|  |  |
| --- | --- |
| 운영체제 | **운영체제란 컴퓨터 시스템의 자원들을 효율적으로 관리하며 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 시스템입니다.**  사용자와 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로 다른 응용프로그램이 유용한 작업을 할 수 있도록 환경을 제공해줍니다. |
| 프로세스  메모리구조 | 1. 코드영역 : 실행할 프로그램의 코드가 저장되는 영역, CPU는 해당코드 영역에 저장된 명령어를 하나씩 가져가서 처리  2. 데이터 영역 : 전역변수와 정적변수가 저장되는 영역으로 프로그램이 시작하는 동시에 할당, 프로그램이 종료되면 소멸  3. 힙 영역 : 사용자가 직접 관리하는 영역이며 메모리 공간이 동적으로 할당 및 해제  (런타임) 동적으로 할당되므로 new로 생성한 object가 많아질수록 영역의 크기 증가  4. 스택 영역 : 함수의 호출에 따른 지역변수와 매개변수가 저장되는 영역 컴파일시  (컴파일) 크기 결정, 함수의 호출과 함께 할당, 종료되면 소멸 |
| 프로세스 | **프로세스란 프로그램이 돌아가고 있는 상태 즉 컴퓨터에서 작업중인 프로그램을 의미합니다.**  ● 프로세스별 각각의 영역을 가지기 때문에 별다른 동기화 작업이 필요하지 않음  ● 한 프로세스에서 오류가 생겨도 다른 프로세스에는 영향을 주지 않음 |
| 쓰레드 | **쓰레드는 한 프로세스 내의 실행 단위, 실행의 흐름을 의미합니다.**  ● Stack 영역만 별도로 가지고 나머지에 대해서는 해당 쓰레드를 포함한 프로세스의  자원을 공유  ● stack 영역 외부의 공유 자원에 대해 접근 시 동기화 작업이 필요 |
| 멀티프로세스,  멀티 쓰레드 | **멀티 프로세스란 두개 이상 다수의 프로세서가 협력적으로 하나 이상의 작업을 동시에**  **처리하는 것이고, 멀티 스레드는 하나의 프로세스에 여러 스레드로 자원을 공유하며 작업을 나누어 수행하는 것입니다.**  멀티 프로세스는 하나의 프로세스가 죽더라도 다른 프로세스에 영향을 주지 않아 안정성이 높지만 멀티 스레드보다 메모리 공간과 CPU 시간을 차지하는 단점이 있습니다. |
| 동기, 비동기 | 동기화란 요청과 결과가 동시에 일어난다는 말로 요청을 하면 시간이 얼마가 걸릴지라도 바로 결과가 주어져야 합니다.  비동기화는 요청과 결과가 동시에 일어나지 않는 것이며 작업을 동시에 처리할 수 있어 처리 속도를 향상 시킬 수 있습니다.  동기화는 설계가 간단하고 직관적이지만 결과가 주어지기 전까지 대기해야 하는 단점이 있고, 비동기화는 복잡하지만 자원을 효율적으로 사용할 수 있습니다. |
| 데드락 | **데드락이란 두 개 이상의 프로세스나 스레드가 서로 자원을 얻지 못해서 다음 처리를 하지 못하는 상태로 무한히 다음 자원을 기다리게 되는 상태를 말합니다.**  데드락의 특징  1. 상호 배제 : 자원은 한번에 한 프로세스만 사용  2. 점유 대기 : 하나의 자원을 점유하고 있으면서 다른 자원을 대기하는 프로세스가 존재  3. 비 선점 : 다른 프로세스에 할당된 자원은 사용이 끝날 때 까지 강제로 뺏을 수 없음  4. 순환 대기 : 프로세스의 집합에서 순환 형태로 자원을 대기하고 있어야함 |
| 프로세스 스케줄링 | **프로세스 스케줄링이란 시스템의 자원을 효율적으로 사용하기 위해 어떤 프로세스를 먼저 처리할지 순서를 정하는 것입니다.**  선점형 스케줄링 기법 : Round Robin, SRF  비 선점형 스케줄링 : Priority Scheduling, FCFS, SJF |
| 선점, 비선점 | **선점형 스케줄링이란 실행중인 프로세스나 스레드를 강제로 중단시키고 다른 프로세스를 실행시키는 것입니다.**  **비선점형 스케줄링은 실행중인 프로세스가 있다며 다른 프로세스들이 CPU를 강제로 뺏을 수 없는 스케줄링 방식입니다.** |
| Blocking Non-Blocking | ???????????????? |
| G C  (가바지 컬렉션) | 가비지 컬렉션이란 동적으로 할당했던 메모리 영역 중 필요 없게 된 메모리 영역을 주기적으로 삭제하는 프로세스를 말합니다. |
