자바 기초 언어 문법-1

김정수

ⓒSaebyeol Yu. Saebyeol's PowerPoint


Q.
질문을 입력하세요

A Table of Contents.

1강 자바 시작하기

2강 변수와 타입

3강 연산자

ⓒSaebyeol Yu. Saebyeol's PowerPoint

1. 자바 시작하기

프로그래밍 언어란?

소스

```java
class Hello {
    public static void main(String[] args){
        System.out.println("Hello World!");
    }
}
```

컴파일러

- 고급 언어
  - 사람이 쉽게 이해할 수 있는 언어
  - ex. C, JAVA, PYTHON
- 저급 언어
  - 기계어에 가까운 언어
  - ex. 어셈블리어

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

자바의 특징

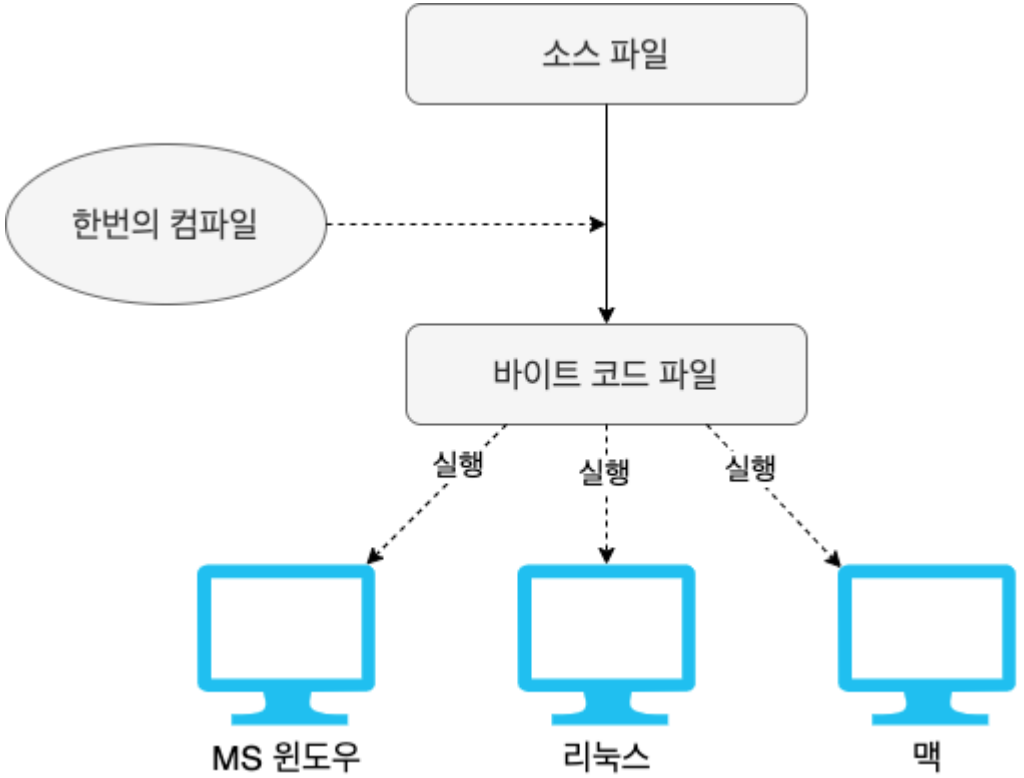
- 이식성이 높은 언어이다.
- 객체 지향 언어이다.
- 함수적 스타일 코딩을 지원한다.
- 메모리를 자동으로 관리한다.
- 다양한 애플리케이션을 개발할 수 있다.
- 멀티 스레드를 쉽게 구현할 수 있다.
- 동적 로딩을 지원한다.
- 막강한 오픈소스 라이브러리가 풍부하다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

이식성이 높은 언어이다

- 이식성이란 서로 다른 실행 환경을 가진 시스템 간에 프로그램을 옮겨 실행할 수 있는 것.



ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

객체 지향 언어이다

- 객체 지향의 핵심은 적절한 책임을 수행하는 역할 간의 유연하고 견고한 협력 관계를 구축하는 것이다.
- 객체지향의 중심에는 클래스가 아니라 객체가 위치하며, 중요한 것은 클래스들의 정적인 관계가 아니라 메시지를 주고받는 객체들의 동적관계다.
- 객체의 역할, 책임, 협력에 집중하라.

함수적 스타일 코딩을 지원한다.

- 함수적 프로그래밍은 대용량 데이터의 병렬처리 그리고 이벤트 지향 프로그래밍에 적합하다.
- 함수형의 특징인 데이터의 불변성으로 인한 스레드 안정성을 의미하는 것이라고 이해했습니다.
- 실제로 페이스북의 경우 함수형 언어(Elrang)를 도입하여 엄청난 부하를 커버했다는 예시가 있다.
- JAVA8부터 함수적 프로그래밍을 위해서 람다식을 지원한다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

메모리를 자동으로 관리한다.

- 개발자가 직접 메모리에 접근할 수 없도록 설계되었으며, 메모리는 자바가 직접 관리한다.
- 객체 생성시 자동적으로 메모리 영역을 찾아서 할당하고, 사용하지 않는 객체는 **GC**를 실행시켜 제거한다.

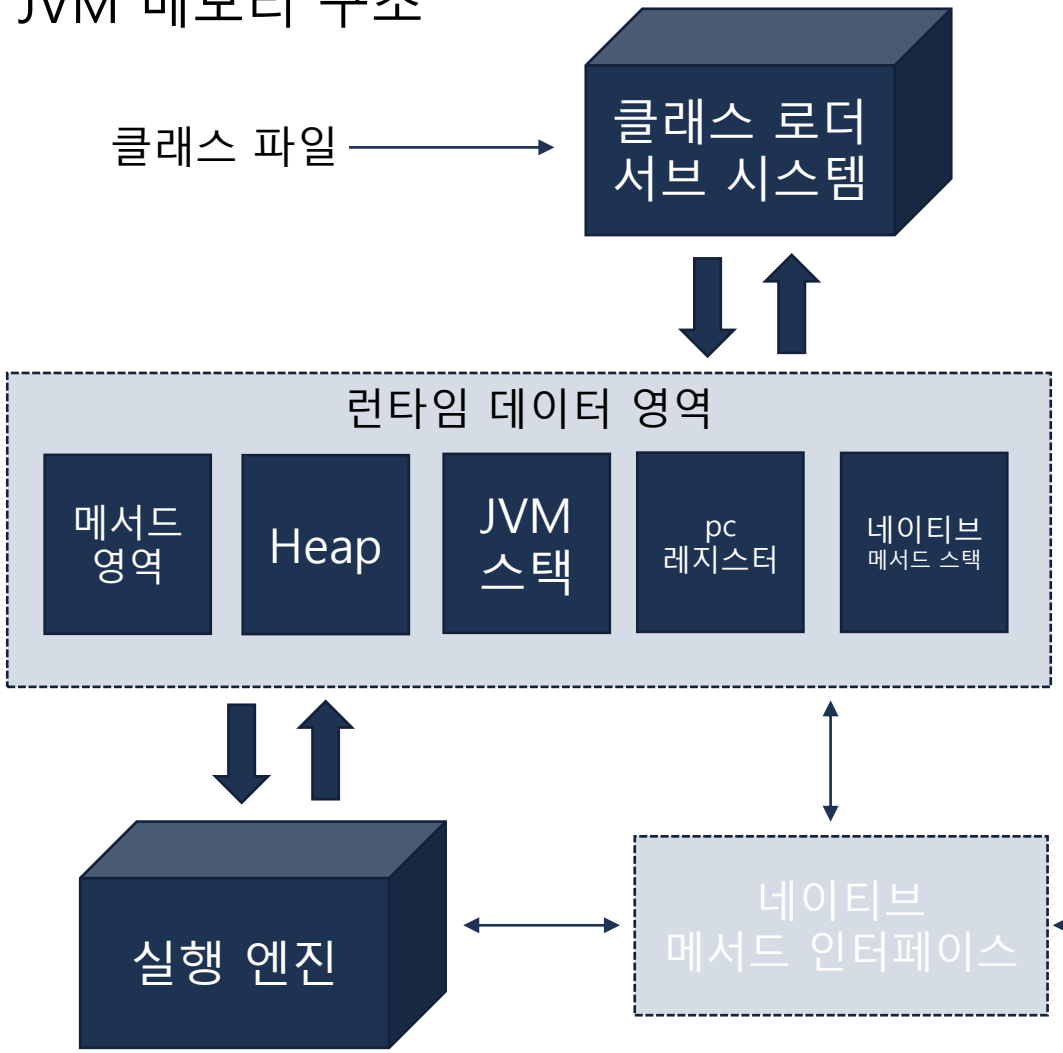
면접질문: GC가 무엇인지, 필요한 이유는 무엇인지, 동작방식에 대해 설명해주세요.

