**SQL**

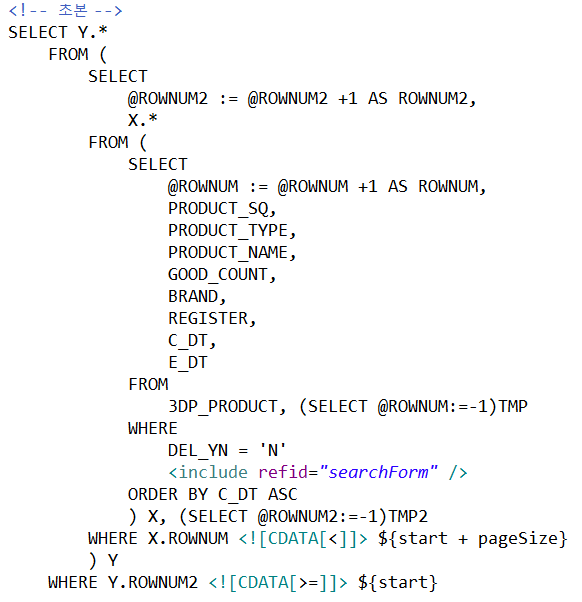
SQL 동작순서

WHERE문 🡪 SELECT문 🡪 ORDER BY 순으로 진행

🡺 ROWNUM과 ROWNUM이 들어간 WHERE문은 함께 들어갈 수 없음

🡺 Order By를 먼저 실행하고 한번 감싸준 뒤 ROWNUM을 수행 함 그 후 한번 더 감싸서 WHERE문 수행

1) ROWBOUND



SQL 성능

A)WHERE DATE\_FORMAT(ST\_LAST\_DATE, '%Y-%m-%d') LIKE DATE\_FORMAT(NOW(), '%Y-%m-%d');

B)WHERE ST\_LAST\_DATE BETWEEN DATE\_FORMAT(NOW(), '%Y-%m-%d') AND NOW();

A의 경우 FULL SCAN을 통해 결과값을 도출하지만 B의 경우 필요한 부분들만 SCAN하여 가져옵니다.

**JAVA**

Paging – ROW NUMBER은 출력되는 결과물에 순차적인 번호를 달아주는 기능

Integer.parseInt(String): int형 기본 자료형 반환 || Integer.valueOf(String): Integer 객체를 반환

StringTokenizer: java.util에 포함되어 있는 자체 클래스, 값이 존재하지 않다면 해당 데이터를 무시하고 실제 값부분만 반환

- StringTokenizer st = new StringTokenizer(br.readLine(), " ");

String.split: String 클래스의 메소드로 추출한 문자를 배열로 저장, 값이 존재하지 않더라도 그 결과를 반환

- split( String regex, int limit ), limit 분해되는 행렬의 최소 개수

Public static void main(**string[] args**)

- String[] args: 명령형 파라미터, 프로그램 시작시 옵션들을 파라미터로 받음(version, help, Xss, Xms, X…etc)

- public: 클래스의 안이든 밖이든 어디에서 호출 될 수 있음

- static: main() 메소드가 메모리에 인스턴스되는 공간의 주소가 모두 똑같음

- void 리턴값이 없음

정적 변수화 메소드(Method)

- static: 메모리의 효율 or 메모리 공유  
 ㅋ

Java.awt.list: List Component, list of information in an AWT GUI app for Displaying

java.util.list: Collection / Data Structure

**@Annotation**

**Paging**

RowBound

**@Controller Parameter 종류**

* HttpSession  
  HttpSession Object는 HttpServletRequest를 통해 가져올 수도 있지만 Http의 세션만 필요하다면 이것을 이용해 직접 받는 것이 나음 하지만 서버에 따라 멀티스레드 환경에서 안전성이 보장되지 않는다.
* RequestParam  
  Http 요청 파라미터를 메소드 파라미터에 넣어주는 애노테이션이다.

