**스택**

문제

* 함수를 이용해서 스마트폰 클래스를 만들어보시오. 전화를 거는 call메서드 추가
* 클래스문법을 이용해서 위의 스마트폰 클래스를 만들어 보시오. 캡슐화를 이용해서 model, phoneNumb프로퍼티를 숨겨보아라.
* This 란 무엇인가
* 상속이란 무엇인가 ?
* 상속을 이용하여 포유류 -> 고양이 -> 러시안 블루 종을 표현해보라
* 다중 상속이란? 그리고 이를 자바스크립트에서는 어떤 방법으로 표현하는가 ?
* 믹스인 기법을 이용하여, 포유류 -> 호랑이, 사자 -> 라이거의 형태의 클래스를 만들어보라 .

**서문**

우리는 곧 있으면 동기 비동기라는 개념을 배우게 됩니다. 이 동기와 비동기라는 메커니즘을 이해하기위해서는 스택과 큐라는 자료구조의 이해가 어느정도 필요하기 때문에 시간을 많이 잡아먹더라도, 커리큘럼에 추가하게 되었습니다.

자료구조는 우리가 알고 있는 배열과 익히지 않은 객체 따위를 이용하여, 자료를 효율적으로 관리하기 위해서 만든 추상적인 구조입니다. 쉽게 말하면 우리가 어떤 구조를 커스텀하여, 특정한 방식으로 관리 하도록 만들 수 있다는 것 입니다.

자료구조에는 스택, 큐, 연결리스트, 덱, 그래프, 해쉬테이블 등이 있는데, 저희가 이 스터디에서 주로 다룰 내용은 **스택과 큐입니다**. 자료구조는 개발자로 성장하기 위해서 필수적으로 배워야 하는 과목이기에 이 스터디에서 스택과 큐 이외에 더 이상 다루지 않는다고, 괄시하지 마시고 혼자서 끝까지 배우시기 바라겠습니다.

스택은 접시를 쌓아놓은 형태의 자료구조 이다. 접시를 쌓기 위해서는 접시를 기존 접시 위에 올려 놓아야 합니다. 그렇다면 쌓여진 접시더미에서 맨 처음 쌓인 접시를 꺼내려면 어떻게 해야 할까요 ? 바로 그 접시위에 쌓여진 모든 접시들을 치워야 해당 접시를 꺼낼 수 있습니다. 스택이라는 자료구조 또한 이런 접시더미와 같은 모습을 취하고 있습니다. 그래서 스택이라는 자료구조는 마지막에 들어 간 자료가 처음으로 꺼내어진다고 해서 LIFO(last in first out)의 형태를 띈다고 말합니다.

**스택의 구조**

자바스크립트의 배열은 사실 타 언어에서 말하는 배열과는 다릅니다. 보통 배열은 처음 선언했던 크기 만큼 아이템을 넣을 수 있습니다. 그러나 자바스크립트 배열은 이런 전통적인 배열의 특성에서 진화된 Array List라고 하는 아이템 개수에 따라서 신축적으로 늘어나는 성질을 가지고 있습니다. 그래서 전통적인 스택의 구조와 같이 만들기 위해서 우리는 클래스를 이용하고 생성자 함수의 인자로 숫자타입을 받아 배열의 크기를 제어 할수 있는 변수인 maxSize에 할당 하도록 하겠습니다. 그리고 top이라는 변수를 두어 스택의 높이를 알 수 있도록 하여,

배열은 프로퍼티 형태로 두지 않고, 캡슐화하여 직접적인 접근을 막도록 하겠습니다.

**스택의 유틸리티**

이 스택의 기본 기능은 push와 pop그리고 peek이 있으며, 효율을 좀 더 주기 위한 isFull,isEmpty함수를 만들도록 하겠습니다.

이 push라는 녀석은 배열의 기본 메서드의 기능과 마찬가지로 배열의 끝에 데이터를 추가하는 기능입니다. 배열에 추가하기 전에 isFull 함수로 스택이 가득 찼는지 확인하고 빈 공간이 있다면 top에 1을 증가시켜 해당 배열에 아이템을 추가 하도록 합니다.

다음은 pop은 배열에 마지막에 추가된 데이터를 제거하고 반환하는 기능입니다. Pop도 push메서드와 같이 isEmpty로 배열이 비었는지 확인하고 해당 배열에서 아이템을 제거하고 반환합니다. 이 때 실제로 아이템을 제거하고 지워주어도 되지만 top을 1 감소 시키면 다음 번 push때 새로들어오는 아이템이 기존의 아이템을 대체하기 때문에 문제 될 것이 없습니다.

마지막으로 peek 이 peek이라는 기능은 스택에 마지막으로 추가된 데이터를 제거하지 않고 반환하는 기능입니다. 이 peek이라는 기능도 스택이 비었는지 확인하고, top에 있는 아이템을 반환합니다.

이 기능들을 클래스를 이용해서 만들어보도록 하겠습니다.

class StackType{

constructor(size){

*//일반 프로퍼티로 사용하면 배열을 컨트롤 하기 쉬우니 캡슐화를 진행하겠다.*

let stack = [];

let top = -1;

const maxSize = size;

let isEmpty = ()=>{

return top === -1;

}

let isFull = ()=>{

return top === maxSize;

}

*//stack의 높이를 계산하는 프로퍼티, 자바스크립트의 유용한 기능들을 덜 쓰기 위한 조치*

this.pop = () => {

if(isEmpty()){

return "스택이 비어있습니다.";

}else{

console.log("스택에서 정상적으로 제거 되었습니다.");

return stack[top--];

}

};

this.push = (item) => {

if(isFull()){

console.log("스택이 가득 찼습니다.");

}else{

stack[++top] = item;

console.log("정상적으로 스택에 추가되었습니다.")

}

}

this.peek = ()=>{

if(isEmpty()){

return "스택이 비어있습니다.";

}else{

return stack[top];

}

}

}

}

let stack = new StackType(4);

stack.push("5");

stack.push("2");

stack.push("1");

stack.push("6");

stack.push("7");

stack.push("7");

stack.push("7");

stack.push("7");

console.log(stack.pop());

**스택의 사용 사례**

재귀 알고리즘

웹 브라우저 방문기록 (뒤로가기)

실행 취소

역순 문자열 만들기

수식의 괄호 검사

후위 표기법 계산

이 스택을 두개 만들어서 한쪽 스택에다가 모두 넣어주면 그 스택은 기존 스택과 반대의 스택이 된다. 만들어보라. 스택의 top을 퍼블릭하게 만들었으니, 이 top프로퍼티를 이용하면 쉽게 다른 스택으로 전부 옮길 수 있다.