- GC는 JVM의 Heap 영역에서 더 이상 사용하지 않는 객체를 효과적으로 처리하는 작업
- GC가 필요한 이유는 자바는 개발자가 직접 메모리를 접근할 수 없기 때문에 객체를 제거하는 작업이 필요합니다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

JVM 메모리 구조



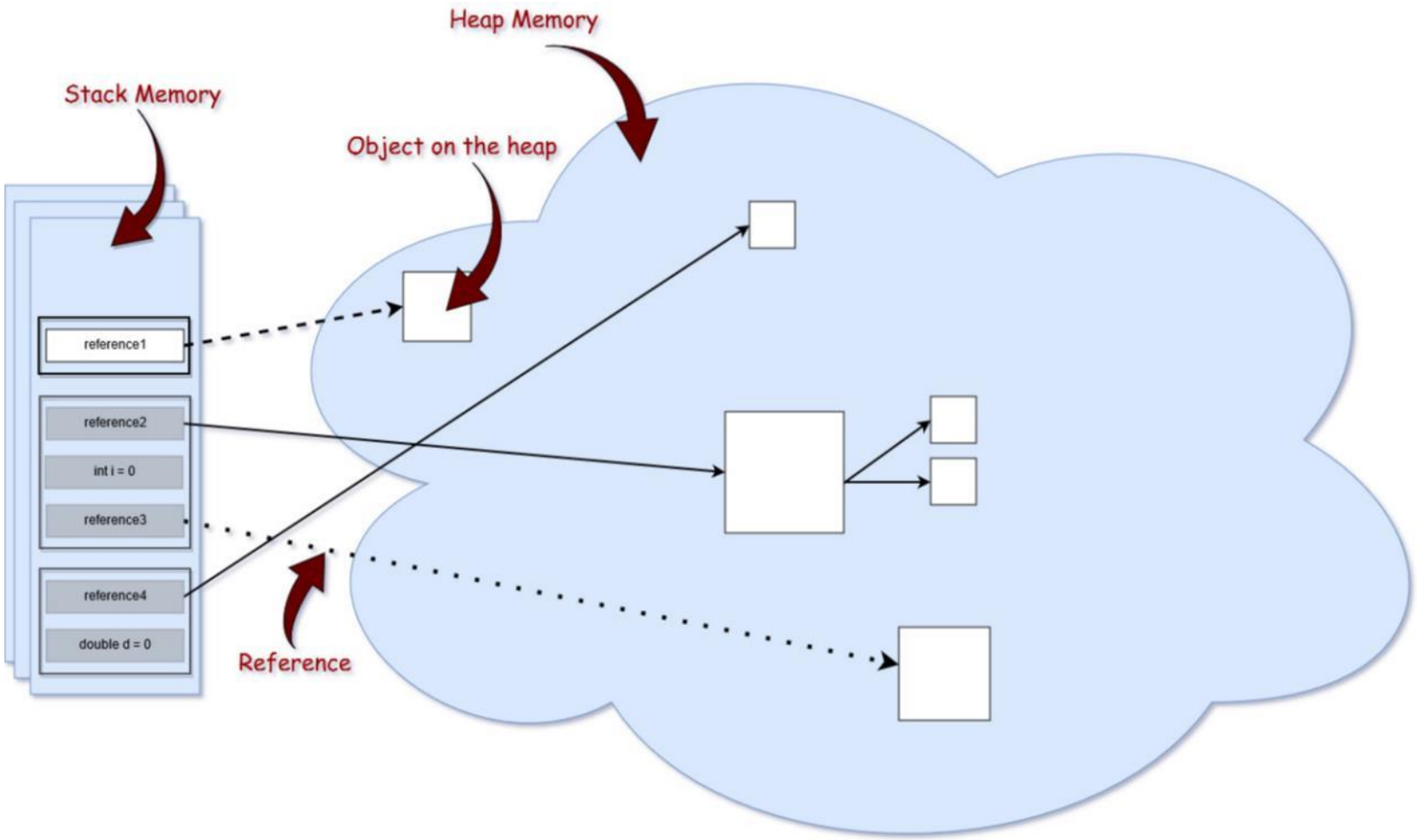
- 클래스 로더 서브 시스템: 클래스나 인터페이스를 JVM으로 로딩하는 기능
- 실행 엔진: 로딩된 클래스의 메서드들에 포함되어 있는 모든 인스트럭션 정보를 실행.
- Heap: new로 생성된 객체와 배열이 생성되는 영역. 스레드에서 공유되는 메모리
- 메서드영역: 클래스의 필드정보, 메서드 정보, constant pool, static변수, final class 변수

- JVM 스택: 지역변수, 임시결과, 파라미터, 리턴값 등 메서드 수행과 리턴에 관련된 정보 저장
- pc 레지스터: 네이티브한 코드를 제외한 모든 자바 코드들이 수행할 때 JVM 인스트럭션 주소를 pc 레지스터에 보관
- 네이티브 메서드 스택: 자바가 아닌 다른 언어로 된 코드를 실행할 때의 스택 정보 저장

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

JVM 메모리 구조



- JVM 스택: 지역변수, 임시결과, 파라미터, 리턴값 등 메서드 수행과 리턴에 관련된 정보 저장
- Heap: new로 생성된 객체와 배열이 생성되는 영역. 스레드에서 공유되는 메모리

