- 자기소개, 마지막 할말

- 질문에 대한 답변을 2~3마디 답변으로 준비하자

- 질문을 유도하는 방법 연습 (내가 원하는 질문으로 이끌어 가게 끔하면 답변하기 쉬워진다.)

\* 전공 면접 필수 과목 \*

1. 운영체제

- 운영체제란?

- 시스템(CPU,메모리 등)의 자원과 동작을 관리하는 소프트웨어

- 프로세스, 저장장치((RAM,HDD하드디스크), 네트워킹, 사용자, 하드웨어를 관리

- 메모리 구조(RAM)

- 메모리 공간 종류 4가지 : Code(실행할 프로그램의 코드), Data(전역변수, 정적변수)

, Heap(사용자의 동적 할당): 런 타임에 크기가 결정됨, 데이터가 동적으로 할당되는 공간

, Stack(함수의 호출정보, 지역변수, 매개변수): 컴파일 타임에 크기가 결정됨

- 프로세스 : 실행중인 프로그램(각각의 프로세스는 독립된 영역을 갖는다)

- 한 프로세스당 개별의 Code, Data, Heap, Stack을 가지고 있음

- 프로세스에서 Code, Data, Heap은 공유되지만 각각의 스레드가 갖고있는 Stack은 공유되지 않음

- 스레드 :

프로세스 안에서 실행되는 흐름 단위(프로세스 안에서 다른 스레드와 메모리와 CPU를 공유해서 사용합니다)

- CPU 스케줄러 : 준비큐 있는 프로세스에 대해 CPU 할당하는 방법

- 선점 : 남의 것(CPU)을 뻇는 스케줄링 방식

- SRT(Shortest Remaining Time) :

진행중인 프로세스가 있어도, 최단 잔여시간인(남은 시간이 짧은)

프로세스를 위해 Sleep 시키고 짧은 프로세스를 먼저 할당, 선점형 SJF 스케줄링이라 불린다.

- RR(Round Robin) :

Time Sharing System을 위해 설계된 스케줄링.

모든 프로세스가 같은 우선순위, Time 컨텀이 잦다면 FCFS와 다를게 없다.

그러므로 Time 컨텀을 짧게 적절히 조절하는 것이 중요 .

- Priority Scheduling(우선순위 스케줄링) :

우선 순위가 높은 프로세스에 CPU를 우선 할당하는 방식.

우선 순위는 시간 제한, 메모리 요구량, 프로세스의 중요성, 자원 사용 비용등에 따라

달라질 수 있음.

우선 순위가 같을 경우, FCFS와 다를게 없다.(비선점, 선점 둘다 사용)

- 비선점 스케줄링 : 남의 것(CPU)을 뻇지 않는 스케줄링 방식

- FCFS(First Come First Served) : 먼저 들어오는 순서대로 처리

- SJF(Short Job First) : 버스트 시간이 짧은 프로세스부터 CPU를 할당 처리

- 가상 메모리 : 사용하는 부분만 메모리에 올리고, 나머지는 디스크에 보관

모든 프로세스에게 메모리를 할당하기에는 메모리의 크기가 한계가 있어서 사용하는 방법.

프로세스에서 사용하는 부분만 메모리에 올리고 나머지는 디스크에 보관하는 기법을 가상메모리라 함.

- 데드락이란? : 프로세스가 자원을 얻지 못해 다음 작업을 못하는 상태 입니다.

\* 데드락 발생조건 \*

- 데드락은 네가지 조건이 동시에 성립 할 때 발생. 하나라도 성립하지 않도록 만든다면 데드락을 해결할 수 있다.

1. 상호 배제(Mutual exclusion) :

자원은 한번에 한 프로세스만이 사용할 수 있어야 한다.

2. 점유 대기(Hold and wait) :

최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용하고 있는

자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 한다.

3. 비선점(No preemption) :

다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 한다.

4. 순환 대기(Circular wait) :

프로세스의 집합 {P0,P1,...Pn}에서 P0는 P1이 점유한 자원을 대기하고 P1은 P2가

점유한 자원을 대기하고 Pn은 P0가 점유한 자원을 요구해야한다.

- 데드락 처리

- 운영체제란?

- 실행파일 생성 과정

- 프로그램 실행과정

- 명령어사이클

- 캐시란?

- 캐시라인

- 메모리구조

- 전역변수와 정적변수의 차이

- Heap, Stack 비교

- 가시성(visibility)과 원자성(atomocity)

- 유효주소, 주소지정방식

- 메모리 할당 알고리즘

- 프로세스, 스레드

- 멀티 프로세스와 멀티 스레드

- 스레드와 프로세스 콘텍스트 스위치 차이 이유

- 프로세스 스케줄러에 대해

- CPU 스케줄러

- 스레싱이란?

- IPC란?

- Race Condition?

- User, Kernel 스레드 차이

- 가상 메모리

- 메모리 단편화

- 페이징기법

- 세그먼테이션 기법ㅏ

- 메모리풀

- 페이지 교체 알고리즘

- MMU란?

- TLB란?

- 페이지 부재시 절차

- 인터럽트란?

- System Call

- 프로세스 제어 명령

- fork(), vfork() 차이

- 시스템콜과 서브루틴 차이

- 블록킹, 논블록킹

- 동기, 비동기

- 동기 IO 처리과정 : 프로세스 A가 디스크에서 어떤 데이터 읽어올때 상황

- 입출력 처리방식

- DMA

- Cycle Stealing

- 데드락

- 발생조건

- 회피기법

- 해결경험

- PCB(Process Control Block) 이란?

- Spin Lock 이란?

- 동기화 종류

- Critical Section?

- Mutex란?

- Semaphore란?

- Mutex, Semaphore 차이

- Monitor란?

- Thread-safe 란?

- Reentrant란?

- 인터럽트, 폴링 개념 및 차이

- Fault Tolerance 란?

- 작동절차

\* 그 외 \*

10. stream API란?

11. Optional 클래스란?

12. OOP의 4가지 특징, 5대 원칙(SOLID)

13. HTTP 응답 코드 유형에 대한 답변 300번대 400번대 500번대

14. OSI 7계층

15. 인터페이스와 추상클래스의 차이

16. ERD :

ERD란 Entity Relationship Diagram의 약어로, 데이터베이스 구조를 한눈에 알아보기 위해서 쓰인다.

DB를 개발하기 전에 보다 많은 아이디어를 도출하고, 데이터베이스 설계의 이해를 높이기 위해 데이터

모델링을 실시한다.

17. 모델링 정규화

- 자바 또는 스프링의 특징

- 스프링의 3대 장점

- 객체지향 3대 원칙

- 소프트웨어 형상관리

2. 네트워크

- \* OSI 7계층? (이건 방법없다 꼭 외우자)

- 인터넷 환경에서 통신하기 위해 네트워킹에 대한 표준을 7계층으로 나눈것

1. 응용 계층 Application

2. 표현 계층 Presentation

3. 세션 계층 Session

4. 전송 계층 Transport

5. 네트워크 계층 Network

6. 데이터 링크 계층 DataLink

7. 물리 계층 Physical

- TCP(Transmission Control Protocol) : 신뢰성은 높지만 느린 프로토콜 (상대가 받았는지 확인 가능)

- 연결형 프로토콜이며,연결이 성공해야 통신이 가능

- 데이터 경계를 구분하지 X

- 신뢰성 있는 데이터를 전송. 데이터의 순서 유지를 위해 각 바이트마다 부여한다. 데이터를 재전송한다.

- 1 대 1 통신

- 전송속도가 느림

- 패킷관리가 필요없다.

\* 특징 \*

- 가상 회선 연결방식, 연결형 서비스를 제공

- 연결의 설정(3-way handshaking)과 해제(4-way handshaking)

- 데이터 흐름제어(수신자 버퍼 오버플로우 방지) 및 혼잡 제어(네트워크 내 패킷 수가 과도하게 증가하는

현상 방지)

- UDP(User Datagram Protocol) : 신뢰성은 낮지만 빠른 프로토콜 (상대가 받았는지 확인 불가능)

- 비연결형 프로토콜, 연결 없이 통신이 가능

- 데이터 경계를 구분, 데이터그램 서비스

- 비신뢰적인 데이터를 전송. 데이터 재전송과 데이터 순서 유지를 위한 작업을 하지 않는다.

- 1 대 1 통신, 1 대 다 통신, 다 대 다 통신

- 전송속도가 빠름

- 패킷관리가 필요하다.

- 스트리밍 같은 서비스에 주로 사용

- IP란?

- Internet Protocol의 약자로 인터넷 망을 통해 패킷을 전달하는 프로토콜

- 인터넷에서 데이터 전달 프로토콜

- 어떠한 확장자로 열것인가에 대한 것도 프로토콜, 인터넷에서 데이터를 주고 받기 위한 약속

- IP Address : 라우터를 통해서 IP주소를 통해서 데이터 주소를 전달하는 것이 IP

\* 특징 \*

- 비신뢰성

- 패킷의 완전한 전달을 보장하지 않는 것

- IP는 프로토콜일 뿐이고 보낸 데이터의 흐름에 관여하지 않기 때문에 보낸 패킷이 제대로 갔는지

보장하지 않는다.

- 중간에 패킷을 가지고 있던 호스트가 강제종료되어 패킷이 소실될 경우 패킷이 목적지까지

가지 못하고 소실될 수 있다.

- 비연결성

- 패킷을 보내는 길을 정하지 않는 것

- 마찬가지로, 패킷을 받을 대상이 연결 상태임을 확인하지 않고 전송하기 때문에 받을 대상(호스트)이

서비스 불능 상태여도 패킷이 전송된다는 문제점이 있다.

- 이를 비연결성이라고 한다.

- DNS란?

- 도메인 주소를 IP주소로 변환해주는 시스템

예) naver.com -> 172.168.000.000

- 다음과 같은 원리로 작동하게 된다.

1. URL 입력하면 ISP가 관리하는 DNS 해석기에 요청을 라우팅 시킨다.

2. DNS 해석기가 루트서버에 top-level의 서버 주소를 요청 -> second-level -> sub DNS server

3. 최종적으로 IP주소를 얻게 된다.

- 로드밸런서란?

- 서버의 부하를 분산시켜주는 시스템

- 로드밸런서의 종류

- L4 : Transport 계층(전송 계층)을 사용, Port 기반 부하 분산 (TCP, UDP)

- L7 : Application 계층(응용 계층)을 사용, 요청(URL) 기반 부하 분산 (HTTP, HTTPS 등)

- DDOS(서버에 대한 과부하를 통해 서버장애를 일으키는 공격)비정상적인 트래픽 공격에

대한 해당 URL을 차단하는 방법으로 방어가 가능

- OSI 7계층에 대해서 설명하시오

- 라우터와 스위치의 차이

- 라우터와 게이트웨이 차이

- 게이트웨이

- TCP/IP 4계층

- OSI 7계층 쓰지않고 TCP/IP쓰는 이유

- TCP, UDP에 대해서 설명

- TCP에서 서버와 클라이언트 함수 호출 순서

- 3way, 4way handshake에 대해 설명

- 4way handshake의 비정상 종료상황

- IP

- ICMP

- IGMP

- ARP

- RARP

- DNS란?

