED에서 첫 발표.
(애플릿 기반의 다이나믹한 웹 환경으로 사람들의 이목 집중)
Write Once, Run Anywhere 라는 슬로건과 함께 자바 1.0 발표
이후 프로그래밍 언어 점유율 27% 가까이를
차지하며 프로그래밍 언어의 강자로 급부상.
주로 웹에서 많이 사용됨
제임스 고슬링, 패트릭 노튼을 필두로 한
GreenTeam 결성,
Oak 프로그래밍 언어 개발에 착수
Oak 완성
1991
1995
~2000



이미지 출처_IMASO https://img.imaso.co.kr/2015/07/01/5595dB_150624_java_chronicle-794x448.jpg



"왜 당시 프로그래머들은 자바를 택했을까?"

이미지 출처_나무위키 https://newcdn.namu.wiki/r/http%3A%2F%2Fi.imgur.com%2FbMOYMTM.jpg

프로그래밍 언어는

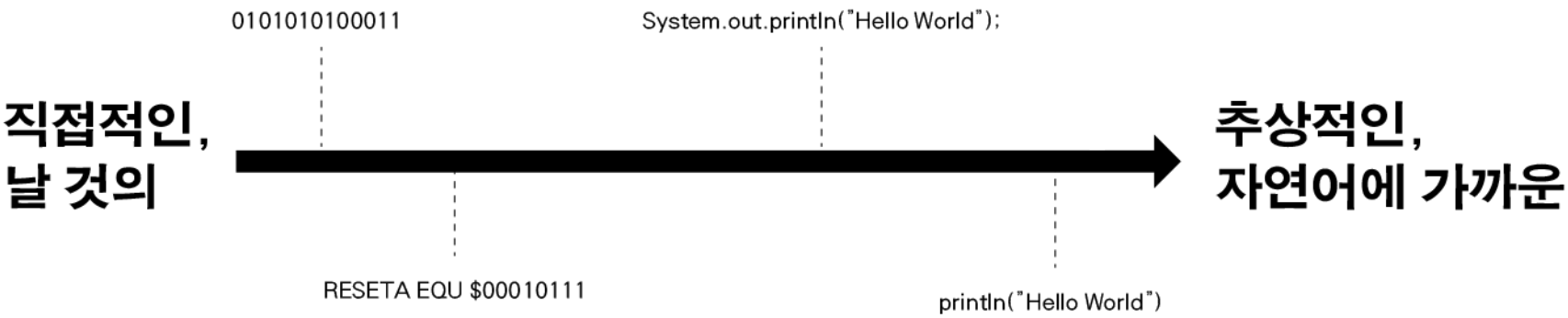
추상 수준을 높여

행사코드 boilerplate code 를 줄이는 방향으로 발전해왔다

프로그래밍 언어는

시간이 흐를수록

본질적인 내용(실제 사용자가 만들고자 하는 것)에 집중해 작성할 수 있게 되었다



행사코드 boilerplate code

지루하게 반복되는 코드,

실제 수행되고자 하는 것과 상관없는 **형식적인** 코드

> “복잡성은 **비본질적인 복잡성**과 **본질적인 복잡성**으로 구분할 수 있다. 프로그래머들은 언제나 본질적인 내용에 집중해야 한다. 상위 수준의 언어가 성취하는 것은 무엇인가? 그것은 비본질적인 복잡성으로부터 프로그램을 자유롭게 만드는 것이다. (중략) 구체적인 기계어 프로그램은 비트, 레지스터, 채널, 디스크와 같은 대상을 염두에 두어야 한다. **상위 수준의 프로그래밍 언어는 이와 같은 구조물들을 추상적인 프로그램 안에서 완전히 사라지도록 만들어 주는 것이다.**”

1986 Fred Brooks

그런 의미에서 자바는

90년대의 획기적인 프로그래밍 언어였다

행사 코드를 줄인 대표적인 기능 GC Garbage Collector

프로그래머가 메모리에 대해 생각하지 않아도 되도록 하여
만들고자 하는 프로그램 자체에만 집중할 수 있도록 함

(C++에서 빈번히 발생하던 메모리 문제를 해결)

1990

GC Garbage Collection

메모리를 관리하는 방법 중 하나

1 자바는 프로그래머가 직접 메모리를 할당하거나 해제하지 않고,
프로그래밍 언어 차원에서 자동으로 메모리를 할당/해제함.