ⓒSaebyeol Yu. Saebyeol's PowerPoint

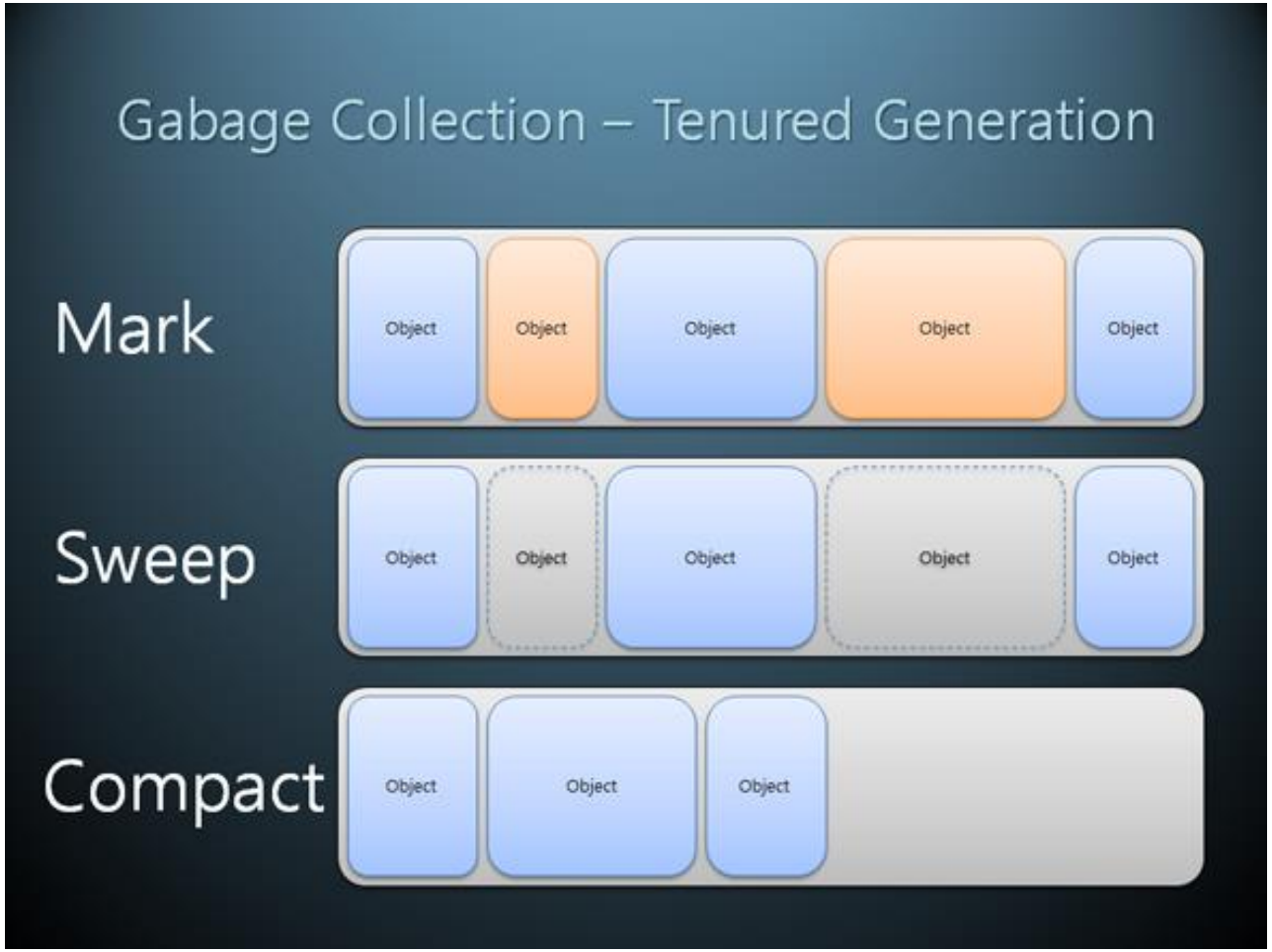
자바란?

GC의 동작 구조(Mark-Sweep-Compact)

- Heap 영역에 존재하는 객체들에 대해 접근가능한지 확인
- GC Root부터 시작해 참조값을 따라가며 접근 가능한 객체들에 **Mark**한다.
- **Mark** 되지 않은 객체(접근할 수 없는 객체)는 제거(**Sweep**) 대상이 되고, 제거 대상을 제거한다.
- 접근할 수 있는 객체를 한 곳으로 모은다.

특징

- 의도적으로 GC를 실행해야 한다
- 어플리케이션과 GC 실행이 병행되어야 한다.