**JVM:** 자바 바이트코드를 실행할 수 있는 주체

레지스터 기반 프로세서: 레지스터를 사용하여 연산하는 방식(ARM 아키텍쳐)

* 함수 호출 시 인자를 스택에 넣어 줘야함
* 컨텍스트 스위치가 발생할 경우 모든 레지스터의 상태를 저장해 줘야함
* 레지스터가 CPU내에 존재하기 때문에 참조시 속도가 매우 빠름

스택 기반 프로세서: 피연산자를 스택에 저장하고 나서 연산을 수행하면 피 연산자를 꺼내어 연산 후 스택에 다시 넣음 (JVM, )

* 인자가 이미 스택에 저장되어 있어 추가적인 작업이 필요하지 않음
* 컨텍스트 스위치가 발생할 경우 스택만 변경시키면 간단히 해결
* 스택을 메모리에 두기 때문에 값을 읽고 쓰기 위해 메모리에 액세스를 해야하는 단점

Context Switching: Ready인 상태의 Process A와 Running인 Process B의 상태를 서로 전이 시키는 것, 이 작업동안 CPU는 아무런 일을 하지 못하고 CPU에 많은 부하를 줌

JVM 수행 과정

1. OS로부터 필요 메모리 할당

2. 자바 컴파일러(javac)가 소스코드(.java)를 읽어 바이트코드(.class)로 변환시킴

3. Class Loader를 통해 class파일들을 JVM으로 로딩’

4. 로딩된 class파일들을 Execution engine을 통해 해석

5. 해석된 바이트코드는 Runtime Data Area에 배치되어 실질적 수행

Class Loader

* JVM내로 클래스(.class)를 로드하고 링크를 통해 작업을 수행하는 모듈
* Runtime시 동적으로 클래스를 로드
* jar파일 내 저장된 클래스들을 JVM위에 탑재하고 사용하지 않는 클래스들을 메모리에서 삭제
* 클래스를 처음으로 참조할 때, 해당 클래스를 로드하고 링크하는데 이 역할 수행

Execution Engine

* 클래스를 실행
* 바이트코드를 JVM내부에서 기계가 실행할 수 있는 형태로 변경
* 바이트 코드를 명령어 단위로 읽어서 실행(한 줄 씩 수행)하기 때문에 속도가 느림, 인터프리터 형식

JIT(Just – In – Time): 사용할 때 컴파일해서 사용

* 인터프린터 방식을 보완하기 위해 도입된 JIT 컴파일러
* 인터프리터 방식을 실행하다 적절한 시점에 바이트코드 전체를 컴파일하여 네이티브 코드로 변경하고 이후 더 이상 인터프리팅 하지 않고 네이티브 코드로 수행
* 반복되는 내용은 한번만 컴파일(byte -> native)

GC

Minor GC

* 새로 생성된 대부분의 객체(Intance)는 Eden 영역에 위치한다. Eden영역에서 GC가 한번 발생한 후 살아남은 객체는 Survivor 영역 중 하나로 이동된다. 이 과정을 반복하다가 계속해서 살아남아 있는 객체는 Old 영역으로 이동시킨다.

Major GC(Full GC)

* Old영역에 있는 모든 객체들을 검사하여 참조되지 않은 객체들을 한꺼번에 삭제한다. 시간이 오래 걸리고 실행 중 프로세스가 정지된다. 이것을 ‘stop-the-world’라고 하는데 Major GC가 발생하면 GC를 실행하는 스레드를 제외한 나머지 스레드는 모두 작업을 멈춘다. GC 작업을 완료한 이후에야 중단했던 작업을 다시 시작한다.