- 작동순서

- 라운드로빈이란? 및 단점

- Proxy Server

- 로드밸런싱이란?

- 서버 느려질때

3. 데이터베이스

- DBMS(DataBase Management System)

- 데이터베이스 내 데이터에 접근하도록 도와주는 시스템

예) 사용자 -> DBMS서버(MySQL, Oracle) -> 데이터

- 질의처리기(Query Processor), 저장시스템(Storage System)

- 트랜잭션 특징 ACID

- 내가 작업해야하는 것들이 모두 한번에 처리되거나 처리되지 않아야 하는 작업들을 모아놓은 것을

트랜잭션이라 한다.

- 데이터베이스의 무결성과 일관성을 위해서 트랜잭션은 4가지의 특징을 만족해야한다.

\* 특징 \*

- A : 원자성(Atomicity)

- 트랜잭션의 작업이 부분적으로 실행되거나 중단되지 않는 것을 보장하는 것을 말함.

- 즉, All or Noting의 개념으로서 작업 단위를 일부분만 실행하지 않는다는 것을 의미

- C : 일관성(Consistency)

- 트랜잭션이 성공적으로 완료되면 일관적인 DB상태를 유지하는 것

- 여기서 말하는 일관성이란, 송금 예제에서 금액의 데이터 타입이 정수형인데 갑자기 문자열이

되지 않는 것을 말합니다.

- I : 격리성(Isolation)

- 트랜잭션 수행시 다른 트랜잭션의 작업이 기어들지 못하도록 보장하는 것을 말함.

- 즉, 트랜잭션 끼리는 서로를 간섭할 수 없다.

- D : 지속성(Durability)

- 성공적으로 수행된 트랜잭션은 영원히 반영이 되는 것을 말함.

- commit을 하면 현재 상태는 영원히 보장.

- NOSQL

- Not Only SQL의 약자로, SQL을 보완한다는 의미

- 스키마? : 데이터를 저장하는 규칙

\* SQL

- RDBMS(관계형데이터베이스),엄격한 스키마 존재 O, 컬럼에 관계를 저장

\* NOSQL

- 스키마가 X, 데이터를 조회하고 삽입하는 속도가 빠름.

- 그래서 대량의 분산데이터를 저장하는데 특화되어 있음.

- 파티셔닝

- 테이블을 컬럼 단위로 나누어 관리하는 기법

- 스키마를 나누고 데이터가 따라 옮겨가는 것을 말한다.

- 하나의 엔티티를 2개 이상으로 분리하는 작업

- 장점 : update나 insert 같은 작업이 분산되어서 성능이 향상.

- 단점 : 테이블간 join 비용이 증가, index를 별도로 파티셔닝 할 수 없다는 단점을 가지고 있다.

- 샤딩

- 테이블을 row 단위,가로 횡 단위로 분산하여 저장하는 방법

- 같은 테이블 스키마를 가진 데이터를 다수의 데이터베이스에 분산하여 저장하는 방법

- Shard Key를 어떻게 정의하느냐에 따라 데이터를 효율적으로 분산시키는 것이 결정됩니다.

1. Hash Sharding

- Shard Key : Database id를 Hashing 하여 결정합니다.

- Hash 크기는 Cluster 안에 있는 Node 개수로 정하게 됩니다.

- 데이터가 많아져서 Cluster를 확장해야 할 때에 전체의 규칙을 바꿔줘야하는 것이 단점

2. Dynamic Sharding

- Naming 그대로 Dynamic 으로 바꿀 수 있다.

- Locator Service를 통해 Shard Key를 얻습니다.

- 테이블의 구간 규칙을 정해서 사용

- Cluster가 포함하는 Node 개수를 늘려 확장을 하기가 용이함.

- DBMS란?

- DBMS 구조

- 질의처리기 작동 순서

- 옵티마이저

- DB 저장 공간

- DBMS Join 원리 종류

- JDBC, ODBC차이

- DDL, DML, DCL?

- Key

- 인덱스

- 클러스터드, 넌클러스터드 인덱스란?

- 클러스터란?

- 시퀀스

- 뷰

- 트리거

- 무결성

- 무결성 보장방법

- 정규화

- 정규형 필요 조건

- 함수적 종속성

- 반정규화

- Anomaly

- SQL Injection

- 파티셔닝이란?

- 샤딩이란

- 리플리케이션이란?

- 클러스터링이란?

- 트랜잭션 특징 ACID

- 트랜잭션 상태

- 트랜잭션 격리 종류

- 낮은 격리단계 선택시 발생 문제

- Persistence Layer란?

- UML

- MySQL 특징

- PostgreSQL 특징

- SQL vs NOSQL

- CAP 이론

- NoSQL 종류

- NoSQL 사용경험

- Redis란?

- 카산드라 특징

4. 자료구조

- 힙

- 최대값 혹은 최소값을 빠르게 찾기 위한 이진트리

- 최소힙 : 부모는 자식보다 작아야한다.

- 최대힙 : 부모는 자식보다 커야한다.

- 높이만큼 시간복잡도가 걸리며 lgN의 시간복잡도를 가진다.

- 힙의 삽입과 삭제는 O(lgN)만큼의 시간복잡도를 갖습니다.

- 이진 탐색 트리

- 왼쪽 자식은 부모보다 작고, 오른쪽 자식은 부모보다 큰 이진 트리

- 이진 탐색 트리는 삽입, 검색, 삭제가 모두 이진트리의 높이만큼 시간복잡도가 걸리며 lgN~N만큼의

시간복잡도를 가진다.

- 그래서 트리가 편향되지 않기 위해서 자가 균형 트리를 사용합니다.

- 자가 균형 트리

- 편향되지 않게 삽입, 삭제를 개선한 이진 탐색 트리

- 트리를 전체적으로 재배치 해주는 방법

- AVL, Red Black

- 이진 탐색 트리는 시간복잡도가 트리의 높이에 따라 결정되므로 편향될 경우 효율이 떨어집니다.

- 그래서 편향되지 않게 삽입과 삭제를 개선한 이진 탐색 트리를 자가균형트리라고 합니다.

- 자가 균형 트리에는 AVL, Red Black 트리가 있습니다.

- 해시

- 해시에 매핑하여 데이터를 저장하는 자료구조

- 리스트는 O(N)만큼 걸리는 구조로 자료 검색에 아주 성능이 떨어지는 자료구조이다. 그래서 해시를 사용.

- 해시는 삽입, 삭제, 조회가 모두 O(1)의 시간복잡도를 갖습니다.

- key는 Hash Function을 통해 hash로 변경된 다음에, value와 매핑되어서 bucket에 저장되게 됩니다.

- 해시 충돌 회피 방법

- 해시에서 하나의 버킷에 여러개가 들어갈 때 그 충돌을 회피하는 방법으로,

Open Addressing과 Chaining이 있습니다.

- Open Addressing : 다른 해시 값에 저장하는 방법

- Chaining : 해쉬 테이블이 원소 하나를 담는게 아니라, Linked List를 담는 방법

- 힙이란

- 힙의 삽입, 삭제

- 힙이 할수있는 것을 균형 이진 트리가 할수 있지않나?

- 이진 탐색 트리

- 이진 탐색 트리 삽입, 탐색, 삭제

- 자가 균형 트리

- AVL

- Red Black Tree

- 해시

- 충돌회피 방법

5. 자바

- 자바 애플리케이션 실행 과정

- 운영체제가 다르더라도 공통적으로 사용가능하게끔 운영체제 위에 짜여진 애플리케이션 -> JVM(자바 버츄얼 머신)

- JAVA Source File(.java) -> JAVA Compiler(javac) -> JAVA Byte Code(.class)

1. 컴파일러가 자바 소스를 바이트 코드로 변환합니다.

2. Class Loader가 바이트 코드를 런타임 데이터 영역에 로드시킵니다.

3. 로딩된 바이트 코드가 실행엔진에 의해서 실행되게 됩니다.

- 자바 메모리 구조(Runtime Data Area)

- 크게 5가지 영역으로 구분된다.

- 스레드별 : PC Register, JVM Stack, Native Method

- PC Register : 현재 실행중인 명령어를 가지고 있는다.

- JVM Stack : 함수에 대한 정보, 매개변수, 지역변수, 리턴정보들이 저장.

- Native Method : 자바 외부 언어로 작성된 코드들이 들어있다.

- 공통 : Method Area, Heap

- Heap : 런타임 중 생성되는 객체들이 동적으로 할당되는 곳.

사용자가 사용하면서 생긴 데이터가 저장되는 공간, 사용데이터가 동적으로 할당되는 곳.

- Method Area : 클래스 별로 전역변수, 정적변수, 메소드 정보들이 저장되는 공간.

- 가비지 컬렉션(GC)

- JVM에서 자동으로 메모리를 관리해주는 모듈

- Heap 메모리를 재활용하기 위해서 더 이상 참조되지 않는 객체들을 메모리에서 제거하는 모듈.

- 개발자가 직접 메모리를 정리하지 않아도 되어서 개발 속도가 향상되는 장점이 있지만 Mark and Sweep

이라는 과정에서 참조되지 않는 객체를 찾는 과정이 있는데, 이때 스레드가 잠깐 중된되며 성능이 떨어진다는

단점이 있습니다.

- 메모리가 꽉 찰 경우에 잠깐 실행을 멈추고 메모리들이 어디에서 참조되는지를 확인하고 참조되지 않으면

메모리 공간을 비움.

- 잠깐 멈추고 안쓰고 있는 것을 찾는 과정을 Mark and Sweep 이라 한다.

- 객체지향 프로그래밍

- 현실 세계의 사물같은 객체를 만들고, 객체에서 필요한 특징을 뽑아 프로그래밍 수행.

- 객체지향 프로그래밍에는 4가지의 특징이 있다.

1. 추상화

- 공통적인 특징들을 뽑아서 개념화 하는 것

예) 물, 콜라 -> 마시는것

2. 캡슐화

- 객체들 안에 숨겨야 할 정보들은 숨기고 내부에서는 잘 연결되도록, 외부와는 많이 의존,

결합하지 않도록 감싸는 것.

- 정보 은닉을 통해 내부적으로 높은 응집도를 가지고 외부에서는 낮은 결합도를 가지는 것을

캡슐화 라고 한다.

3. 상속성

- A와 B에 겹치는 작업에 대해서 A를 B에 포함시켜 작업의 재사용을 용이하게 하는 것이 상속성이다.

4. 다형성

- 다양한 형태, 하나의 요청에 각각의 형태,객체를 다르게 해주는 것을 다형성이라 한다.