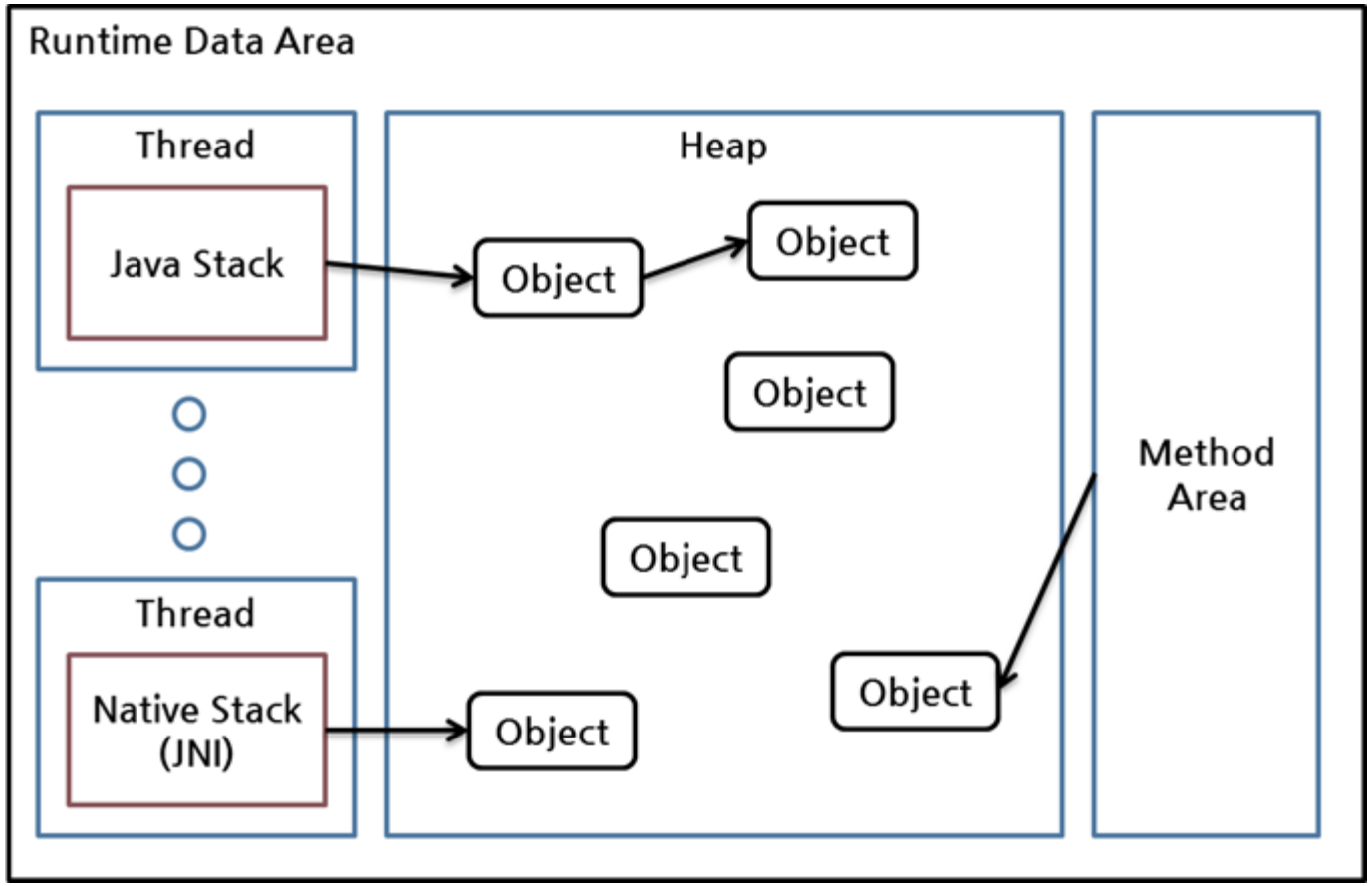


ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

GC Root가 될 수 있는 대상

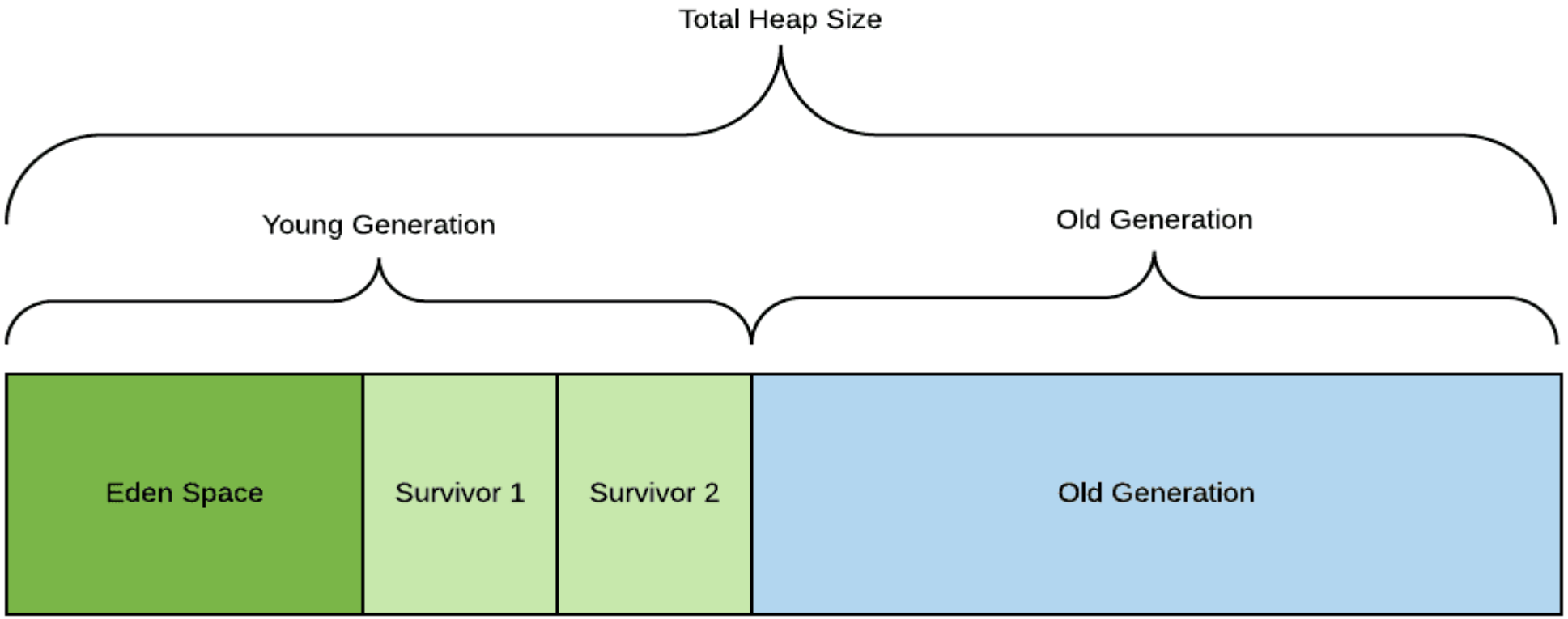
- JVM 메모리의 Stack 영역에 존재하는 참조 변수
- Method Area의 static 데이터
- JNI에 의해 생성된 객체들



ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

GC를 실행시킬 기준



ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

Stop The World

- GC 실행을 위해 GC를 실행하는 쓰레드를 제외한 나머지 쓰레드를 멈추는 것
- 어플리케이션 실행을 멈추는 것이기에 Stop The World 시간을 줄이는 것이 GC 튜닝의 목표

GC 방식

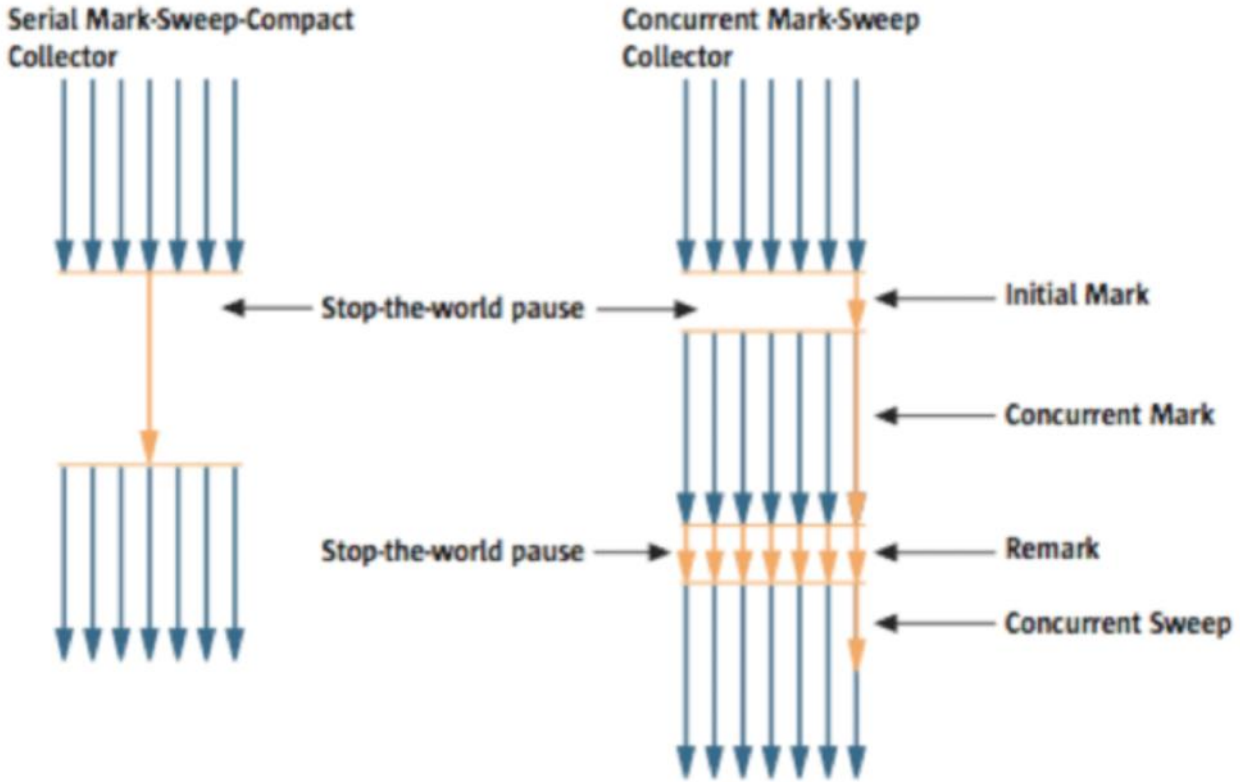
- Serial Collector
- Parallel Collector
- Parallel Compacting Collector
- Concurrent Mark-Sweep Collector
- Garbage First Collector

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

CMS (Concurrent Mark-Sweep) Collector

- Initial Mark(STW):
  GC root가 참조하는 객체만 마킹
- Concurrent Mark:
  참조하는 객체를 따라가며,
  지속적으로 마킹
- Remark(STW): 전 단계에서 변경된 객체에 대해 다시 표시.
- Concurrent Sweep: 접근할 수 없는 객체 제거.
- Young, Old 모두 Compaction X

