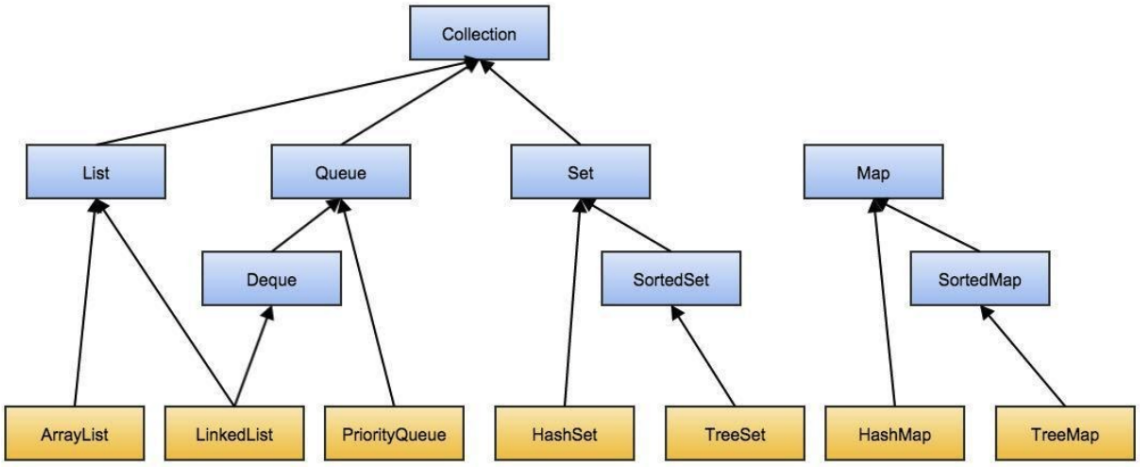
Java Basic-Jan26

7. Collection



blue: Interface

yellow: implementation class

List is an interface

- ArrayList
- LinedList

a. ArrayList based on Array, resizable-array
   - can be resized (double the size)
   - based on **consecutive** Memory space
     - access element by O(1)

       know the first location address of array, and add the offset, then get the corresponding element

It is a list and a Collection.

It implements all the methods specified in List and Collection.

b. LinkedList(double LinkedList)

- stored in in-consecutive Memory space
- higher overhead than ArrayList

Vector is similar as the ArrayList but Thread safe.

Stack based on List, also Thread safe, FILO

Thread safe has been implement internally. (sometime we do not need, which could be lots of overheads). Vector and Stack may be slower.

Deque → ArrayDeque

Head / Tail

Deque

Type of operation	First element		Last element	
	throw exception	return special value(null)	throw exception	return special value(null)
Insert	addFirst(e)	offerFirst(e)	addLast(e)	offerLast(e)
Remove	removeFirst()	pollFirst()	removeLast()	pollLast()
Examine	getFirst()	peekFirst()	getLast()	peekLast()

1. Deques are Queues. So it has Queue's APIs: offer/poll/peek. Not recommended.
2. When used as a stack. Use only one end. i.e. offerFirst/pollFirst/peekFirst.

Set

- HashSet

- TreeSet
- LinkedHashSet

Properties of Set:

- unique: set don’t allow duplicated values

HashSet

- unorder: all the elements in set are unordered

TreeSet:

- all elements in TreeSet are ordered.
- implement by RedBlackTree

LinkedHashSet:

keep insertion order of set

Map:

- HashMap
- LinkedHashMap
  - keep the insertion order, similar as LinkedHashSet
- TreeMap
  - order the map entries(entry) based on the key
- HashTable(Thread safe)
- ConcurrentHashMap(Thread safe)

Queue

- FIFO

Heap(**completed tree**)

- PriorityQueue
- minheap
- maxheap

Feature of Heap

array

- int[], String[], Object[]
- 2D array (2 dimensional)

Binary Tree

left child and right child can be null

The most frequence interview questions is including

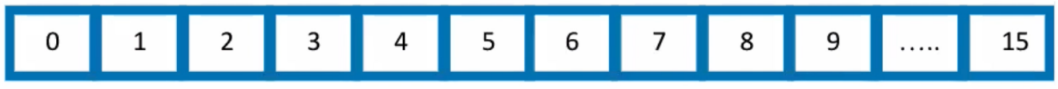
What’s the different between List ad Set?