|  |  |
| OSI 7계층 | **응용 계층** : 최종 목적지로 서비스를 수행하는 계층  **표현 계층** : 데이터 형식을 정의합니다.  **세션 계층** : 데이터가 통신하기 위한 논리적 연결을 담당하는 계층  **전송 계층 [세그먼트]** : 사용자들에게 신뢰성 있는 데이터를 전달하는 계층 ( TCP, UDP)  **네트워크 계층 [패킷]** : IP를 지정하고 라우터로 경로를 선택해 네트워크를 통해 데이터 전달하는 계층  **데이터 링크 계층 [프레임]** : 신뢰성 있는 전송을 위해 에러 검출, 흐름제어를 담당 물리적인 전송  **물리 계층 [비트]** : 데이터를 전기 신호로 변경하는 계층 |
| DNS,  DDNS | **DNS란** 도메인 네임 시스템의 약자로 IP주소를 사용자가에게 편리한 문자로 표현된 도메인주소로 바꿔주고 반대의 역할도 수행주는 시스템을 말합니다.  **DDNS란** Dynamic 도메인 네임 시스템으로 유동적으로 변하는 IP에 도메인주소를 매칭시켜주는 시스템을 의미합니다. |
| TCP / UDP | **TCP는** Transmission Control Protocol의 약자로 신뢰성 있는 통신을 위해 사용하는  프로토콜로 높은 신뢰성을 보장하지만 UDP보다는 속도가 느립니다. 3way, 4way handshaking으로 서버와 클라이언트가 1:1로 통신을 하고 흐름제어와 혼잡제어가 이루어 집니다.  **UDP는** User Datagram Protocol의 약자로 비연결형 프로토콜입니다.  손상된 데이터에 대해서 재전송하지 않고 신뢰성이 낮아서 스트리밍 같은 서비스에서 사용되며 1:1, 1:N, N:M연결이 가능합니다. |
| IP | **IP는** Internet Protocol의 약자로 데이터 패킷이 네트워크를 통해 이동하고 올바른 대상에 도착할 수 있도록 데이터 패킷을 라우팅하고 주소를 지정하기 위한 규칙의 집합입니다.  비연결성과 비신뢰성의 특징을 가지고 있습니다. |
| 로드밸런서 | **로드밸런서**는 서버에 가해지는 부하를 분산해주는 장치 또는 기술을 통칭합니다.  클라이언트와 서버풀 사이에 위치하며 트래픽을 관리해줍니다. |
| HTTP  /  HTTPS | **HTTP란** Hyper Text Transfer Protocol의 약자로 월드 와이드 웹의 토대이며 하이퍼텍스트 링크를 사용하여 웹페이지를 로드하는데 사용됩니다.  **HTTPS란** HTTP의 보안상 문제를 해결하기 위해 나온 프로토콜입니다.  Body부분만 암호화합니다. |
| HTTP Method | **GET** : 조회  **POST** : 요청된 데이터를 처리하고 자원 생성  **PUT** : 요청된 자원이 있으면 갱신 없으면 생성  **PATCH** : 수정  **DELETE** : 삭제 |
| GET / POST | **GET** 방식이란 요청을 전송할 때 필요한 데이터를 Body에 담지 않고 쿼리스트링을 통해 전송하는 방식을 말합니다.  **POST** 방식은 전송해야 할 데이터를 HTTP메시지의 Body에 담아서 전송합니다.  GET 방식은 길이 제한이 있고 보안에 취약하고 멱등성을 보장합니다.  POST방식은 길이 제한이 없고 멱등성을 보장하지 않습니다. |
| 멱등성 | **멱등성이란** 여러 번 적용하더라도 결과가 달라지지 않는 성질을 의미합니다. |
| 정적 라우팅  /  동적 라우팅 | **정적 라우팅이란** 네트워크 경로를 수동으로 구성하고 선택하는 것을 말합니다.  네트워크 정체가 나타날 수 있으며 네트워크의 유연성을 저하시켜 성능을 제한합니다.  **동적 라우팅이란** 네트워크 조건에 따라서 런타임에 라우팅 테이블을 만들고 업데이트합니다.  목적지까지 가장 빠른 경로를 찾으려고 시도합니다.  트래픽 볼륨, 네트워크 장애 등 변화하는 조건에 대처가 가능합니다. |
| 쿠키  /  세션 | **쿠키란** 브라우저 로컬에 저장되는 키와 값이 들어있는 작은 데이터 파일입니다.  유효시간을 정할 수 있으며 브라우저가 종료와는 상관없이 시간에 영향을 받습니다.  **세션이란** 사용자의 데이터를 서버에 저장되는 데이터파일입니다.  브라우저를 종료할 때 까지 인증상태를 유지하비다. |