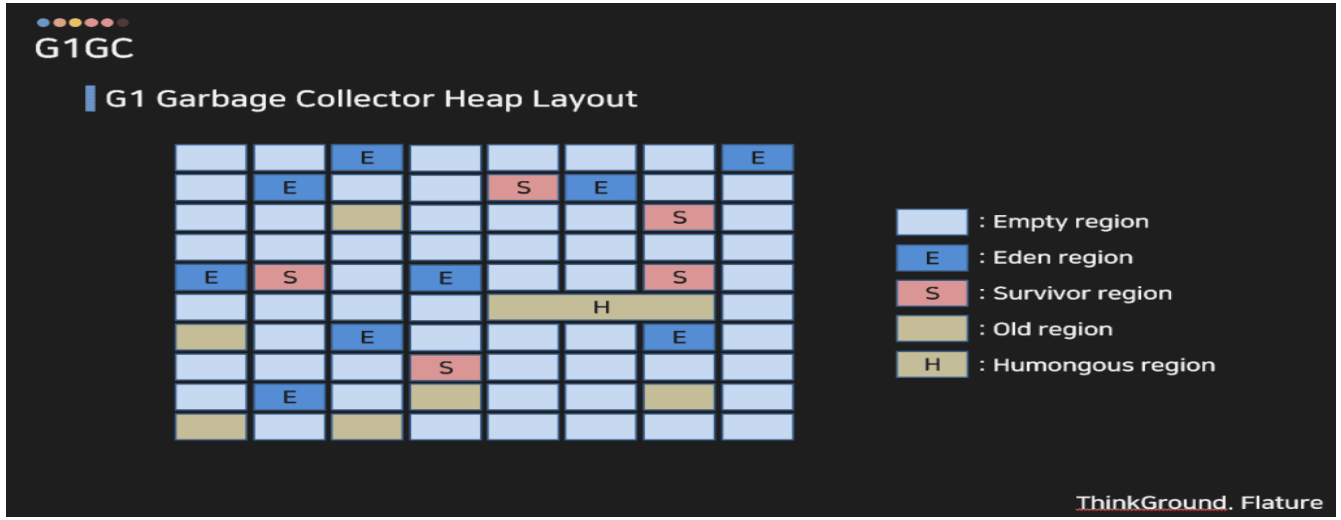


ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

G1(Garbage First) Collector

- Young과 Old를 물리적으로 구분X
- Humongous region은 객체의 크기가 큰 경우 사용하는 영역(50%)
- 런타임에 따라 영역별 region개수조절



ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

G1 Collector의 동작 단계(Old)

1.Initial Mark : Old Region에 존재하는 객체들이 참조하는 Survivor Region을 찾는다(STW)

2.Root Region Scan : 위에서 찾은 Survivor 객체들에 대한 스캔 작업을 실시한다. Young GC 전에 수행

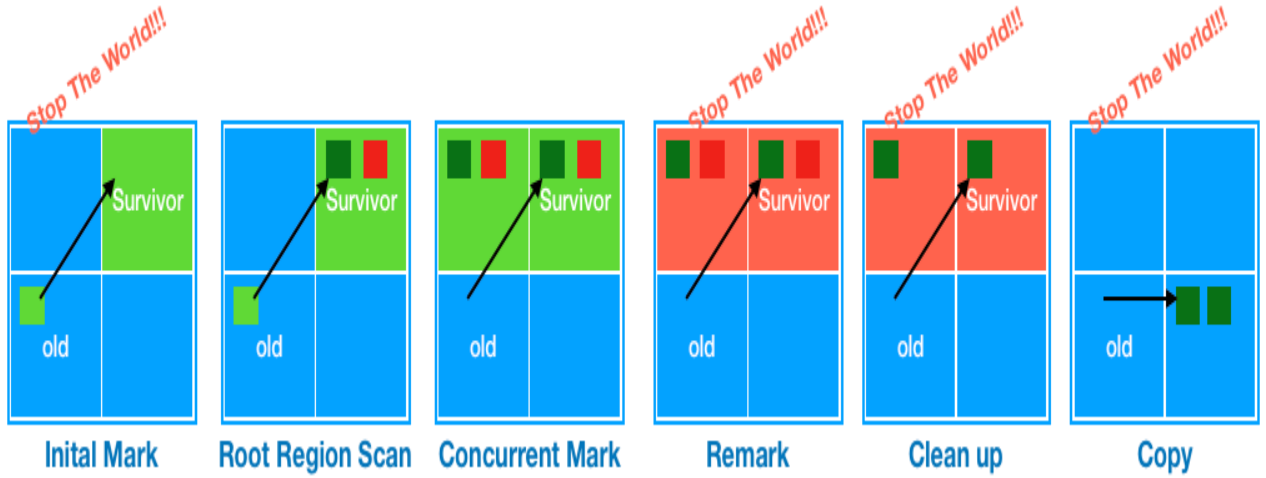
3.Concurrent Mark : 전체 Heap의 scan 작업을 실시하고, GC 대상 객체가 발견되지 않은 Region은 이후 단계를 제외한다. Young GC수행시 잠시 멈춘다.

4.Remark : 애플리케이션을 멈추고(STW) 최종적으로 GC 대상에서 제외할 객체를 식별한다

5.Cleanup : 애플리케이션을 멈추고(STW) 살아있는 객체가 가장 적은 Region에 대한 미사용 객체를 제거한다

6.Copy : GC 대상의 Region이었지만, Cleanup 과정에서 완전히 비워지지 않은 Region의 살아남은 객체들을 새로운 Region(Available/Unused) Region에복사하여 Compaction을 수행한다

7.살아있는 객체가 아주 적은 Old 영역에 대해 [GC pause(mixed)] 를 로그로 표시하고, Young GC가 이루어질 때 수집되도록 한다



ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

동적 로딩을 지원한다

- 애플리케이션이 실행될 때 모든 객체가 생성되는 것이 아니라 객체가 필요한 시점에 클래스를 동적로딩해서 객체생성.
- 개발 완료 후에 유지보수가 필요할 경우 해당 클래스만 수정하면 된다.

멀티 스레드를 쉽게 구현할 수 있다.

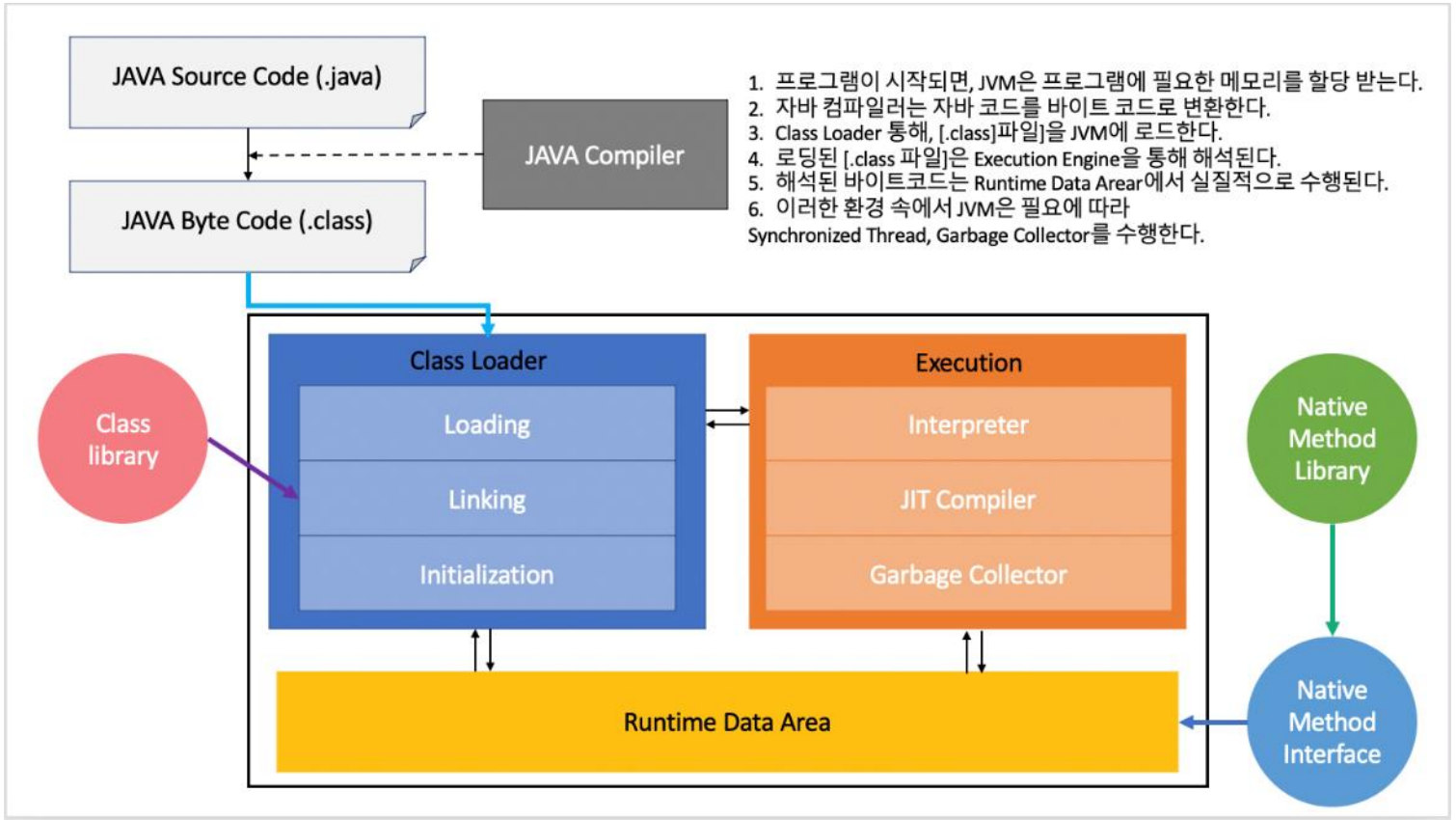
- Thread 클래스를 상속받는 방법
- Runnable 인터페이스를 구현하는 방법.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