2 메모리 해제는 가비지 컬렉터(Garbage Collector)가 정기적으로 수행.

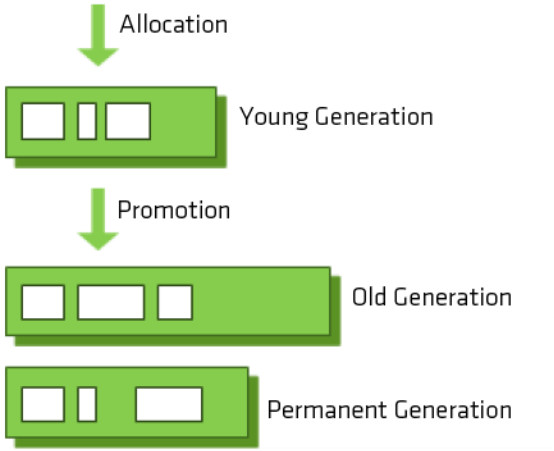
3 프로그래밍이 편하다는 이점이 있으나, 효과적인 메모리 관리가 어렵기 때문에
결국 일정 규모 이상의 어플리케이션 구현시 가비지 컬렉터 및 메모리에 대한 공부가 필요해짐.

4 GC를 수행하는 방식은 구현 알고리즘에 따라 나뉘며
JDK 7을 기준으로 총 5가지 방식이 존재함.

- Serial GC
- Parallel GC
- Parallel Compacting GC
- Concurrent Mark & Sweep GC
- G1(Garbage First) GC

GC Garbage Collection

메모리를 관리하는 방법 중 하나

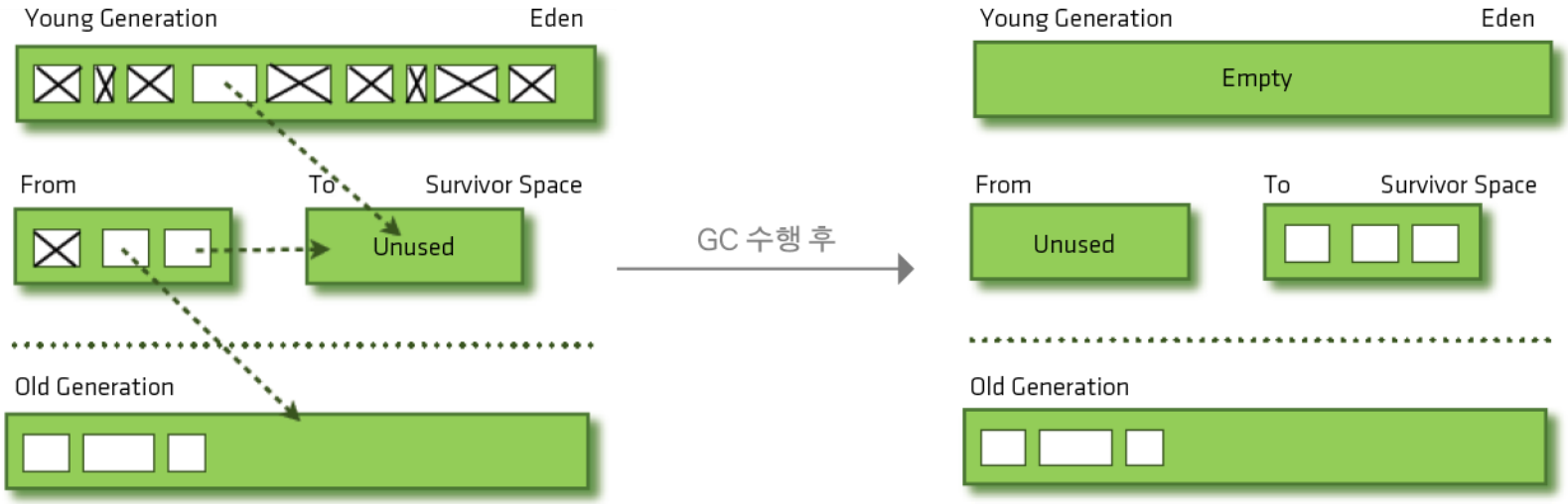


가비지 컬렉터 영역별 데이터 흐름

이미지 출처_네이버 D2 http://d2.naver.com/helloworld/1329

GC Garbage Collection

메모리를 관리하는 방법 중 하나



GC 전과 후 비교

이미지 출처_네이버 D2 http://d2.naver.com/helloworld/1329

JVM Java Virtual Machine

자바를 모든 플랫폼으로부터 독립적일 수 있도록 만든 가상 에뮬레이터.

