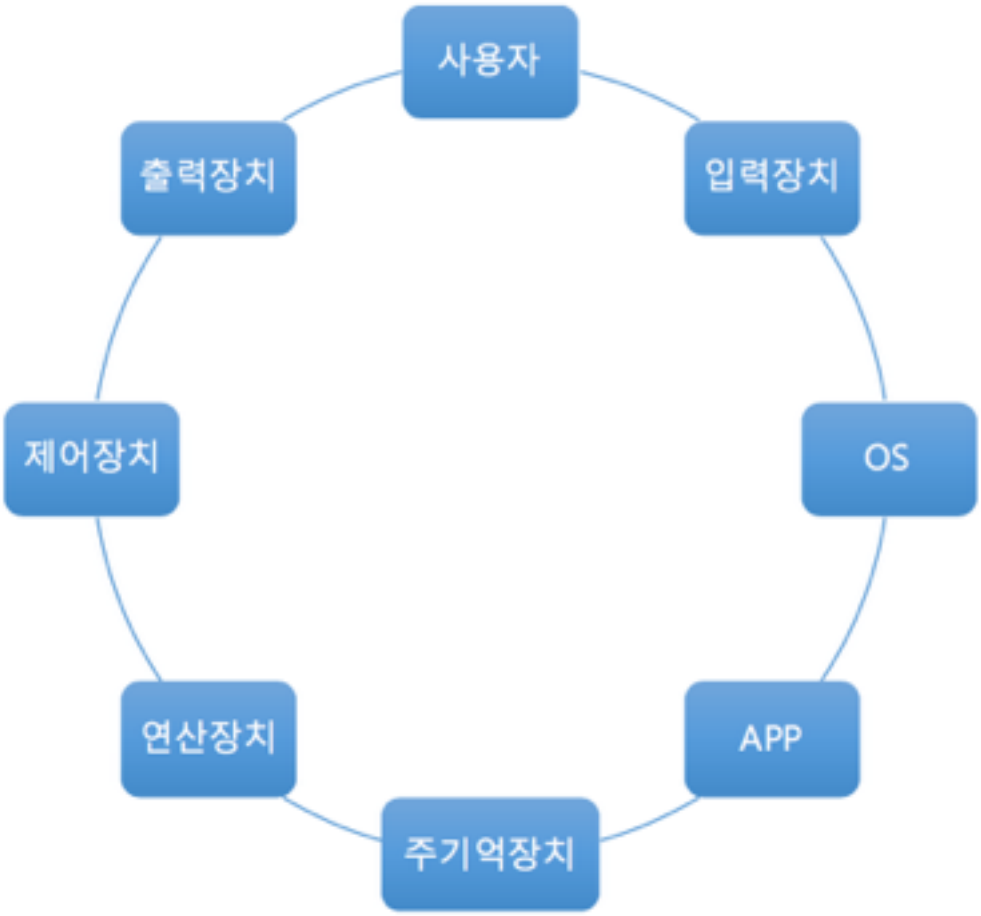
프로그래머: 프로그래밍을 하는 사람
프로그래밍: 컴퓨터가에 일련의 명령어를 짜는 행위
컴퓨터프로그램: 일련의 명령어 모음
컴퓨터: 전자계산

컴퓨터의 역사
진공관: 최초 연산을 위해 켜짐/꺼짐으로 연산을 했던 장치
트렌지스터: 진공관의 발전된 형태
집적회로: 회로를 하나의 칩으로 하여 현재의 반도체(cpu)

데이터의 표현
정수, 실수, 문자 - 최소한의 자료구조

컴퓨터 구조
폰 노이만 구조: 현재 우리가 사용하는 대부분의 컴퓨터의 구조



이 구조를 통해서 알게된 것은 운영체제가 내가 아는 것 보다 훨씬 더 많은 일을 수행한다는 것이다.
예를 들면 클릭을했을때 어느 위치를 클릭했는지(버튼인지, 입력상자인지, 콤보상자인지) 그리고 입력된 정보를 처리할 수 있도록 ram, cpu에 전송 하고 다음에 컴퓨터가 어떤 행위를 해야하는지 등등 이러한 부분을 운영체제가 해주지 않았다면, 프로그래머가 모든 행위에 대한 명령어를 일일히 입력해야하고 하드웨어에 대한 제어도 직접적으로 실행해야 한다.