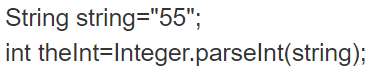
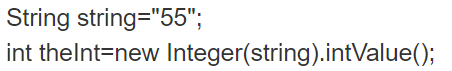
GC의 대상 선정 기준

* GC는 힙 내의 객체 중에서 Garbage를 찾아내고 찾아낸 Garbage를 처리해서 힙의 메모리를 회수한다. 참조되고 있지 않은 객체(Intance)를 Garbage라고 하며 이를 판단하기 위해 Reachability라는 개념을 사용한다. 어떤 힙 영역에 할당된 객체가 유효한 참조가 있으면 Reachability, 없다면 unReachability로 판단한다. 하나의 객체는 다른 객체를 참조하고, 다른 객체는 또 다른 객체를 참조할 수 있기 때문에 참조 사슬이 형성되는데, 이 참조 사슬 중 최초에 참조한 것을 Root Set이라고 칭한다. 힙 영역에 있는 객체들은 총 4가지 경우에 대한 참조를 하게 된다.

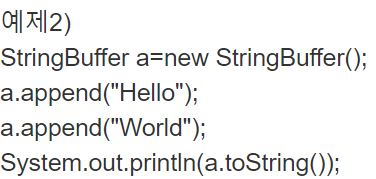
Programming technique for improvement of performance

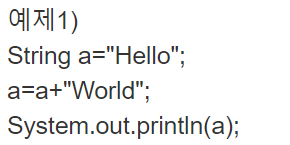
1. Avoiding Garbage Collection(static method의 사용)

- 가능한 Object의 Instance가 필요 없는 static method를 사용한다.

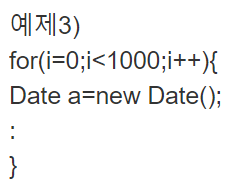
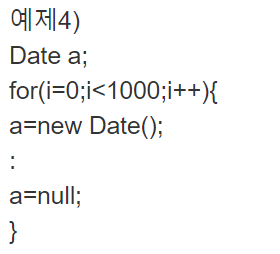


2. Avoiding Garbage Collection(임시 Object 생성 금지)

 - String Object의 append를 위해 (+) 연산을 사용하는 것을 들 수 있음, (+) 연산자 대신 StringBuffer 클래스를 사용한다.



어떤 메소드는 Object의 복사본을 반환하는 경우가 있다. 대표적으로 스트링 클래스의 trim() 메소드가 수행되면 기존의 Object는 수집되고, 기존 Object의 복사본이 사용된다. 즉 루프 내부에서 Object를 생성하는건 가급적 피해야 함



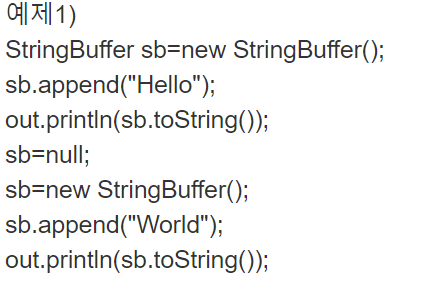
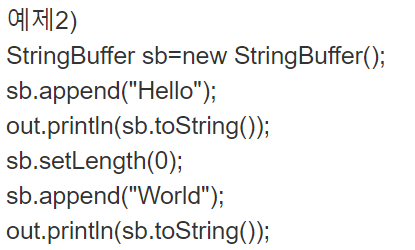
3. Avoiding Garbage Collection(primitive data type 사용)

- Date Class, String Class 값들 중 int나 long으로 표현할 수 있는 경우 굳이 String으로 표현하지 않아도 됨, 주의할 점은 Object를 기본데이터로 Castring하는 작업이 오히려 더 걸릴 수 있음.

4. Object 재사용(Pool Management)

- Object 재사용 기법으로 흔히 Pool Management 기법을 사용한다. 이는 임의 개수의 Object를 미리 생성해 두고 이를 Vector 클래스를 사용해서 관리하는 방법, 해당 Object를 사용하기 위해서 Pool의 Vector에 있는 Object를 하나 가져오고 다 사용하면 다시 Pool에 반납.

