#### activity life cycle

#### 

## -. Thread 동기화의 종류와 각각의 장단점

1.메소드에 synchronized를 추가

2.code block에 synchronized를 추가

-> 가능한 좁은 범위, 1번의 경우 메소드가 길어질 경우 문제가 될 수 있음, dead lock 발생할 수 있음

3. java.util.concurrent.locks 패키지 사용

## - RESTful API에 대해 설명

이처럼 다양한 장비와 프로그램을 하나의 서버로 대응하기 위해 REST API가 개발되었습니다. 모바일 기기, OS, 브라우저 등 다양한 단말기와 프로그램을 지원하는 API 형태를 REST API라고 표현합니다.

URI와 HTTP 메소드를 이용해 객체화된 서비스에 접근하는것

|  |  |  |
| --- | --- | --- |
| 메서드 | 의미 | Idempotent |
| POST | Create | No |
| GET | Select | Yes |
| PUT | Update | Yes |
| DELETE | Delete | Yes |

## Http 통신시 Post/Get 방식의 차이점

* GET은 주소줄에 값이 ?뒤에 쌍으로 이어붙고 POST는 숨겨져서(body안에) 보내진다.
* GET은 URL에 이어붙기 때문에 길이제한이 있어서 많은양의 데이터는 보내기 어렵고 POST는 많은 양의 보내기에도 적합하다.(역시 용량제한은 있지만)
* 즉 http://url/bbslist.html?id=5&pagenum=2 같이 하는 것이 GET방식이고 form을 이용해서 submit을 하는 형태가 POST입니다.

GET은 가져오는 것이고 POST는 수행하는 것입니다.

select 기능을 원한다면 GET, UPDATE기능을 원한다면 POST

검색결과등 고정적인 주소 및 링크 주소로 사용될 수 있다면 GET

GET은 캐시 가능, POST는 캐시 남지 않음

## Hash Table에 대한 설명.

key와 value를 갖는 자료 구조

hash function이란 함수와 array로 만들어져 있는 hash table이 조합되어서 만들어저 있는 자료구조다. O(1)에 가까운 퍼포먼스.

성능은 적절하게 자료를 분산시킬 수 있는 hash function에 달렸다.

Collision의 문제가 발생할 수 있다.

java에선 동기화가 필요할 경우 hashTable, 아닐 경우 hashMap을 사용

## 

## 특정 Tree search 알고리즘에 대한 설명

## - 프로세스, 스레드, 메모리, vector/list, 퀵소트의 원리 등 프로그래밍에 있어서 중요한 기본적인 지식을 갖추고 있는지를 물어봄

|  |  |
| --- | --- |
| Process | - 실행 중인 프로그램  - 리소스와 쓰레드로 구성  - 모든 프로세스는 하나 이상의 쓰레드를 가지고 있다. |
| Thread | - 경량화된 프로세스이다.  - 프로세스 내에서 실제 작업을 수행함 |

-프로세스는 완벽히 독립적이기 때문에 메모리 영역(Code, Data, Heap, Stack)을 다른 프로세스와 공유하지 않는다.

-쓰레드는 해당 쓰레드를 위한 스택을 생성할 뿐 그 이외의 Code, Data, Heap영역을 공유한다.

퀵소트 : 퀵 정렬은 *n*개의 데이터를 정렬할 때, 최악의 경우에는 [**O**](https://ko.wikipedia.org/wiki/%EB%8C%80%EB%AC%B8%EC%9E%90_O_%ED%91%9C%EA%B8%B0%EB%B2%95)(*n*2)번의 비교를 수행하고, 평균적으로 **O**(*n* [log](https://ko.wikipedia.org/wiki/%EB%A1%9C%EA%B7%B8) *n*)번의 비교를 수행한다. 최악의 경우는 pivot이 계속 최대 또는 최소값이 나올 경우

## - 문자열(String)과 관련된 간단한 코딩

## - 자소서 내용과 관련된 질문: 왜 이직을 하려고 하는가, 왜 해당 언어를 사용하는가, 이용하거나 알고있는 네이버 서비스에 대해 말해보시오, 진행했던 프로젝트에 대한 설명과 질의응답 등

## - 어떤 것을 개발하고 싶은가?

## vector와 list의 차이점과 특징

**-java의 arrayList가 일반적인 vector, LinkedList가 일반적인 list라고 보면 될 듯.**

**Vector** - 구버전 호환용으로 잘 사용되지 않고 동기화 처리가 내부적으로 일어나서 다른 객체들 보다 무겁다. 고로 왠만하면 다른 List클래스를 사용하는것이 좋습니다.

**ArrayList** - 배열의 복사에 의한 데이터 저장처리를 내부적으로 행하며, 각 데이터에 대한 인덱스를 가지고 있기 때문에 검색이 매우 빠르지만 많은 데이터의 추가 / 삭제시에는 배열의 복사가 빈번하게 일어나 성능이 떨어지는 단점이있습니다.

