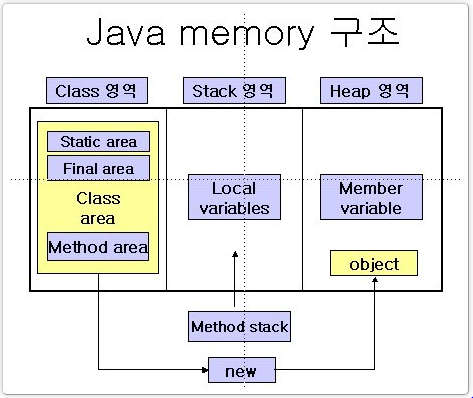
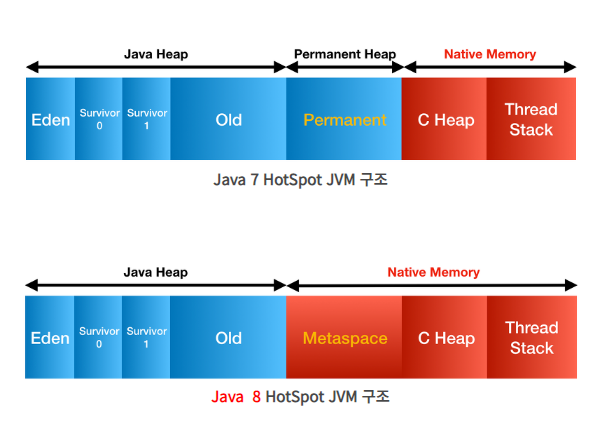
**DEV #4 – 2020.04.16**

**>> JVM 메모리 구조를 조사하고 설명하세요.**

* JVM은 자바 가상 머신으로 자바 프로그램을 실행 시킨다는 것은 JVM을 실행시키고 그 위에서 자바프로그램을 실행시킨다는 의미.

---JVM 메모리 구조---





Perm영역은 Class의 Meta 정보, Method의 Meta정보, Static변수와 상수 정보가

저장되는 공간. 흔히 메타데이터 저장 영역이라고 한다.

이 영역은 Java 8 부터는 Native Memory 영역으로 이동하여 Metaspace영역으로

명명되었으며, 다만 static object는 Heap영역으로 옮겨져서 GC의 대상이 최대한 될

수 있도록 하였다.

**Java Compiler**

: Java Source 파일을 JVM이 해석할 수 있는 Java Byte Code로 변경한다.

**Java Byte Code**

: Java Compiler에 의해 컴파일 된 결과물(.class)

**Class Loader**

: JVM 내로 .class 파일들을 로드하여 Runtime Data Areas에 배치한다.

**Execution Engine**

: 로드 된 클래스의 Byte code를 해석한다.

**Runtime Data Areas**

: JVM이라는 프로세스가 프로그램을 수행하기 위해 OS에서 할당 받은

메모리 공간.

1. Method Area(Static Area)

: 정적 변수, 생성자, 메소드, 멤버 변수가 저장되는 공간

1. Heap Area

: 동적으로 생성된 객체(new 연산자로 생성된 객체 또는 인스턴스)가

저장되는 공간

1. Stack Area

: 메소드가 호출될 때 Stack Area에 할당되고 FILO 구조를 가짐

스레드가 생성되면 각 스레드는 스레드마다 하나의 스택을 할당 받음

메소드 수행 시 발생되는 지역변수도 여기에 저장됨

1. PC Register

: 현재 수행중인 JVM 명령 주소를 가짐

1. Native Method Stack Area

: Java 이외의 언어로 된 코드를 위한 스택

JNI(Java Native Interface)를 통해 호출되는 다른 언어의 코드를 수행하기 위해 존재

**Garbage collector**

: 자바 프로그램이 실행 중에 동적으로 생성한 객체가 모두 사용되었는지의

여부를 판단하고 사용하지 않는 메모리를 해제한다.

Eden 영역의 메모리가 꽉 차면 Survivor 1/2 영역이 차례로 채워지고

지워지고를 반복한다.(Minor GC) Survivor 영역도 꽉 차게 되면 Old 영역까지

넘어오게 되고 Old 영역도 다 찰 경우 Full GC가 발생해서 메모리를 해제한다.

이 때 stop the world가 일어난다. stop-the-world란 JVM이 GC를 실행하기

위해서 애플리케이션 실행을 잠시 멈추는 것이다.

GC튜닝이란 이 stop-the-world 시간을 줄이는 것이다.