5. Object 재사용(Method를 사용한 Object 초기화)

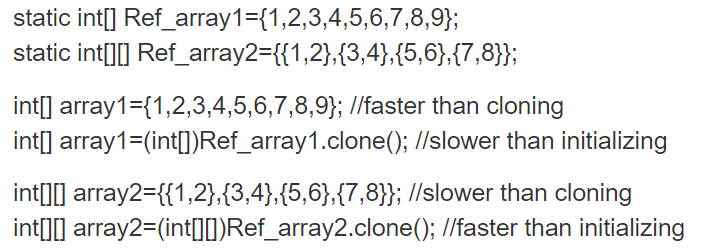


예제 1과 같은 경우 2번의 초기화 과정을 거침, 예제 2와같이 사용할 수 있음

6. Static Instance Variable의 사용에 의한 단일 클래스 인스턴스 구현

7. Clone()메소드 사용으로 Object 생성에 따른 Overhead 제거

- 대부분의 Method는 clone()이 존재하는데 clone() Method가 호출되면 Object의 복사본을 반환하는데, 대신 클래스의 생성자를 호출하지 않기 때문에 생성자 호출의 OverHead를 피할 수 있음.



8. “Extends” vs “Implements”  
 - “Implements”의 경우에는 특정 메소드를 구현하고 인터페이스에 정의된 모든 메소드를 코딩해야 하기 때문에 코드의 낭비를 초래하는 반면, “Extends”의 경우에는 슈퍼 클래스에 정의된 메소드들 중 필요한 메소드만 Overriding하면 된다.

9. I/O Performance의 개선

- 자바에서는 자료를 읽거나 쓰기 위해 stream을 사용하는데 자바는 Readers and Writers와 Input and Output stream이다. Reader and Writers는 high-level의 I/O를 지원하고 Input And Output Stream은 low-level의 I/O를 지원한다. 속도 향상을 위해서는 Buffered Stream을 이용한다. Buffered stream을 사용할 경우 버퍼의 기본값은 2K이다. 이 값은 조정될 수 있으나, 자료의 용량이 클 경우 메모리가 많이 필요하기 때문에 Buffered stream을 사용할 경우 여러가지 사항을 고려해야한다.

// stream 이란?

* 컴퓨터 상의 모든 프로그램, 입출력 기기들 사이에서 통신하는 방식
* Inputstream: 바이트의 흐름을 받아들이거나 내보내는 객체

Runtime Data Area

PC Register

* Thread가 시작될 때 생성되며 생성될 때마다 생성되는 공간으로 스레드가 하나씩 존재

JVM 스택 영역

* 프로그램 실행과정에서 임시로 할당되었다가 메소드를 빠져나가면 소멸되는 데이터를 저장하기 위한 영역
* 각종 형태의 변수, 임시 데이터, 스레드, 메소드 정보들을 저장

Native Method Stack

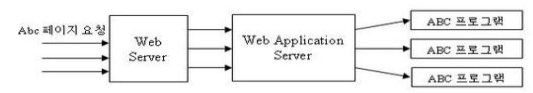
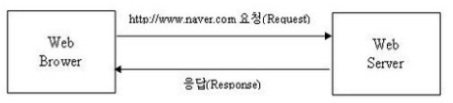
* 바이트 코드가아닌 실제 기계어로 작성된 프로그램을 실행시키는 영역
* JAVA Native Interface를 통해 바이트 코드로 전환하여 저장

Method Area(= Class Area = Static Area)

* 클래스 정보를 처음 메모리 공간에 올릴 때 초기화 되는 대상을 저장하기 위한 공간

Heap(힙 영역)

JNI(Java Native Interface)란?