JVM

- Loading
  - .class 파일을 읽어 아래 내용들을 메소드 영역에 저장.
    - 클래스와 부모 클래스 정보
    - class파일이 Class, Interface 관련여부
    - 변수나 메소드 정보
- Linking
  - .class 파일의 정확성 보장
  - 적절한 포맷인지 유효한 컴파일러에 의해 생성되었는지 확인
  - 클래스가 필요한 메모리 할당.
  - 레퍼런스 연결
- Initialization
  - static 필드가 코드에 명시된 값으로 초기화

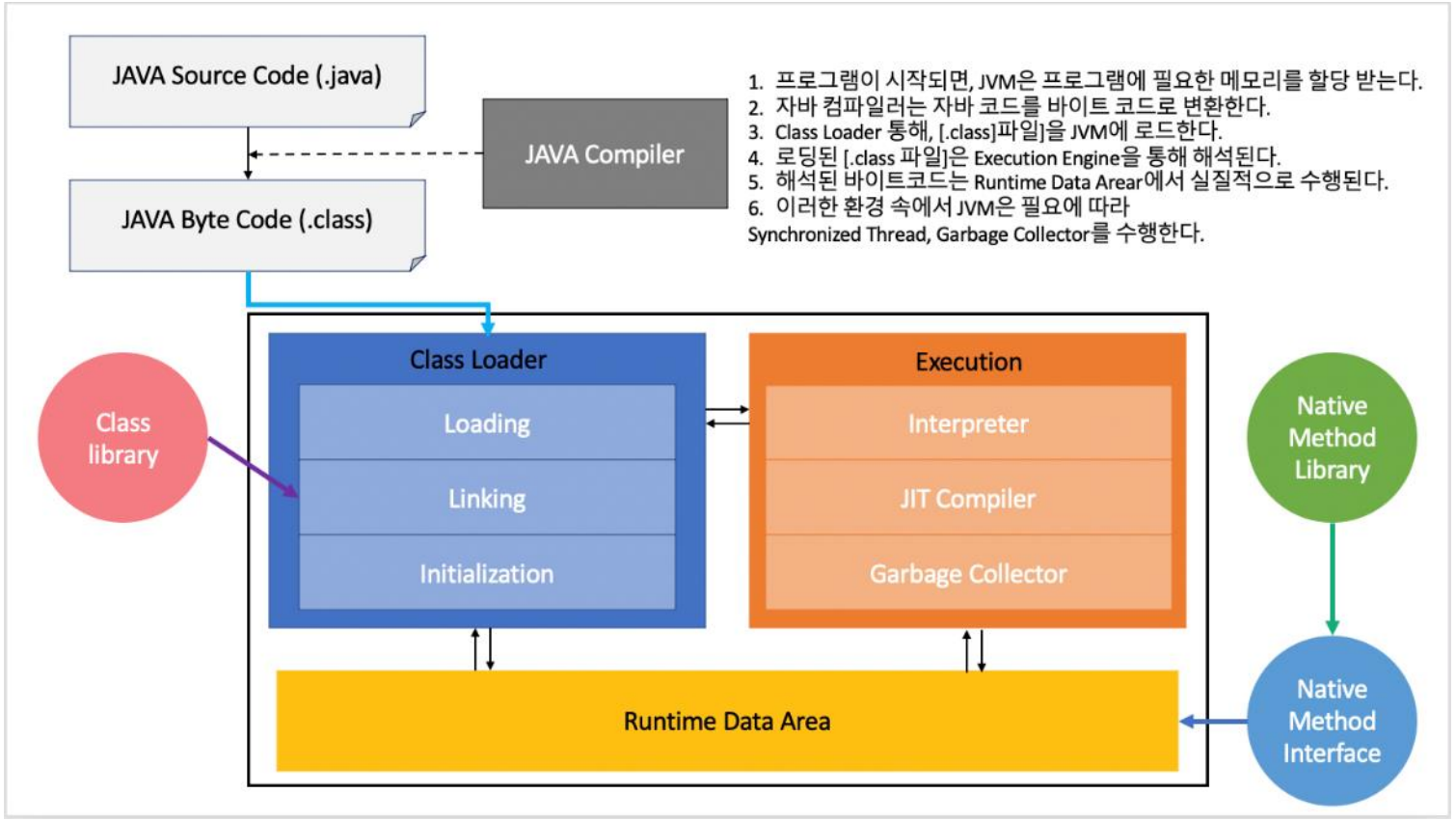


ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

JVM

- Interpreter
  - 바이트 코드를 한줄씩 읽으면서 수행
- JIT(Just In Time) Compiler
  - 자주 쓰는 코드는 컴파일하여 나중에 메서드 호출시 컴파일된 코드 실행
  - 컴파일 기준
    - 수행카운터: 메서드 시작시 증가
    - 백에지 카운터: 메서드 루프여부
- 자바는 인터프리터, 컴파일 방식 병행.
- JVM(java.exe)은 운영체제에 종속적
- Compiler(javac.exe)
- Hello.java 실행시
  - javac Hello.java
  - java Hello
    - .class를 붙이지 않는다.



ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

자바 개발 환경 구축

- JDK: JVM+라이브러리 API+컴파일러 등의 개발도구
- JRE: JVM+라이브러리API

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

주석

주석은 컴파일과정에서 제외되기 때문에 많이 작성해도 문제 없다.

주석에 들어가면 좋은 내용

- 작성자
- 작성일과 시간
- 프로그램 버전과 변경 이력
- 주요 코드에 대한 설명

주석은 문자열 내부를 제외한 모든 곳에 작성 가능.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

자바란?

이클립스

워크스페이스 하위 폴더인 .metadata에 이클립스 변경값들이 저장.
이후에 워크스페이스에 접속시 동일한 작업환경을 복원하기 위함.