- 추상클래스 vs 인터페이스

- 추상클래스 : abstract로 지시자로 정의되며 추상메소드가 하나 이상 포함

- 부모클래스를 자식클래스에서 상속받아서 재사용 하기 위해서 사용하는 것이 추상클래스 이다.

- public abstract class Animal extends Creature{

위와 같은 형태로 extends를 통해 부모의 클래스를 자식에서 상속받음.

- 인터페이스 : interface 지시자로 정의하며 모든 메소드가 추상 메소드로 정의

- 자바 8 이후 부터는 인터페이스에도 메소드가 구현이 가능해지며 점점 구분이 모호해져 가고 있다.

- 모든 메서드가 추상 메서드로 정의되게 된다.

- 인터페이스로 구현하면 같은 명의 메소드들을 모두 구현해야한다.

- 인터페이스로 구현한 객체들은 모두 같은 동작을 할 수 있다! 라는 것을 보장하는 것이 인터페이스 이다.

- public class Turtle implements Swimable{

위와 같은 형태로 implements로 부모의 클래스를 상속받게 됨.

- 추상클래스와 인터페이스의 차이는 그 존재의 목적에 있다.

1. 추상클래스는 상속받아서 기능을 재활용하고 확장시키는데 목적이 있다.

2. 인터페이스는 함수의 구현을 강제해서 구현한 객체들이 같은 동작을 하는것을 보장하는 것에

그 목적이 있다.

- 객체지향프로그래밍이란?

- 객체지향 프로그래밍 4가지 특징

- 객체지향 설계 5대 원칙

- 추상 클래스와 인터페이스의 차이

- 객체지향, 함수형의 차이

- 오버로딩 오버라이딩

- 자바 애플리케이션, JVM 실행 과정

- 자바 메모리 구조

- 클래스 로드 방법

- 클래스 로드 과정

- 가비지 컬렉션이란?

- 가비지 컬렉션 일어나는 과정

- GC 모니터링이란

- 컬렉션 프레임워크

- Vector와 ArrayList의 차이

- HashSet, TreeSet, LinkedHashSet 차이

- HashMap, LinkedHashMap, HashTable, TreeMap 차이

- Servlet 개념, 동작순서, 생명주기

- 정적변수와 전역변수의 차이

- 접근제어자에 대해 설명

- 접근제어자 사용하는 이유

- 직렬화(Serialize)란?

- Wrapper Class

- String은 래퍼클래스인데 == 비교시 값 같게 나오는 이유

- 제네릭이란?

- final이란?

- 자바 8버전, 11버전의 특징과 차이점

6. 스프링

- SPRING

- 객체를 관리할 수 있는 컨테이너를 제공(싱글톤 기법)

- spring MVC(model, view, control)

- 순수 자바

- 자바의 오픈소스 애플리케이션 프레임워크 중 하나로 스프링의 기본철학은 특정 기술에 종속되지 않고

객체를 관리할 수 있는 프레임워크를 제공하는 것 입니다.

- 컨테이너로 자바 객체를 관리하면서 의존성 주입과 제어의 역전을 통해 결합도를 낮추게 됩니다.

- DI(Dependency Injection)

- 의존성 주입, 컨테이너가 의존관계를 자동으로 연결

- DI는 Dependency Injection의 약자로 의존성 주입을 의미합니다.

- 객체간의 의존관계를 미리 설정해두면 스프링 컨테이너가 의존관계를 자동으로 연결해 줍니다.

- 직접 의존하는 객체를 생성하거나 검색해서 가져올 필요가 없어서 결합도가 낮아지는 장점이 있습니다.

- IOC(Inversion of Control)

- 자바에서 직접 제공하지 않고 외부에서 가져오는 것을 라이브러리라고 한다.

- 전체 로직이 어떻게 짜여지고 움직이는지를 관리하는 흐름이 이미 정해져 있고, 개발자는 부분의 코드만

채워 넣는다.

- IOC는 Inversion of Control의 약자로 제어의 역전을 의미합니다.

- 제어권이 사용자에게 있지 않고, 프레임워크에 있어서 필요에 따라 사용자의 코드를 호출하게 됩니다.

- 스프링에서는 인스턴스의 생성부터 소멸까지 개발자가 아닌 컨테이너에서 대신 관리하게 됩니다.

- ORM(Object Relational Mapping)

- ORM는 Object Relational Mapping의 약자

- 관계형 데이터베이스(RDBMS)를 OOP 언어로 변환해주는 기술

- 비즈니스 코드가 DB 테이블에 바로 접근하게 도와줍니다.

- JPA

- ORM을 위해 자바에서 제공하는 API

- 자바 객체와 DB 테이블을 Mapping하는데 구현체로는 하이버네이트가 있다.

- JPA는 인터페이스 이다.

- 대표적으로는 하이버네이트가 있다(implement로 구현).

- ORM, JPA, Hibernate 장단점

장점

- 비즈니스 로직(내가 짜는 코드)에 집중하고 , 객체중심의 개발을 할 수 있게 됩니다.

- 메소드 호출만으로 쿼리를 수행해서 생산성 향상, 유지보수 비용 감소

- 특정 DB에 의존하지 않는다.

단점

- 직접 SQL을 호출하는 것 보다는 성능이 조금 느려짐.

- 아주 복잡한 쿼리는 메소드, 하이버네이트로 구현이 어렵다.

- 스프링이란?

- IoC란?

- DI란?

- 디스패처 서블릿이란?

- 디스패처 서블릿으로 인한 web.xml 역할 축소

- AOP란?

- AOP 용어 설명

- AOP 사용 경험 : 공통 로그 구현

- 필터와 인터셉트 차이

- 스프링 MVC 구조 흐름

- 스프링과 스프링 부트 차이

- 스프링 어노테이션

- 어노테이션 종류

- 스프링 버전별 기능

- MVC 1과 MVC 2의 차이

- ORM

- JPA

- ORM, JPA, Hibernate 장단점

- Mybatis란?

- Mybatis, JPA 차이

- Spring JDBC

- DAO, DTO 란?

7. 웹 WEB

- 브라우저에 url치면 일어나는 일

- 해당 도메인에 대한 IP주소(DNS) -> Http request 메세지 -> 프로토콜 스택 -> LAN -> 허브, 스위치, 라우터

-> 프로바이더ISP -> pop -> 인터넷 핵심부 -> 고속라우터 -> 방화벽 -> 캐시 -> 웹서버 -> WAS 응답

1. 브라우저에서 DNS 서버에서 도메인명으로 IP주소를 가져오게 됩니다.

2. HTTP Request 메세지 작성

3. OS의 프로토콜 스택에 메시지 전송을 의뢰

4. 프로토콜 스택이 LAN에 제어정보를 붙인 패킷을 LAN 어댑터에 넘김.

5. LAN 어댑터는 전기신호로 변환시켜 LAN 케이블로 송출.

6. 송신한 패킷은 허브, 스위치, 라우터를 경유

7. 프로바이더에게 전달

8. 패킷은 수많은 엑세스 회선을 통해서 pop를 거쳐 인터넷 핵심부에 들어가게 된다.

9. 많은 고속 라우터들 사이로 패킷이 상대방 서버까지 도달.

10. 서버측의 LAN에 도착하게 되면 방화벽이 패킷을 검사.

11. 이상이 없을 경우 캐시서버가 먼저 응답 데이터가 있는지 확인하게 된다.

12. 없을 경우에는 웹서버에 전송.

13. 패킷이 웹서버에 도착하면 이제 프로토콜 스택은 패킷을 추출, WAS에 전달.

14. WAS는 응답 메시지를 만들어서 다시 클라이언트에게 보내게 된다.

15. 위와 같은 방법대로 다시 클라이언트에게 전달.

- 쿠키와 세션

- HTTP는 상태와 연결에 대한 정보를 저장하지 않아, 이를 도와주는 것

- 쿠키

- 사용자의 정보가 기록된 텍스트 파일.

- 사용자의 브라우저에 저장, 통신할 때(HTTP)헤더에 포함되어서 전송.

- 쿠키의 정보가 노출될 수 있기 때문에 보안에 취약하다는 특징이 있음.

- 세션

- 사용자의 정보를 서버에 저장.

- 브라우저가 종료될 떄까지 유지.

- 서버에 저장되기 때문에 보안이 강하다는 특징을 가지고 있다.

- REST API, RESTful이란?

- REST 기반으로 서비스 API를 구현한 것

- REST라는 것은 자원(내가 요청할 건)의 표현(URI), 즉 이름으로부터 자원의 정보를 주고받는 것을 의미합니다.

- 자원에 대한 행위는 HTTP Method로 표현한 것이 REST API 입니다.

- HTTP Method(CRUD)

- Create : Post

- READ : Get

- Update

- Delete

- RESTful

- REST의 원리를 잘 따르는 시스템.

- 자원을 URL로, 행위에 맞는 적절한 HTTP method를 사용한 것이 RESTful 한 메소드 입니다.

- RESTful 하지 않은 경우의 예

- CRUD 기능을 모두 POST 만으로 처리한 것을 RESTful 하지 않다고 할 수 있습니다.

- HTTP 응답코드

- 클라이언트에서 서버에 응답을 요청했을 때 받게 되는 메세지를 보지 않아도 알 수 있게 하는 것

- 100번대 : 조건부 응답, 요청을 받아서 처리중을 의미.

- 200번대 : 성공. Successful Response. Request가 성공적으로 처리되었음을 의미.

- 300번대 : Redirection, 사용자를 다른 곳으로 이동. 클라이언트를 지정된 위치로 이동.

- 400번대 : 클라이언트의 오류, 보내는 사람이 요청을 잘못 보냈을 때.

HTTP 요청이 잘못되거나 권한이 없을 때.

- 500번대 : 서버의 오류. 서버가 요청을 제대로 수행하지 못할 때 발생.

- HTTPS

- 암호화 프로토콜을 사용하여 HTTP 통신을 안전하게 하는 프로토콜

- HTTP의 3가지 문제점

1. 평문 통신을 해서 도청이 가능 (보안에 취약)

2. 통신상대를 확인하지 않아서 위장이 가능

(내가 보낼 때 상대방이 맞는지 확인하지 않고 보내기 때문에 이상한 곳으로 아이디 패스워드를 보낼 수 있음.)

3. 상대방이 나한테 보낸 응답을 중간에 누군가가 수정할 수 있기 때문에 완전성이 보장이 안됨.

변조가 가능.

- 이를 보안하기 위해서 사용하는 것이 HTTPS.

- 구체적으로는 HTTP에서 통신하는 소켓을 암호화 프로토콜을 사용해서 TCP와 통신하도록 함.

- 암호화 프로토콜을 사용함으로써 HTTPS는 암호화, 증명서, 변조를 방지할 수 있다.

- 사용하는 암호화 프로토콜 : SSL, TLS

- 브라우저에서 URL치면 발생하는 일

- 쿠키, 세션

- 쿠키 동작방식

- 세션 동작방식

- 쿠키와 세션의 차이

- 캐시

- URI, URL, URN에 대해서

- REST, REST API, RESTful 이란?