* 자바가 다른 언어(Native Code, C/C++/Assembly)로 만들어진 어플리케이션과 상호작용할 수 있는 인터페이스 제공
* JNI로 작성된 Native Code는 VM을 거치지 않고 CPU가 바로 실행
* JVM이 Native Method를 적재하고 수행할 수 있도록 해줌
* JVM이 호스트 운영체제상의 입출력, 그래픽스, 네트워킹 쓰레드 같은 기능들을 작동하기 위한 Local System Calls를 수행할 수 있도록 한다.
* WAS(Web Application Server)  
  브라우저에서 사용자의 요청을 처리하는 서버 쪽 프로그램  
   - Apache, iis, 웹로직, Tomcat, Jboss, WildFly, Jeus
* 웹 서버(Web Server)  
  웹에서 서버 기능을 처리하는 프로그램, HTTP 프로토콜 기반으로 요청 서비스를 처리하여 응답
* WAS(Web Application Server)  
  웹 서버의 기능들을 구조적으로 분리하여 처리, Web Server + Container  
  대부분의 WAS는 이 기능 이외에 Transaction, Traffic, 보안, DB Connection Pool… 제공, 그렇기에 J2EE 서버라고도 불림
* J2EE(Java To Enterprise Edition)  
  자바 기술로 기업환경의 어플리케이션을 만드는데 필요한 스펙들을 모아둔 스펙 집합  
  🡪 Servlet: Client가 보내는 HTTP 요청을 처리하는 서버측 자바 프로그램  
  🡪 JSP(Java Server Pages): HTML+Java 코드를 써서 사용자에게 정보를 보여줌  
  🡪 EJB(Enterprise Java Beans): 비즈니스 로직이나 데이터, 메시지를 처리할 수 있음  
  🡪 Remote Method Invocation(RMI): 프록시를 써서 원격에 있는 Java 객체의 매소드를 실행 시킴  
  🡪 Java Naming Directory Interface(JNDI): Java 객체에 이름을 붙여 단일 Interface 제공
* WEB  
  기본적으로 Client/Server 방식의 형태, CGI, ASP, PHP 등이 존재함
* Servlet(Server Applet)  
  🡪 JAVA언어를 기반으로 동적인 Contents를 생성하는 기술, JAVA 코드안에 HTML, 효율성이 떨어짐
* JSP(Java Server Pages)  
  JAVA 언어 기반으로 만들어짐, ASP, PHP처럼 HTML 태그 사이에 동적 Contents생성을 담당할 JAVA 코드가 들어있는 형태, 동적 Contents를 생성하기 위해 스크립트 언어 형식으로 프로그램을 작성할 수 있어 개발자에게 쉬운 개발을 가능케 함  
  🡪 JSP가 Servlet으로 변환되고 그 후 변환된 Servlet이 Compile 되고 실행됨
* CGI(Common Gateway Interface)
* Web Application  
  웹을 기반으로 실행되는 프로그램

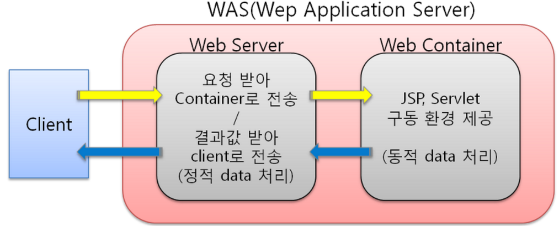
**Database**

* JDBC (Java Data Base Connectivity)  
  Data Base와 연결하기 위한 Java Interface
* DBCP (Database Connection Pool)  
  Data Base와 Connection을 맺고있는 객체를 관리하기 위한 Connection Pool
* ORM(Object Relation Mapping)  
  iBatis, JPA(Java Persistence API), NHibernate