**LinkedList** - 다음 자료의 위치정보를 가지며, 내부적인 인덱스 가지고 있지않다. 데이터의 추가 / 삭제는 위치정보의 수정만으로 가능하기 때문에 많은 정보의 추가 / 삭제처리가 필요할때 유용하지만 데이터가 많은 경우의 검색시 처음 자료부터 순차적으로 찾아 나가야 하기 때문에 느려지는 단점이 있습니다.

4. linked list와 map, hashmap의 특징

## Singleton을 multithread를 고려해서 구현

처음부터 인스턴스를 생성하던지, getInstance()에 syncronized 적용하기

public class Singleton {

private static Singleton uniqueInstance = **new Singleton();**

private Singleton() {}

public static Singleton getInstance() {

return uniqueInstance;

}

}

public class Singleton {

private static Singleton uniqueInstance;

private Singleton() {}

public static **synchronized** Singleton getInstance() {

if (uniqueInstance == null) {

uniqueInstance = new Singleton();

}

return uniqueInstance;

}

}

## HTTP, HTTPS 차이점

HTTPS는 공개키 방식으로 클라이언트와 서버간의 통신을 암호화 해서 사용하는 프로토콜.

HTTP에 비해 서버에 암호화와 관련된 부하가 생기며, 인증서를 구입해야 한다.

## SPRING 개념

## GC

GC는 GC 대상이 되는 객체들을 힙에서 제거하기 위해 JVM에서 제공되는 매커니즘이다. C와 다르게 별도의 메모리 관리를 할 필요가 없다.

Garbage Collector라고 불리는 데몬 쓰레드에 의해 수행된다.

객체들은 스택, 힙 중 힙에 생성된다. 이 중 객체의 모든 참조가 null 일 경우 GC 대상이 된다.

# AJAX

비동기 JavaScript와 XML을 말합니다. 간단히 말하면, 서버측 Scripts와 통신하기 위한 XMLHttpRequest객체를 사용하는 것을 말합니다. 서버측으로 다양한 형식(JSON, XML, HTML 및 일반 텍스트 형식 등)의 정보를 주고 받을 수 있습니다.

$.ajax({

url:'./time2.php',

type:'post',

data:$('form').serialize(),

success:function(data){

$('#time').text(data);

}

})

## Dynamic Programming이 무엇인지 설명하시오

1) DP는 피보나치 수열을 이용해 설명하였습니다

## 안드로이드에서 쓰레드 간에 통신하는 방법은

주로 메인UI 스레드와 연산작업을 위한 백그라운드 스레드간의 통신과 관련있다.

Handler를 이용해 스레드에서 메인UI로 message를 보내는 방식으로 통신한다.

이를 좀 더 쉽게 사용할 수 있는 AsyncTask 도 있다.

## 손코딩 문자열 순서 바꾸기, 문자열 배열 가능 경우의 수 출력 등

6. 정렬이나 탐색 등의 알고리즘 중에서 자유롭게 하나 골라서 코드로 작성하고 설명

**TreeMap, HashMap, LinkedHashMap의 차이점은 ?**

TreeMap은 정렬 저장

HashMap 은 그냥 저장

LinkedHashMap은 입력된 순서대로 저장

특별한 이유가 없다면 검색 성능이 좋은(O(1)) HashMap 을 사용하자

키값이 일정한 수준대로 iterate 하려고 한다면 TreeMap 을 사용하자. 하지만 랜덤 접근에 대해서는 O(logn) 성능을 지니므로 많은 데이터를 넣을경우 좋지 않은 성능이 나올수 있다.

입력 순서가 의미있다면 LinkedHashMap 을 사용하자. 랜덤 접근에 대해 O(n) 성능을 지니므로 많은 데이터를 입력할 경우 사용하지 않는것이 좋겠다.

클로저는 무엇인가?

클로저는 독립적인 (자유) 변수를 가리키는 함수이다. 또는, 클로저 안에 정의된 함수는 만들어진 환경을 ‘기억한다’.