- REST API 장점, 단점

- HTTP 1,2,3의 차이

- HTTP 응답코드

- 301, 302 차이

- 401, 403 차이

- HTTPS

- SSL 동작방식

- HSTS란

- SSL Stripping이란?

- GET과 POST의 차이

- 검색엔진이 뭐에요

- 웹 크롤러는?

- HTTP 멱등성

- 토큰 기반 인증

- JWT

- OAuth

- CORS란?

- CORS 해결방법

- JSON, XML의 차이

- MIME이란?

- AWS란?

OOP의 4가지 특징 (Object Oriented Programming) => 객체 지향 프로그래밍

우리가 실생활에서 쓰는 모든 것을 객체라 하며, 객체 지향 프로그래밍은 프로그램 구현에 필요한 객체를 파악하고 상태와 행위를 가진 객체를 만들고

각각의 객체들의 역할이 무엇인지를 정의하여 객체들 간의 상호작용을 통해 프로그램을 만드는 것을 말합니다.

즉, 기능이 아닌 객체가 중심이며 "누가 어떤 일을 할 것인가?"가 핵심이다.

추상화(abstract)

-공통의 속성이나 기능을 묶어 이름을 붙이는 것을 추상화라고 할수 있다.

-객체 지향적 관점에서 클래스를 정의하는 것을 추상하라고 할수 있다.

예를 들어 사자, 고양이 강아지가 있을때 우리는 이것을 각각의 객체라고 하며,

객체들의 공통적인 동물이라고 표현할 수있는데 이떼 동물로 묶는 행위를 추강화라고 한다.

캡슐화(encapsulation)

-특정 객체가 독립적으로 역할을 수행하기 위해 필요한 데이터와 기능을 하나로 묶는 것을 캡슐화라고 한다.

-쉽게 말해 모듈화를 의미한다. 이러한 캡슐화를 통해 정보를 객체 안에 포함시키고, 그정보에 대한 접근은

허용하지 않는 대신,

필요에 따라 확인할수 있는 인터페이스를 외부에 공개함으로써 정보 은닉 효과도 자연스럽게 따라온다.

상속(inherutance)

-상위 개념의 특징을 하위 개념이 물려받는것을 상속이라고 한다.

-상속은 코드의 재사용성을 높이고 확장성을 높여준다.

다형성(polymorphsim)

-같은 모양의 코드가 다른 행위를 하는 것을 나타낸다. 자바에서는 overriding, overloading이 그방법이다.

=>오버라이딩(overriding) - 상위 클래스에 있는 메소드를 하위 클래스에서 재정의 하는 것을 말한다.

=>오버로딩(overloading) - 매개변수의 개수나 타입을 다르게 하여 같은 이름의 메소드를 여러 개 정의하는 것을

말한다.

OOP의 5대 원칙 (SOLID)

SRP (단일 책임 원칙 - single responsebility principle)

-쉽게 말하면 하나의 클래스에 역할과 책임을 너무 많이 주지 말라는것이다.

-클래스에 모든 기능을 다 때려넣지 말고 목적과 취지에 맞게 속성과 메서드를 구성함으로 관련된 책임만

주어야한다.

OCP (개발 패쇄 원칙 - open/close principle)

-자신의 확장에는 열려있어야 하며, 주변의 변화에 대해서는 닫혀 있어야한다.

LSP (리스코프 치환 원칙 - Liskov substitution priciple)

-하위 클래스의 인스턴스는 상위 클래스의 인터슨터 역할을 하는데 문제가 없어야한다.

-인터페이스와 클래스 관계, 상위 클래스와 하위 클래스 관계를 얼마나 논리적으로 설계했는냐이다.

-하위 클래스가 상위클래스의 역할을 대신할 때 논리적으로 맞아 떨어져야한다.

ISP (인터페이스 분리 원칙 - interface segregaion principle)

-상황에 맞는 메서드만 제공하라는 것이다.

-어떤 거현 클래스는 자신이 구현하지 않은 인터페이스는 사용하지 않아야한다.

DIP (의족 역전 원칙 dependency inversion principle)

-추상 클래스 또는 상위 클래스는 구체적인 구현 클래스 또는 하위 클래스에게 의존적이면 안된다는것이다.

=================================================================================

동기 vs. 비동기

우선 차이점부터 설명하자면, 동기는 '직렬적'으로 작동하는 방식이고 비동기는 '병렬적'으로 작동하는 방식이다.

즉, 비동기란 특정 코드가 끝날때 까지 코드의 실행을 멈추지 않고 다음 코드를 먼저 실행하는 것을 의미한다.

동기(synchronous)'란 무엇일까?

직렬적으로 태스크를 수행하는 방식이다.

즉, 요청을 보낸 후 응답을 받아야지만 다음 동작이 이루어지는 방식이다. 어떠한 태스크를 처리할 동안 나머지 태스크는 대기한다.

실제로 cpu가 느려지는 것은 아니지만 시스템의 전체적인 효율이 저하된다고 할 수 있다.

비동기 (asynchronous)'란 무엇인가?

병렬적으로 태스크를 수행하는 방식이다.

요청을 보낸 후 응답의 수락 여부와는 상관없이 다음 태스크가 동작하는 방식이다.

a 태스크가 실행되는 시간 동안 b 태스크를 할 수 있으므로 자원을 효율적으로 사용할 수 있다.

이때, 비동기 요청시 응답 후 처리할 '콜백 함수'를 함께 알려준다.

따라서 해당 태스크가 완료되었을 때, '콜백 함수'가 호출된다.

=================================================================================

Java의 특징을 설명해주세요.

Java는 객체지향 프로그래밍 언어입니다.

기본 자료형을 제외한 모든 요소들이 객체로 표현되고, 객체 지향 개념의 특징인 캡슐화, 상속, 다형성이 잘 적용된 언어입니다.

장점

JVM(자바가상머신) 위에서 동작하기 때문에 운영체제에 독립적이다.

GabageCollector를 통한 자동적인 메모리 관리가 가능하다.

단점

JVM 위에서 동작하기 때문에 실행 속도가 상대적으로 느리다.

다중 상속이나 타입에 엄격하며, 제약이 많다.

=================================================================================

JVM의 역할에 대해 설명해주세요.

JVM은 스택 기반으로 동작하며, Java Byte Code를 OS에 맞게 해석 해주는 역할을 하고 가비지컬렉션을 통해 자동적인 메모리 관리를 해줍니다.

=================================================================================

Java의 컴파일 과정에 대해 설명해주세요.

개발자가 .java 파일을 생성한다.

build를 한다.

java compiler의 javac의 명령어를 통해 바이트코드(.class)를 생성한다.

Class Loader를 통해 JVM 메모리 내로 로드한다.

실행엔진을 통해 컴퓨터가 읽을 수 있는 기계어로 해석된다.(각 운영체제에 맞는 기계어)

=================================================================================

추상 클래스와 인터페이스를 설명해주시고, 차이에 대해 설명해주세요.

추상클래스는 abstract키워드로 선언되어 있거나 클래스에 하나 이상의 추상메소드가 포함되어있는 클래스입니다.

추상메서드가 없어도 추상클래스로 선언할 수 있으나 하나라도 있으면 무조건 추상클래스로 선언되어야 합니다.

기존의 클래스에서 공통적으로 사용될 수 있는 부분을 추상화 하여 추상클래스를 만듭니다.

단독으로는 객체를 생성할 수 없으며, 상속을 위한 상위 클래스로 활용하기 위해 사용합니다.

즉, 추상 클래스의 추상 메서드를 하위 클래스가 구체화하여 기능을 확장하는 데 목적이 있습니다.

인터페이스는 interface 키워드를 사용하여 선언하며, 추상메서드의 집합이라고 할 수 있습니다.

내부에 구현되어있는것이 전혀 없기 때문에 인터페이스를 implement한 클래스에서 추상메서드를 구현하여 사용합니다.

구현 객체의 같은 동작을 보장하기 위한 목적으로, 공통적으로 사용되는 기능을 명시하고 구현을 강제하기 위해 사용됩니다.

차이점

서로 다른 목적을 가지고 있습니다.

추상클래스는 추상메서드를 하위클래스가 구체화 하여 기능을 확장하는데 목적이 있습니다.

인터페이스는 정의된 기능 구현을 강제하여 상속받은 클래스들의 같은 동작을 보장하는데 목적이 있습니다.

추상클래스는 단일 상속이지만 인터페이스는 다중상속이 가능합니다.

=================================================================================

싱글톤 패턴에 대해 설명해주세요.

싱글톤 패턴은 단 하나의 인스턴스를 생성해 사용하는 디자인 패턴입니다.

인스턴스가 1개만 존재해야 한다는 것을 보장하고 싶은 경우와

동일한 인스턴스를 자주 생성해야 하는 경우에 주로 사용합니다. (메모리 낭비 방지)

싱글톤 패턴의 대표적인 예시는 Spring Bean 입니다.

스프링의 빈 등록 방식은 기본적으로 싱글톤 스코프이고, 스프링 컨테이너는 모든 빈들을 싱글톤으로 관리합니다.

=================================================================================

가비지 컬렉션(Garbage Collection)에 대해 설명해주세요.

가비지 컬렉션은 JVM의 메모리 관리 기법 중 하나로 시스템에서 동적으로 할당됐던 메모리 영역 중에서 필요없어진 메모리 영역을 회수하여 메모리를 관리해주는 기법입니다.

가비지 컬렉터란?

더이상 참조되지 않는 메모리인 가비지를 청소해주는 JVM의 실행 엔진의 한요소 입니다.

JVM은 new와 같은 연산에 의해 생성된 객체들 중에서 더이상 참조되지 않는 객체를 정리해줍니다.

가비지 컬렉터는 Heap 영역을 위주로 탐색하며 메모리를 정리해 줍니다.

=================================================================================

자바의 메모리 영역에 대해 설명해주세요.

자바의 메모리 공간은 크게 Method 영역, Stack 영역, Heap 영역으로 구분되고, 데이터 타입에 따라 할당됩니다.

-메소드(Method) 영역 : 전역변수와 static변수를 저장하며, Method영역은 프로그램의 시작부터 종료까지 메모리에 남아있다.

-스택(Stack) 영역 : 지역변수와 매개변수 데이터 값이 저장되는 공간이며, 메소드가 호출될 때 메모리에 할당되고 종료되면 메모리가 해제된다.

LIFO(Last In First Out) 구조를 갖고 변수에 새로운 데이터가 할당되면 이전 데이터는 지워진다.

-힙(Heap) 영역 : new 키워드로 생성되는 객체(인스턴스), 배열 등이 Heap 영역에 저장되며, 가비지 컬렉션에 의해 메모리가 관리되어 진다.

=================================================================================

클래스와 객체에 대해 설명해주세요.

클래스는 객체를 만들어내기 위한 설계도 혹은 틀 이라고 할 수 있고, 객체를 생성하는데 사용합니다.