자바 소스 파일(~.java)는 src 디렉토리에, 바이트 코드 파일(.class)는 bin 디렉토리에 저장.
이클립스의 Package Explorer에는 src 디렉토리만 보이고 윈도우 탐색기를 통해 bin 디렉토리 확인 가능.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

2. 변수와 타입

변수와 타입

변수

- 변수는 하나의 값을 저장할 수 있는 공간으로 한 가지 타입의 값만 저장가능.
- 변수선언시 타입과 이름을 결정해야 한다.
- 변수 이름은 메모리 주소에 붙인 이름으로 변수 이름으로 메모리 주소에 접근한다.
- 변수이름은 camelCase를 사용한다.
- 변수는 초기화되어야 읽을 수 있다.
- 메소드 블록 내에서 선언된 변수를 로컬 변수라고 하며 메소드 실행이 끝나면 메모리에서 자동으로 없어진다.
- 변수는 선언된 블록 내에서만 사용 가능하다.
- 어떤 진수로 입력해도 동일한 값이 2진수로 저장

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

리터럴

- 소스코드에서 직접 입력된 값
- 0으로 시작하는 리터럴은 8진수로 간주
  ex. 02
- 0X, 0x로 시작하는 리터럴은 16진수로 간주
  ex. 0x5
- E, e가 있는 리터럴은 10진수 지수와 가수로 간주
  ex. 0.12E-5 // $0.12X10^{-5}$
- 문자(char)는 '로 묶는다.
- 문자열(String)은 "로 묶는다.
  문자열 내부에 이스케이프 문자 사용 가능

이스케이프 문자	용도	유니코드
'₩t'	수평 탭	0x0009
'₩n'	줄 바꿈	0x000a
'₩r'	리턴	0x000d
'₩"'	"(큰따옴표)	0x0022
'₩''	'(작은따옴표)	0x0027
'₩₩'	₩	0x005c
'₩u16진수'	16진수에 해당하는 유니코드	0x0000 ~ 0xffff

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

데이터 타입

타입		범위	기본값	크기(byte)
논리형	boolean	true, false	FALSE	1
정수형	char	0 ~ 65,535	₩u0000	2
	byte	-127 ~ 128	0	1
	short	-32768 ~ 32767	0	2
	int	-2,147,483,648 ~ 2,147,483,647	0	4
	long	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	0L	8
실수형	float	1.4E-45 ~ 3.4028235E38	0.0f	4
	double	4.9E-324 ~ 1.79769313486231 57E308	0	8

실행 중에 저장할 수 있는 값의 범위를 초과하면 최소값부터 다시 반복저장된다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

데이터 타입(정수형)

char(2byte)

- 자바는 모든 문자를 유니코드로 처리.
- 0 ~ 127까지는 ASCII문자 할당.
- 유니코드는 음수가 없기에 char 타입에 음수값 저장 불가
- 특정 문자의 유니코드를 알면 char변수에 10진수나 16진수로 저장가능.
  ex. char c = '₩u0041'
- char 변수를 int 타입 변수에 저장하면 유니코드를 알 수 있다
  ex. char c = 'A'; int unicode = c;
- 단순히 초기화할 경우 ' '와 같이 공백을 하나 넣어서 초기화해야 한다.

String

- String은 기본 타입이 아닌 클래스 타입이고, String 변수는 참조 변수.
- 그렇기에 문자열을 String 변수에 대입하면 String 객체가 생기고, String 객체의 주소가 변수에 저장된다.
- 빈문자로 초기화 가능.
  ex. String str = "";

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

데이터 타입(정수형)

short(2byte)
- C언어와의 호환을 위해서 사용한다.

int(4byte)
- 자바는 정수연산을 하기 위한 기본타입.
- byte나 short 타입의 변수를 연산하면 결과는 int 타입이 된다.
- 메모리가 부족하지 않다면 정수를 저장할 때 일반적으로 int 타입사용.

long(8byte)
- long 타입 변수를 초기화할 때는 정수값 뒤에 'L'을 붙여야 한다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

데이터 타입(실수형/논리형)

float(32bit=1(부호)+8(지수)+23(가수))

double(64bit=1(부호)+11(지수)+52(가수))

- 자바는 실수 리터럴의 기본 타입을 double로 간주.
- float타입 변수에 저장하려면 리터럴 뒤에 'f', 'F'를 붙여야 한다.

boolean(8bit)

- true/false
- 조건문이나 제어문 실행 흐름 변경에 사용.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

타입 변환(자동)

- 작은 크기를 가진 타입이 큰 크기를 가진 타임에 저장할 때 발생
- 크기의 기준은 사용하는 메모리 크기
- byte(1)<short(2)<int(4)<long(8)<float(4)<double(8)
  - float이 long보다 표현할 수 있는 범위기 크기에 더 큰 타입으로 표현
- 자동 타입 변환 전과 후의 값은 동일하다.
- 정수 타입이 실수 타입으로 변환되는 것은 무조건 자동변환이며 변환 후의 값은 .0이 붙은 실수값이다.
- 자동 타입 변환의 예외는 byte->char.
  - Char에는 음수를 저장할 수 없기 때문에 byte를 char로 자동변환 불가.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

타입 변환(강제)

- 강제적으로 큰 데이터 타입을 작은 데이터 타입으로 쪼개어서 저장하는 것을 강제 타입 변환(Casting)이라고 한다.
- 강제 타입 변환시 캐스팅 연산자()를 사용.
  ex. int value = 230; byte bvalue = (byte) value;
- 강제 타입 변환시 끝에서부터 작은 크기만 저장.
- 실수->정수 변환시 소수점 이하 부분은 버려지고 정수 부분만 저장.
- 강제 타입 변환시 값이 손상되면 안되기 때문에 MIN_VALUE, MAX_VALUE를 이용해 최대값, 최소값을 벗어나는지 확인하고 변환한다.
- 정수->실수 변환시 정수 값(32bit)이 float의 가수(23bit)로 표현할 수 있는 범위를 넘어가면 손실이 일어나기에 double(52bit)로 변환하는 것이 안전하다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

변수와 타입

연산식에서의 타입 변환(자동)

- 서로 다른 타입의 피연산자를 연산할 경우 피연산자 중 크기가 큰 타입으로 자동변환 후 연산.
- 다른 타입 피연산자를 연산시 작은 타입으로 연산해야 한다면 강제 타입변환을 해야 한다.
- 정수 연산은 int 타입이 기본이므로 정수연산의 결과는 int타입이다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

3. 연산자

연산자

연산자, 연산방향/우선순위

- 연산식은 반드시 하나의 값을 산출하며 다른 연산식의 피연산자 위치에 올 수 있다.

연산자 종류	연산자	피연산자 수	산출값 타입	기능 설명
산술	+, -, *, /, %	이항	숫자	사칙연산 및 나머지 계산
부호	+, -	단항	숫자	음수와 양수의 부호
문자열	+	이항	문자열	두 문자열을 연결
대입	=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=	이항	다양	우변의 값을 좌변의 변수에 대입
증감	++, --	단항	숫자	1 만큼 증가/감소
비교	==, !=, >, <, >=, <=, instanceof	이항	boolean	값의 비교
논리	!, &, |, &&, ||	단항 이항	boolean	논리적 NOT, AND, OR 연산
조건	(조건식) ? A : B	삼항	다양	조건식에 따라 A 또는 B 중 하나를 선택
비트	~, &, |, ^	단항 이항	숫자 blooean	비트 NOT, AND, OR, XOR 연산
쉬프트	>>, <<, >>>	이항	숫자	비트를 좌측/우측으로 밀어서 이동

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

연산자, 연산방향/우선순위

1. 단항, 이항, 삼항 연산자 순으로 우선순위를 가진다.
2. 산술, 비교, 논리, 대입 연산자 순으로 우선순위를 가진다.
3. 단항과 대입 연산자를 제외한 모든 연산의 방향은 왼쪽에서 오른쪽이다
4. 복잡한 연산식에는 괄호()를 사용해서 우선 순위를 정해준다.

연산자	연산 방향	우선 순위
증감(++, --), 부호(+, -), 비트(~), 논리(!)	←	높음
산술(*, /, %)	→	
산술(+, -)	→	
쉬프트(<<, >>, >>>)	→	
비교(<, >, <=, >=, instanceof)	→	
비교(==, !=)	→	
논리(&)	→	
논리(^)	→	
논리(|)	→	
논리(&&)	→	
논리(||)	→	
조건(?:)	→	
대입(=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=)	←	낮음

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

단항 연산자

부호 연산자(+, -)

- 부호 연산자의 산출 타입은 int 타입이다.