클로저는 외부함수(포함하고 있는)의 변수에 접근할 수 있는 내부 함수를 일컫습니다. 스코프 체인(scope chain)으로 표현되기도 합니다. 클로저는 세가지 스코프 체인을 가집니다: 클로저 자신에 대한 접근(자신의 블럭내에 정의된 변수), 외부 함수의 변수에 대한 접근, 그리고 전역 변수에 대한 접근. 이렇게 3단계로 구분할 수 있습니다.

내부 함수는 외부 함수의 변수뿐만 아니라 파라미터에도 접근할 수 있습니다. 단, 내부 함수는 외부 함수의 arguments 객체를 호출할 수는 없습니다. (하지만, 외부 함수의 파라미터는 직접 호출할 수 있습니다.)

#### **기본적인 클로저 예제:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | function showName(firstName, lastName) {  var nameIntro = "Your name is ";  // 이 내부 함수는 외부함수의 변수뿐만 아니라 파라미터 까지 사용할 수 있습니다.  function makeFullName() {  return nameIntro + firstName + " " + lastName;  }  return makeFullName();  }  showName("Michael", "Jackson"); // Your name is Michael Jackson |

클로저는 Node.js의 비동기, 논-블록킹 아키텍처의 핵심기능으로 활용되고 있습니다. 클로저는 jQuery에서도 빈번히 사용되며, 거의 모든 자바스크립트 코드에서 볼 수 있습니다.

#### **jQuery의 전형적인 클로저 사용예:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $(function() {  var selections = [];  $(".niners").click(function() { // 이 클로저는 selections 변수에 접근합니다.  selections.push(this.prop("name")); // 외부 함수의 selections 변수를 갱신함  });  }); |

|  |
| --- |
| function hello(name) {  var \_name = name;  return function() {  console.log('Hello, ' + \_name);  };  }  var hello1 = hello('승민');  var hello2 = hello('현섭');  var hello3 = hello('유근');  hello1(); // 'Hello, 승민'  hello2(); // 'Hello, 현섭'  hello3(); // 'Hello, 유근' |

// 여기서 메모리를 release 시키기 클로저의 참조를 제거해야 한다.

hello1 = null;

hello2 = null;

hello3 = null;

# 바이너리 파일이랑 텍스트 파일 처리 속도 차이가 왜 나는가?

바이너리보다 텍스트로 저장할 경우 더 많은 저장 공간이 필요(개행표현)하기 때문에, disk I/O가 더 많이 일어난다. 따라서 텍스트보다 바이너리가 빠르다고 할 수 있다.

# 뮤텍스와 세마포어의 차이는 무엇인가?

공유자원에 대한 접근권한, Lock이라는 키를 한 개만 가지고 있는 것은 뮤텍스(mutex)이다.

반면에, Lock이라는 키를 여러 개 가질 수 있는 것은 세마포어(Semaphore)이다.

# 뮤텍스란 무엇인가?

일종의 Locking 매커니즘이다. lock을 가지고 있을 때만 공유 데이터에 접근이 가능하다.

화장실에 갈 때 키를 가진 사람만이 갈 수 있다. 일을 다 본 후에는 키를 반납하고 그 다음 사람이 갈 수 있다. 이러한 메커니즘을 말한다.

유의할 점은 lock에 대한 소유권이 있다는 점. 열쇠를 획득한 사람만 반납 할 수 있다.

# 세마포어란 무엇인가?

세마포어는 동시에 리소스에 접근 할 수 있는 “허용 가능한 counter의 개수”를 말한다.

예를 들면 병원에 있는 어느 한 병실에 5명까지 들어 갈 수 있다고 한다면, 5명까지 들어 갈 때 counting을 하고 5명 이후에는 밖에서 기다려야 한다. 한 사람이 나오게 되면 다음 사람이 들어 갈 수 있게 되는 것이다.

그 count수에 따라서 1개이면 binary semaphore, 여러 개이면 counting semaphore라고 한다.

binary semaphore는 뮤텍스와 개념적으로 같다.

# Java volatile은 변수의 가시성(Visibility)을 보장한다.

volatile은 작업의 요소를 뺀 가시성, 즉 **읽기, 쓰기**동작(연산은 아님)에 대해서만 동기화가 됩니다. syncronized는 이럴 때 작업자체를 원자화해버립니다.

volatile이 할 수 없는 일이죠.

###### 