* + Java Development
* + Collections
  + Diferencias entre Linked List y Array List.
    - *Recuperar* data esta mejor en Array List. Porque esta recuperando directo con el index.
    - *Añadir y Borrar*informacion esta mejor en Linked List. ArrayList nececita hacer bit shifting por estes operaciones.
    - Rendimiento esta mejor cuando haciendo cambios en una Linked List. Pero es mejor para almacenar y recuperar los datos desde Array List.
  + Diferencias entre List y Set
    - Un Set tiene registros unicos, pero el cambio un List puede tener varias mismos registros .
  + Differences between Queues and Stacks
    - Queue – FIFO, Stack – LIFO
  + How HashMaps / Hashtables work (hashing, collissions, Object#hashcode)
    - Cada Key y Value par tiene un hashcode lo que generado basado en el par. Cuando mas que uno par tiene el mismo hashcode, ese se llama collission.
* + JVM pooled objects
  + Strings
    - Pooled / inmutable object: una vez se ha creado, no puede ser modificado. Cuando se pretende a cambiar el valor, ese proceso se ha ejecutado en un otro nuevo lugar en memoria.
  + primitive data types
* + Concurrency
* + Concepts
  + Deadlock
  + Livelock
  + Starvation
  + Race Condition
* + Abstractions
* + SYNCHRONIZED

*Sincronización se ha usado para resolver el problemo de inconsecuencia*

* + en nivel de class

Cuando un static synchronized method se ha invocado, el hilo esta usando el lock de objeto class el lugar de lock de instancia.

Lock(Bloque) de un class y la instancia son seperados .

* + at instance level

Cuando Synchronized se ha aplicado a un metodo o bloque, se crea un sección de critica, y solamente uno hilo puede acceder a este sección en un bloque del tiempo, se lo usa, y lo libera. Y ese proveendo a los hilos a acceder el recurso y cambiar valores en el mismo tiempo

*Volatile is making an object to be seen to all of the threads.*

*Synchronized, makes this available also.*

* + Object#wait, Object#notify, Object#notifyAll
  + Thread#join

Se ha usado para hacer el hilo currente a esperar hasta que el hilo adjuntando termina su tarea.

+ Concurrency API

* + + java.util.concurrent.locks
  + Lock
    - Haciendo un objeto inalcanzable para los otros hilos, a habilitar solamente hacer cambios a el currente hilo
  + Condition
    - + java.util.concurrent
  + CyclicBarrier

*await()*

Es un cantidad especificado para hacer todos hilos a esperar los que vienen a el punto especificado entre ese hilo, y cuando cumple el cantidad, los siguen a tus procesos.

* + CountDownLatch

*countdown(), await(), getCount()*

El hilo, lo que tiene metodo await de CountDownLatch, tiene que esperar hasta el cantidad especificado se ha termina con el metodo countdown.

*CountDownLatch ctl = new CountDownLatch(5);*

* + Java 8
  + Lambdas
  + Method References

+ Interfaces

* + Default Methods
  + Static Methods
  + Interfaces vs. Abstract Classes

+ API

* + Streams
  + Optional
  + CompletableFuture
* + Garbage Collections
* + Generations
  + + Young
  + Eden
  + Survivor
  + Virtual
  + + Tenured
  + Virtual
* + GC Algorithms
  + Serial Collector
  + Parallel Collector
  + Concurrent Mark Sweep (CMS) Collector
  + Garbage-First Garbage Collector
* + Profiling
  + visualmv
  + + load testing
  + jmeter
  + locust io
* + Frameworks / Specs
* + Spring
  + IOC
  + MVC
  + Transactions
  + Cache
  + Boot
* + ORM
  + Hibernate
  + JPA
* + Servlets / JSP
* + 3.0+
  + Async Requests
  + JSTL
* + Architecture
  + SOLID principles
  + GoF patterns
  + EAI patterns
  + SOA patterns
  + Java design patterns
    - Singleton Design Pattern
    - Builder Design Pattern
    - Strategy Design Pattern
  + Microservices patterns
  + Reactive patterns
* + Distributed Apps
  + ACID transactional properties
  + BASE and Eventual Consistency
  + CAP Theorem
  + FLP Impossibility
  + + Consensus
  + Paxos
  + Raft
* + Tools / Frameworks
  + Spring Cloud
  + Hystrix (Netflix)
  + Apache Zookeper
* + DevOps
  + Docker
  + Kubernetes
  + Chef / Puppet
  + Bash
* + Methodoligies
  + Agile
  + Scrum