객체는 설계도(클래스)를 기반으로 생성되며, 자신의 고유 이름과 상태, 행동을 갖습니다.

여기서 상태는 필드(fields), 행동은 메소드(Method)라고 표현합니다.

객체에 메모리가 할당되어 실제로 활용되는 실체는 '인스턴스'라고 부릅니다.

=================================================================================

생성자(Constructor)에 대해 설명해주세요.

생성자는 클래스와 같은 이름의 메소드로, 객체가 생성될 때 호출되는 메소드입니다.

명시적으로 생성자를 만들지 않아도 default로 만들어지며, 생성자는 파라미터를 다르게하여 오버로딩할 수 있습니다.

=================================================================================

Synchronized에 대해 아는 대로 말해주세요. (동기화)

여러 개의 쓰레드가 한 개의 자원을 사용하고자 할 때, 현재 데이터를 사용하고 있는 쓰레드를 제외하고 나머지 쓰레드들은 데이터에 접근할 수 없게 막는 개념입니다.

데이터의 thread-safe를 하기 위해 자바에서 Synchronized 키워드를 제공해 멀티 쓰레드 환경에서 쓰레드간 동기화를 시켜 데이터의 thread-safe를 보장합니다.

Synchronized는 변수와 메소드에 사용해서 동기화 할 수 있으며, Synchronized 키워드를 남발하게 되면 오히려 프로그램의 성능저하를 일으킬 수 있습니다.

=================================================================================

String, StringBuffer, StringBuilder의 차이를 설명해주세요.

String은 불변의 속성을 가지며, StringBuffer와 StringBuilder는 가변의 속성을 가집니다.

StringBuffer는 동기화를 지원하여 멀티 쓰레드 환경에서 주로 사용하며,

StringBuilder는 동기화를 지원하지 않아 싱글 쓰레드 환경에서 주로 사용합니다.

String, StringBuffer, StringBuilder의 차이와 장단점

=================================================================================

접근 제한자(Access Modifier)에 대해 설명해주세요.

변수 또는 메소드의 접근 범위를 설정해주기 위해서 사용하는 Java의 예약어를 의미하며, 총 4 가지 종류가 있습니다.

public - 접근 제한이 없다. (같은 프로젝트 내 어디서든 사용 가능)

protected - 해당 패키지 내, 다른 패키지에서 상속받아 자손 클래스에서 접근 가능하다.

(default) - 해당 패키지 내에서만 접근 가능

private - 해당 클래스에서만 접근 가능

=================================================================================

클래스 멤버 변수 초기화 순서에 대해 설명해주세요.

static 변수 선언부 : 클래스가 로드 될 때 변수가 제일 먼저 초기화 된다.

필드 변수 선언부 : 객체가 생성될 때 생성자 block 보다 앞서 초기화 된다.

생성자 block : 객체가 생성될 때 JVM이 내부적으로 locking( thread-safe 영역 )

=================================================================================

static에 대해 설명해주세요.

static 키워드를 사용한 변수나 메소드는 클래스가 메모리에 올라갈 때 자동으로 생성되며 클래스 로딩이 끝나면 바로 사용할 수 있습니다.

즉, 인스턴스(객체) 생성 없이 바로 사용 가능합니다.

모든 객체가 메모리를 공유한다는 특징이 있고, GC 관리 영역 밖에 있기 때문에 프로그램이 종료될 때까지 메모리에 값이 유지된 채로 존재하게 됩니다.

=================================================================================

Inner Class(내부 클래스)의 장점에 대해 설명해주세요.

1. 내부 클래스에서 외부 클래스의 멤버에 손쉽게 접근할 수 있다.

2. 서로 관련 있는 클래스를 논리적으로 묶어서 표현함으로써, 캡슐화를 증가시키고, 코드의 복잡성을 낮출 수 있다.

3. 외부에서는 내부 클래스에 접근할 수 없으므로, 코드의 보안성을 높일 수 있다.

=================================================================================

Error와 Exception의 차이를 설명해주세요.

Error는 실행 중 일어날 수 있는 치명적 오류를 말합니다.

컴파일 시점에 체크할 수 없고, 오류가 발생하면 프로그램은 비정상 종료되며 예측 불가능한 UncheckedException에 속합니다.

반면, Exception은 Error보다 비교적 경미한 오류이며, try-catch를 이용해 프로그램의 비정상 종료를 막을 수 있습니다.

=================================================================================

CheckedException과 UnCheckedException의 차이를 설명해주세요.

CheckedException은 실행하기 전에 예측 가능한 예외를 말하고, 반드시 예외 처리를 해야 합니다.

대표적인 Exception - IOException, ClassNotFoundException 등

UncheckedException은 실행하고 난 후에 알 수 있는 예외를 말하고, 따로 예외처리를 하지 않아도 됩니다.

대표적인 Exception - NullPointerException, ArrayIndexOutOfBoundException 등

RuntimeException은 UncheckedException을 상속한 클래스이고, RuntimeException이 아닌 것은 CheckedException을 상속한 클래스 입니다.

=================================================================================

Set과 Map의 타입이 Wrapper Class가 아닌 Object를 받을 때 중복 검사는 어떻게 할건지 설명해주세요.

hashCode() 메소드를 오버라이딩하여 리턴된 해시코드 값이 같은지를 보고 해시코드 값이 다르다면 다른 객체로 판단하고,

해시코드 값이 같으면 equals() 메소드를 오버라이딩하여 다시 비교합니다. 이 두 개가 모두 맞으면 중복 객체입니다.

=================================================================================

Vector와 List의 차이를 설명해주세요.

벡터는 데이터 삽입시 원소를 밀어내지만 리스트는 노드를 연결만 하기 때문에, 삽입 삭제 부분에서 리스트가 시간복잡도의 우위를 가집니다.

벡터는 랜덤부분접근이 가능하지만 리스트는 더블링크드리스트(노드가 양쪽으로 연결)로 되어있기 때문에 랜덤 접근이 되지 않습니다.

검색적인 측면에서는 벡터가 우위에 있습니다.

벡터는 리스트와 달리 항상 동기화되는 장점이자 단점을 가지고 있습니다. 멀티 쓰레드 환경에서 안전하게 객체를 추가하고 삭제할 수 있지만,

단일쓰레드 환경 일때도 동기화를 하기 때문에 List보다 성능이 떨어집니다.

=================================================================================

제네릭에 대해 설명해주시고, 왜 쓰는지 알려주세요.

제네릭은 데이터의 타입을 하나로 지정하지 않고 사용할 때마다 범용적이고 포괄적으로 지정한다는 의미입니다.

제네릭 타입을 사용함으로써 잘못된 타입이 사용될 수 있는 문제를 컴파일 과정에서 제거할 수 있어 에러를 사전에 방지할 수 있습니다.

=================================================================================

java의 컴파일과정을 설명하세요

컴파일이란 고급언어로 작성된 .java파일을 .class파일로 변환하는 과정을 말합니다.

컴파일 과정

첫번째로 프로그래머가 java언어로 소스코드를 작성합니다.

두번째 javac 컴파일러를 사용해 .java소스 파일을 컴파일하고, 바이트코드로 변환된 .class파일을 생성합니다.

세번째 컴파일된 바이트코드인 .class파일을 클래스로더에 전달합니다.

네번째로 클래스 로더는 동적 로딩을 통해 필요한 클래스를 로딩 및 링크하여 런타임 데이터 영역인 jvm의 메모리에 올립니다.

마지막으로 실행엔진은 jvm 메모리에 올라온 바이트코드들을 인터프리터 방식 혹은 jit 컴파일러 방식으로 실행합니다.

=================================================================================

객체지향이 무슨뜻?

어떤 프로그램의 일부분에 해당하는 작은 부품, 즉 객체를 먼저 만들고 이렇게 만들어진 여러 객체들을 조립해서

하나의 완성된 프로그램을 만드는 프로그래밍 방법론을 뜻한다.

객체란 현실세계의 실체 및 개념을 반영하는 상태와 행위를 정의한 데이터의 집합

객체지향 프로그래밍이란 각자의 역할을 지닌 객체들끼리 서로 메세지를 주고받으며 동작할 수 있도록 프로그래밍 하는 것

=================================================================================

업캐스팅 & 다운캐스팅

슈퍼 클래스의 변수에 서브 클래스의 객체가 들어가는 것을 업캐스팅이라 하고, 업캐스팅 된 변수의 타입을 서브 클래스로 변경하는 것을 다운 캐스팅이라고 합니다.

서브 클래스 객체는 슈퍼 클래스의 메소드 명세를 상속 받기 때문에 슈퍼 클래스의 변수에 들어가 슈퍼 클래스 인 것처럼 사용될 수 있고,

업 캐스팅 된 변수의 타입이 다시 서브 클래스로 돌아와 본인의 클래스 객체인 것처럼 사용할 수 있습니다.

=================================================================================

클래스, 객체, 인스턴스의 차이

클래스는 객체지향 프로그래밍에서 특정 개체의 속성과 행위를 정의하는 설계도라고 할 수 있습니다.

클래스는 멤버변수와 메서드로 구성되며 멤버변수는 해당 클래스의 객체들이 가질 수 있는 상태를 정의하고,

메서드는 객체들이 수행할 수 있는 행위를 정의합니다.

객체는 클래스에 의해 정의된 설계도를 따라 만들어진 실체로, 객체는 클래스의 인스턴스입니다.

객체는 메모리에 할당되며 각 객체는 고유한 상태를 가지고 클래스에 정의된 행위(메소드)를 수행할 수 있습니다.

인스턴스는 클래스의 객체를 의미합니다. 클래스를 통해 객체를 생성할 때마다 새로운 인스턴스가 생성되는 것입니다.

인스턴스는 클래스에 정의된 속성과 행위를 가진 객체로 각 인스턴스는 고유한 상태를 가집니다.

객체가 특정 클래스의 인스턴스라는 것은 해당 객체가 해당 클래스의 설계도에 따라 생성되었다는 것을 의미합니다.

즉, 클래스는 객체의 설계도이며, 객체는 클래스에 정의된 속성과 행위를 가진 실체입니다.

인스턴스는 클래스의 객체를 나타내는 용어로 사용되며 클래스로 부터 생성된 객체를 의미합니다.

=================================================================================

Call by Reference와 Call by Value의 차이

Call by Value방식은 함수에 인자를 전달할 때 인자값을 복사하여 전달하는 방식입니다.

이 방식에서는 원본 데이터와 전달된 복사본이 서로 다른 메모리 주소를 가지기 때문에,

함수 내에서 인자의 값을 변경해도 원본 데이터에는 영향을 미치지 않습니다. C, C++, Java, Python 등이 call by value 방식을 사용합니다.

Call by Reference방식은 함수에 인자를 전달할 때 인자의 메모리 주소를 전달하는 방식입니다.