HashMap

- How does java implement hashmap?
- How does it deal with Null Key and Null Value?
- How does it deal with Duplicates?
- TreeMap vs LinkedHashMap?

What is a HashMap?

- node: use key-value pair as a node(Entry) to store data
- bucket(slot):



1. Find the correct index.
2. Create a new Node.
3. Insert into the bucket.

bucket

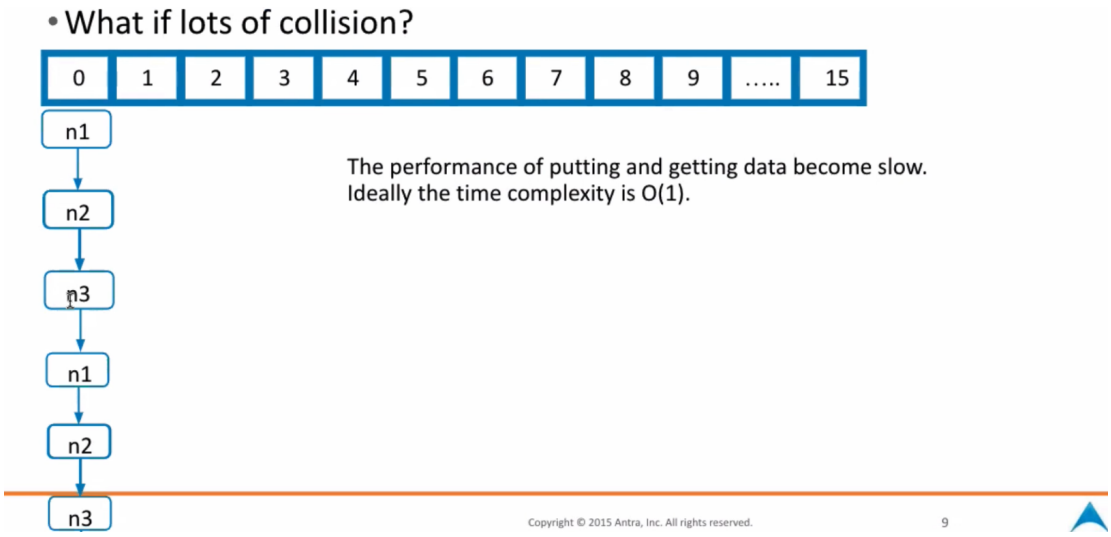
Hash Collision:

Collision

- Two Keys have the same hashCode.
- But they are different Keys like:
  - KeyOne has : int hashCode(){ return 1 + x};
  - KeyTwo has : int hashCode(){ return 3 – y};
  - If x = 1, y = 1, both will return 2.
  - This will result in the same index, same bucket will be used to same the Entry.

Collision will lead to be lower the performance.

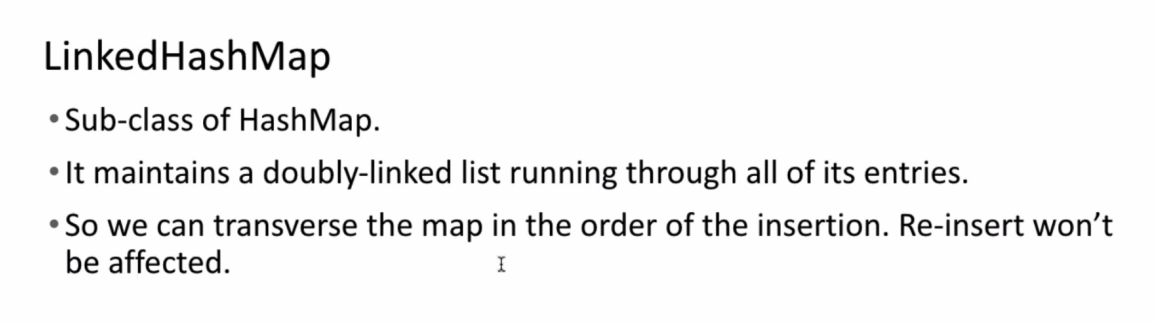
E.g. When all entries(nodes) in a same bucket(worest hash function), get() method will be O(n).