|  |  |
| DBMS | **DBMS란** DataBase Management System의 약자로 데이터베이스 관리 시스템으로 여러 사용자가 데이터 베이스에 접근해 사용할 수 있도록 해주는 소프트웨어를 의미합니다. |
| 스키마 | **스키마란** 데이터베이스의 구조와 제약조건에 관한 전반적인 명세를 기술한 메타데이터의 집합입니다. |
| 뷰 | **뷰는** 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 테이블에서 유도된, 메모리에는 물리적으로 존재하지 않는 가상 테이블입니다. |
| 키 | 슈퍼키 ( Super Key )  후보키 ( Candidate Key )  기본키 ( Primary Key )  대체키 ( Alternate Key )  외래키 ( Foreign Key ) |
| 트랜잭션 | **트랜잭션이란** 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기위한 작업의 단위  또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미합니다.  1, **Atomicity (원자성)** : 연산 전체가 모두 실행 or 모두 취소  2. **Consistency (일관성)** : 성공적으로 완료하면 일관성 있는 데이터베이스 상태 유지  3. **Isolation (고립성)** : 두 개 이상의 트랜잭션이 발생할 때 영향을 주면 안된다.  4. **Durability (지속성)** : 커밋 된 트랜잭션의 내용은 영구히 반영된다. |
| DELETE  TRUNCATE DROP | **DELETE** : 데이터는 지우지만 테이블 용량은 줄어들지 않고 원하는 데이터만 삭제가능  **TRUNCATE** : 전체데이터를 모두 삭제하는 방식, 테이블 용량이 줄어든다.  **DROP** : 테이블 자체를 완전히 삭제하는 방식 |
| DDL DML DCL | **DDL (조작어)** : DB내의 자료검색, 삽입, 갱신, 삭제를 위한 언어  ( SELECT, INSERT, UPDATE, DELETE)  **DML (정의어)** : 데이터베이스 구조를 정의, 수정, 삭제하는 언어  ( ALTER, CREATE, DROP)  **DCL (제어어)** : 데이터에 대해 무결성유지, 병행 수행 제어, 보호와 관리를 위한 언어  ( COMMIT, ROLLBACK ) |
| DB 인덱스 | **Index**란 테이블을 처음부터 끝까지 검색하지 않고 인덱스를 검색해서 해당 자료의 테이블을  접근하는 방법을 의미합니다. |
| 정규화 | **1정규화** : 테이블의 칼럼이 하나의 값을 갖도록 테이블을 분해하는 것  **2정규화** : 기본키가 2개이상의 속성으로 이루어진 경우, 종속성을 제거한다.  **3정규화** : 기본키를 제외한 컬럼간의 종속성을 제거한다. |
| 수직 파티셔닝  /  수평 파티셔닝  (샤딩) | **파티셔닝은** 데이터베이스 내에서 데이터를 논리적으로 나누는 것이며,  **샤딩은** 데이터 베이스를  물리적으로 분산시키는 것입니다.  파티셔닝은 주로 데이터 관리와 성능 최적화를 위해 사용되고 샤딩은 데이터베이스 시스템의 확장성을 향상시키기 위해 사용됩니다. |
| ORM | **ORM** 이란 Object -RDBMS – Mapping의 약자로 객체클래스와 RDB에서 사용하는 데이터 테이블을 서로 매핑 시킨 것을 의미합니다. |
|  |  |
| JVM | **JVM이란** Java Virtual Machine으로 운영체제와 Java Application의 사이를 중재해주는 가상머신입니다. |
| Java 실행과정 | 자바 소스코드(.java)를 컴파일러에 의해 바이트코드 파일(.class)로 변환  Class Loader를 통해 바이트코드 파일(.class)을 Runtime Data Area로 로딩  로딩된 파일을 Execution Engine를 통해서 해석 및 실행 |