증감 연산자(++, --)

- ++1과 i=i+1은 바이트 코드가 동일하기에 연산속도 동일하다.

증감 연산자	설명
++x	먼저 피연산자의 값을 1 증가시킨 후에 해당 연산을 진행함
x++	먼저 해당 연산을 수행하고 나서, 피연산자의 값을 1 증가시킴
--x	먼저 피연산자의 값을 1 감소시킨 후에 해당 연산을 진행함
x--	먼저 해당 연산을 수행하고 나서, 피연산자의 값을 1 감소시킴

비트 반전 연산자(~)

- 정수타입의 피연산자에만 사용하며 모든 비트값을 반전시킨다(0->1, 1->0)
- 피연산자는 연산을 수행하기 전에 int 타입으로 변환하고 산출값도 int 타입이다.
- Integer.toBinaryString() 메소드는 정수값을 32비트 이진 문자열로 리턴하며 0인 비트가 있으면 모두 생략하고 나머지 문자열만 리턴한다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

이항 연산자

산술 연산자(+, -, *, /(나누기), %(나머지))

- boolean을 제외한 모든 기본 타입에 사용 가능
- long타입을 제외한 정수 타입 연산은 int 타입으로 산출.
- 피연산자 중 하나라도 실수타입이면 실수 타입으로 산출
- 자바는 리터럴 간의 연산은 타입 변환 없이 해당 타입으로 계산

  ex. char c1 = 'A' + 1; (O) char c3 = c1+1; (컴파일 에러) => char c3 = (char) (c1+1);

올바른 계산을 위한 검토사항

- 값을 미리 검정하기
- 실수 타입 피하기
- 특수값 처리에 신경쓰기

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

오버플로우 탐지

- 산출 타입의 범위를 벗어난 값이 산출되었을 경우, 오버플로우가 발생하고 쓰레기값을 얻을 수 있다.
- 데이터를 바로 연산하기 전에 예외 발생 코드(ArithmeticException)로 검증하고 연산하는 메서드를 이용하는 것이 좋다.

정확한 계산은 정수 사용

- 정확하게 계산해야 할 때는 실수타입을 사용하지 않는 것이 좋다.
- 부동소수점 타입은 소수값을 정확하게 표현할 수 없기 때문에 근사값으로 계산한다.

NaN과 Infinity 연산

- 정수 타입을 0으로 /,%연산을 하면 컴파일은 정상적으로 되지만, 실행시 예외 발생
- 실수 타입을 0.0, 0.0f로 /연산하면 Infinity(무한대), %연산을 하면 NaN(Not a Number)를 가진다.
- Infinity, NaN과 산술연산하면 무조건 Infinity, NaN이 나오므로 연산 전에 Double.isInfinite(), Double.isNaN()으로 검증하고 연산한다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

입력값의 NaN 검사

- 부동소수점으로 변환이 가능한 문자열을 입력받을 때 "NaN"을 입력받으면 Double.valueOf()로 double 타입으로 변환해도 NaN이 저장된다.
- NaN인지 조사할 때 Double.isNaN()이 아닌 ==을 사용하면 안된다. NaN은 != 연산자를 제외한 모든 비교 연산자는 false를 리턴.

문자열 연결 연산자(+)

- 문자열과 숫자가 혼합된 + 연산식은 왼쪽에서 오른쪽으로 연산한다.

비교 연산자(==, !=, >, >=, <, <=)

- 비교연산을 수행하기 전에 타입 변환을 통해 피연산자의 타입을 일치시킨다.
- Double과 float을 비교할 경우 부동소수점 타입은 오차가 있으므로 피연산자의 타입을 하나로 통일하거나 정수로 변환해서 비교해야 한다.

  ex. 0.1 == 0.1f (false)

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

비교 연산자(==, !=, >, >=, <, <=)

- String은 ==, !=만 사용 가능하며 두 연산자는 String 객체가 참조하는 객체가 동일한지, 다른지를 판단한다.
- 자바는 문자열 리터럴이 동일하면 동일한 String 객체를 참조하도록 되어있다.
- new 연산자로 생성하면 새로운 객체의 주소값을 갖는다.
  ex. String str1 = "1"; String str2 = "1"; String str3= new String("1");
  str1 == str2;(true) str1 == str3;(false)
- String 객체의 문자열을 비교하기 위해서는 equals() 메소드를 사용해야 한다.
  ex. str1.equals(str2)

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피연산자 모두가 true일 경우에만 연산 결과는 true
	true		false	false	
	false		true	false	
	false		false	false	
OR (논리합)	true	|| 또는 |	true	true	피연산자 중 하나만 true이면 연산결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피연산자가 하나는 true이고 다른하나가 false일 경우에만 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
NOT (논리부정)		!	true	false	피연산자의 논리값을 바꿈
			false	true	

논리 연산자(&&(and), &(and), ||(or), |(or), ^(xor), !(not)

- &&와 ||가 &와 |랑 같은 결과를 내지만 좀더 효율적으로 동작한다.
- &&은 앞 피연산자가 false면, ||는 앞 피연산자가 true면 뒤 피연산자를 보지 않고 산출한다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

비트 논리 연산자(&, |, ^, ~))

- 0과 1로 표현이 가능한 정수 타입만 연산 가능
- 피연산자를 int 타입으로 자동 타입 변환한 후 연산 수행.

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피연산자 모두가 true일 경우에만 연산 결과는 true
	true		false	false	
	false		true	false	
	false		false	false	
OR (논리합)	true	|| 또는 |	true	true	피연산자 중 하나만 true이면 연산결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피연산자가 하나는 true이고 다른하나가 false일 경우에만 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
NOT (논리부정)		!	true	false	피연산자의 논리값을 바꿈
			false	true	

비트 이동 연산자(<<, >>, >>>)

<<	A << Z	정수A는 비트 단위로 왼쪽으로 Z비트만큼 시프트한다. 빈 비트에는 0으로 채운다. 결과는A * 2^Z(2의Z승)
>>	A >> Z	정수A는 비트 단위로 오른쪽으로 Z비트만큼 시프트한다. 빈 비트에는 0으로 채운다. 결과는A / 2^Z(2의Z승)
>>>	A>>>Z	정수A는 비트 단위로 오른쪽으로 Z비트만큼 이동한다. 빈 비트에는 0으로 채운다. 부호 비트를 고려하지 않기 때문에 A가 음수일 경우 시프트 결과는 양수로 나타난다.

ⓒSaebyeol Yu. Saebyeol's PowerPoint

연산자

대입 연산자(=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, <<<=)

- 연산자 중 가장 낮은 연산 순위이므로 가장 나중에 수행
- 연산 진행 방향이 오른쪽에서 왼쪽.

대입 연산자	표현식	연산자의 의미
=	A = B	B의 값을 A에 대입
+=	A += B	A = A + B와 동일
-=	A -= B	A = A - B와 동일
*=	A *= B	A = A * B와 동일
/=	A /= B	A = A / B와 동일
%=	A %= B	A = A % B와 동일
&=	A &= B	A = A & B와 동일
|=	A |= B	A = A | B와 동일
^=	A ^= B	A = A ^ B와 동일
<<=	A <<= B	A = A << B와 동일
>>=	A >>= B	A = A >> B와 동일
>>>=	A >>>= B	A = A >>> B와 동일

삼항 연산자(~ ? ~ : ~)

- 조건식(피연산자1) ? 값or연산식(피연산자2) : 값or연산식(피연산자3)
- 조건식이 true면 삼항 연산자 결과가 피연산자2
- 조건식이 false면 삼항 연산자 결과가 피연산자3

ⓒSaebyeol Yu. Saebyeol's PowerPoint


Q&A

PPT 템플릿 출처: 새별의 파워포인트
http://bit.ly/saebyeol

©Saebyeol Yu. Saebyeol's PowerPoint