1

자바로 만든 모든 코드는 컴파일시 기계어로 만들어지지 않고,

JVM에서 해석 가능한 바이트 코드로 만들어짐. (hello.java -> hello.class)

2

자바 프로그램을 실행하는 시점에 코드를 읽어 수행함.

OS 내부에서 별도의 에뮬레이터가 프로그램을 수행하는 것이므로, 상대적으로 퍼포먼스가 떨어짐.

최근에는 실행 시점에 인터프리터 방식이 아닌 JIT(Just In Time) 방식으로 동작하게 되어 속도가 많이 향상됨.

Write Once,

3

모든 코드를 바이트 코드 형태로 배포하여 사용할 수 있다는 점은

기업 입장에서는 사내에서 작성된 코드 보안 측면에서 용이하기 때문에 선호하는 편.

Run Anywhere

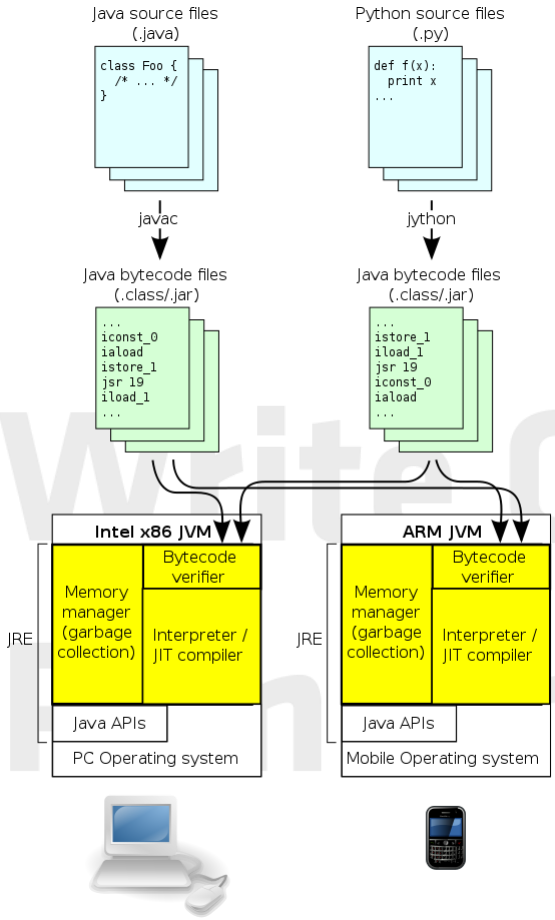
4

20년에 가까운 시간 동안 최적화 되어왔기 때문에, 다른 가상 머신들을 압도하는 퍼포먼스/안정성을 확보함.

최근 발표되는 언어들은 JVM에서 실행 가능한 것을 하나의 사양으로 가지는 경우가 많음. (예 : Clojure, Scala)

JVM Java Virtual Machine

자바를 모든 플랫폼으로부터 독립적일 수 있도록 만든 가상 에뮬레이터.



이미지 출처__위키미디어 https://commons.wikimedia.org/wiki/File:Java_virtual_machine_architecture.svg?uselang=ko

그 외의 자바가 가진 특징들

객체지향적 프로그래밍 언어

강력한 제네릭 타입 기능 지원

고수준의 추상화가 이루어진 언어 중 비교적 높은 생산성을 가짐

높은 안정성

.

.

.

못다한 이야기들

프로그래밍 언어의 생명 주기와 커뮤니티의 관계

모던 자바의 새로운 시작, 자바 8의 등장

함수형 프로그래밍이 주목 받는 이유

.

.

.

참고서적 / 자료

폴리글랏 프로그래밍 __ 임백준 저

네이버 D2 ‘Java Gabage Collection; __ http://d2.naver.com/helloworld/1329

IMASO ‘자바는 어떻게 자바가 됐나’ __ https://www.imaso.co.kr/news/article_view.php?article_idx=20150701094406

자바(Java) 탄생의 역사와 썬마이크로시스템즈 __ http://www.emh.co.kr/content.pl?sun_and_java

JVM Overview ‘자바의 역사와 철학’ __ http://stunstun.tistory.com/200

위키백과 __ https://ko.wikipedia.org

나무위키 __ https://namu.wiki

이윤성(sharlean@naver.com)