if hash function is good or not be changed, another way to improve the performance is changing the node-list(size > 8) to a TreeNode(Red-Black Tree) - O(logn)

threshold(门槛): if size is over the threshold, the bucket size can be resized

LinkedHashMap:



TreeMap:

- Nodes are sorted.
- Uses Tree as the backing data structure.

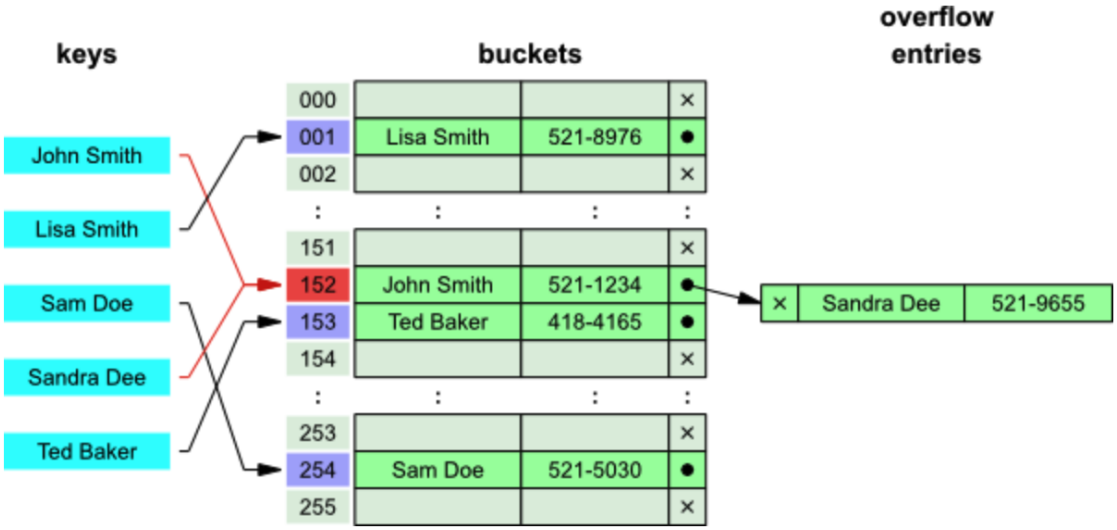
HashSet:

- Uses HashMap as its backing data structure/implementation.
- The Keys in HashMap act as the Value inside HashSet.
- A dummy object will be inserted into the hashmap.
- HashSet == the Keys in HashMap

Two ways for Hash Collision:

- separate chaining
- open addressing
  - every bucket only can store one node

- **Collision Control**
  - Collision - two keys mapped to the same bucket
  - **Separate Chaining**(Close Addressing) - the element of each of the buckets is actually a single linked list.
  - **Open addressing** - put the key-value pair into the “next” available bucket.
    - How to define next? Linear/quadratic/exponential probing, hash again
    - Challenge: handling removed keys in the map
    - Not used by Java, but by some real life systems.
  - If different Keys are determined to use the same bucket, they will be chained in the list.



HashMap vs HashTable vs ConcurrentHashMap

HashTable and ConcurrentHashMap are all based on HashMap.

HashTable and ConcurrentHashMap are Thread Safe.

Hashtable use sync keyword locking all objects. When locking all objects, there will be only one thread can access the whole hashTable - lower performance.