| JVM 메모리 구조 | 1. Method Area : 클래스에 대한 정보를 저장 클래스, 인터페이스, 메서드 저장  2. Heap Area : 런타임시 동적으로 할당하여 사용하는 영역  3. Stack Area : 스레드마다 존재하여 스레드가 시작할 때 할당 지역변수, 매개변수  4. PC Register : 스레드가 생성될 때마다 생성되는 영역 스레드가 현재 실행할 프레임의 주소 저장  5. Native Method Stack : C / C++등의 Low Level 코드를 실행하는 스택 |
| ==와 equals | **==은** 참조비교로 두 객체가 같은 메모리 공간을 가리키는 지를 확인하는 연산입니다.  **Equals는** 두 객체의 내부 값이 같은 지 내용을 비교합니다. |
| Java의 접근 제한자 | 1. **private** : 자기 자신 클래스 내에서만 접근 가능  2. **default** : 동일한 패키지 내에서만 접근 가능  3. **protected** : 동일한 패키지 내에서만 접근 가능 + 상속을 이용한 접근 가능  4. **public** : 접근에 제한이 없음 |
| 원시 자료형 (Primitive Type)  /  참조 타입  (Reference Type) | **Primitive Type은** 기본 타입의 크기가 작고 고정적이기 때문에 Stack영역에 저장됩니다.  **Reference Type은** 데이터의 크기가 가변적이고 동적이므로 Heap영역에서 관리됩니다.  데이터는 Heap영역에서 관리되지만 주소 값은 Stack 영역에 저장됩니다. |
| 자바의 제네릭 | **제네릭은** 데이터의 타입을 일반화하는 것을 의미합니다. 클래스나 메서드에서 사용할 데이터의 타입을 컴파일시에 미리 지정하는 방법입니다. |
| 추상 클래스  /  인터페이스 | **추상 클래스는** 추상 메서드를 선언해 놓고 상속을 통해 자식 클래스에서 메서드를 완성하도록  유도하는 클래스로서 미완성 설계도라고 표현합니다.  **인터페이스도** 추상 클래스와 비슷하게 다른 클래스를 작성하는데 도움을 주는 목적으로 작성합니다. 다만 인터페이스는 추상클래스보다 추상화 정도가 높아 추상클래스와 다르게 구현부가 있는 일반 메서드, 일반 변수 멤버 등을 가질 수 없습니다. 기본설계도라고 부릅니다. |
|  |  |
| String  String Builder  String Buffer | **String**은 final char[] 형태이기 때문에 변경이 불가능합니다. 따라서 + 연산을 사용하면 두 String을 연결한 새로운 객체를 만들어서 저장하기 때문에 메모리 관리측면에서 치명적입니다.  **String Builder와 Buffer는** 가변성을 가지는 클래스로 문자열을 변경하는 것이 가능합니다.  다만 Builder는 동기화를 지원하지 않지만 속도가 빠르고 Buffer는 동기화를 지원하기 때문에 멀티 스레드 환경에서 사용이 가능합니다. |
| new String (“”)  “” 차이 | “”으로 문자열을 초기화 하게 되면 Java Heap 메모리의 String pool에 저장되게 됩니다.  반면에 new String(“”)은 String pool이 아닌 Heap영역에 각각 생성됩니다.  따라서 Heap영역에 새로운 주소지를 참조하게 됩니다. |
| Java 리플렉션 | **리플렉션이란** 컴파일 시간이 아닌 런타임 시간에 메모리에 올라간 클래스나 메소드의 정의를  동적으로 찾아서 조작할 수 있는 기술을 의미합니다. |
| Java Optional | “NullPointerException”을 피하기 위해 null여부를 검사하게 되는데  Optional<T> 클래스를 사용해 NPE발생을 방지하도록 도와줍니다. |
| 자바 8  /  자바 11 | **자바 8** : 람다 표현식, Stream, interface와 default, Optional  **자바 11** : String, File 클래스의 메서드 추가, 람다 표현식 var키워드 사용가능 |