**Linux  
-** 데비안: 패키지 설치 및 업그레이드 등을 apt로 쉽게 설정 가능 / 서버 **-** 우분투: 업데이트 주기가 짧은 편 / 데비안 계열 / 데탑 서버 개발  
- 민트: 성능이 좋지않은 태블릿으로 간소하게 웹서핑 정도에 유리 / 우분투 기반 / 데탑  
- 레드햇(RHEL): 안전성, 업데이트, Q&A에서 유리, 기업↑ / 서버, 데탑 / 라이센스  
- CentOS: 레드햇을 배포판으로 만든 리눅스, 서버가 안정적이지만 서비스 지원X / 서버,개발  
- 페도라(Fedora): 레드햇에 추가시킬 기능들을 테스트하는 리눅스, 버그↑ / 개발  
- 칼리(Kali): 모의해킹, 패킷 분석 등 해킹에 효율적 / 모의해킹, 패킷분석, 취약점 찾기

* SSH(Secure Shell)  
  네트워크 상의 다른 컴퓨터에 로그인하거나, 원격 시스템에서 명령을 실행하고 파일을 복사할 수 있도록 해 주는 응용 프로그램 또는 그 프로토콜
* Telnet(Telecommunication Network)  
  인터넷을 통하여 원격 호스트 컴퓨터 접속 시 지원되는 인터넷 표준 프로토콜
* FTP(File Transfer Protocol Server) Server  
  네트워크 상의 컴퓨터들간에 파일을 교환하기 위한 표준 인터넷 규약
* SFTP(Secure File Transfer Protocol)  
  PC와 서버간의 데이터 전송을 암호화하여 보안이 강화 된 FTP

ETC

* Bootstrap  
  Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.  
  트위터에서 만든 프레임워크로
* WebPack
* Console vs Terminal vs Shell  
  🡪 Console: 장비에 붙어있는 입출력 장치  
  🡪 Terminal: 원격이나 더미 터미널에서 쉘을 여는 도구  
  🡪 Shell: 커널과 유저가 대화하는데 쓰이는 유일한 통로
* Jboss(JBossAS, JBoss Application Server)  
  🡪 오픈소스 Java EE(Java Enterprise Edition) 서버, 보통 Java EE 서버를 WAS(Web Application Server)라고 부르는데 이것들에는 BEA WebLogic, Tmax JEUS, IBM WebSphere, Adobe JRUN, JBoss, Apache Geronimo
* Apache Tomcat과 JBossAS  
  🡪 Apache Tomcat은 Java EE표준에 포함되어 있는 JSP, Servlet, JSTL등과 같은 웹 어플리케이션 개발을 위한 표준을 구현한 웹 컨테이너입니다. Java EE 표준에는
* WAS Restart  
  apache란?  
   - 웹 서버, http 서버, http요청을 처리함  
  tomcat이란?  
   - WAS서버, Web서버 + 웹컨테이너

ASP환경는 WAS대신 IIS를 사용하는데 MS계열 서비스라면 WAS의 기능을 담당하는 부분이 존재

EJB(Enterprise Java Bean)이란

**Data Format**

1. JSON(JavaScript Object Notation)은 무엇인가? <http://ohgyun.com/43>

* 경량의 데이터 교환 포맷
* 기계들을 위한 분석과 생성이 쉬움
* 언어에 독립적으로 프로그래밍 됨
* XML < JSON
  + On-the-wire 데이터 포맷인 XML보다 가볍고 빠름
  + JSON types: string number array Boolean || XML types: String
  + JavaScript 코드를 위해 Native 코드 포맷

2. JSON Object

* Name/value 쌍으로 구성
* JSON 내에서도 리스트는 순서가 존재, Set은 순서가 없음

1. AJAX(Asynchronos Javascript And XML)란 무엇인가?

* XML(Extensible Markup Language): 데이터 형식의 일종, 확장 가능한 ‘표시’언어

Native code(UnManaged Code)

* 컴퓨터의 기계어로 동작하는, OS에 직접 컴파일 되는 코드
* 직접 메모리 할당
* C, C++

Managed Code

* 구동을 위해 Interpreter(Java Virtual Machine, Net Framework)같은 프로그램 필요
* Garbage Collection 존재, 자동으로 메모리 할당, 해제를 담당
* C#, Java
* 언어 독립적
* Native에 비해 속도가 느림,