운영체제(추가 문서를 보고 정리한 내용)

시스템 하드웨어를 관리할 뿐 아니라, 응용소프트웨어를 실행하기 위하여 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어이다
(컴퓨팅 플랫폼: 소프트웨어가 구동 가능한 하드웨어 아키텍처나 스프트웨어 프레임워크의 종류를 설명. 예를들면 컴퓨터의 아키텍쳐, 운영체제, 프로그램 언어, 관련 라이브러리 혹은 GUI를 설명
플랫폼: 소프트웨어 응용 프로그램들을 돌리는데 쓰이는 하드웨어와 소프트웨어의 결합. 플랫폼을 하나의 운영체제 혹은 컴퓨터 아키텍쳐라 하며 이 두가지를 통칭해서 말하기도 한다)

운영체제는 응용 프로그램과 컴퓨터 하드웨어 사이의 중재 역할함.

커널(보안, 자원관리, 추상화)
운영체제에서 가장 중요한것은 커널이다
일반인이 보지 못하는 낮은 수준의 프로세스를 제어함.
메모리를 얼마나 읽고 쓸 것인지, 어느 프로세서를 실행시킬것인지, 모니터, 키보드, 마우스와 같은 장치를 통해 어떤 정보를 주고 받을것인지, 네트워크를 통해 수신한 정보를 해석하는 역할을 한다.
- 추상화: 하드웨어에 직접 접근하는 것은 복잡한 문제를 초래할 수 있는데 이러한 복잡함을 깔끔하고 일관성 있는 인터페이스를 하드웨어에 제공하기 위해서 추상화를 진행한다. 이 추상화 작업은 프로그래머가 여러 장ㅇ비에 작동하는 프로그램을 개발할때 도움이 된다.

커널은 드라이버를 이용해 CPU, 그래픽 드라이버와 같은 하드웨어를 제어하고 여러 응용프로그램들이 갖는 다른 하드웨어 위해서 돌아가도록 호환성을 보장하기 위한 API를 제공한다

운영체제의 목적
사용자에게 컴퓨터의 프로그램을 쉽게 사용할 수 있게 만든다.
컴퓨터 시스템 하드웨어 및 소프트웨어 자원을 여러 사용자간에 효율적으로 할당, 관리, 보호 한다.
제어프로그램, 프로그램의 오류, 잘못된 자원 사용을 감시하고 입출력 장치등의 자원에 대한 연산과 제어를 관리한다.
파일관리: 파일 읽고, 쓰기, 메모리의 어디서부터 어디까지가 파일인지 파악하는 기능 등.

내가 화면에서 클릭한 내용을 프로그램에 알려주는것
프로그램이 실행될때 ram에 올라가는데 올라가도록 도와줌
cpu에 올라가는 순서를 정함
cpu의 결과값을 프로그램에 알려줌
결과를 출력장치로 출력함

가상메모리: 보조기억장치를 Ram(주기억장치)처럼 사용한다

운영체제
요즘은 운영체제와 응용프로그램의 경계가 많이 모호해지고 있다.
linux, dos, mac os, window, iOS, android, 붉은별
탄생계기: 진공관으로 할때는 운영체제라는 개념이 없었다.
관리자 역할을 해줄 수 있는게 필요해서 운영체제가 나옴
안드로이드도 리눅스 커널을 가지고 있다.

unix- bsd를 통해서 mac os x 탄생, ios도 이걸 통해서 탄생하였다.
windows-nt -> xp부터 사용을 시작함
linux를 포함해 unix-like os라고 함.
리눅스에 유닉스 소스가 들어가있진 않다

유닉스는 신뢰성을 기반으로 만들어짐. -> 서버, 중요한작업에 적합함.
윈도우 편의성을 기반으로 만들어짐. -> 개인
편의성이란? 범용적으로 많이 사용할 수 있다.

무료소프트웨어, 오픈소프트웨어 무엇이 차이일까?
리눅스는 프리, 오픈소프트웨어이다.
유닉스 환경에서는 리눅스의 명령어를 사용할 수 있다. - 명령어들이 윈도우에선 안됨

유닉스가 연구소에서 쓰던 운영체제이다. -> 안정적인 이유
GNU(그누): 프리소프트웨어 프로젝트 유닉스와 비슷하게 만들어진, 90년대 초반에 리눅스를 흡수함

그래서 운영체제가 뭔데?