ConcurrentHashMap has/allow 16 threads operating at the same time.

Read operation doesn’t require lock. Any number of threads can read the concurrentHashMap. If there is write-operation, it can use 16 threads

operating.

- In ConcurrentHashMap, the Object is divided into a number of segments according to the concurrency level. The default concurrency-level of ConcurrentHashMap is 16.

Why is 16? Because we have 16 buckets.

When concurrentHashMap is initialized, the table will be initialized as **16 buckets. Everyone has one thread.** When need to lock, lock the bucket.

HashTable is object level lock, lock whole array.

ConcurrentHashMap is bucket level lock, lock one bucket. It never affect other bucket.

ConcurrentHashMap has a relatively higher performance and thread safe.

hashmap
- key-value
- allows one null key, which will be in the first bucket(index 0)

hashtable
- key-value
- thread safe
- object level lock
- don't allow null key

concurrentHashMap
- key-value
- thread safe
- bucket level lock

HashSet <- HashMap
- HashMap : <Key, Value>
- HashSet: <Key, Dummy Object> == hashmap.keySet();

```
Set<Integer> set = new HashSet<>();
|- Interface-|     |- Implement Class-|

//-------------
set.add(1);
set.add(1);
System.out.print(set.size()); // 1

//-------------
Set<newType> set = new TreeSet<>();// lack a comparator (define the order)
// 1. implements Comparable<> and Override the compareTo()
//
// 2.

// if there is not a comparator or implement comparable,
// Exception: ClassCastException
```

Nots: TreeSet, TreeMap. TreeMap is same.

Stack previses Thread Safe on the Object level, which comes lots of overhead - not better performance. → Deque

add from head and tail.

ArrayDeque (better performance)

Implement by circular array

stored in consecutive space

memory location aspect

Deque also can be implemented as a double LinkedList.

Deque<> de = new LinkedList<>();