함수 호출시에 인자로 전달되는 변수의 레퍼런스를 전달하며 해당 주소를 통해 원본 데이터를 직접 수정할 수 있습니다. C++, Swift, PHP 등이 call by reference 방식을 지원합니다.

꼬리) java의 기본 타입은 call by value고 참조타입은 call by reference다?

놉 java는 call by value로 작동합니다. java의 참조변수에는 원본 객체에 대한 참조를 값으로 복사하여 가지고 있습니다.

즉 변수가 가지는 값이 레퍼런스이므로 인자로 넘길 때 call by value에 의해 변수가 가지고 있는 레버런스가 복사되어 전달되는것 입니다.

함수 내에서 객체의 속성을 변경하게 되면 원래 객체에 영항을 줄 수 있지만 함수 내에서 전달된 참조변수 자체를 변경하면 호출자의 원래 참조에는 영향을 주지 않습니다.

=================================================================================

널과 엠티의 차이점?

Null

변수에 아무것도 할당되지(참조하지) 않은 상태를 의미합니다.

Empty

empty(비어있는) : 문자열의 길이가 0임을 의미합니다.

< 2023.08.14 >

- 용어 공부 많이~

- 유튜브 IT기술면접 영상~

- 신입사원을 위한 기술면접 수준 파악하기~

- 단어, 용어 검색 많이~

- 지금까지는 자바 문법위주로 했다면,이제는

금융 IT쪽에서 사용할 법한 업무기능 실습을 하게 될것임~

- 금융관련 용어, 단어에 대해서 공부를 많이 해야할것~

- 프론트, 백엔드 구분없이 다 할줄 알아야 한다.

- 금융IT는 백,프론트 구분 하지 않는다~

- 금융과 제일 중요한 것은 신뢰성, 안정성,

UI에서 문제 생겼을 시 전담할 상용 솔루션을 사용함

- 소프트웨어 공학,클린코딩,리팩토링,알고리즘,디자인패턴

\* 아키텍터 \*

- 아키텍처가 되려면 뭘 공부해야할까?

- 엔지니어부터 해라~ㅎ :-p

- 최소한의 엔지니어적인 역량을 갖추고 있는 것이 중요(기본소양)

- PM(프로젝트 매니저),PL(프로젝트 리더)

- 내가 70~80까지 코딩할 생각으로 인생 로드맵을 짜고 공부해야한다.

- 3년 뒤, 나의 연봉을 어떻게 만들어갈지 지금부터 잘 구상하고

준비해야한다.

- 여기까지가 첫 잔소리~

- 오늘 하루동안은 잔소리~

- 10월 16일 입사 후 11,12월 간단한 실습을 보낸 뒤

12월 이후 다시 교육과정을 해야한다. 그 뒤에 CBP 시험을 볼 예정

!!!! 매우 중요 !!!!

- CBP 코어뱅킹(이론,실기)시험을 반드시 합격해야 입사할 수 있다.

- 이번에는 교육과정이 3주 밖에 안되어 있기 때문에,

CBP 시험 과정이 포함되어 있지 않다. 입사 후~ㅎㅎㅎ

- 기술 면접 구술로 내가 아는 것을 입으로 설명하는 것

표현하는 방법, 말하는 태도가 매우 중요

- 발전 가능성을 중요하게 체크

하나를 가르쳤을 때 둘을 깨우칠 수 있을지

- 소프트웨어 공학에 대한 공부를 해야한다.

- 깨진 창문이론(사소한 하나하나가 모이면 큰 결과를 불러온다)

★알아볼 용어★

1. 예대마진 (예금과 대출을 하여 남는 이득)

2. 여신(대출,신용,증권,카드,돈을 빌려주는 것) /

수신(저축,예금)

3. 1,2,3 금융권의 차이

4. 채널(Channel) : 껍데기

5. SI와 솔루션SI의 차이점 검색

< 2023.08.16 >

1. 객체지향 설계원칙 중 단일 책임원칙에 대해서 설명해주세요

2. 빅테크 기업의 의미를 설명해 주세요.

3. 알고 있는 개발자의 종류를 적어주세요.

4. Java의 연산자 "=="와 "equals()"의 차이점을 설명해 주세요.

==연산자와 equals()메소드의 가장 큰 차이점. == 연산자는 비교하고자 하는 두개의 대상의 주소값을 비교하는데,

String클래스의 equals 메소드는 비교하고자 하는 두개의 대상의 값 자체를 비교한다는 점이다.

5. Java의 실수형 데이터 타입을 적어 주세요.

1. float

2. double

6. Java와 Spring의 관계를 설명해 주세요.

7. IDE에 대해서 설명해 주세요.

- 통합 개발 환경(Integrated Development Environment, IDE)이란

공통된 개발자 툴을 하나의 그래픽 사용자 인터페이스(Graphical User Interface, GUI)로

결합하는 애플리케이션을 구축하기 위한 소프트웨어입니다.

8. 테스트 주도 개발(TDD)에 대해서 설명해주세요.

9. 애자일 방법론에 대해서 설명해 주세요.

애자일(Agile) 방법론이란? 애자일(Agile)은 '기민한, 민첩한'이라는 뜻으로 일정한 주기를 가지고

빠르게 제품을 출시하여 고객의 요구사항, 변화된 환경에 맞게 요구를 더 하고 수정해나가는 탄력적인 방법론을 말합니다.

작업 계획을 짧은 단위로 세우고 제품을 만들고 고쳐 나가는 사이클을 반복함으로써 고객의 요구 변화에 유연하고도 신속하게 대응하는 개발 방법론이다.

10. 프론트앤드와 백앤드의 차이점을 설명해주세요.

==================================================================================

\* 신분상승의 기회! :-D \*

- 정보처리기사 -> PMP(국제공인 자격증) -> 정보시스템감리사 -> 정보관리기술사

(학사) (석사) (박사)

==================================================================================

- 요구사항(요건, Requirement)

성명

휴대폰번호

이메일주소

집주소

- 요구사항 추가

A,B반을 따로 공지할 필요가 있음.

=> 열(컬럼)을 추가

- 요구사항 추가2

혈액형별로 공지할 필요가 있음

==================================================================================

< 1교시 >

- 데이터 모델의 종류 : 개념 / 논리 / 물리

- 엔티티 : 예시) 비상연락망 자체가 하나의 엔티티, 만약에 A,B반으로 sheet를 나눴다면 2개의 엔티티로 나눈 것

- ERD : Entity Relation diagram

- 논리 데이터모델링의 절차에 대한 용어 공부

==================================================================================

< 2교시 >

- 엔티티(Entity)와 관계(Relationship)와 속성(Attribute)을 데이터 모델링의 3요소라 한다.

- 식별자 : 유니크하게 구분되는 학번 혹은 반 코드

- UML : 통합 모델링 언어

< UML을 사용하는 이유 >

하나의 시스템을 개발하기 위해 보통 여러 사람이 참여합니다.

참여인원 중에는 개발자, 사용자가 있고 때로는 설계자, 분석가가 있을 수도 있습니다.

이 때 개발하려는 시스템과 이 시스템의 기능, 구조 등을 참여인원 모두에게 설명하려면 어떻게 해야 할까요?

물론 말로도 설명이 가능하겠지만, 기능과 구조를 문서화하여 이해시키는 것이 훨씬 효율적일 것입니다.

하지만 이를 표준화된 양식을 사용하여 작성하지 않았을 때, 내부 작성 양식에 따라 작성한 문서를 사용자, 개발자,

설계자 모두가 이해하긴 힘듭니다. 또한 내부 작성 양식을 만들고, 적응하고 이해하는 데 자원이 소모되기 때문에

표준화된 양식을 사용하는 것이 바람직합니다.

UML에 따라 작성한 문서는 표준 양식에 의해 작성된 문서이기 때문에, 참여인원들이 이해하기 편하며,

효율적으로 의사소통 할 수 있습니다. 또한 한번에 이해하기 힘든 복잡한 시스템을 시각적으로 모델링하여

구조를 보다 알아보기 쉽습니다.

< UML의 종류 >

- 유스케이스 다이어그램: 사용자(Actor)의 관점에서 시스템의 기능, 상호작용과 그들간의 관계를 표현합니다.

- 클래스 다이어그램: 클래스의 속성, 메서드, 관계를 표현합니다.

- 액티비티 다이어그램: 프로그램의 시작과 끝을 정의하고, 작업의 수행과정을 단계적으로 정의합니다.

- 시퀀스 다이어그램: 객체간의 상호작용을 시간의 흐름에 따라 나타냅니다.

- 커뮤니케이션 다이어그램: 객체간의 상호작용을 메시지의 관점으로 나타냅니다.

- 상태 다이어그램: 객체가 취할 수 있는 상태와 상태변화를 표현합니다.

- 컴포넌트 다이어그램: 소프트웨어 컴포넌트들과 그들의 관계, 구조를 표현합니다.

- 배포 다이어그램: 프로그램의 아키텍처를 물리적인 관점에서 설계합니다.

- 패키지 다이어그램: 관련있는 요소들을 하나의 패키지로 묶고, 패키지 사이의 관계를 표현합니다.

- 명명규칙을 지켜야 한다

- 현장에서 보편적 사용하는 업무용어 사용, 명사 또는 명사구 사용

- 모델에서 유일한 이름이어야한다. 이름은 짧게

- 기본(원장) / 상세 / 내역 / 이력 / 관계 / 목록 -> 원장테이블의 보조

- 기본 : 중복되지 않는 가장 순수한 원천인 이름

- 명명규칙을 지키지 않아도 프로그램은 돌아간다.

- 그렇지만 여러명이 협업을 할때에 공통된 규칙을 정하지 않고 모델링을 했을 시에 혼동이 발생하기 때문에

정해진 약속, 규칙을 지키는 것이 좋은 프로그램을 짜는 시작이다.

- Key : 유니크하게 중복되지 않는 값이 제일 눈에 띄는 앞에 위치하도록 하는것이 코딩 센스

- 복합 키

- 데이터의 정합성, 결합성이 떨어지게 되면 데이터모델링의 신뢰도가 떨어진다.

- 엔티티 어커런스(Entity Occurrence) : 인스턴스의 집합을 엔티티 어커런스라 한다.

- 엔티티에 소속된 사물의 예, 엔티티 인스턴스 라고도 함.

- 엔티티 타입(엔티티), 엔티티 어커런스

- Super Type 엔티티 : 합집합 개념, 공통의 속성을 위한 하나의 엔티티.

- Sub Type 엔티티 :

- 개별 속성을 위한 복수개의 엔티티, 반드시 Super Type에 속하여야 함.

- 잠재적으로 null 값을 가지는 속성은 Sub Type으로 정의

\* null 과 Empty는 다른 것이다. null은 말그대로 없을 무, Empty는 껍데기는 존재

< 2023.08.17 >

- 관계(Relationship)

- 엔티티 간에 존재하는 상호 연관성을 "Cardinality"(경우의 수)와 "Relationship Name" 으로 표기.