**>> “openjdk 1.8” 이 설치된 centos 7 구동용 Dockerfile 및 docker image를 생성하세요.**

* **openjdk 설치는 yum을 사용하고, 검색방법/설치방법을 익히세요. (yum cheat sheet)**
* **yum 사용하기 전 ( yum update && yum upgrade )를 먼저 실행합니다.**

🡺 컨테이너 접속 후

# yum update -y && yum upgrade -y

* **“/javawork” 디렉토리를 workdir로 설정합니다.**

🡺 Dockerfile에서

WORKDIR /javawork

* **docker image 명은 “centos7\_openjdk8”로 지정합니다.**

🡺 현재 컨테이너 상태를 이미지로 만듬

docker commit dong\_centos7 centos7\_openjdk8:1

**>> 빌드된 이미지 (centos7\_openjdk8)로 컨테이너 구동 옵션 추가**

* **컨테이너 workdir로 설정된 경로(/javawork)를 host pc 임의 경로와 연결하세요.**

🡺 docker run -d --name centos7\_web

-v //c//Users//Dong//test:/javawork centos7\_openjdk8:1 tail -f //dev//null

**>> “Hello World”를 출력하는 코드를 작성하고, compile 후 실행하세요.**

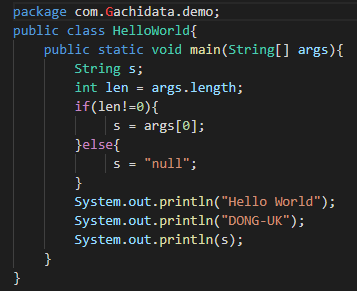
🡺 docker exec centos7\_web java –version

( centos7 컨테이너에 설치된 java version 확인 -> 1.8.0\_242)

java –version

( 윈도우에 설치된 java version 확인 -> 1.8.0\_171)

alias java=’docker exec centos7\_web java’

 (alias를 이용해서 centos7\_web에서의 java 명령어를 등록)

alias javac=’docker exec centos7\_web javac’

code HelloWorld.java

javac HelloWorld.java

(HelloWorld.java 파일을 컴파일 한다.)

java HelloWorld

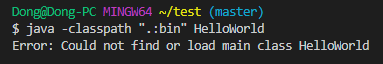
(java 파일을 실행한다.)

javac –d ./bin HelloWorld.java

(HelloWorld.java 파일을 현재 디렉토리에 있는 bin 디렉토리에 컴파일한다.)

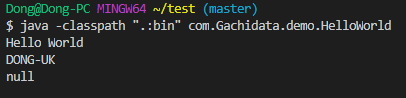
java –classpath “.:bin” HelloWorld

(현재 디렉토리에서 classpath를 지정해서 HelloWorld를 실행한다.

 . : 현재 디렉토리를 참조

**:** : 리눅스는 classpath 지정 시 : 로 연결함

java –classpath “.:bin” HelloWorld (오류남)

 왜? package를 com.Gachidata.demo 지정했는데

패키지명까지 다 적어서 실행해야 함.)

**>> “public static void main(String[] args)” 의 각 키워드를 설명하세요.**

public

: 접근 제한자이며 제약이 강한 순서로 private -> protected -> public 순이며

defualt는 클래스 내부와 동일 패키지에서 접근할 수 있음. 여기서 public은

어디서든 해당 객체를 참조할 수 있다는 의미

static

: 어떠한 값이 메모리에 한번 할당되어 프로그램이 끝날 때 까지 그 메모리에

값이 유지된다는 의미. 여기선 static 메소드를 위한 키워드로서 객체의 생성

없이 호출이 가능하고 일반 객체에서는 호출이 불가능함. 또한 static 메소드 안에서는 인스턴스 변수 접근이 불가능.

void

: 해당 메소드가 실행된 후 리턴되는 값이 없고 해당 메소드가 호출된

부분으로 이동.

main

: 자바 프로그램을 시작할 때의 시작 지점을 의미

String[] args

: 문자열 데이터가 배열에 저장되기 위한 공간

**>> classpath 의 개념을 조사하고 설명하세요.**

* JVM이 프로그램을 실행할 때, 클래스 파일(.class)을 찾는 기준이 되는 파일 경로

보통 classpath를 고정해서 사용함 ( 환경변수 세팅 )

--- 세팅 방법 ---

내 PC 우클릭 속성 -> 고급 시스템 설정 -> 고급 (환경변수) -> 새로 만들기 ->

변수명 (JAVA\_HOEM), 경로값 (jdk의 경로) -> path 설정(편집) -> 새로 만들기 (%JAVA\_HOME%\bin\)