fine to use stack ?! 0:47:00

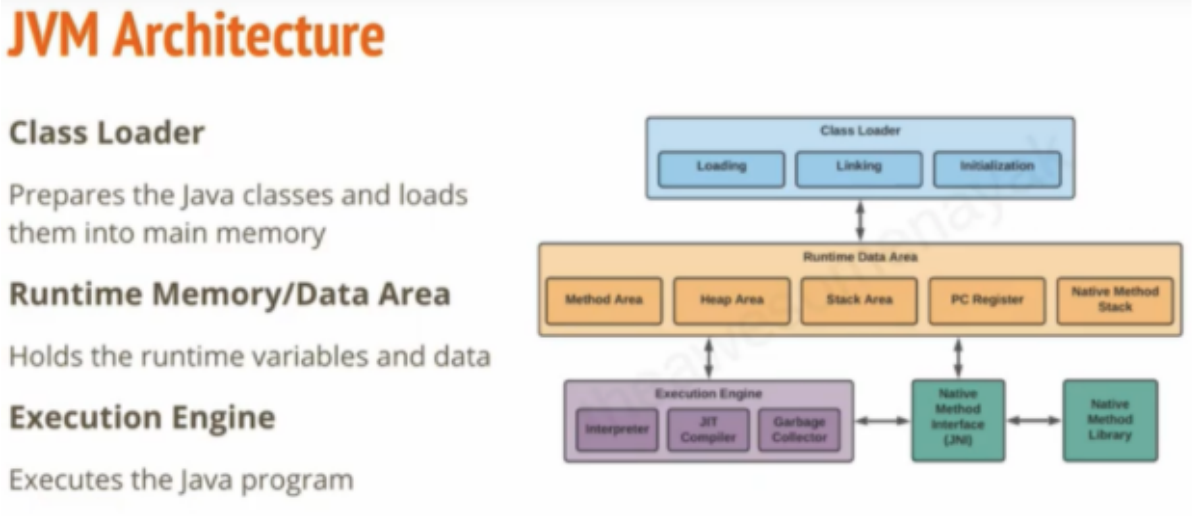
8. Comparator vs. Comparable

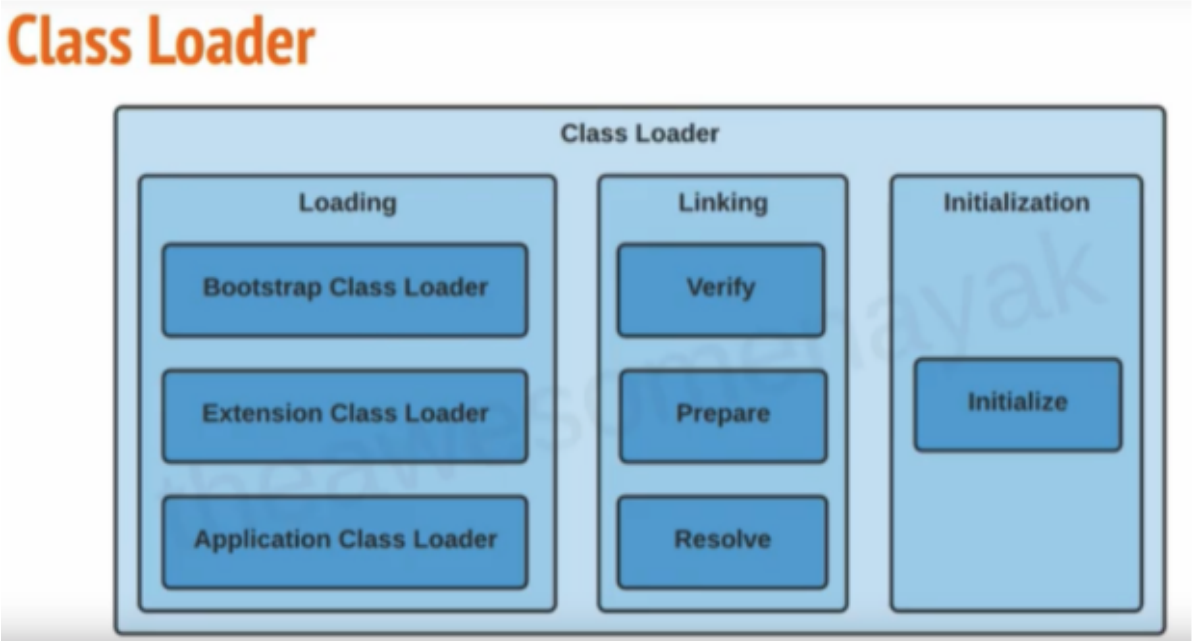
Comparator defines external order.

Comparable defines internal order.

9. JVM

Java Virtual Machine





- Class Loader
- Runtime Data Area
- Java Native Interface(JNI)
- Native Method Library