- 행위에 의한 관계

\* 금융에서는 모든게 계약으로 통한다. \*

- 고객 : 신규한다. -> 계약 : 신규된다.

- 신규한다 신규된다는 서로 상반된 표현.

- 소유에 의한 관계

- 상품 : 갖는다 -> 조건 : 속한다.

- 물건의 종류를 "Cardinality"(경우의 수) 라고 한다.

- 데이터의 관점에서 행동이나 소유 관계에 의해서 종속이 정의된다.

- 관계는 업무규칙(Business Rule)을 기술.

- 공동명의라는 것은 두사람을 하나로 묶어서 설명한다.

- 계좌라는 입장에서는 어떠한 특정 한명만을 가진다.

- 엔티티의 그림 설명

- 1 대 N의 그림을 설명할 때 N 쪽으로 까마귀 발톱을 향하게 한다.

- 관계는 데이터상으로 관리대상이 되는 관계를 의미한다.

- 두 엔티티 간의 관꼐는 2개 이상일 수 있다.

- 관계명(Relationship Name)

- 두 엔티티 사이의 관계는 선으로 표시

- 관계선의 좌우, 상하에 관계명을 기록(시계방향으로 기록)

- 관계명 명명규칙

- 애매한 동사는 사용 x, 현재형으로 표현, 명확하게 표현

- 관계의 기수(경우의 수)

- 기수(Cardinality)의 발생 유형

- 최대 기수 3가지 유형

: 두개의 엔티티 간 관계에서, 한 엔티티의 한 어커런스를 기준으로 상대 엔티티의 관계 된 어컬너스가

최대로 몇개까지 있을 수 있는가

- 최소 기수 3가지 유형

: 두개의 엔티티 간 관계에서 , 한 엔티티의 한 어커런스를 기준으로 상대 엔티티의 관계된 어커런스가

적어도 몇개까지 있을 수 있는가

- 0 : 발생할 수도 있고 발생 안할 수도 있는 경우(Optional)

- 1이상 : 꼬 발생해야 하는 경우(Mandatory)

==================================================================================

- 관계검증1 : 중복제거

- 원형관계(Circular Relationship) 제거

- 사슬관계(Chain Relation) 제거

- 관계검증2 : M : N 관계 단순화

- 관계를 별도 엔티티로 분리 : 숨겨진 정보로부터 연관 엔티티 추출

- 다대다(N:N) 관계는 나빠!:-( 엔티티를 정규화해서 풀어줘야되!

- 관계검증3 : 1:1 관계정제(1:1관계는 테이블이 존재할 필요가 없다)

- 데이터 모델링 과정에서 엔티티 간의 관계가 1:1로 대응되었다는 사실은 해당 엔티티들의

기본키가 동일하다를 의미하며 1:1 관계를 가진 모델은 3가지 방법으로 표현할 수 있다.

1. 하나로 통합

- Mandatory 관계

- 기본키가 동일

- 대부분의 속성과 관계가 동일

2. 그대로 유지

- Optional 관계이ㅏ면서 대부분의 속성 및 관계가 상이한 경우

- 대응되는 속성이 값이 서로 상이한 경우

3. 새로운 Super Type 엔티티 추가

- 기본키가 동일하고 속성의 일부만 상이한 경우

- 관계별 엔티티 종류

- 관계를 기준으로 한 엔티티 종류

1. 기본형 엔티티(Fundamental Entiry)

2. 속성형 엔티티(Attributive Entiry)

3. 관련형 엔티티(Associative Entiry)

- 속성(Attribute)

- 엔티티가 관리해야 되는 데이터 항목으로 더 이상 분리될 수 없는 최소의 데이터 보관 단위

- 향후 물리 데이터 모델링 작업을 통하여 컬럼으로 변환

- 속성의 명명규칙

- "수식어 + 도메인 명" 형식을 사용하여 표준화

예) 데이터 항목 이름이 다르고 의미가 같은 경우 : 계좌개설일, 계좌신규일

데이터 항목 이름이 같고 의미가 다른 경우 : 일자(계좌신규일, 계좌해지일)

- 속성 타입(Attribute Type)

- 근본적으로 속성과 같은 의미

- 엔티티가 갖는 속성의 유형을 의미함

- 테이블/타이틀 : 엔티티 타입, 칼럼 : 속성타입, 칼럼값 : 엔티티 인스턴스

- 속성 인스턴스(Attribute Instance)

- 속성 타입의 값(Value)을 의미 함

- 도메인(Domain)

- 어떤 한 속성에 허용되는 업무적을호 의미있는 인스턴스 값의 집합

- 속성이 가질 수 있는 값에 대한 업무적 제약요건으로부터 파악된 특성

- 속성에 대한 데이터 Type과 크기, 제약사항을 지정한 것

- 모든 업무영역에서 같은 도메인 집합을 사용

- 도메인의 활용(중요성)

- 속성 값의 비교 기준

- 공통 검증 규칙

- 도메인의 종류

- 열거형 도메인 : 코드와 같이 인스턴스 값이 정해져 있다. 개수가 한정되어 있는것

- 제약형 도메인 : 인스턴스 값이 하한가와 상한가와 같이 값의 범주가 정해져 있으나

특정한 값을 정할 수 없고 개수도 제한이 없는 것

-도메인의 구성요소

- 도메인명, 데이터 타입, 자릿수, 인스턴스 기술서(열거형 도메인 경우)

- 속성 배치1

- 기본키에 종속되는 속성을 해당 엔티티에 배치

- 가능한 범위 내에서 부모 엔티티 쪽으로 배치

- 속성이 여러 개의 값을 가질 수 있는 경우, 새로운 자식 엔티티를 만들어 자식 엔티티의 속성으로 배치

- 속성 배치2

- 관계를 구체화 시켜주는 속성이 존재할 경우 관계를 엔티티로 전환시켜 해당 속성 배치

- 속성 배치3

- Super Type(추상클래스) 엔티티, Sub Type 엔티티 내의 속성 배치

- Super Type 엔티티에 Sub Type 엔티티의 구분항목을 속성으로 등록

- Sub Type 엔티티 모두에 포함되는 속성을 Super Type 엔티티로 이동

- 데이터를 정규화(찢) 하면 할 수록 모델이 안정화 된다.

-> 한번만 잘 만들어 놓으면 나중에는 유지보수가 쉬워진다.

-> NULL 값이 없어진다.

- 식별자(Key)

- 식별자란 한 엔티티 내에서 각각의 어커런스를 유일하게 구분할 수 있는 단일 속성 또는 속성 그룹을 말함

- 관련 용어

1. 후보키(Candidate Key)

: 엔티티 내에서 각각의 어커런스를 유일하게 구분할 수 있는 속성으로 하나 또는 하나 이상의

속성으로 구성됨, 즉 기본키가 될 수 있는 후보 속성 임

예) 주민등록번호

2. 기본키(Primary Key), 주요키, 일차키

: 어느 엔티티 어커런스를 식별하기 위한 속성을 의미 함

< 조건 >

- 유일하게 구분이 되어야함(Unique)

- 반드시 데이터 값이 존재해야 함(Not Null)

- 가능하면 변하지 않아야 함(Never Change)

- 가능하면 업무적으로 활용도가 높은 것으로 정의하여야 함

- 가능하면 길이가 짧은 것을 선정하여야 함

예) 사번등을 Primary Key로 지정, 이름은 중복(동명이인)이 있을 수 있어서 지정안함.

3. 대체키(Alternate Key)

: 대체키는 후보키 중에서 기본키로 선정되지 않은 속성을 말함

즉, Primary Key 이외에 엔티티 인스턴스를 Unique하게 식별하는 Key임

4. 복합키(Concatenate Key), 연결키

: 2개 이상의 속성으로 Primary Key를 구성한 Key

예) 고객번호 + 주문번호

5. 외부키(Foreign Key)

: 어떤 엔티티 A가 다른 엔티티 B와 관계를 갖는 경우, B 엔티티와 관계때문에 속성으로 갖게 되는

B 엔티티의 Primary Key를 말 함.

6. 합성키(Compound Key)

: 엔티티와 엔티티의 관계가 Many to Many 관계일 때 그 관계를 나타내는 엔티티의 Key

예) 서로 다른 PK를 합성, 부서ID + 프로젝트NO

- 주제영역(Subject Area)

- 엔티티를 도출하고 관리하는 단위영역

- 표현방법(Notation)

- ER 모델의 표기법

- ER 모델의 표기법은 다양하고, 모델링 도구 업체별로 다름

< 2023.08.18 >

- 논리모델과 물리모델의 차이

- 테이블 레이아웃은 물리모델의 결과물

- VARCHAR2 : 오라클에 있는 데이터타입

- 데이터 모델러가 제일 싫어하는 것? : NULL과 데이터의 중복~

- 논리모델에서 물리모델로 넘어갈 때에 성능을 고려하여

- 서브타입 모델의 변화

- 엔티티 합체와 분해

- 비정규화

- PK확정

- 테이블 파티션 확정 : 일(day)단위로 파티션을 설정할 수도 있다.

- 데이터 저장방법 확정

- 인덱스 설계

- 뷰 설계

- 시스템 속성 추가

- dbms : 데이터를 적재하는 방식(가장 최근에 쌓인 데이터의 성능을 높인다.)

- JOIN 되는 2개의 테이블을 굳이 하나로 합쳐서 만들지 않는 이유는 데이터의 관점에서는 모델의 엔티티를

분리해서 관리하는 것이 낫기 떄문에, view라는 테이블을 만들어서 두개의 테이블을 합쳐지는 결과물로 보여지게

해서 편의성,실용성을 높인다.

예) 뷰에서 select \* from view 쿼리문을 작성 하면 실제 물리적인 테이블은 2개이지만, 용도에 따라

가상의 테이블에 합쳐진 결과물을 통해 하나처럼 보여진다.

- 스키마 : 테이블에 접근하기 위한 권한, 권한을 가진 사람만 특정 해당 테이블에 접근할 수 있다.

데이터의 접근을 관리하는 기법, 통제하는 기법. 종류는 논리 스키마와 물리 스키마가 있다.

- 논리 스키마 : 가상 스키마. 실제 DB에는 없지만 가상으로 만들어 진 view 테이블, 논리모델은 dbms에 종속 x

- 물리 스키마 : 실제 DB에 존재하는 물리적인 테이블, 물리모델은 dbms에 종속 o

- 업무데이터를 이해한다는 것

- 누가, 언제, 어떤 상품을 어느 회사로부터 얼마에 구매했는가?

- 선행관계, 종속성 규칙 등을 관찰

- 업무데이터 분석 과정

- 업무의 이해 -> 업무 데이터의 이해 -> 데이터 구조 모델링 -> DB 생성

- 데이터 개념 필요성

- 개념 데이터를 정의하지 않고 엔티티를 도출해서는 고품질의 데이터 모델을 작성할 수 없음

- 명사를 대상으로 엔티티 정의에 맞는 것을 도출한다 하여도 후보 엔티티를 비슷한 성격으로 분류하고 통합하여

일관된 사상을 갖는 모델을 효율적으로 작성하려면, Bottom-up으로 작업하기 전에 데이터 개념 정립이 필요함.