H/w s/w를 이어주는 교두보 역할
하드웨어 관리, 프로그램의 오류나 잘못된 자원 사용을 감시 I/O 장치등의 자원에 대한 연산과 제어를 관리
사용자에게 컴퓨터의 프로그램을 쉽고 효율적으로 실행할 수 있는 환경제공
(가상 시스템 서비스 제공)
H/w의 자원관리
한정된 메모리를 어디에 얼마만큼 할당하는지를 제어한다

그래서 프로세스를 관리한다(자원관리)
프로세스: 실행되고 있는 프로그램 - 주기억장치 안에 프로그램이 실행되면 그 자체가 RAM에 들어간다.
-> 실행되고 있는 프로그램

프로세스 상태
생성, 준비, 실행, 대기, 종료

주기억장치 관리
메모리안에서 일어나는일도 운영체제가 관리하게 된다.
프로그램이 ram에 너무 많이 로드가 되었을때
가상메모리 개념으로 보조기억장치를 이용해서 ram의 역할을 수행한다
(조금 덜 중요할거 같은 프로그램을 보조기억장치로 옮긴다)

가상메모리를 많이 쓰면 오히려 성능이 더 저할될수도 있다.
<-> ram disk와 반대의 상황이다.
메모리덤프: 메모리에 있는 내용을 보조기억장치로 복사를 해온다 -> 디버깅할때 사용함
(최대절전모드)

파일관리

ntfs, fat 파일을 어떤 방식으로 읽고 쓸것이냐

운영체제의 정체성은 커널이다. 커널이 없다면 그건 운영체제가 아니다
커널 - 보안, 자원관리, 추상화
디바이스와 연결해주는 핵심이 커널이다. 안드로이드 루팅도 커널에 접근하여 값을 수정하는 것이다.

펌웨어: 하드웨어가 움직이는데 필요한 최소한의것 - 요즘에 와서는 운영체제와 동떨어진 게 아니다
바이오스 - 메인보드가 해야하는 최소한의 동작을 한다.
휴대전화에 들어가는 것도 옛날엔 펌웨어였다
(임베디드 소프트웨어와 완전 다른건 아니다)
디바이스 드라이버가 펌웨어를 대신한다 - 펌웨어는 부착되서 나오는데 드라이버는 부착되서 나오는게 아니다

알고리즘
문제해결을 위한 절차/방법
어떠한 문제를 해결하기 위한 여러 동작들의 모음

자료구조
효율적으로 자리를 정리하는것, 데이터를 구조적으로 표현하는 방식
모양을 바꾼다 -> 공간을 효율적으로 바꾸기위해함

알고리즘 + 자료구조 = 환상의 궁합

테트리스 - 요리조리 블럭을 돌린다(알고리즘)
효율적으로 부피를 줄이고 꺼낼순서에 맞게 저장한다(자료구조)

다시 자료구조
정수를 32bit에 0과 1로 표현을 한다 -> 가장 기초적인 자료구조
배열 - random access
리스트 - linked list

다시 알고리즘

일을 처리하는 순서/방법 - 분류가 어렵다
대표적 알고리즘 - **정렬, 탐색, 재귀**
정렬알고리즘: 자료가 난잡하게 나열이 되어있을때 기준을 세워 정렬하는것
최악, 최선, 평균을 고려해서 알고리즘을 선택해야한다

comparison array access
선택정렬: 7875 15526 - 한번 쭉 훑어보고 제일 작은걸 제일 앞에
순서대로 하나씩 비교해서 정렬한다
버블정렬: 7875 24761 - 큰거를 찾아서 제일 뒤에 갖다놓는다(선택정렬과 비슷) 거품처럼 보글보글 올라옴
삽입정렬: 17119 51116 - 적절한 위치를 찾아서 계속 삽입을 한다
병합정렬: 5057 16509 - 범위를 나눠서 정렬을 하고 합하면서 다시 정렬한다
퀵정렬: 16721 24113 - 평균점을 잡아서 작으면 앞, 크면 뒤 반을나눠서 계속 이렇게하여 수렴시킨다
(통상적으로 가장 빠르다고해서 퀵정렬임)

http://panthema.net/2013/sount-of-sorting 알고리즘시간에 실습을 할것임

알고리즘을 실행 시켰을때 그 효율성이 어느정도인지 예측할 필요가 있다
시간복잡도, 공간복잡도(메모리 저장 막대를 저장해 둘 메모리의 양)
h/w가 넉넉하다면 시간복잡도가 우선적으로 고려된다

점근표기법: 대문자 O표기법 (최악의 경우를 고려하는 표기법)

선택정렬 - O(n^2)
버블정렬 - O(n^2)
삽입정렬 - O(n^2)
병합정렬 - O(nlogn)
퀵정렬 - O(nlogn)