| DAO DTO VO | **DAO란** Data Access Object의 약자로 데이터베이스에 접근하기 위한 객체이며 데이터베이스 접근을 하기 위한 로직과 비즈니스 로직을 분리하기 위해 사용합니다.  **DTO란** Data Transfer Object의 약자로 계층간 데이터 교환을 위한 자바 빈즈입니다.  **VO는** Value Object로 Read-Only 속성을 지닌 값 오브젝트입니다. |
| 오버 라이딩  /  오버 로딩 | **오버 라이딩이란** 부모클래스로부터 상속받은 메소드를 자식 클래스에서 재정의 하는 것입니다.  부모클래스의 메소드를 재정의하는 것이므로 메소드의 이름, 매개변수, 리턴 값이 모두 같아야 합니다.  **오버 로딩이란** 자바의 한 클래스 내에 이미 사용하려는 이름과 같은 이름을 가진 메소드가 있더라도 매개변수의 개수 또는 타입이 다르면 같은 이름을 사용해서 메소드를 정의할 수 있습니다. 리턴 값만 다른 것은 오버로딩이 불가능합니다. |
| 단위 테스트 | **단위 테스트란** 테스트 가능한 가장 작은 소프트웨어를 실행하여 예상 동작을 검증하는 테스트  테스트 대상 단위는 클래스, 메소드가 있습니다.  테스트 단위의 크기가 작을수록 복잡성이 낮아지고 테스트 활용이 더 쉬워집니다.  대표적인 단위테스트 프레임워크는 Junit이 있습니다.  **TDD** ( Test Driven Development), 테스트 주도 개발은 선 개발 후 테스트를 진행하는 방식이 아닌  선 테스트 후 개발 방식의 프로그래밍 방법을 말합니다. |
| 통합 테스트 | **통합 테스트란** 단위 테스트보다 더 큰 동작을 위해 여러 모듈을 모아 의도대로 협력하는 지 검증하는 테스트입니다. 단위 테스트보다 더 많은 코드를 테스트하기 때문에 신뢰성이 떨어지고  에러 발생시 확인하기 쉽지 않아서 유지보수가 힘들다는 단점이 있습니다. |
| 인수 테스트 | **인수 테스트란** 사용자 시나리오에 맞춰 수행하는 테스트입니다.  단위 / 통합 테스트와 달리 비즈니스에 초점을 둡니다.  주로 API를 확인하는 방식으로 이뤄집니다. |
| Enum(열거형) | **Enum 클래스**란 Enumeration의 약자로 열거형 클래스입니다.  서로 관련이 있는 상수들끼리 모아 상수들을 정의한 것입니다. |
| 리팩토링 | **리팩토링이란** 코드의 의미와 결과의 변경 없이 코드의 구조를 바꾸는 것으로  주로 가독성을 높이고 유지보수를 편하게 하기 위해 사용하며 버그를 없애거나 기능을 추가하는 행위는 아닙니다. |
| Array  /  ArrayList  /  LinkedList | **Array는** 크기가 고정되고 데이터의 순서를 의미하는 Index가 존재하여 랜덤 액세스가 빠르지만 요소 추가 및 삭제가 느립니다.  **ArrayList는** 크기가 가변적이며 데이터의 순서를 의미하는 Index가 존재하여 랜덤 액세스가 빠르며 동적으로 크기를 조절할 수 있습니다.  **LinkedList는** 각 노드를 서로 연결시켜 만든 구조로 요소 추가 및 삭제가 빠르지만 랜덤 액세스가 느리며, 순차적으로 탐색해야 합니다. |
| Stack  /  Queue | **Stack은** FILO (First In Last Out)의 특성을 가진 자료 구조입니다. 가장 마지막에 추가된 요소가 가장 먼저 나가게 됩니다.  **Queue는** FIFO (First In First Out)의 특성을 가진 자료 구조입니다. 처음에 추가된 요소가 먼저 나가게 됩니다. |
| 해시 | **해시**는 자료를 효율적으로 저장하고 검색하기 위한 자료 구조입니다.  Key – Value로 쌍을 사용하여 데이터를 저장하고 특정 키를 사용하여 데이터를 빠르게 검색하는데 사용됩니다. Key 값을 이용해서 데이터를 검색하기 때문에 Key값은 중복이 불가능합니다.  해시 충돌 회피 기법으로는 체이닝 방식과 Open Addressing이 있습니다.  **체이닝 방식**은 하나의 Value 값을 가지는 것이 아닌 Value에 LinkedList를 이용해서 여러 개의 값을 저장하는 방법입니다.  **Open Addressing**은 중복되는 Key값을 바꿔서 저장하는 방식입니다. [ |
| Static  /  Dynamic |  |
| 클래스 |  |
|  |  |
|  |  |