Class Loader

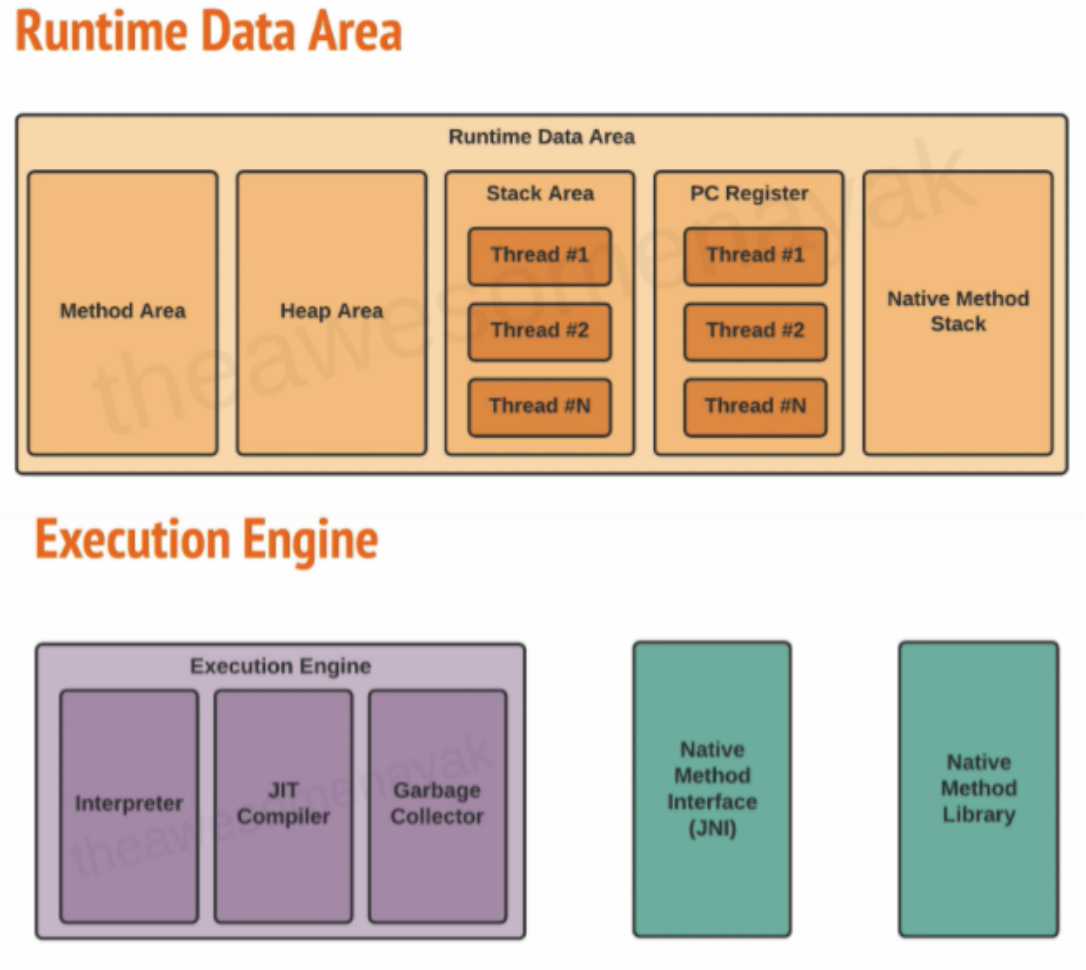
- Loading - load everything into JVM

  All the classes and dependences will be loaded.

  - Booststrap
    - load the Java packages like java.lang.* and java.net.*
    - which not need to use import, and by C/C++ (low level)
  - Extension
    - load the extension classes like JDBC driver
  - Application
    - load classes in application classpath

  super relationship (Booststrap → Extension → Application)
- Linking - three stages (compile)
  - verify
    - verify the correctness of all .class files
  - prepare
    - load? all static fields, classes and Interface
  - resolve
    - replace symbolic reference to directly reference
- Initialization
  - initialize everything

- static fields
- final variables
- static methods
- static classes



Runtime Data Area

- Method Area - all the class level data
  - String/Integer constant pool
  - all classes

- Heap Area
  - new some object will be in the heap
- Stack Area
  - different kinds of stack frames
    - each stack frames contains local variables, operating stacks and some frame data
- PC Register (program counter)
  - addresses of current execution JVM instruction
- Native Method Stack
  - create stacks and call other libraries written by C/C++ ...
  - JVM implementations cannot load native methods and cannot rely on conventional stacks .

Execution Engine

- Interpreter - byte code
- JIT Compiler - improve the performance of Interpreter
  - improve the performance of Java applications by compiling platform-neutral Java byte code into native machine code at run time.
- Garbage Collector

Native Method Interface(JNI) - Interface, to access NML

- bridge, make java invoke the Native Method Library

Native Method Library

- written by other languages (C/C++, the lower level languages)

Homework 2.1

- what is JVM? do some research

- play around comparator, comparable, data structures

What is JIT?

Stack Area

what is call stack

what is heap Area

what is native method stack?

why call it native