이런 부분을 비교하여 어떤 알고리즘을 써야 좀 더 빠를까를 고민하게된다

데이터베이스 - 데이터를 모아둔 창고
여러 사람이나 여러 컴퓨터에서 사용할 데이터를 모아둔 집합소 - 통합데이터 저장소
0과 1이 모여있는 많은 데이터
자료구조 - ram에서 이루어질 내용 (보조기억장치에서 가져와서 어떻게 그것을 메모리에서 표현할지를 고려함)
데이터베이스 - 보조기억장치에서 이루어질 내용
어떻게 저장하고 빼올것인가

관계형: 어떤 관계, 어떤 모형으로 저장이 되어있으니 이렇게 저장을 해라
테이블이란 단위로 구성이 된다
키-값형
객체형
문서형
컬럼형

DBMS: DB의 운영체제 mysql, oracle, mariadb, mongodb
DB를 운영할 수 있는 소프트웨어

구글 검색

“” : 반드시 포함, 어순까지 일치하는 결과
- : 이 단어는 빼고 검색
* : 기억나지 않는 단어 또는 용어등을 대신해 사용(wild card)
(가는 *이 고와야 오는 *이 곱다)
~ : 유사한 의미 검색
site: 특정 사이트 내 검색
(site:yagom.net 프로그래밍)
link: 특정 링크가 포함된 결과
(link:youtube.com 연결되는 페이지가 유튜브인경우)
related: 해당 사이트와 유사한 사이트
intext: 본문 내 검색
intitle: 웹페이지 제목으로 검색
filetype: 파일형식 지정
filetype: doc 2015전화부, xls 23기명단 이런식으로

네트워크
프로그래밍 기초 용어, 환경, 언어
보안/암호화
소프트웨어 공학

네트워크: 통신망, 여러 개체가 연결되어 정보를 주고받는 경로,
노드, 링크로 구성된 모든 종류의 시스템을 대상
용어들 위주로 알아보는 시간.

LAN: 층, 건물 근거리 통신
WAN: 국가나 대륙별
client:
server:
이 두 가지 개념은 상대적인것이다. 어느것 하나가 처음부터 끝까지 서버이거나 클라이언트일 수는 없다.
대체적으로 서비스를 요청해서 정보를 받는것이 client이고 요청을 받고 그 요청에 의한 결과로 정보를 보내주는것이 server이다.

단방향성
peer to peer: 서로가 서버도되고 클라이언트도 되는 개념 p2p
network architecture: 정보를 제공하는

network topologies: star, ring, bus

network communications technology:

extranet -> 인트라넷의 반대개념, 외부 조직원들이 사용할 수 있는것

www : 어느 서버에 들어가도 정보를 가져올 수 있다.
초반엔 핵 연구소에서 자료를 원활하게 공유하기 위해 만든 개념
웹에 접근할 수 있는 모든 서버에서 정보를 원활하게 받아들일 수 있는 것.
서버와 서버 , 클라이언트를 모아둔것

프로토콜(protocol)
protocol: 장비 사이에서 메세지를 주고받는 모든 약속들
ex) http, https, ftp, smtp 등등 주고받는 자료에 따라서 프로토콜은 굉장히 다양하다.

ftp: file transfer protocol 파일을 전송하기 위한 프로토콜
telnet: 원격로그인을 위한 프로토콜이지만 암호화가 되지 않아 정보 노출등의 위험으로 요즘은 잘 사용하지 않는다.원격으로 다른 컴퓨터에 접속할 때 사용하는 통신규약(명령어로 제어함)
ssh: secure shell telnet의 대용으로 암호화가 된다.

URL: Uniform Resource Locator
[Protocol]://[Host]:[port]/[path]

http: hyper text 문서(html)): 하이퍼링크를 이용해 문서를 이동할 수 있는 프로토콜

smtp: Simple mail transfer protocol e-mail 발송 프로토콜

Hostname: 네트워크에 연결된 장치에 부여되는 고유한이름
ip주소, mac주소(하드웨어 장치에 부여되는 고유의 이름), 도메인주소(ip주소와 mapping)

ip: Internet Protocol Address
컴퓨터 네트워크에서 서로 통신하기위해 갖고있는 주쇠.

DNS: ip주소를 도메인으로, 도메인을 ip로 바꿔준다.
호스트의 도메인 이름을 네트워크 주소로 바꾸거나 그 반대의 변환을 수행한다.

port: 가상의 통신-> 논리적인 개념
컴퓨터까지 도달할 땐 ip가 필요하고 그 컴퓨터에 접속하고자하는 프로토콜은 포트를 이용해 접속한다.

ssl: socket secure layer 웹서버와 브라우저 사이이의 보안을 위한 프로토콜
암호화된 통신을 제공하함.