- FSDM(Financial Services Data model)

- 핵심 내용

- 금융기관의 데이터는 크게 9가지의 Data 개념(Concept)을 근간으로 함

- 금융기관의 데이터는 9가지 Data 개념에 관한 데이터와 9가지 Data 개념간의 관계에 관한 정보로 구성

- 9가지 Data 개념에 관한 데이터는 Classification 정보와 Description 정보로 구성 됨

- 9 Concept

1. 이해 관계자(IP) :

- 금융기관과 접촉을 하고 관심이 있는 모든 당사자

2. 계약(AR) :

- 상품, 서비스 등의 가치를 교환하는데 관련된 룰이나 의무/권리를 나타내는 둘 이상의

개인이나 조직간의 잠재적, 실제적인 약정

3. 조건(CD) :

- 업무처리에 관한 구조화된 요건과 전제조건

4. 상품(PD) :

- 금융기관이 제조, 판매하는 금융상품 또는 서비스

5. 장소(LO) :

- 금융기관이 정보를 유지하고자 하는 장소, 목적지, 또는 지역

6. 분류(CL) :

- 특정한 업무정보를 분류하는 카테고리의 구조

7. 경영방침/전략(BD) :

- 금융기관의 업무/재무 계획, 정책, 절차와 일정 등 업무를 수행하고자 하는 방식과

경영환경에 대한 관계당사자의 지침/전략

8. 이벤트(EV) :

- 업무 수행의 한 부분으로 데이터에 영향을 미치는 사건

9. 자원(RI) :

- 업무를 수행하는 동안 금융기관이 운영, 소유하거나 사용하는 유형/무형의 가치가 있는 것

==================================================================================

\* ITA (정보기술 아키텍처) : 범정부 ITA 통합관리 시스템

- 다시 Entity

- 범주화(Classification)

- 추상화(Abstraction)

- CRM :

CRM이란 Customer Relationship Management의 약자이며, 한국어로 고객 관계 관리를 뜻

CRM은 매출과 직결되는 솔루션

- Entity 모델링의 고충

- 데이터 집합을 정의하기 쉽지 않다.

- 데이터의 본질을 볼 줄 알아야 한다.

- 추상화 수준의 결정

- 하위 트랜잭션 데이터로 상위 논리집합 발견하기

- 엔티티의 본질

- 수학적 개념 : 엔티티는 집합이다.

- 철학적 개념 : 엔티티는 본질이다.

- 속성 분석

- 검토할 사항

- 속성이 충분히 세분화되었는가?

- 속성이 일관성 있고 배타적인 값을 갖는가?

- 속성이 같은 어커런스에 대해 복수의 값들을 갖는가?

- 복수 어커런스가 이 속성에 대해 같은 값을 갖는다면, 그 값들이 업무에 의미가 있는 것인가?

- 이 속성이 도출 가능한 것이라면 그 값이 모든 어커런스에 적용 가능한 것인가?

- 검토1 : 세분화

- 속성이 충분히 세분화되었는가?

- 분리가능 한 속성은 최대한 분리해야 함.

- 검토2 : 배타성

- 속성이 일관성 있고 배타적인 값을 갖는가?

- 값의 배타성을 조사하여 배타적이지 않은 경우, 속성을 분리하여야 함

- 검토3 : 시간의 흐름

- 속성이 같은 어커런스에 대해 복수의 값들을 갖는가?

- 특정 시간에?

- 계속해서 변하는가? 과거의 기록이 중요한가?

- 만약 그렇다면 별도의 엔티티로 도출하여야 한다.

- 검토4 : 의미 있는 중복 값

- 복수 어커런스가 이 속성에 대해 같은 값을 갖는다면, 그 값들이 업무에 의미가 있는 것인가?

- 만약 그렇다면 별도의 엔티티로 분리하여야 함

예)엔티티 : 담보설정

- 검토5 : 도출가능한 속성

- 이 속성이 도출 가능한 것이라면 그 값이 모든 어커런스에 적용 가능한 것인가?

- 도출 가능한 속성은 제외함

- 정규화(Normalization)

- 정의 :

효율적인 관리를 위한 데이터구조를 도출하기 위하여, 속성간의 관계를 분석하여 별도 엔티티

도출 여부를 결정해 나가는 절차에 관련된 이론

- 효과 :

정규화의 기본 원칙은 하나의 테이블에는 중복된 데이터가 없도록 하는 것이며 효과는 다음과 같음

- 데이터 중복에 따른 정합성 오류 발생가능성 제거

- 성능 개선

- 정규화 규칙

- 효율적인 관리를 위한 데이터구조를 도출하기 위하여, 속성간의 관계를 분석하여 별도 엔티티

도출 여부를 결정해 나가는 절차에 관련된 이론

- 제 1 정규화 : 반복 또는 복수 값을 갖는 속성의 제거, 반복되는 패턴을 찾는다.

- 반복그룹의 속성을 별도의 엔티티로 분리가능

- 작업 절차

- 반복하는 속성 그룹으로 새로운 엔티티 작성

- 새로운 엔티티 키(Concatenate Key) 및 관계를 정의

- 제 2 정규화 : 기본키에 종속 되지 않는 속성의 제거(기본키의 일부에 종속되는 속성의 제거),

복합키가 있어야지만 정규화 가능

- 기본키에 종속되지 않는 모든 속성을 분리하여 새로운 엔티티를 구성

- 복합키(Concatenated Key)가 있는 엔티티에 적용됨

- 작업 절차

- 속성중에 Key의 일부에만 종속하는 속성을 분리

- 새로운 엔티티를 정의한다.

- 새로운 엔티티 키 및 관계를 정의

- 제 3 정규화 : 기본키가 아닌 속성에 종속적인 속성의 제거

- 엔티티의 속성 중에 기본 키가 아닌 속성에 종속하는 모든 속성을 분리하여 새로운

엔티티를 생성

- 작업 절차

- 속성 중에 Key가 아닌 속성에 종속하는 속성을 분리한다.

- 새로운 엔티티를 정의한다.

- 새로운 엔티티 키 및 관계를 정의

메타데이터의 관리

- 데이터 표준화

- 표준화란? : 이음동의어와 동음이의어를 관리하는 것

- 메타데이터란? :

- 데이터들의 데이터

- 통일시켜 일관되게 사용되도록 하는 것

- 주로 데이터를 담는 최소 단위인 속성의 표준화

- 메타관리 시스템에서 관리하는 수많은 시스템을 어떻게 통제할 것인가

- 표준용어 : 표준 단어들의 조합, 복합어의 허용 제한 필요

- \* 인덱스(Index) : 데이터를 더 빨리 찾아오기 위해서 구조화 해놓은 것(색인)

- 인덱스 기법 : 데이터를 적재할 때 무엇을 기준으로 적재할지?

예) 주민등록번호 오름차순 적재, 이름 가나다순 내림차순 적재

- 컬럼마다 인덱스 기준으로 나누지 않는 이유는 메모리를 굉장히 많이 차지하기 때문

- Full Scan : 인덱스를 잡는것 과는 상관없이 천만건이 있으면 정직하게 천만건을 다 탐색하는 것을 말한다.

예) 무식하게 하나하나 다 뒤져보는....쿼리속도가 아주 극악으로 떨어지는 결과를 초래한다.

(정렬이나 탐색방법을 통해서 빠르게 찾는방법을 이용해야지!)

- 메타데이터 관리 시스템

- 코드

- 코드인 것과 코드가 아닌 것, 식별자(보통 PK)와 식별자가 아닌 것을 구분해야 한다.

- 코드 : 오직 릴레이션의 튜플들에 대해 특정 기준 중심의 분류 / 집합.

- 표준화 : 식별자에는 ID나 번호, 코드에만 코드

- 개별코드와 통합코드

- 개별코드 : 유형만 관리, 일반화 가능

- 코드가 하나 생길 때 마다 테이블, 엔티티를 하나씩 추가 해줘야함.

- 코드가 날씬한 느낌

- 통합코드 : 실체 관리

- 코드가 하나 추가 되더라도 테이블이 추가되지 않고 엔티티에 한줄 추가.

- 코드가 뚱뚱한(온갖 엔티티가 다 들어가 있으니까~) 느낌

- 사용하는데 있어서 득과 실이 있으므로 사용에 고민해 보아야함.

- 데이터를 통합 관리함으로써, 등록 / 참조 / 분배 등 관리상 장점

- 코드 사용빈도가 하나의 테이블에 응집 - 메모리 성능상의 이점 기대

- 어플리케이션 아키텍처를 고려한 선택 필요

- 모델링 프로세스 실무

- 모델링 / 테이블 작성 프로세스

\* 형상관리 프로그램 : SVN, Git Hub

- 데이터의 품질관리

- 품질관리의 기준

- 모델링에는 정답이 없다. 평가 및 측정 또한 개발 후 운영되어지면서 증명

- 현황에 대한 다양한 기준을 마련하여, 해당 기준 충족여부를 통해 품질 가늠

- 모델링 품질 기준

1. 표준화 관점 : 업무에서 결정된 표준항목의 준수 여부로 판단

- 표준용어와 표준 도메인 사용여부 확인

- 네이밍룰과 필수 입력항목의 준수여부 확인

2. 모델링 방법론 관점 : 일반적인 모델링 룰의 준수 여부로 판단

- 중복여부 확인

- 관계선 지정 룰 준수 여부 확인

- 인덱스 룰 확인

- 정규화 여부 확인

3. 업무적 관점 : 기업의 업무를 정확하게 표현했는지의 여부로 판단

- 담당 업무 DA들에 의한 업무 표현의 정확도 확인

- 역할별 품질관리의 범위

- CIO / EDA (개괄적 관점)

- DA (개념적 관점)

- 모델러 (논리적 관점)

- DBA (물리적 관점)

- 사용자 (운용적 관점)

- 데이터모델링 도구

- 데이터모델링 지원 툴

- eXERD : 토마토 시스템

- DA# : 엔코아

- MS Visio : 그림그리는데 특화 된 툴

- 인증제도

- 한국데이터산업진흥원(K-data)에서 시행

- 데이터 가치 증대를 위한 데이터 품질 / 관리 / 보안 체계에 대해 심사, 인증하는 제도

- 인증대상

- 데이터품질인증 : 유동, 활용, 개방 데이터의 품질 향상을 위한 인증

- 데이터관리인증 : 효과적인 데이터 활용을 위한 데이터 거버넌스 정책, 데이터 관리 체계 인증

- 데이터보안인증 : 데이터 보안 위협의 선제적인 대응과 안전한 데이터 비즈니스를 위한 인증

\* 정보처리기사, SQLD 자격증 따라~ 꼭 따라~ \*