암호화:
보안: 컴퓨터에 관련된 보안
정보를 보호하고 보안을 유지함,
기밀성, 무결성, 가용성
암호학_ 수학자들의 몫, 수학적인 계산이 많이 필요하다
대칭키: 키를 가지고 서로 암호화 복호화를 함
공개키: 하나의 키는 자기만 갖고있다. 나머지 키 하나를 공개적으로 제공함
해시: SHA, MD5
임의의 데이터를 고정된 길이의 데이터로 매피앟여 원래의 입력값과의 간계를 찾기 어렵게 만드는것

소프트웨어공학: 소프트웨어의 개발, 운용, 유지보수 등의 생명 주기 전반을 체계적이고 서숙적이며 정량적으로 다루는 학문
https://ko.wikipedia.org/wiki/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EA%B3%B5%ED%95%99
프로그래밍: 프로그램을 만드는 것.

소프트웨어 공학
소프트웨어공학?
개발, 운영, 유지보수, 폐기까지 체계화를 위한 방법들

스프트웨어 개발 생명주기 모델: 어떻게 개발할 것인가에 대한 흐름.

폭포수 모델: 단계->단계
요구사항이 변경되면 앞에 진행한 작업들이 무용지물로 변한다는것이 문제

프로토타이핑: 원활한 커뮤니케이션을 이해 프로토타입을 프로그래밍하여 발주자와 확인을 하며 진행함
의사소통에 많은 시간이 필요하지만 폭포수모델에 비해 실패율이 낮다.
다만, 프로토타입을 원칙적으로 폐기해야하나 그것을 그대로 사용하는 경우가 많아 유지보수가 힘들다.

나선형모델: 폭포수모델의 주기를 빠르게 기능 하나씩 개발한다. 미리만든 기능을 수정하면 정말 힘들다.
위험분석 과정에 좀 더 면밀한 진행ㅇ 피리요하다.

소프트웨어 개발 방법론?
소프트웨어를 어떻게 개발할 것인가에 대한 전체적인 흐름.
생산하는데 필요한, 반복적인 과정을 처리하는것.
흔히들 말하는 객체지향도 스프트웨어 개발 방법론에 해당한다.

애자일 개발 프로세스:
좋은것을 빠르고 낭비없게 만드는 방법들의 집합, 빠르게 만드는 모든 방법들.

UML: 객체 지향으로 개바랗ㄹ때 산출물을 명세화, 시각화 하여 한눈에 볼 수 있는 문서화가 가능하다
이 작업을 통해서 처음부터 메소드들을 만들 수 있다(기능까지 나오는건 아니지만)

TDD: 다양한 환경의 테스트를 코드로 구현하여 테스트하는 방식.
테스트 코드를 만드는데 많은 시간이 필요하지만 여러번 테스트를 해야한다면 오히려 시간을 단축할 수 있다.

형상관리: 소프트웨어개발 시 유지보수 과정에서 발생하는 코드, 문서, 인터페이스 등 각종 결과물에 대해 형상을 만들고 그것을 기록하여 개발시 발생하는 문제들에 대처한다.

버전견롸: SVN, git 몇몇 버전들을 기록하여 되돌아 가거나 협업시 프로그래머는 프로그래밍에, 디자이너는 오로지 디자인에 집중할 수 있으며, 배포버전, 테스트버전, 개발진행중 버전등으로 나누어 관리가 가능하며, 기능별로 새롭게 버전을 관리하여 개발에 많은 도움을 준다.

Library: 특정 기능을 수행할 수 있는 클래스 또는 함수의 집합체
API: 어떤 프로그램의 함수를 다른 프로그램에서 사용할 수 있게 제공해줌
framework: 구조적으로 고정된 부분을 재사용 할 수 있또록 함.

Reference Document: lib, api가 어떤것을 가지고 있는지 문서로 설명하여 개발을 도움

객체지향 프로그래밍: 프로그램 명령을 하나하나의 객체로 보는것
가장 좋은점은 이것을 객체화 하여 구분할 수 있고 코드를 재사용하여 시간을 절약하는데 도움이된다.

클래스와 객체
클래스를 설계도라 하면 이 설계도를 바탕으로 자동차를 만드는데 색상, 특화 등의 세세한 부분을 다르게하여 설정할 수 있다.